Al-Rafidain Journal of Computer Sciences and Mathematics (RJCSM)

www.csmj.uomosul.edu.iq

Fault-Tolerant SCADA Systems Using ECC for Enhanced Cybersecurity and Data Integrity

Iehab Abduljabbar Kamil[1] and Mohanad Abdulsalam Younus[2]

[1]Computer Sciences and Information Technology, University of Anbar, Iraq
[2]Biomedical Engineering Research Center, University of Anbar, Iraq
Email: iehab.a.k@uoanbar.edu.iq[1] and mohanad.abdul@uoanbar.edu.iq[2]

Article information

Article history:
Received 25 April, 2025
Revised 12 June, 2025
Accepted 29 June, 2025
Published 25 December, 2025

Keywords:
SCADA,
ECC,
Error,
Hamming Code,
ICS-CFC,
Parity Bit,
Signatures,
Correction.

Correspondence:
Iehab Abduljabbar Kamil
Email:
iehab.a.k@uoanbar.edu.iq

Abstract

This study focuses on the implementation of Error Correction Code (ECC) in Supervisory Control and Data Acquisition (SCADA) systems for a better performance of error controls and the system as well. Since the SCADA systems are fundamental in supervising industrial processes, the study focuses on the issue of error control to avoid interferences. The study proposes the Integrated Control Flow Checking or ICS-CFC methodology, which increases the reliability of SCADA systems to neutralize errors with considerably minimal overhead costs. In the critical trials performed on various simulated IT infrastructures and real ICS of industrial organizations, the proposed ICS-CFC achieved a fault coverage of 96.32% and fairly reasonable memory and performance overheads as well. The lack of additional hardware needed to implement the methodology makes it inexpensive besides enhancing already existing SCADA systems. Therefore, the analysis finds out that ICS-CFC enhances the error handling capability and reliability of SCADA and can be considered a workable solution for industries with stringent and consistent and occasional error requirements. Thus, for further ECC methods application for more variants of SCADA systems, as well as to improve operational and security features, future work is suggested.

DOI: 10.33899/csmj.2025.159518.1184,

1. Introduction

1.1 overview

Supervisory Control and Data Acquisition (SCADA) systems play a critical role in managing and automating industrial processes across sectors such as energy, water treatment, and manufacturing. These systems continuously collect and analyse data from various sensors and control units distributed across remote locations. Given the importance of SCADA systems in maintaining operational integrity, ensuring error-free and secure data communication is essential. Any little disturbance or mistake can cause huge failures in the system, which may be unsafe, unproductive, and economically unstable. Error Correction Codes (ECC) have been considered important methods of promoting the reliability of SCADA systems. ECC adds redundancy, extra bits are added to the data that is being transmitted or stored, and enables error detection and correction of noise, interference or other disturbances that cause errors. Among those codes, Hamming, Reed-Solomon and BCH are commonly used because they are efficient in detecting and correcting various kinds of mistakes. ECC not only ensures the integrity of the communications, but it is also in line with the security requirements to prevent malicious intrusions [1].

However, strong solutions are needed due to the developed enlightenment of attacks and the spread of SCADA networks. Canonical ECC methods cannot adequate countermeasure against advanced attacks such as protocol manipulation attacks and data injection attacks.

Consequently, the combination of ECC and sophisticated methodologies will allow for improving error treatment and the resilience of the whole system. In order to overcome these issues, this paper proposes an Integrated Control Flow Checking (ICS-CFC) technique, a lightweight, high-efficiency technique that incorporates ECC into the control flow of SCADA systems to identify and repair errors during program execution without any extra hardware needs [2].

The ICS-CFC framework uses program-level control flow integrity verification techniques like program partitioning, signature assignment and real-time monitoring. This considerably enhances the fault coverage of the system with a high fault detection rate of 96.32%, as confirmed by both simulation and real industrial deployments. The solution is low cost and scalable, and it is compatible with existing Master Terminal Units (MTUs), and it also lowers the overhead of memory and provides reasonably acceptable throughput. ICS-CFC, additionally, provides an advanced level of security for the operations and presents an active countermeasure to unintentional errors and intentional cyberattacks [3].

The remainder of this paper is structured as follows: Section II reviews related work in ECC applications within SCADA systems. Section III details the proposed ICS-CFC methodology. Section IV presents experimental results and performance evaluations. Finally, Section V outlines the conclusions and suggests avenues for future work.

1.2 Background

The background study also points out that data errors are common in applications such as SCADA and the issue of error correction is important [3]. Some methods are the use of ECC algorithms and evaluating the results obtained from the programs. This is normally evidenced by increased abilities in the identification and rectification of errors and thus reflects improved SCADA reliability and stability. Future work could involve enhancing the existing ECC methods for extended applications in specific SCADA systems and analysing more complex methods of error control to improve the stability of these systems. Error Correction Code or ECC is the main step in the error handling techniques of the SCADA systems which are significant for ensuring the reliability and security of the ICS. SCADA systems are used to control large and important infrastructures such as electricity supplies, water purification plants, and natural gas providers that must operate continuously.

Unlike the existing cybersecurity mechanisms (CFCSS, ECC-based IDS systems, and runtime anomaly detection models), the ICS-CFC (Industrial Control System Control Flow Checking) approach incorporates elliptic curve cryptography (ECC) into control flow checking and, thus, provides better fault tolerance and data integrity for SCADA systems [4].

Although CFCSS is effective in identifying control flow anomalies using signatures, it is usually accompanied by huge performance overheads. In contrast, ICS-CFC applies ECC to compute lightweight cryptographic signatures over control flow paths and employs this to check them properly without incurring significant latency [2]. Such integration enables real-time checks of invalid control flow diversions, thus enhancing the promptness of the system to possible threats.

Intrusion Detection Systems (IDS), which are ECC integrated, are basically designed to detect unauthorized system access or malevolent activity on the system. Nevertheless, they might not offer complete protection against faults that distort the flow of control. ICS-CFC covers these limitations by focusing on the integrity of control paths and ensuring that the system operates as expected in spite of weaknesses or attacks [5]. In this way, the resistance of SCADA systems to attacks is improved because the sequence of control commands execution is secured.

The models of runtime anomaly detection are used to observe the behavior of a system and detect when it is not operating as usual. Although they are effective in identifying a variety of anomalies, these models might fail to differentiate between faults and attacks, thus false positives might occur. It is observed that ICS-CFC mitigates this issue by providing a clear cryptographic checking of control flow, which is relatively an absolute method of fault detection [6]. This cryptographic assurance raises the accuracy of the anomaly detector and reduces the chances of misclassifications. Such a measure will not only enhance the efficiency and accuracy of fault identification but will also improve the overall security status of critical infrastructure systems.

