Experimental Validation of Cyber-Resilient Electrical Drive Systems via Encrypted Controller Implementation

M. Baldi [1], E. Brasili [1], F. Chiaraluce[1], R. El Mechri [1], G. Ippoliti [1] and G. Orlando[1]

***Abstract*— In modern industrial applications, the implementation of protective measures for electric drive systems is paramount to maintaining system reliability. As these systems rely on networked communication among their interconnected control units, they face a significant risk of exploitation by cyber adversaries. This study investigates secure end-to-end communication within a DC motor drive system. The methodology employs a practical networked control system, implemented on a Raspberry Pi platform, which integrates a motor drive and an encrypted controller. The functionality of the encrypted controller extends beyond securing communication links to also protecting the confidentiality of its internal parameters. The experimental results confirm that encryption successfully obfuscates both the internal controller parameters and the transmitted signals. Moreover, the experimentally evaluated processing overhead of the encrypted controller is shown to be compatible with real-time operation.**

***Index Terms*— Encrypted Control; Networked Control Systems; Industrial Cybersecurity;**

I. Introduction

The design of modern electric drive systems incorporates networked communication capabilities to facilitate remote supervision and operation. These systems enable precise control over machinery in diverse applications, spanning from fundamental industrial equipment, such as conveyor belts, to advanced systems like complex robotic manipulators. Such precision is essential for modern industrial production and process automation [1], [2], [3]. Integration of electric drives within automation architectures is essential for achieving high-fidelity control. These drives are vital components in numerous industrial settings — including, but not limited to, lifts, pumps, and machine tools — which are progressively transitioning to computer-based monitoring and management. The evolution toward Industry 5.0, characterized by the integration of cyber-physical systems (CPS), artificial intelligence (AI), and the Internet of Things (IoT), results in increasingly interconnected and networked operational environments [4], [5]. The integration of networked architectures, while promoting higher efficiency and facilitating data analytics, inevitably results in an enlargement of the system's threat vector. This structural change exposes electrical drive systems to novel and exploitable cybersecurity vulnerabilities [6], [7]. Effective cybersecurity measures are essential to mitigate the potential impact of cyber threats on industrial drive systems, particularly those that could result in operational failures, equipment damage, or compromise personnel safety.

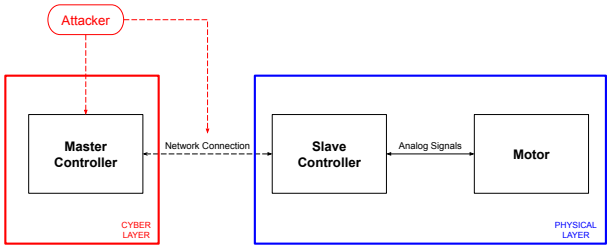

Fig. 1: Threat model of a distributed control system

These threats manifest themselves in various forms, including denial of service (DoS) [8], man-in-the-middle (MITM) attacks [9], code injection [10], ransomware [11], and targeted intrusions [12]. The fundamental objective of such attacks is to compromise the system's core security properties: confidentiality, integrity and availability. Effective countermeasures against these cyber attacks are imperative to avoid economic losses and, above all, to ensure the protection of critical infrastructures.

The man-in-the-middle (MITM) attack is a particularly potent cyber threat, posing a substantial risk to the operational integrity and reliability of electrical drive systems [13]. The MITM attack involves an adversary covertly intercepting and potentially modifying communication exchanged between two legitimate entities [14]. By manipulating the data transmitted to the electrical drive, an adversary can disrupt its nominal operation, resulting in potential equipment damage and the introduction of operational inefficiencies. The injection of incorrect commands into the machinery can lead directly to system malfunction or unsafe operation. Since industrial drive systems control critical processes, unauthorized alteration of control signals also poses severe safety risks to both personnel and equipment. For instance, an attacker could manipulate control signals to induce abrupt changes in drive parameters (e.g., speed or torque), which may subsequently trigger mechanical failure.

The system model we are considering, schematically represented in Fig. 1, is suitable for modeling a modern distributed control infrastructure, in which there are:

- one or more *master* controllers, which have high computing power and can implement advanced control

*This work was partially supported by project SERICS (PE00000014) under the Italian Ministry of University and Research National Recovery and Resilience Plan, funded by the European Union - Next Generation EU.

