WiPiEC Journal

# Works in Progress in Embedded Computing Journal

June 2026 · Volume 12 · Issue 1 · ISSN 2980-7298

*https://wipiec.digitalheritage.me*

# Contents

# Exposing Vulnerabilities in NMEA Gateways: Insights from Shodan and Honeypot Experiments

Jeroen Pijpker Marten Struijk Fadi Mohsen

*Abstract*—**As connectivity increases through the Internet of Things (IoT) and Industry 4.0. Previously isolated systems gained remote access capabilities and became more exposed to cyberattacks. For example, in the maritime domain, the Global Maritime Transportation System (GMTS) is considered a high-potential target. Attacking a GMTS system with malware has been shown to influence ships or disrupt onboard operations. Another significant component of the ship network is an NMEA gateway. Prior research has shown evidence of NMEA gateways being exposed to the Internet, and our previous work experimentally demonstrated four practical attack vectors against such gateways: GPS spoofing, AIS injection, autopilot manipulation, and resource exhaustion. However, it remains unclear whether they have been targeted by adversaries or how an attacker could exploit them.**

**In this work, an NMEA gateway honeypot is designed, implemented, and deployed. The design of the honeypot is inspired by using Shodan, which is used to identify real and exposed NMEA gateways. Our Shodan results show that Internet-exposed NMEA gateways are widely spread. For instance, the refined $GPRMC-based Shodan query identified 4,305 unique endpoints that transmitted NMEA messages during the observation period, of which 1,542 were analyzed in detail to identify their vulnerabilities and other parameters, such as the attack window. As per the honeypot, although no attacks against the specific NMEA gateway were captured, the honeypot logs captured other types of attacks, such as automated scanning and reconnaissance efforts. These findings indicate that NMEA gateways could become real targets in the near future if not configured or secured properly.**

***Index Terms—Industrial Cyber-Physical System; Maritime Cybersecurity; NMEA; Honeypot; ICS Security***

## I. INTRODUCTION

Industrial control systems (ICS) are everywhere in our modern world. These systems play a crucial role in controlling our infrastructure and processes. They are found in many different industries across many different applications and often serve as the connection between the digital and physical world. A couple of examples include the power grid, nuclear reactors, robots in factories, maritime systems, etc. Historically, ICS systems operated in an isolated environment, which offered a degree of protection from remote cyber threats. However, with the rise of the Internet of Things (IoT) and Industry 4.0, this is no longer the case. Often, remote access and data gathering turn out to be very useful features. In turn, this makes these systems prone to being remotely exploited by malicious parties, especially since they are usually not designed to deal with these kinds of threats. Most of the security patches are 'bolt-on' instead of designing new secure systems from the ground up. Generally, bolt-on security is considered a weaker option to secure a system [1].

Manuscript received May 12, 2026; revised May 15, 2026; accepted June 7, 2026. Published June 15, 2026.
Issue category: Regular
Paper category: Regular
DOI: 10.64552/wipiec.v12i1.128

These systems are now everywhere. With the steady increase in the number and importance of these systems, so have attacks on them. Some examples of the most impactful security incidents include Stuxnet [2], but also the 2015 attack on the Ukrainian power network [3]. Lesser known are the attacks on the Global Maritime Transportation System (GMTS). The GMTS is similarly vulnerable to cyberattacks, and documented incidents indicate that this threat is growing and relatively recent [4], [5]. To create a knowledge base, the Maritime Cyber Attack Database (MCAD) was developed [6] showing a steady increase in maritime systems being attacked/targeted. The scale of the GMTS is illustrated by a merchant fleet consisting of 105,500 vessels above 100 gross tons [7], and it continues to grow. The amount (measured in millions of tons) of goods loaded keeps growing too [8]. With the push of Industry 4.0, ships continue to see more digitization. Whereas ships used to be isolated systems, they often feature connections with the outside world these days.

To protect ICS environments within the GTMS, it's crucial to understand how an attacker performs their attack, and ideally, recognize and respond to it as quickly as possible before any harm is done. This is where honeypots and honeynets become very useful. A honeypot is a deceptive tool that mimics authentic operational systems to attract and engage adversaries [9]. This interaction enables them to gather information about the bad actors' tactics, techniques, and procedures (TTPs). A honeynet is a network of honeypots. Some great progress has been made in the research and development of Industrial honeypots. For example, HoneyICS [10], and HoneyPLC [11].