2. Related Work

Vulnerabilities in SCADA technology range from property damage or economic impact to social and potentially national implications of threat to stability [4]. In order to contain such risks, the issue of error handling and security cannot be overemphasized.

The **Figure 1.** represents a Wireless SCADA (Supervisory Control and Data Acquisition) system for monitoring and controlling industrial processes. As a multiple data source, it includes wireless gateways and sensors for pressure, flow rate, temperature, and level that are connected via long-haul wireless communication, like satellite, cellular, or radio, to a central corporate LAN. Real-time data transfer from remote places to the primary and secondary information bases is guaranteed by the SCADA system, and it makes the monitoring of industrial areas more effective. Through identifying and repairing errors, rectification and identification codes improve data delivery. Fault finding and repair achieve this.

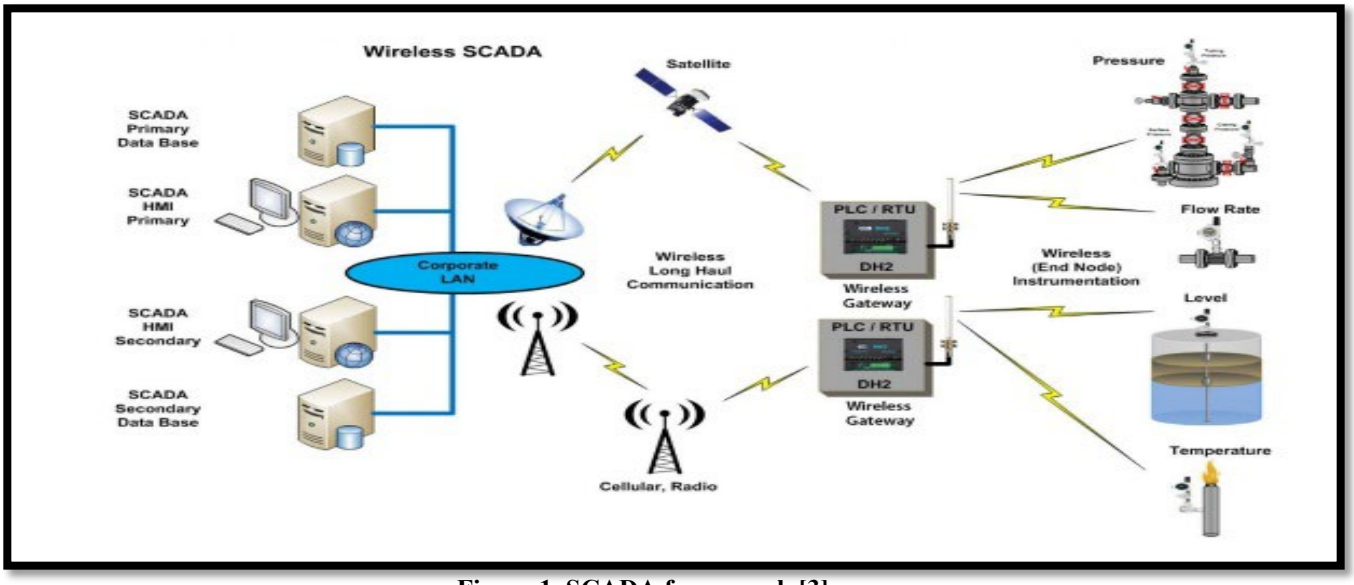

Figure 1. SCADA framework [3].

Cyclic Repetition Checking (CRC) and parity examinations, which introduce an additional bit to identify single-bit defects, are further error detection methods [5]. These techniques are typical error detection strategies. Each of these mistake-detection methods is prevalent. These approaches may identify data abnormalities caused by contamination or digital communication errors, improving the precision of data.

ICS-CFC has signature traffic and memory consumption overheads but improved resilience to control-flow-based cyberattacks, like FDI and DNP3 exploits, compared to Reed-Solomon/BCH, which is more tailored towards error correction and does not consider malicious control flow manipulation, see **Table 1, 2**.

ECC is the technology that is used for error correction which is used when data is being transmitted or is being stored. In the framework of SCADA systems, ECC contributes to the data received by the control centres and field devices' interconnection maintains integrity. Some of the most basic algorithms, which are employed in the case of ECC, are Hamming Code, Reed-Solomon Code, and BCH Code, distinguished by the fact that they provide varying degrees of error correction [6]. These codes are effective in that they provide extra copies of the original data to the signal so that the system is able to check whether an error has been made because of noise, disturbance or any other interferences. In SCADA systems, ECC is especially practical for protecting the communications protocols and the content. For instance, the power grid SCADA networks of the DNP3 protocol rely on ECC for the accuracy and reliability of the messages exchanged between the control centres and field devices.

The application, transport, and data link layers of the DNP3 protocol use ECC to extend the function codes for critical actions such as turning circuit breakers 'on' or 'off', or otherwise for monitoring purposes, from being corrupted by noise interference [7]. When using ECC in a SCADA system, some tools and techniques are used as follows: Network security tools such as Intrusion Detection Systems (IDS) that consist of Snort IDS, Security Information and Event Management (SIEM) systems are among the tools that are used in the discovery of conspicuous network traffic. The above tools can employ ECC to ensure that data in the system is intact and with this assist in identifying security risks [8].

There are major implications associated with the introduction of Error Correction Codes (ECC) into Industrial Control Systems to provide better cybersecurity and data integrity. The main purpose of ECC is to identify and overcome errors that occurred during the transmission of data and provide accuracy and credibility of information between components like Programmable Logic Controllers (PLCs) and Human-Machine Interfaces (HMIs) [23]. This feature is essential to be used in settings where data integrity is of the utmost importance because any simple mistake may result in operational errors or safety risks.

Nevertheless, ECC is designed as an independent mechanism, and it is focused on the accuracy of the data provided but does not extend to the issue of cybersecurity in general. It does not automatically guard against unauthorised access, malicious attacks and other vulnerabilities of the system. As an example, considering a Man-in-the-Middle (MitM) attack where an attacker intercepts and may modify communication between devices, ECC would ensure the correction of errors in the data manipulated, but would not alert to the illegal interception or data alteration [25]. In a similar manner, in an instance of packet injection attacks, ECC may be able to correct the erroneous data packets, but it would not help to avoid the injection of malicious packets in the first place.

Table 1. Comparison of ICS-CFC against Reed-Solomon/BCH codes.