[1]Authors are with *Dipartimento di Ingegneria dell'Informazione* Università Politecnica delle Marche 60131 Ancona, Italy m.baldi@univpm.it, e.brasili@pm.univpm.it, f.chiaraluce@univpm.it r.elmechri@staff.univpm.it, g.ippoliti@univpm.it, g.orlando@univpm.it

logic; due to their exposure to the network, they represent the main attack surface of the system;
- one or more motors or other types of mechanical devices, which are subject to control;
- a *slave* controller near each motor, which transfers control signals to it and acquires measurement signals from it.

Since a network connection must be used between the master and slave controllers, it must be cryptographically protected to prevent the attacks described above. A classic solution in this regard consists of securing network communications using protocols based on pre-shared keys or classic asymmetric encryption, such as Transport Layer Security (TLS). In that case, however, the data within the master controller would be in unprotected and readable. Consequently, should an adversary attain unauthorized access — for instance, by compromising the control equipment — sensitive system assets become vulnerable to exfiltration and data manipulation. This critical vulnerability extends beyond control signals and system information, and can even compromise the encryption keys used to protect network communications.

A solution that addresses this problem is the one proposed in [15], [16], [17], which secures the master controller computation using encryption techniques with homomorphic properties, including the well-known RSA [18] and ElGamal [19] cryptosystems. According to this approach, the slave controller is the sole owner of encryption keys, and it only exchanges encrypted data with the master controller. The latter can perform its calculations directly in the encrypted domain, by exploiting the homomorphic properties of the encryption algorithms used by the slave controller. In this way, the master controller does not have access to cleartext data, which offers an additional layer of protection against intrusion attempts. A further advantage of this approach is the elimination of the requirement for the master controller to store any secret keys for control input calculation. This is achieved because the control input is computed directly from the encrypted measurements and the encrypted controller parameters, bypassing the need for an internal decryption process. Consequently, the master controller exclusively handles ciphertexts, a feature that significantly enhances the overall security of the networked control system.

The objective of this work is twofold: first, to verify the performance of the encrypted control scheme through its implementation on a practical networked control system utilizing a Raspberry Pi and a DC motor; second, to assess the real-time processing latency of the resulting control system. The specific contribution lies in the experimental realization and quantitative assessment of RSA-based homomorphic encrypted control on physical hardware, demonstrating its compatibility with real-time operation in an industrial drive context.

The structure of the paper is as follows. Section II presents the design of the controller and its encryption, while Section III describes the experimental control platform. Section IV reports and comments on the experimental results concerning the execution overhead of the encrypted controller for several key sizes. Finally, Section V provides some conclusive remarks.

II. Encrypted Control Scheme

The proposed method for regulating the speed of the DC motor employs a PI controller, with its parameters selected by means of a pole-placement design procedure. Assuming that the transfer function of the plant is known, the closed-loop characteristic polynomial of the system is set equal to a desired polynomial of the same order. This procedure yields a unique solution for the parameters of the PI controller.

A. Controller design

Given that the plant under control is a DC motor, its low-frequency dynamics may be represented by a single-pole transfer function. Neglecting electrical transients, the dominant dynamics are mechanical, yielding the simplified process model

$$P(s) = \frac{b}{s+a} \tag{1}$$

The PI controller is represented by

$$C(s) = \frac{K_P s + K_I}{s} \tag{2}$$

where K_P is the proportional gain and K_I is the integral gain. The closed-loop transfer function is expressed as

$$\begin{aligned} T(s) &= \frac{C(s)P(s)}{1 + C(s)P(s)} \\ &= \frac{\dfrac{b(K_P s + K_I)}{s(s+a)}}{1 + \dfrac{b(K_P s + K_I)}{s(s+a)}} \\ &= \frac{b(K_P s + K_I)}{s(s+a) + b(K_P s + K_I)} \end{aligned} \tag{3}$$