Honeypots and nets have already been proposed in the context of the GMTS. One proposal explores the concept of an actual 'ship' honeynet, in which the attacker believes he is attacking an actual ship. It concludes that gathering real-world data is a worthwhile effort [12]. Another proposal focuses on port security and deploys a honeypot in that network to detect intruders [13]. However, neither of these works deployed a honeypot in the wild, exposed to the Internet.

In our earlier work, we proposed and experimentally evaluated four concrete attack vectors against NMEA gateways in a controlled environment [14]. The work identified four primary attack vectors: GPS spoofing, AIS injection, autopilot manipulation, and resource exhaustion. This journal article extends that work in several ways: (i) performs an Internet-scale reconnaissance of NMEA data exposure using a refined Shodan query, (ii) quantifies the availability of the possible attack window of Internet-facing NMEA gateways, and(iii) presents the design, implementation, and deployment of a NMEA gateway honeypot to study how real adversaries currently interact with these devices.

The main contributions of this work are as follows:

- Evidence through using Shodan OSINT techniques showing that more than 4300 Internet-facing endpoints expose NMEA data over TCP, and that a subset of those gateways remain reachable for hours after Shodan discovery, resulting in a short but actionable attack window for adversaries.
- Design and deployment of a high-interaction NMEA multiplexer/gateway. The NMEA honeypot realistically emulates a commercially available multiplexer/gateway, including AIS messages and an HTTP configuration web interface.
- Empirical analysis of Internet traffic towards the NMEA gateway honeypot to assess the presence (or absence) of protocol-aware attacks.
- Translation of previously demonstrated NMEA-based attacks into cyber-physical attack scenarios for exposed Internet-facing gateways (GPS/AIS spoofing, autopilot manipulation, and NMEA resource exhaustion), and an assessment that multiple of these attacks are feasible within the measured attack window.
- Mapping of known NMEA-based attack vectors to observed exposure and availability patterns of Internet-exposed NMEA gateways, leading to recommendations on hardening configuration, segmentation, and monitoring before attackers deploy specialized tooling.

The remainder of this paper is structured as follows: In Section II, an overview of relevant background topics is given. In Section III, an overview of related works is provided. Section IV presents the methodology, encompassing the identification of NMEA gateways online. Next, in Section V, the results of the experiments are provided, followed by the discussion in Section VI. The paper is concluded in Section VII.

## II. Background

In this section, background information about ship networking protocols and equipment is provided, with a focus on the NMEA standards maintained by the National Marine Electronics Association (NMEA)[15].

### A. Ship networking protocols and equipment

Modern ships feature fast networks that span the entire ship. An example of such a network can be seen in Figure 1. The figure shows how ship networks should ideally be, but nowadays, there is, in many cases, a lack of network segmentation [16], [17].

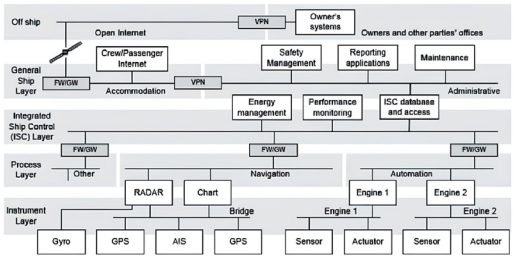



*Figure 1 Layered Ship network data network architecture [18].*

The different layers found in Figure 1 use different protocols. In the maritime sector, Industrial Control Systems (ICS) are usually used to connect the various technologies onboard. The following protocols and their infrastructure are usually found on ships: Modbus, NMEA 0183, NMEA 2000, and NMEA OneNet. These four technologies are used on most ships to control them and relay information between devices and sensors.

- NMEA 0183 is a serial, one-talker/many-listener standard at a 4,800 baud data bus. The data are in printable ASCII format [19].
- NMEA 2000 is a CAN-based standard for marine electronics, a multi-master bus that supports bidirectional communication and plug-and-play configuration [20].
- NMEA OneNet is an Ethernet and IPv6-based standard that bridges all previous NMEA networks [21].