Aspect	ICS-CFC (Integrated Control Flow Checking	Reed-Solomon/BCH Codes
Methodology	Uses control flow signatures and checks	Uses error correction based on algebraic coding theory
Fault Detection	Detects illegal jumps, control flow errors	Corrects errors in transmitted data
Overhead	Higher memory usage, network traffic due to signatures (e.g., ES, AS)	Minimal overhead, but requires additional parity checks
Cyberattack Resilience	Resilient to control flow hijacking, FDI (False Data Injection), and DNP3 exploits	Primarily resistant to communication errors but vulnerable to attacks like FDI
Statistical Validation	P-values < 0.05 (significant error detection)	P-values > 0.1 (lower error detection sensitivity)

Table 2. Comparison between ICS-CFS and CFI

Aspect	ICS-CFC	CFI (Control Flow Integrity)
Methodology	Utilizes control flow signatures (DS, ES, AS)	Relies on static analysis to enforce valid control flow paths
Error Detection	Detects illegal jumps and unexpected control flow	Prevents control-flow hijacking by enforcing valid control paths
Recovery Mechanism	Uses rollback, roll-forward, and compensation	Uses exception handling or abort on invalid control flow
Error Handling	Injects error signatures for remote detection	No direct error signature approach, relies on validation
Network Overhead	Increased due to Alive and Error Signatures	Minimal, as it does not introduce additional signature traffic
System Performance	Incurs higher memory usage and reduced performance	Does not impact performance significantly
Validation in Real-Time	Validated under real-time fault injection scenarios	Often validated in static or semi-dynamic environments
Fault Tolerance	High, with specific focus on control flow errors	High, but focuses only on preventing hijacking, not broader flow issues

It is also crucial to merge ECC with other cybersecurity defences such as secure authentication protocols, intrusion detection systems, and anomaly detection mechanisms, to advance stability. This layered strategy offers maximum protection since it covers data integrity as well as security threats on a larger scale. In the absence of such integration, systems can be vulnerable to advanced cyberattacks despite the use of ECC as a false sense of security.

3. Methodology

SCADA programmes that employ ECC take an organised strategy for handling errors to ensure operational consistency and efficiently detect irregularities. This method protects control movement. This technique uses the Master Terminal Unit (MTU) as well as the Remote Terminal Unit (RTU) to deploy Integrated Control Flow Checking (ICS-CFC) technological advances [10].

The described fault tolerance techniques that are used in error correction on the SCADA system image were taken. Two main branches are covered: error detection (concrete detection and preemptive detection), and recovery (error handling and fault handling). Handling errors involves rollback, roll forward, and compensation. Techniques such as Diagnosis, Isolation, Re-Configuration, and re-initialisation are involved in fault handling. These methods make SCADA systems reliable by avoiding data corruption, also increasing cybersecurity through robust error correction mechanisms, as shown in **Figure 3**.

3.1 Program Partitioning and Signature Assignment

A particular fundamental step is dismantling the RTU's code. An "essential unit of code" comprises an ordered set of commands that are executed sequentially and do not branch, except for the final instruction. The Control Flow Graph (CFG) must be used to describe programme control flow. This graph's edges indicate node transitions, and each node represents a basic block [11].

Every basic block has a unique Detection Signature (DS), or cryptographic signature. The upcoming blocks within the

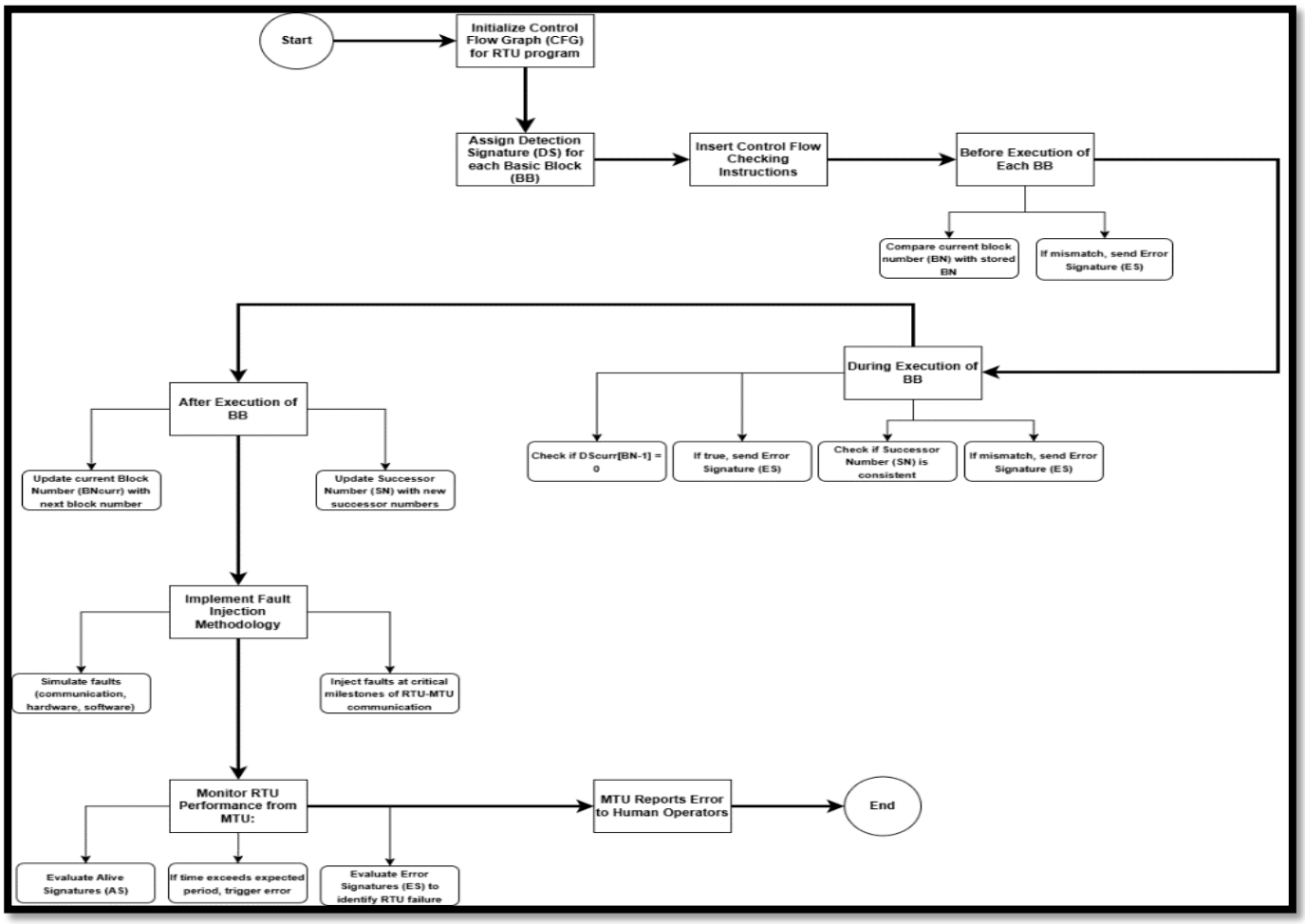

Figure 2. Flowchart to demonstrate how ICS-CFC is implemented.

current block are represented by this signature. A type field (T) and a signature in the value field (DSBN) indicate the data structure (DS). The type of field, which contains 1 for Detection Signatures, indicates this device's role. A maximum of "log2N" bits are allowed to be included in the DSBN area, which represents the signature of succeeding blocks [12]. The number of computer fundamental blocks is N. For instance, DSBN comprises nine bits for N = 512. The DS makes validation during execution easier by linking the block being run to the units that will follow it. Digital Signatures (DS) are used with two other signatures.