In order to place the closed-loop poles at the desired values, the following equation is established

$$s(s+a) + b(K_P s + K_I) = s^2 + 2\xi\omega_n s + \omega_n^2 \tag{4}$$

where the left-hand term denotes the characteristic polynomial of the closed-loop system, while the right-hand term specifies the target polynomial. From (4) we can derive the following matching conditions:

$$a + bK_P = 2\xi\omega_n, \qquad bK_I = \omega_n^2 \tag{5}$$

Starting from (5) the parameters K_P and K_I can thus be determined based on the known model parameters a and b, as well as the chosen natural frequency ω_n and the damping coefficient ξ of the desired polynomial. Hence, the coefficients of the controller can be determined through the following expressions

$$K_P = \frac{2\xi\omega_n - a}{b} \tag{6}$$

$$K_I = \frac{\omega_n^2}{b} \tag{7}$$

In the continuous-time domain, the PI Controller can be written as

$$u(t) = K_P e(t) + K_I I(t) \tag{8}$$

where $e(t)$ is the control error and $I(t) = \int_0^t e(\tau)\, d\tau$.

Let the controller be implemented digitally with sampling period T_s. Using the forward Euler approximation for the integral state,

$$\frac{I[k+1] - I[k]}{T_s} \approx e[k] \tag{9}$$

which gives

$$I[k+1] = I[k] + T_s e[k] \tag{10}$$

The discrete-time control law becomes

$$u[k] = K_P e[k] + K_I I[k] \tag{11}$$

B. Controller encryption

Homomorphic encryption allows a third party to perform computation on encrypted data without disclosing information on unencrypted data. In this scenario, there is an interest in not revealing the control values to a remote PI controller. Through Fully Homomorphic Encryption (FHE) it is possible to perform both additions and multiplications in the encrypted domain. Unfortunately, however, current FHE schemes have a high computational cost, and are therefore unsuitable for real-time control. Consequently, following an approach already proposed in the literature [17], we rely on Partially Homomorphic Encryption (PHE) cryptosystems, which allow for either multiplications or additions in the encrypted domain, but with less computational overhead. In particular, we opted for the RSA cryptosystem, which is a public key cryptosystem, and as such it can be described by three functions $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$:

- $\mathsf{KeyGen}(\lambda)$: on input a security parameter λ, outputs a public key (e, n) and a private key d where
  - n is the public modulus, which is the product of two large primes p and q, $\log_2(n) \approx \lambda$;
  - e is the public exponent, a positive integer such that $e < (p-1)(q-1), \gcd(e, (p-1)(q-1)) = 1$;
  - d is the private exponent, a positive integer such that $e \cdot d \equiv 1 \mod (p-1)(q-1)$.
- $\mathsf{Enc}(m)$: on input an integer message $m < n$ outputs $m^e \mod n$;
- $\mathsf{Dec}(c)$: on input an integer ciphertext $c < n$ outputs $c^d \mod n$.

RSA is homomorphic with respect to multiplications; in fact, given two plaintexts m_1, m_2:

$$\begin{aligned}\mathsf{Dec}(\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2)) &= \mathsf{Dec}(m_1^e \cdot m_2^e \mod n) \\ &= \mathsf{Dec}((m_1 \cdot m_2)^e \mod n) \\ &= (m_1 \cdot m_2)^{e \cdot d} \mod n \\ &= m_1 \cdot m_2\end{aligned}$$

The security of RSA is based on the hardness of the *factorization problem*, which given an integer asks to find its unique prime factorization. In fact, an adversary able to factorize n would be able to find p and q, and consequently find the secret key d. Although the factorization problem can be solved in polynomial time with a quantum computer, and post-quantum alternatives are already being standardized, RSA remains widely used in commercial and industrial applications. The security parameter suggested for RSA by the National Institute of Standards and Technology (NIST) is 2048 bits.

In this work, RSA is employed to encrypt the PI controller's internal parameters, its reference, and its input, such that, by exploiting the multiplicative homomorphism of RSA, the proportional and integral multiplications are performed in the encrypted domain, that is,

$$P = \mathsf{Enc}(K_P) \cdot \mathsf{Enc}(e) \mod n \tag{12}$$

$$I = \mathsf{Enc}(K_I \cdot T_s) \cdot \mathsf{Enc}(e) \mod n \tag{13}$$

The encrypted multiplications are executed on the *master* unit, while the additions are performed, in the domain of plaintext data, on the *slave* unit — the plant-side unit responsible for signal encryption and decryption. Comprehensive details regarding the system components and their respective functionalities are presented in Section III.