Each of these protocols has its unique equipment and uses different hardware. A popular choice that is sometimes seen within the Integrated Bridge System (IBS) is to use NMEA over Ethernet [22]. This has the added benefit that no extra cables are required as Ethernet is usually already provided to the systems, but it compromises some of the security separating the layers provides.


Figure 2 Actisense PRO-MUX-2 is a NMEA 0183 Multiplexer designed for marine environments. It enables combining data from up to eight NMEA 0183 talkers and routing them to six listeners, featuring advanced filtering, isolations on all inputs/outputs, and Ethernet configuration capabilities for complex vessel navigation networks. [23]

### B. NMEA Gateways

To make communication between the various protocols possible, gateways are introduced. NMEA gateways translate messages between shipboard protocols (such as NMEA 0183 or NMEA 2000) and IP-based networks and are often deployed as multiplexers that bridge multiple serial talkers to TCP or UDP streams. An example of such hardware is shown in Figure 2.

### C. NMEA Sentences

The NMEA 0183 sentence starts with either a ’!’ or ’$’. The ’!’ character indicates that the message has a special type of encapsulation. After that, the talker ID follows, and the type of message. The data fields are separated by a ’,’. The number of fields depends on the type of message. After the data fields, an ’*’ follows, which is followed by a checksum based on the previous sentences (excluding the first character). The checksum is computed by XORing all previous bytes. The line ends with ’\r\n’. As an example, the NMEA GPS message that holds the minimal navigational data of the GPS can be seen here:

```
$GPRMC,132233.41,A,5324.6357,N,00611.8764,E,0001.0,26
5.6,280524,0.0,W,A,S*6C
```

An Automatic Identification System (AIS) messages transported via NMEA follow the same high-level grammar but differs in that one of the fields used contains encoded payload data. A typical AIS NMEA sentence is:

```
!AIVDM,1,1,,,1CaL?OhP00PLHAFNSjRh0?v02000,0*25
```

The detailed specification of AIS NMEA encoding is available in the GPSD AIDVM documentation [24].

## III. Related Work

In this section, prior research on (maritime) cyber-physical systems is reviewed. The section is structured as follows: cybersecurity of maritime CPS, attacks on ship networks and NMEA-based systems, OSINT-based exposure of maritime OT assets, and honeypots and the maritime context.

### A. Cybersecurity of maritime CPS

In the maritime domain, cyber-physical systems refer to the integrated combination of IT (Information Technology) and OT (Operational Technology) systems and the interface with the human operators that collectively enable the monitoring, control, and execution of safety-critical and mission-critical functions of maritime assets [25].

In the work of Tan and Jonos [26], a cyber-physical attack is defined as: A cyber-physical attack can either be a physical attack that adversely affects cyberspace, or an attack that originates in cyberspace and has a physical outcome.

### B. Attacks on Ship Networks and NMEA-based Systems

Prior work has demonstrated that exposed NMEA-over-IP gateways provide a technically feasible entry point for compromising maritime CPS. Struijk Pijpker and Mohsen [14] systematically categorized and experimentally validated four attack vectors against an NMEA gateway, namely GPS spoofing, AIS injection, autopilot manipulation, and resource exhaustion through malformed message flooding. These attack scenarios are further organized into three higher-level classes: data spoofing, remote control via data injection, and resource exhaustion. The results of the work provide evidence of protocol-level weaknesses and their potential impact. The experiments are conducted in a controlled environment and do not represent the real world.

Hemminghaus, Bauer, and Padilla created a malware toolkit called Bridge Attack Tool (BRAT). BRAT was designed to compromise systems on board a ship [27]. The tool was able to execute different attacks on ship networks. The attacks primarily affected the availability and integrity of the systems.

The work that is closely related to the NMEA gateway honeypot is HoneyShip [28]. HoneyShip was designed by closely monitoring real-world maritime VSAT systems identified via Shodan, enabling it to emulate an exposed VSAT system on the internet. The honeypot design incorporated detected CVEs, allowing attackers to be closely monitored while interacting with HoneyShip. The dataset collected in this research is publicly available [29]. It includes Shodan and Nmap scans of exposed VSAT systems, and logs from a high-interaction honeypot emulating a Cobham Sea Tel terminal.