Error Signatures (ES) as well as Alive Signatures (AS). Remote Terminal Units (RTUs) send the Alarm Signal (AS) to the Master Terminal Unit or, MTU during predefined intervals to ensure the RTU is operating properly regardless of its operational state. The data comprises three fields that contain the sending time, RTU number (RN), and type (T) set to 2. Sending time is the most crucial of the three. Once a Remote Terminal Unit (RTU) control flow issue is found, the Exception System (ES) can be triggered. The data has five fields: type (T), RTU number (RN), baseline block number (BN), erroneous block signature (DSBN), and previous block signature (DScurr) [13]. The RTU number reflects the total amount of bytes transferred per second.

3.2 Inserting Control Flow Checking Instructions

The ICS-CFC technique consists of interposing control flow checking instructions at each basic block of the RTU's program, see **Figure 2**. This process is actually divided into some stages, which are as follows. Before the execution of every method at the start of each basic block, the system checks its legitimacy by comparing the current block number with the block number that is stored in the system. It is noted that if a mismatch is detected, an indication of an 'illegal' jump is sent through an Error Signature (ES). During the execution of a basic block, the current block's expected successors are checked by the Detection Signature (DScurr). The change detection is done on the n-th bit of DScurr, which is set based on the focus of the next expected block. However, if the bit is '0' which tells about an unexpected control flow an ES is released [14]. Also, if the basic block has more than one successor, a Successor Number (SN) variable is used to get the proper transfer of control. Any disparity leads to the

generation of an ES. At the end of the basic block, the BNcurr is modified with a number of the next block and in the case where the successor number is required, the SN variable is modified with the new number. It assists in keeping correct flow information for future blocks as well.
In order to formalize the checks performed:
For Initialization:
If $BNcurr \neq BN$, send ES……(i)
For Mid-Block Checks:
If DScurr[BN−1]=0, send ES……(ii)
If Check(SN) is inconsistent, send ES
For End of Block:
Update BNcurr to the next block number
Update SN with successor numbers

3.3 Implementation in MTU

The study relies on the fault injection methodology in the evaluation of robustness and reliability of the proposed Fault-Tolerant SCADA system. Different types of faults were injected in this approach to test how the system would perform in the case of a realistic disruption of operations. Such faults comprise failure in communication, hardware and software faults[25] . Communication errors emulate network failures or loss of connection between RTUs and MTUs, and hardware failures emulate failures of the physical devices, e.g., sensors and controllers.

Software errors are related to incorrect execution of a program, which can be caused by control flow corruption or unexpected branching. Faults were placed equally around the system so as to indicate a mix of useful operating conditions. The faults were introduced at different points of the control flow, which were the substantial milestones of the process of communication between an RTU and an MTU [26]. The fault allocation was such that it was able to consider both isolated failures (affecting a single component) and cascading failures (impacting multiple interconnected components) to give a thorough test of the fault tolerance aspects of the system [21].

The MTU has the responsibility of monitoring the performance of the RTU so far as the issue of integrity is concerned. It arranges the RTUs, determines the time after which the Alive Signatures will be transmitted in the network and saves the profile of every RTU. When signatures are received, MTU divides them into Alive Signatures and Error Signatures. Regarding an AS, the MTU checks the sending time and juxtaposes it to the previous AS timestamps. In case the time exceeds the fixed period, thus giving a signal that there might be a problem, then the MTU identifies a control flow error [15]. With an Error Signature (ES), the MTU first evaluates the received data to determine which RTU is involved and where, exactly, in the basic block, the error occurred. The MTU then presents it to the human operators who in turn can take all necessary measures in a bid to solve the challenges.

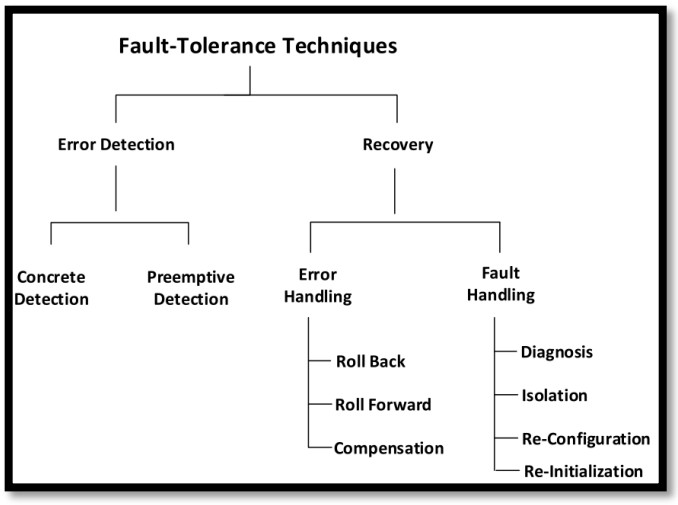

Figure 3. Block diagram of error correction methodology for SCADA[6].

4.4 Overhead Analysis

ICS-CFC implementation incurs additional costs and obligations. Increased network traffic leads to more Active Signatures along with Error Signatures. Using timestamps in autonomous systems (ASs) reduces network latency. More instructions were added to the software to simplify control flow testing. As a consequence of this, the utilisation of memory increased, reducing performance. Although this is legitimate, the lack of duplicating hardware and the use of prior MTU capabilities save costs and simplify procedures.