C. Conversion from real value to plaintext

An issue that has been neglected in Section II-B for simplicity, is that of encoding the control system variables into the plaintext space. RSA requires the input to the encryption function to be a positive integer smaller than the public modulus n. Control system variables take real values that can be both positive and negative. Therefore, before encrypting the variables that must be sent to the PI controller, a fixed-point encoding is necessary. This is performed by scaling the values by a power of 10 corresponding to the required precision, and then rounding to the nearest integer. If the value is negative, a sign reversal is performed. Upon receiving the values back from the controller, the values are scaled back and the sign is inverted when needed.

III. Experimental Testbed

We developed a DC motor-driven mechanical control system (illustrated in Fig.2) to serve as a testbed for examining and assessing the proposed controller encryption scheme. The results presented focus on assessing the applicability, quantifying the performance degradation induced by the encryption process, and determining the real-time computational overhead.

The experimental platform utilizes a DC motor whose shaft velocity is measured by an integrated tachometric generator. Table I summarizes the characteristics of the DC Motor employed in the experimental activities. Control and data exchange between the system's components are managed by two Raspberry Pi units communicating over an Ethernet link (see Table II for Raspberry Pi units specifications).

Characteristic	Value	Unit
Nominal Torque, locked rotor	2.15	Nm
Peak Torque, locked rotor	8	Nm
Maximum Speed	6000	rpm
Rotor Inertia	1100	$10^{-6}\,\mathrm{kgm}^2$
Mechanical time constant	16	ms
Electrical time constant	6.10	ms
Speed constant	0.58	rad/s/Nm

TABLE I: Selema Series 30 Motor + Tachometric generator Specifications

	Raspberry Pi 3 B (master)	Raspberry Pi 3 B+ (slave)
CPU	1.2 GHz quad-core A53	1.4 GHz quad-core A53
RAM	1 GB LPDDR2	1 GB LPDDR2
Ethernet	100 Mbps	300 Mbps
OS	Raspbian GNU/Linux 8.0	Raspbian GNU/Linux 11
Kernel	4.4.50-v7+	6.1.21-v7+
GCC	4.9.2	10.2.1

TABLE II: Raspberry Pi Platforms Specifications

Both Raspberry Pi units are configured with a real-time Raspbian-Linux environment. The control system tasks are distributed such that one unit ('*master*'), performs the controller homomorphic multiplications, and the second ('*slave*') manages the plaintext addition to compute the control voltage, the cryptographic operations, sensor input/reading, and the actuation of control commands for the DC motor. The system plant consists of a DC motor whose dynamics are assumed to be linear. The system input is the voltage applied to the motor driver, and the output is the velocity feedback provided by the tachometer. A transfer function was derived for the plant through black-box identification, resulting in:

$$P(s) = \frac{46}{s + 42} \tag{14}$$

The most suitable controller parameters identified are $K_P = 0.00135$ and $K_I = 0.035$. It is important to emphasize that, since these parameters are computed entirely offline, they are preloaded in encrypted form into the master to enable the computation of the control law. The RSA encryption scheme was implemented using key lengths of 512 and 1024 bits, representing security levels that closely approximate practical and realistic settings. This choice allows for the evaluation of control performance under different encryption strengths, providing insight into the trade-offs between computational load, dynamic response and system security.

Experimentally, the control loop interval was adjusted to accommodate the increased computational load associated with longer RSA keys. As key length increased, the encrypted operations became more time-consuming, necessitating longer intervals between successive control cycles to ensure proper execution and maintain system stability. For the unencrypted controller, a control loop interval of 20 ms was selected. When using RSA encryption, the interval was increased to 25 ms for 512-bit key and 80 ms for 1024-bit key. These control intervals were determined empirically as the minimum values ensuring stable and reliable system operation under the considered conditions.