### C. OSINT-based Exposure of Maritime OT Assets

The work of Amro [30] used Open-Source Intelligence (OSINT) tools to detect maritime components emitting NMEA messages on the internet, identifying 4,942 hosts and 331 possible maritime components. In contrast, our work not only performs an OSINT-based reconnaissance of internet-exposed NMEA gateways but also deploys a NMEA gateway honeypot to observe how Internet-based adversaries currently interact with such a device.

### D. Honeypots in the Maritime Context

In the context of maritime systems, multiple papers have been published proposing the use of Honeypots. One of the papers developed a Digital Twin-assisted Honeypot for Cybersecure Smart Seaports. The researchers noted that virtual honeypots must be more realistic to entice attackers, necessitating better high-fidelity. By deploying a digital twin, they created an attractive environment for an attacker. Their solution is called TwinPot, which seems promising for detecting external attacks in smart seaports [13].

In the context of a maritime system exposed to the internet as a honeypot, some work has already been done. McCombie and Pijpker proposed a ship honeynet project that simulates key components of the integrated bridge system to collect data on cyber threats targeting vessels [31]. In another work, Pijpker and McCombie extended this concept with a Ship honeynet explicitly designed to gather structured cyber threat intelligence for vessels.

Brouwer et al. [28] presented HoneyShip, a high-interaction honeypot emulating a maritime VSAT-system to capture attack behavior targeting vessels. The honeypot was deployed on the internet, and the development of HoneyShip was guided by OSINT analysis. The HoneyShip captured thousands of interactions, mainly automated probing and exploitation attempts against known vulnerabilities.

In summary, prior work demonstrated that honeypots and related technologies are effective for studying cyber threats against maritime systems, and that OSINT can reveal exposure of maritime systems on the Internet. Existing studies have focused on VSAT terminals, smart seaports, or conceptual ship honeynets, and NMEA exposure has been analyzed only using OSINT. This work addresses that gap by combining OSINT on NMEA gateways with a NMEA gateway honeypot, enabling empirical observation of how adversaries currently interact with exposed NMEA devices.

## IV. Methodology

The study combines active reconnaissance using Shodan for exposed NMEA gateways and the deployment of a custom-

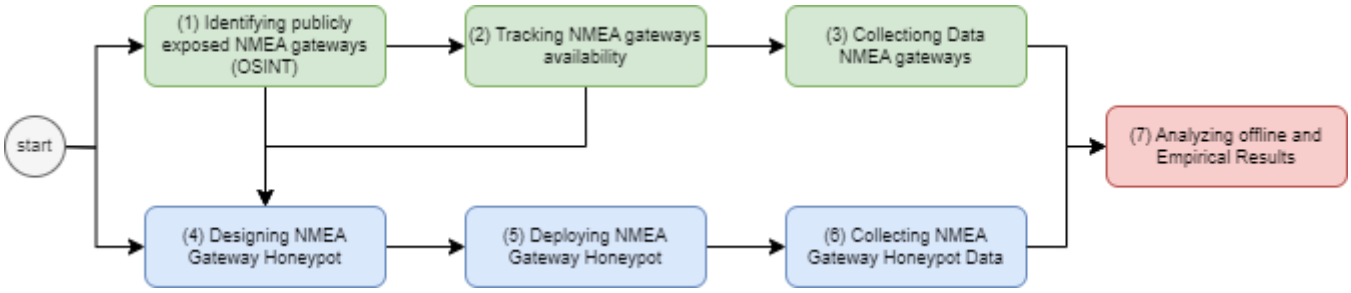



*Figure 3 Workflow of the presented study. Publicly exposed NMEA gateways are first identified via OSINT using Shodan, gateway availability is tracked, and data is collected. In parallel, an NMEA gateway honeypot is designed and deployed, and adversary interaction data are collected. The captured datasets are analyzed offline to derive empirical results.*

*built, high-fidelity NMEA gateway honeypot. The overall workflow, illustrated in Figure 3, consists of two parallel tracks that converge in an offline analysis phase.*