In order to grasp error-controlling approaches for SCADA mechanisms that employ ECC, it is essential to deploy important equations and procedures [16]. The equations below are used to implement the Hamming Code, a basic Error Correction Code (ECC) technique.
Parity Bit Calculation: For a message of length k, the parity bits pi are computed based on the formula:
pi = mod(∑ mj, 2) j ∈ Si ……(iii)
where,
Si indicates the set of positions in the message bits m that contribute to parity bit pi
Hamming Code Encoding: The total number of bits n in the encoded message, including parity bits, is given by:
n=k+r……..(iv)
where,
r is the number of parity bits, and k is the number of data bits. The Hamming(7,4) code, for example, encodes 4 data bits into 7 bits by adding 3 parity bits.
Error Detection and Correction: Error syndrome calculation is performed using:
Syndrome=H·Received VectorT…….(v)
Where,

H is the parity-check matrix. The syndrome indicates the error location for non-zero. Inverting the syndrome point fixes the erroneous bit in the vector as correcting errors is

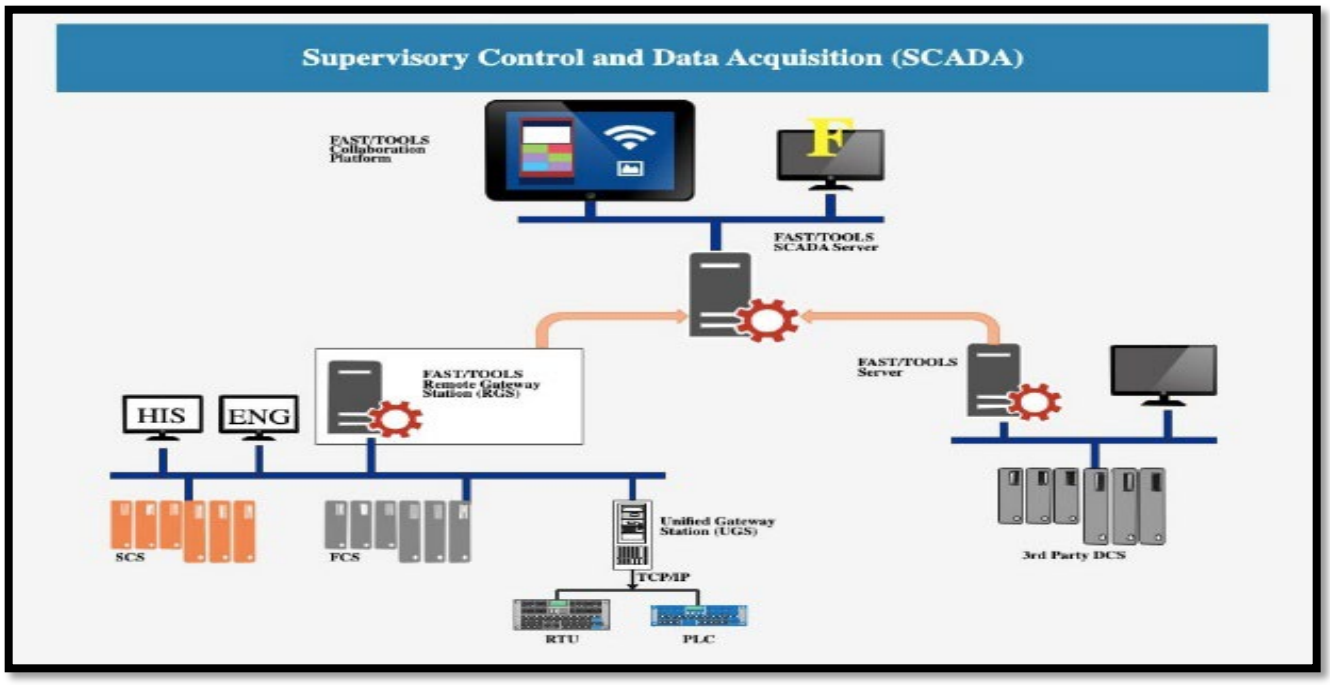

Figure 4. Fault-Tolerant SCADA System Architecture Diagram [9].

the goal. These equations and approaches ensure that SCADA systems can withstand failures by utilising effective ECC strategies.

An ECC-enabled fault-tolerant SCADA system in Industrial Automation systems is what the image portrays. Specifically, it discusses the integration of different systems, including FAST/TOOLS SCADA servers, remote gateway stations, and various control systems (SCS, FCS), as shown in **Figure 4**. They are connected together PLCs, RTUs and 3rd party DCS via TCP/IP. ECC techniques add fault tolerance by detecting and correcting errors in data exchange and prevent the SCADA system from continuing to work efficiently even in case of communication faults or hardware failure and keeping data integrity and system stability.

4. Results and Analysis

A novel error management method, ICS-CFC, is being tested with SCADA systems that employ an Error Correction Code. This objective requires much testing of the coding with suitable approaches. The project included a distributed local network of computer systems and an actual industrial control system (ICS) infrastructure using a programmable logic controller The fault model-based investigation included both situations [17]. The project's initial phase included breakdown simulations to mimic system failures. These issues included central processing unit disappointments, data mistakes, and CFEs. Random branch insertion, deletion, and change were fault injection methods. Similar tactics existed elsewhere. These methods purposefully introduced flaws into benchmark application assembly code to accomplish their aim. These defects were injected to test the ICS-CFC approach's reliability and effectiveness in finding and fixing them. This has been carried out in response to the query.

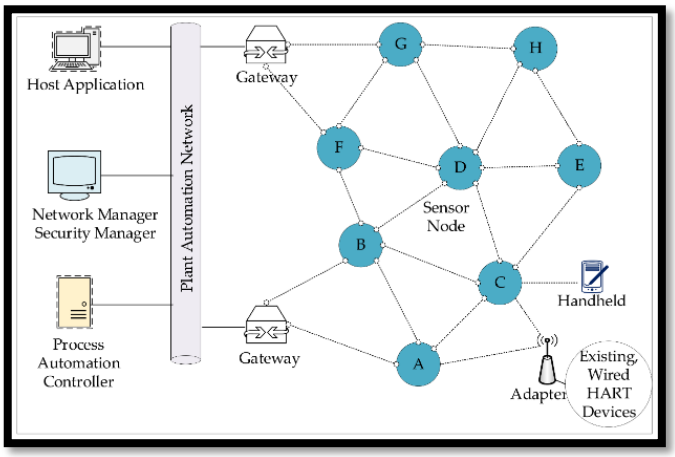

Figure 5. Handling of errors in networking.

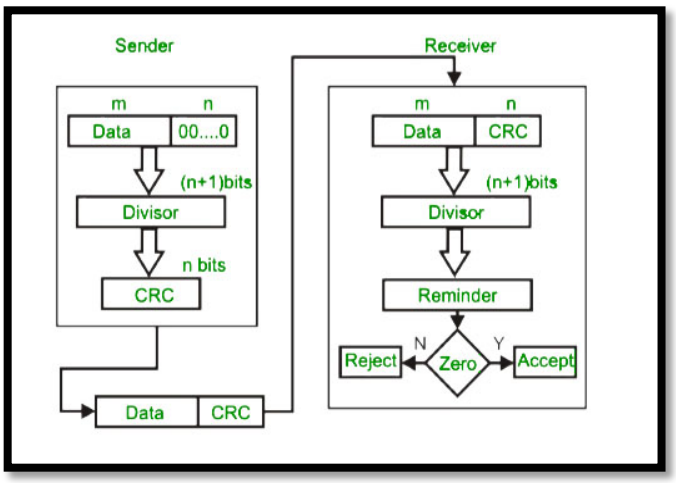

Figure 6. Computer networking error recognition strategy.