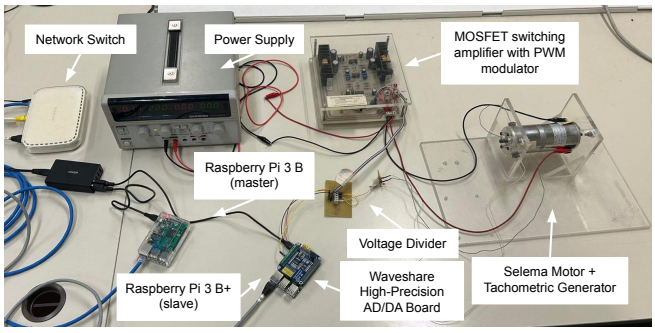

Fig. 2: Experimental setup

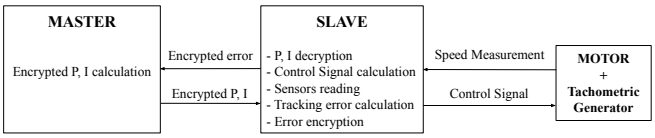

Fig. 3: Master-slave control loop with homomorphic encryption

Fig. 2 depicts the experimental setup, while Fig. 3 presents a flow diagram describing the encrypted control loop. The slave, utilizing a Waveshare high-precision AD/DA module, reads the rotational velocity from the tachometer. This unit is responsible for calculating the velocity error, encrypting the error signal, and subsequently transmitting the encrypted value to the master across the network. The master exploits homomorphic properties to create a multi-signal encrypted representation of the control input, which is then returned to the slave. Finally, the slave decrypts the received signals to yield the control input, which is then converted into a voltage command by the AD/DA board and applied to the motor driver circuit. All control and encryption logic deployed in the testbed system was coded in C-language. Homomorphic operations were performed using the OpenSSL library, taking advantage of its implementation of the RSA cryptosystem and integer modulo operations.

IV. Performance Assessment

Despite the computational intensity inherent to the introduction of the encryption scheme, which results in a slower control loop, the degradation in system performance remains limited. Fig.4 compares the motor speed responses under three control configurations: unencrypted, encrypted with a 512-bit RSA key, and encrypted with a 1024-bit

RSA key. Fig.5 illustrates the control signal profiles for the three considered configurations. Minor delays are observed as key length increases, yet the controller maintains effective regulation in all cases. This demonstrates that secure control can be achieved without substantially compromising system dynamics, even under significant computational load arising from the encryption and decryption processes as well as from the execution of homomorphic multiplication operations. The observed delays are primarily due to the computational and memory overhead of the Raspberry Pi units, as detailed in Table III, which reports the mean times of encryption, decryption, inter-device communication and control cycle execution, averaged over the entire simulation.

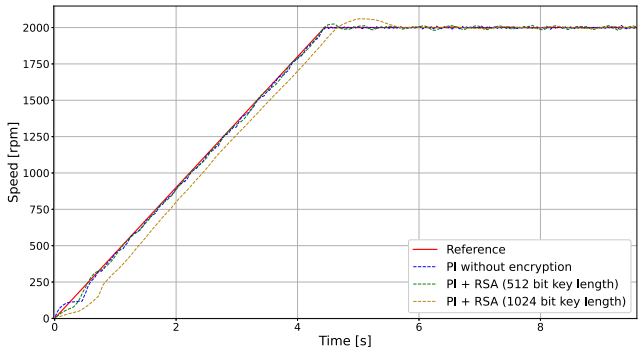

Fig. 4: Motor Speed signals comparison: reference speed (red continuous line), unencrypted PI (blue dashed line), encrypted PI with RSA (512 bit key length) (green dashed line), encrypted PI with RSA (1024 bit key length) (yellow dashed line)