*Shodan-based Reconnaissance*

The green track in Figure 3 identifies, tracks, and collects NMEA gateways that are exposed on the Internet. In previous research, it was shown that many NMEA-exposing connections can be found on Shodan using queries that contain NMEA tags [30]. However, upon manual inspection of the results, many were found to be non-operational, which limits their usefulness to an attacker. To improve the query, this work focuses on a single NMEA sentence starting with $GPRMC, which indicates that a GPS device is emitting minimal navigation data. A Python script issues daily Shodan queries. In total, Shodan was queried 28 times, of which 23 measurements were performed consecutively on a daily basis. To estimate how long these NMEA gateways remain reachable from the Internet, a second Python script performs a daily connectivity check to the reported NMEA gateway ports. By doing this, an estimated attack window before the device goes offline (if it ever does) can be calculated, which in turn tells how useful these devices might be for a potential attacker.

For each endpoint, the date and time of when the queries were done, when Shodan discovered the host for the first time, and every time an attempt to make a connection to the device was made have been recorded. For some of the results, only scans that are on the same day as their update 'timestamp' on Shodan (i.e., new hosts) were taken into account. By doing this we ensure that entries are treated the same regardless of when we performed our query. This allows multiple scans to be done while still being able to build a bigger, overarching dataset as well. The subsequent analysis of the collected data takes temporal factors into account.

*A. Honeypot based Experimentation*

The blue track in Figure 3 covers the development, deployment, and operation of the NMEA gateway honeypot. The high-fidelity honeypot is designed to emulate an Actisense PRO-MUX-2 multiplexer that bridges multiple serial talkers to IP-based listeners in a realistic shipboard setting. The implementation was built using Java Spring Boot and exposes two services: an HTTP configuration dashboard on port 80 and an NMEA-over-TCP service on port 60001, both services reflecting the look and feel and basic workflow of the real device.

To provide plausible navigation dynamics to an attacker, NMEA traffic is generated from publicly available AIS vessel data sources, converted into GPS and AIS sentences (e.g., $GPRMC, $AIVDM), and replayed at a realistic update interval, with minor disruptions. In Figure 4, the AIS-based message generation is shown.

In our previous work, four practical attack vectors against Internet-exposed NMEA gateways were experimentally evaluated in a controlled environment [14]. These attacks included GPS spoofing, AIS injection, autopilot manipulation, and resource exhaustion. These attacks from our earlier work helped validation of the NMEA gateway honeypot used in this work. Spoofing-based attacks exploit the lack of authentication in the Gateways' NMEA stream by injecting GPS or AIS sentences that override legitimate sensor data, for example, in the ECDIS system. Control-Oriented Attacks target NMEA messages that directly influence actuators, such as autopilot waypoints or heading commands, and inject crafted sentences to steer the vessel. By manipulating these control NMEA messages, an attacker can attempt to alter the course and rudder angle and potentially redirect the ship without the crew's awareness. Resource exhaustion attack targets the communication pipeline. NMEA 0183 over serial connections has limited throughput, and the NMEA gateways have limited buffering capacity. By sending high-frequency or malformed messages over TCP, attackers can flood the system, resulting in dropped messages or delayed transmission of critical data.

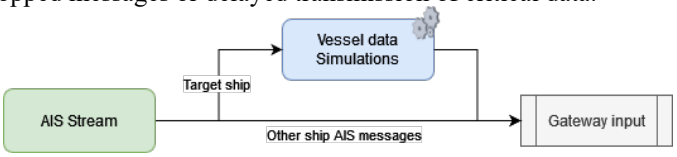



*Figure 4 AIS-based message generation. The AIS Stream provides data from the target vessel. For the selected vessel, the AIS data helps to generate believable NMEA messages.*

In the implementation of the NMEA Honeypot, several NMEA messages were implemented. This is partly based on the messages in the VDR recording of the work from Calvi and Hakefjord [32].

For validation purposes, the NMEA gateway honeypot was first evaluated in a controlled test environment. The following attacks were executed: brute-forcing the administrator's web interface, passively spying on NMEA traffic, and injecting specially crafted NMEA messages as outlined in the related work section. The honeypot successfully recorded all relevant interaction traces, enabling these attacks to be clearly observed during subsequent analysis. Checkpot [33] was used to verify that the NMEA gateway honeypot was operating as intended. It is a honeypot checker utility developed to help security researchers verify that their honeypots are properly set up to attract high-quality traffic [34].