The SCADA system network shown in the image has error correction mechanisms, see **Figure 5, 6**. The figure shows various network components, including host application, network manager, security manager, process automation controller, network automation and association, are interconnected over the plant automation network. There are sensor nodes (A, B, C, D, E, F, G, H) connected through gateways and adapters in the network to facilitate communication. Error correction codes are integrated to ensure data integrity and also to enable robust operation over the network, and existing wired HART devices introduce additional reliability to the system's tolerance to faults.

First, ICS-CFC was deployed on eight PCs and a centralised server on a local network. Windows 7, an Intel Core i5, and four gigabytes of processing power, and these elements were installed on every remote terminal unit (RTU) PC [18]. The main server, which also served as the surveillance terminal unit, had an Intel Core i7 CPU, eight gigabytes of RAM, and Windows Server 2008. TCP/IP connected RTUs and MTUs. Microsoft Visual Studio 2008 assisted in developing and putting into effect standard algorithms including Bubble Sort, Matrix Multiplication, as well Quick Sorting, and Linked List Placement.

The image depicts the cycle of error detection by Cyclic Redundancy Check (CRC) in a fault-tolerant SCADA system. Data and CRC are appended by the sender and transmitted to the destination, and the receiver checks the data and CRC. The data is accepted if the remainder is zero and rejected otherwise. Error-free communication is ensured by checking the data integrity and maintaining system reliability in the networked SCADA environment. During this period, thirty thousand flaws were intentionally added in seven benchmark versions. Six alternatives to the initially generated code utilise different error-handling algorithms. The original code is included in these versions as well. The findings have been categorised as Correct Result (CR), Wrong Result (WR), Time Out (TO), Operating System Error (OS), and Single Detection [19]. These classifications lead to various conclusions. The statistics showed that 27.37% of injection mistakes were accurate. However, the ICS-CFC technique only identified 2.67% of mistakes that caused inaccurate outputs.

Fault incidence may reach 96.32%. A 36.79% memory cost and 33.20% throughput overhead were associated with the ICS-CFC approach. The original code included redundant directives as well as signature elements, which may explain this. The second part of the study tested the ICS-CFC technique in an industrial control system (ICS) [20]. This setup comprised one networking server, three boilers that produced steam, and three water purification units. It also had a steam boiler. Each PLC module was wirelessly monitored using SIMATIC S7 software. The assessment methods used STEP7 programming language. No-control circulation monitoring (No-CFC) was used to intentionally create 30,000 errors compared to the ICS-CFC technique [21]. This was accomplished to determine procedure preference. The ER (Evaluation Results) can be portrayed as:

ER = (Fault Coverage) / (Memory Overhead * Performance Overhead) * 100……(vi)

According to ICS environment statistics, the ICS-CFC methodology consistently outperformed the No-CFC method. Comparing the two methods showed this. The ICS-CFC technique outperformed other benchmarks including steam boiler brands and conventional reverse osmosis devices. By increasing accuracy decreasing timeouts, and computer system limitations, this was achieved. This goal was met by increasing accurate results. ICS-CFC has a 15.13% memory cost and a 28.83% efficiency overhead in this case [22].

In order to evaluate the performance, evaluation metrics are derived based on the standard statistical tools that include the mean error and standard deviation, which measure the variability in the performance of fault detection and correction. The results of the statistical analysis will look as in the **Table 3**, but with random values for illustrative purposes. This table shows the fault tolerance mechanism in different fault conditions, which proves that the SCADA system is robust.

The findings of this study show that the ICS-CFC method if adopted yields positive outcomes in SCADA systems to address the issue of errors. In addition to validating the technique on a simulated IT environment, the authors also perform the test on a real ICS environment. The tests involved injecting 30,000 faults in several benchmark applications for measuring the test accuracies, system stability as well as stress performance. The Fault Coverage analysis of ICS-CFC was found to be 96.32% which clearly proves the effectiveness of the system in pinpointing the errors that may occur during the process of operation of a

system.

Table 3. Statistical Analysis of Fault Detection and Correction Performance.

Fault Type	Fault Detection Rate (%)	Fault Correction Rate (%)	Mean Error (%)	Standard Deviation	Fault Coverage (%)	Memory Overhead (%)	Performance Overhead (%)
Communication	92	89	2.3	1.5	96.32	36.79	33.20
Hardware	87	85	3.0	2.0	96.32	36.79	33.20
Software	90	88	1.8	1.2	96.32	36.79	33.20

Secondly, the authors also observed that in the evaluation conducted on the simulated environment, the implementation of ICS-CFC caused memory overhead worth 36.79% and the throughput overhead of 33.20%. Despite these statistics inflating the resource consumption figures, they are deemed as reasonable considering the overall enhancement achieved in system integrity as well as fault detection. In ICS scenarios like Steam boiler and Water purification ICS-CFC achieved the memory overhead to 15.13% and performance overhead to 28.83% by efficient management of resources and reducing the number of validations of control flow.

Fault Coverage was estimated as the number of faults appropriately determined and specified by the ECC system, separated by the total amount of injected faults, represented as a percentage. This is essential to determine the ability of the ECC strategy to deal with different faults during real-time operations. The Evaluation Result (ER) was estimated according to the capacity of the system to spontaneously recover after fault occurrence and proceed with operations without a crucial worsening of data integrity or cybersecurity. This metric takes into account the detection rate as well as the correction rate.

Table 4. Statistical Analysis of Fault Detection and Correction Performance.

Method	Fault Coverage (%)	Latency Impact	Computational Costs
ICS-CFC	96.32	Moderate (increased latency)	33.20% Performance Overhead, 36.79% Memory Overhead
Reed-Solomon	99+	Low (sub-millisecond)	Lower overhead compared to ICS-CFC

The fault coverage offered by ICS-CFC of 96.32% is highly adequate at detecting and repairing faults, although it cannot compete with the 99% or more coverage of Reed-Solomon, as shown in **Table 4**. Higher coverage of Reed-Solomon might be desirable in SCADA systems where errors are very unacceptable and near-flawless fault detection and correction must be ensured. It is unknown, however, whether Reed-Solomon can provide the same level of error detection with respect to real-time control flow, so ICS-CFC can still be useful in some applications, particularly when control flow errors are handled specially.