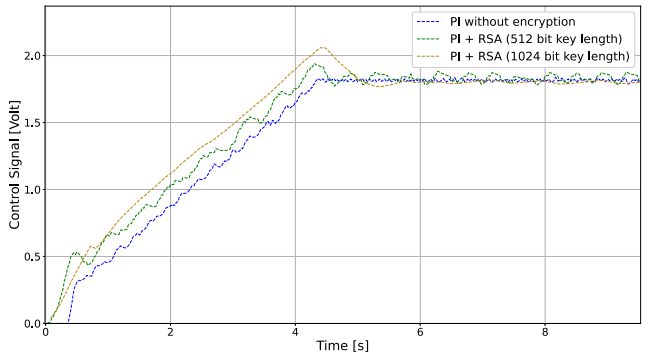

Fig. 5: Control signals comparison: unencrypted PI (blue dashed line), encrypted PI with RSA (512 bit key length) (green dashed line), encrypted PI with RSA (1024 bit key length) (yellow dashed line)

From the control loop time values (reported in the last column of Table III), it is possible to observe how the mean execution time of the control cycle grows with key length, highlighting a direct correlation between cryptographic strength and computational latency. Specifically, an increase in the RSA key length of 512 bits lead to an increase in execution time of approximately 12 ms per control cycle. While longer key lengths inherently improve security, they also impose stricter timing constraints in real-time control applications. Consequently, the key length must be carefully selected to ensure that the overall execution time remains compatible with the sampling period of the control loop. In this respect, the experimental results validate the temporal overhead of the proposed encrypted control scheme and confirm its compliance with deterministic control requirements when appropriate control loop intervals are adopted. These results indicate that handling large RSA keys and intermediate encrypted data contributes to the extended cycle duration, yet the control performance remains acceptable, confirming the feasibility of implementing encrypted control on resource-constrained hardware.

Key length	**Encryption time (ms)**	**Decryption time (ms)**	**Communication time (ms)**	**Control loop time (ms)**
No encryption	-	-	0.4602	18.0432
512	0.2770	2.4964	0.9005	21.1896
1024	0.7670	13.3198	1.3996	33.1628

TABLE III: Average computational times for encryption, decryption, and communication at different key lengths

To further quantify performance, an analysis of the Integral Absolute Error (IAE) and Mean Squared Error (MSE) indices of the speed signals was conducted for the steady-state interval between 6 and 9 seconds of the test (Fig.6). The calculated values of the aforementioned indices are presented in Table IV. For the unencrypted PI controller, the IAE and MSE values are 10.81 and 18.97, respectively, representing the baseline performance. When encryption is applied with a 512-bit RSA key, both indices increase significantly (IAE = 21.66, MSE = 72.37), indicating a noticeable degradation in tracking accuracy. The improved IAE and MSE observed in the 1024-bit key configuration (11.28 and 21.91, respectively), despite its higher computational burden, can be attributed to the longer control-loop interval adopted in this setup. Specifically, the increased interval reduces the sensitivity of the control action to measurement noise, effectively introducing an implicit low-pass filtering effect that enhances steady-state tracking accuracy, at the cost of a slightly slower transient response. Nevertheless, the 1024-bit configuration still guarantees satisfactory overall control performance, demonstrating that stronger cryptographic protection can be achieved without significantly compromising system stability and tracking capability.

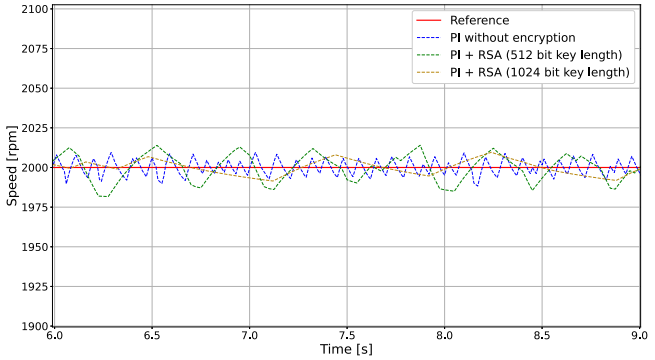

Fig. 6: Zoom of the motor speed signals in steady state

Controller	**IAE**	**MSE**
Unencrypted PI	10.81	18.97
PI + RSA (512 key length)	21.66	72.37
PI + RSA (1024 key length)	11.82	21.91

TABLE IV: IAE and MSE values for the analyzed configurations

V. Conclusion

An encrypted PI control approach based on closed-loop pole placement has been presented, employing the RSA algorithm as cryptographic scheme. Experiments were conducted with both 512-bit and 1024-bit keys, selected to approximate a realistic security scenario and to assess the impact of key length on computational load and control performance. Despite the substantial computational burden imposed by the encryption scheme implementation, the control performance remained acceptable. Although a delay in the control action was observed, it remained within tolerable limits for the considered application.

Furthermore, the computational latency of the encrypted control scheme was evaluated as a function of key length. This analysis established a clear correlation between cryptographic strength and processing overhead, characterizing the temporal transitions across various key dimensions.

Future work may involve the investigation of alternative PHE schemes, or the adoption of an FHE scheme, contingent upon an appropriate upgrade of the hardware platform, as the current system lacks sufficient computational capability to support more demanding cryptographic operations. The use of FHE, in particular, would enable the exploration of more advanced control strategies, potentially extending the applicability of encrypted control to more complex systems.

References

[1] J. Dheeba, V. Oberoi, R. R. Singh, and V. G. Karthik, "Securing Electrical Drive Systems Against Man-in-the-Middle Attacks Using S-Box Optimized AES Encryption," *IEEE Access*, vol. 13, pp. 114 716–114 735, 2025.

[2] L. Szentirmai, *Considerations on the Industrial Drives*. Dordrecht: Springer Netherlands, 2000, pp. 687–722.

[3] D. Kalel and R. Raja Singh, "Iot integrated adaptive fault tolerant control for induction motor based critical load applications," *Engineering Science and Technology, an International Journal*, vol. 51, p. 101585, 2024.

[4] R. R. Singh, G. Bhatti, D. Kalel, I. Vairavasundaram, and F. Alsaif, "Building a digital twin powered intelligent predictive maintenance system for industrial ac machines," *Machines*, vol. 11, no. 8, 2023.

[5] D. Zhong, Z. Xia, Y. Zhu, and J. Duan, "Overview of predictive maintenance based on digital twin technology," *Heliyon*, vol. 9, no. 4, p. e14534, 2023.

[6] B. Yang, L. Guo, F. Li, J. Ye, and W. Song, "Vulnerability assessments of electric drive systems due to sensor data integrity attacks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3301–3310, 2020.

[7] K. I. Cert. (2023) Threat landscape for industrial automation systems statistics for h1 2023. [Online]. Available: https://ics-cert.kaspersky. com/publications/reports/2023/09/13/threat-landscape-for-industrial- automation-systems-statistics-for-h1-2023/

[8] M. vall O. Mohamed, A. Y. Abdelaziz, and F. K. Abo-Elyousr, "Blockchain-based approach for load frequency control of smart grids under denial-of-service attacks," *Computers and Electrical Engineering*, vol. 116, p. 109150, 2024.

[9] M. Thankappan, H. Rifa-Pous, and C. Garrigues, "Multi-channel man-in-the-middle attacks against protected wi-fi networks: A state of the art review," *Expert Systems with Applications*, vol. 210, p. 118401, 2022.

[10] H. A. Noman and O. M. F. Abu-Sharkh, "Code injection attacks in wireless-based internet of things (iot): A comprehensive review and practical implementations," *Sensors*, vol. 23, no. 13, 2023.

[11] K. Ashwini and K. Nagasundara, "An intelligent ransomware attack detection and classification using dual vision transformer with mantis search split attention network," *Computers and Electrical Engineering*, vol. 119, p. 109509, 2024.

[12] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157 761–157 779, 2021.

[13] N. M. Nambiar, R. P, and R. Lal, "FPGA implementation of multibit LFSR as key generator for AES encryption," in *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2022, pp. 231–237.

[14] O. Bazgir, S. Gali, and T. Nikoubin, "Area-power and Energy Efficient Substitution box (S-box) in Advanced Encryption Standard (AES)," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, ser. GLSVLSI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 263–267.

[15] K. Kogiso, R. Baba, and M. Kusaka, "Development and examination of encrypted control systems," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018, pp. 1338–1343.

[16] K. Ishikawa, K. Nagasawa, K. Kogiso, and K. Sawada, "Experimental validation of encrypted controller implemented on raspberry pi," in *2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, 2016, pp. 1–6.

[17] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 6836–6843.

[18] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, Feb. 1978.

[19] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.