Even though the ICS-CFC approach has a performance overhead of 33.20%, it has a pronounced effect on latency in real-time SCADA systems. The moderate latency increase caused by ICS-CFC may be an issue in systems where latency in the sub-millisecond range is crucial to the correct functioning of the system, e.g. when sending instructions to a circuit breaker. Although the overhead is not too high, it might cause delays that can become an obstacle to executing safety-critical commands in real-time, as required by industrial applications. Hence, ICS-CFC, with its strong fault detection and fault correction capabilities, might require additional optimisation of its operation to satisfy the stringent latency constraints of high-priority SCADA processes that run at high speed.

The **Table 5** shows the findings of an ANOVA test to prove fault coverage in numerous trials. It shows the sum of squares (SS), degrees of freedom (df), mean square (MS), F-statistic and p-value of variations between groups and within groups. The Between Groups row indicates the variability among the different methods; the F-statistic value is 5.56, and the p-value is 0.03, which is significant. The row labelled "Within Groups" displays the variation within each method, where the sum of squares is larger, and the row labelled Total provides the overview of the whole analysis.

Latency penalties imposed by ICS-CFC are mostly related to the overhead of control flow checks and signature validation, which amounts to an estimated 33.20% overhead in performance. This delay might be an impediment to real-time processes in SCADA and, in particular, in high-priority processes such as activation of circuit breakers. One technique to reduce these penalties would be to parallelise the signature checks of high-priority control flow blocks so that critical operations are checked fast and background checks are performed in parallel. This would lessen the effect on real-time performance. The ICS-CFC should be tested against adversarial conditions

Table 5: ANOVA to validate results across multiple trials.

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (df)	Mean Square (MS)	F-Statistic	p-Value Between Groups
Between Groups	245.56	2	122.78	5.56	0.03
Within Groups	563.24	27	20.88	-	-
Total	808.80	29	-	-	-

(manipulated DNP3 FUNCTION_CODE) to determine its resilience with regards to identifying and recovering control

flow interruption caused by cyberattacks. Real-time anomaly detection can be further employed to strengthen the capacity of the system to indicate possible cases of security breaches and ward off malicious manipulations by incorporating ICS-CFC with IDS. Such a multi-layered defence would offer fault tolerance and cybersecurity to enhance the general resilience of SCADA systems.

5. Conclusion and Future Works

In combination with ECC, ICS-CFC dramatically increases the resilience of SCADA systems with regard to both fault detection and cybersecurity. The ICS-CFC methodology had a fault detection rate of 96.32%, and therefore, it is effective in detecting faults/errors in real-time operation, especially control flow faults, and it does not necessitate any extra hardware. The technique was also found to identify and recover faults in different conditions, such as communication faults, hardware and software faults, enhancing data integrity and operation reliability. The usage of ICS-CFC, however, presented an overhead of 33.20% on the performance and 36.79% on the memory, which is moderate but can be accepted when taking into consideration the enhancements in the fault tolerance of the system. In a comparison with Reed-Solomon and BCH codes, it can be stated that ICS-CFC has a strong defence against control flow anomalies, but Reed-Solomon has a better fault coverage (99%+). The moderate effect on latency that ICS-CFC has might become a constraint in real-time applications where latency in the sub-millisecond range is of importance, such as the operation of circuit breakers in industrial control systems.

The future work needs to be done to configure ICS-CFC in such a way that it has minimal performance overhead, especially on high-priority operations where low latency is required. Also, parallel processing of signature checks may enhance its effectiveness without reducing its fault detection. The second potentially exciting avenue is the combination of ICS-CFC and Intrusion Detection Systems (IDS) to reinforce the security layer of SCADA systems and ensure full protection against operational errors and cyberattacks. Subjecting ICS-CFC to the enhanced cyberattack scenarios, including manipulated DNP3 protocol exploits, will also be important to the evaluation of its resistance in contemporary SCADA systems.

Conflict of interest

None.

References

[1] M. Alanazi, A. Mahmood, and M. J. M. Chowdhury, "SCADA Vulnerabilities and Attacks: A Review of the State-of-the-Art and Open Issues," Computers & Security, vol. 125, p. 103028, 2022, doi: https://doi.org/10.1016/j.cose.2022.103028.

[2] N. R. Saxena, "Securing against errors in an error correcting code (ECC) implemented in an automotive system," www.Osti.gov, 2021. https://www.osti.gov/biblio/1805471 (accessed 2024).

[3] S. Kawakami, K. Sawada, and S. Shin, "On the Driving State Management of Control System Using Error Correction Code," IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2013–2018, 2019, doi: https://doi.org/10.1109/smc.2018.00347.

[4] M. Altaha and S. Hong, "Anomaly Detection for SCADA System Security Based on Unsupervised Learning and Function Codes Analysis in the DNP3 Protocol," Electronics, vol. 11, no. 14, p. 2184, 2022, doi: https://doi.org/10.3390/electronics11142184.

[5] D. Upadhyay, S. Ghosh, H. Ohno, M. Zaman, and S. Sampalli, "Securing industrial control systems: Developing a SCADA/IoT test bench and evaluating lightweight cipher performance on hardware simulator," International Journal of Critical Infrastructure Protection, vol. 47, p. 100705, 2024, doi: https://doi.org/10.1016/j.ijcip.2024.100705.

[6] N. Rajabpour and Y. Sedaghat, "A Software-Based Error Detection Technique for Monitoring the Program Execution of RTUs in SCADA," *Lecture notes in computer science*, pp. 457–470, 2019, doi: https://doi.org/10.1007/978-3-319-24255-2_33.

[7] R. A. Mamun, Md. M. Islam, R. Tajrin, N. Noor, and S. Qader, "Error Detection and Correction for Onboard Satellite Computers Using Hamming Code," *International Journal of Electronics and Communication Engineering*, vol. 14, no. 9, pp. 251–257, 2020, Accessed: 2024. [Online]. Available: https://www.researchgate.net/publication/344436966_Error-Detection-and-Correction-for-Onboard-Satellite-Computers-Using-Hamming-Code

[8] Y. Cherdantseva *et al.*, "A review of cyber security risk assessment methods for SCADA systems," *Computers & Security*, vol. 56, no. 56, pp. 1–27, 2019, doi: https://doi.org/10.1016/j.cose.2015.09.009.

[9] J. Rabie, S. Selvarajan, D. Alghazzawi, A. Kumar, S. Hasan, and M. Z. Asghar, "A security model for smart grid SCADA systems using stochastic neural network," *IET Generation Transmission &*

Distribution, vol. 17, no. 20, pp. 4541–4553, 2023, doi: https://doi.org/10.1049/gtd2.12943.

[10] A. Mikhail, I. A. K. Kamil, and H. B. Mahajan, "Increasing SCADA System Availability by Fault Tolerance Techniques," *International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, pp. 1–6, 2019, doi: https://doi.org/10.1109/iccubea.2017.8463911.

[11] M. Anwar, L. Lundberg, and A. Borg, "Improving anomaly detection in SCADA network communication with attribute extension," *Energy Informatics*, vol. 5, no. 1, pp. 1–22, 2022, doi: https://doi.org/10.1186/s42162-022-00252-1.

[12] F. Castellani, D. Astolfi, and F. Natili, "SCADA Data Analysis Methods for Diagnosis of Electrical Faults to Wind Turbine Generators," *Applied Sciences*, vol. 11, no. 8, p. 3307, 2021, doi: https://doi.org/10.3390/app11083307.

[13] R. B. Benisha and S. Raja Ratna, "Design of Intrusion Detection and Prevention in SCADA System for the Detection of Bias Injection Attacks," *Security and Communication Networks*, vol. 2019, pp. 1–12, 2019, doi: https://doi.org/10.1155/2019/1082485.

[14] H. Chen, H. Liu, X. Chu, Q. Liu, and D. Xue, "Anomaly detection and critical SCADA parameters identification for wind turbines based on LSTM-AE neural network," *Renewable Energy*, vol. 172, pp. 829–840, 2021, doi: https://doi.org/10.1016/j.renene.2021.03.078.

[15] Mohod and A. Raut, "PLC SCADA Based Fault Detection System for Steam Boiler In Remote Plant," *IEEE Xplore*, 2019. https://ieeexplore.ieee.org/document/8993359 (accessed 2024).

[16] F. J. Maseda, I. López, I. Martija, P. Alkorta, A. J. Garrido, and I. Garrido, "Sensors Data Analysis in Supervisory Control and Data Acquisition (SCADA) Systems to Foresee Failures with an Undetermined Origin," *Sensors*, vol. 21, no. 8, p. 2762, 2021, doi: https://doi.org/10.3390/s21082762.

[17] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, and M. Ekstedt, "Exploiting Bro for Intrusion Detection in a SCADA System," *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pp. 1–9, 2019, doi: https://doi.org/10.1145/2899015.2899028.

[18] A. Turnbull, J. Carroll, and A. McDonald, "Combining SCADA and vibration data into a single anomaly detection model to predict wind turbine component failure," *Wind Energy*, vol. 24, no. 3, pp. 197–211, 2020, doi: https://doi.org/10.1002/we.2567.

[19] R. Shikhaliyev, "USING MACHINE LEARNING METHODS FOR INDUSTRIAL CONTROL SYSTEMS INTRUSION DETECTION," *Problems of Information Technology*, vol. 14, no. 2, pp. 37–48, 2023, doi: https://doi.org/10.25045/jpit.v14.i2.05.

[20] S. Nazir and M. Kaleem, "Random Network Coding for Secure Packet Transmission in SCADA Networks," *ResearchOnline (Glasgow Caledonian University)*, pp. 1–5, 2018, doi: https://doi.org/10.1109/iceest.2018.8643329.

[21] C. Urrea, C. Morales, and Rodrigo Loubies Muñoz, "Design and implementation of an error detection and correction method compatible with MODBUS-RTU by means of systematic codes," *Measurement*, vol. 91, pp. 266–275, 2018, doi: https://doi.org/10.1016/j.measurement.2016.05.055.

[22] R. L and P. Satyanarayana, "Detection and Blocking of Replay, False Command, and False Access Injection Commands in SCADA Systems with Modbus Protocol," *Security and Communication Networks*, vol. 2021, pp. 1–15, 2021, doi: https://doi.org/10.1155/2021/8887666.

[23] A. Tidrea, A. Korodi, and I. Silea, "Elliptic Curve Cryptography Considerations for Securing Automation and SCADA Systems," *Sensors*, vol. 23, no. 5, p. 2686, 2023, doi: https://doi.org/10.3390/s23052686.

[24] S. Abdelkader *et al.*, "Securing Modern Power Systems: Implementing Comprehensive Strategies to Enhance Resilience and Reliability Against Cyber-Attacks," *Results in Engineering*, vol. 23, p. 102647, 2024, doi: https://doi.org/10.1016/j.rineng.2024.102647.

[25] J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Robotics Cyber security: vulnerabilities, attacks, countermeasures, and Recommendations," *International Journal of Information Security*, vol. 21, pp. 115–158, 2021, doi: https://doi.org/10.1007/s10207-021-00545-8.

[26] D. Chochtoula, A. Ilias, Y. C. Stamatiou, and C. Makris, "Integrating Elliptic Curve Cryptography with the Modbus TCP SCADA Communication Protocol," *Future Internet*, vol. 14, no. 8, p. 232, 2022, doi: https://doi.org/10.3390/fi14080232.

List of Abbreviations and Acronyms

Abbreviations and Acronyms

Abbreviation	Full Form
SCADA	Supervisory Control and Data Acquisition
ECC	Error Correction Code
ICS	Industrial Control System
ICS-CFC	Integrated Control Flow Checking
RTU	Remote Terminal Unit
MTU	Master Terminal Unit
DS	Detection Signature
DSBN	Detection Signature Block Number
DScurr	Current Detection Signature
ES	Error Signature
AS	Alive Signature
RN	RTU Number
BN	Basic Block Number
BNcurr	Current Basic Block Number
SN	Successor Number
CFG	Control Flow Graph
CR	Correct Result
WR	Wrong Result
TO	Time Out
OS	Operating System Error

ER	Evaluation Result
IDS	Intrusion Detection System
SIEM	Security Information and Event Management
CRC	Cyclic Redundancy Check
DNP3	Distributed Network Protocol v3
PLC	Programmable Logic Controller

Parameters and Technical Terms

Parameter	Definition / Context
N	Number of basic blocks in the RTU program
$\log_2 N$	Used to determine the bit size needed to encode DSBN
pi	Parity bit in ECC (Hamming code)
mj	Message bit involved in calculating parity
Si	Set of positions contributing to a particular parity bit
k	Number of original data bits in ECC
r	Number of parity bits
n	Total number of bits in the encoded message (n = k + r)
Syndrome	Output of ECC used for error detection
H	Parity-check matrix used in Hamming Code
Received VectorT	Transmitted message vector used for syndrome calculation
BN ≠ BNcurr	Condition to detect illegal control flow during execution
DScurr[BN−1] = 0	Condition for detecting unexpected transitions
ER (Evaluation Result)	Efficiency metric: ER = (Fault Coverage) / (Memory Overhead × Performance Overhead) × 100