The NMEA Gateway was deployed on a publicly reachable host with a fixed IP address and is left accessible for a period that overlaps with the Shodan tracking phase. The honeypot is not actively advertised; discovery is left to internet-wide scanning and opportunistic probes. The configuration is hardened to prevent misuse.

All the interactions with the NMEA Gateway Honeypot are collected for offline analysis. The NMEA gateway honeypot collected valid HTTP requests to the HTTP server, along with their payloads. This information is sufficient to determine what an attacker explores on the website and to distinguish between a human and a bot. The NMEA gateway honeypot logged all connections, disconnections, and messages received. The connection details include the source IP and source port. The disconnect event contained the same information, along with the duration of how long the connection was alive. After the Shodan measurement and NMEA gateway honeypot deployment phases, all collected data is analyzed offline to derive empirical results.

Finally, the results from both tracks are interpreted: exposure metrics and attack window estimate from the Shodan analysis provide context for the honeypot interactions, while the honeypot traffic reveals how Internet clients behave towards a realistic NMEA gateway honeypot.

## V. Results

This section will describe the empirical results of the research conducted. The results are divided into two different sections. The results of the Shodan analysis can be found in section V-A, and the results of the honeypot deployment in section V-B.

### A. Shodan Results

This section presents the empirical results of the OSINT-based identification of tracking-exposed NMEA gateways using Shodan, following the workflow shown in the first track of Figure 3.

Using the refined minimal required navigation data query *$GPRMC*, multiple Internet-facing hosts sending NMEA sentences over TCP were identified during the observation period. In total, the specially formatted *$GPRMC* Shodan query identified 4,305 unique Internet-facing endpoints that transmitted NMEA, confirming that NMEA data streams are widely exposed on the public Internet. Not all of these endpoints are equally suitable for availability and attack window analysis. Due to Shodan's query limitations, it was not feasible to re-scan all 4,305 IP addresses within a single day. Of the gathered IPs, 1,542 were eligible for analysis in the subset.

For each host in the subset of 1,542, an attempt is made to establish a TCP connection, and the received stream is inspected for GPS and potential AIS NMEA sentences. All connected devices have a valid GPS NMEA tag. AIS NMEA messages were only found on one instance. Identifying AIS NMEA messages can help a potential attacker determine whether the target is a real maritime device rather than a decoy.

The Attack window analysis only uses the results that were found within 24 hours of publication on Shodan. This limited the dataset used for this analysis to 1,542 unique hosts. This ensured that all data points are treated equally and have a common starting point. Every 24 hours, a host is re-scanned to see if it is still available. All hosts have an equal amount of times scanned (or more). Availability is at about 40% within the first two hours of Shodan discovery. Figure 5 also shows that the availability drops rapidly below 25% within a couple of days, and the number of hosts that still accept connections has dropped close to 0%.

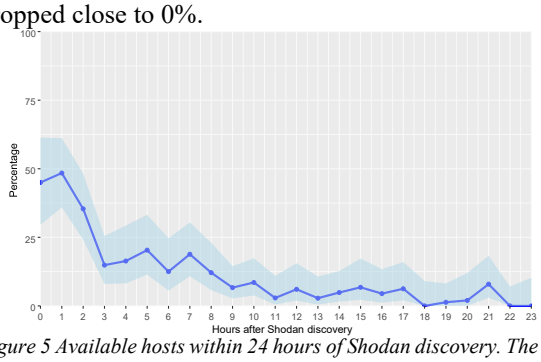



*Figure 5 Available hosts within 24 hours of Shodan discovery. The graph shows that the availability of hosts is around 40% within two hours after Shodan has published the result. After two hours there is already a drop to 25% at it declines until almost zero within 24 hours.*

### B. Honeypot Deployment Results Analysis

Over a period of time, the honeypots were deployed on different platforms. One instance was deployed on a Raspberry Pi and the other instances on a VPS. Combined, the honeypot attracted 2,548 unique IPs to the exposed HTTP dashboard of the NMEA gateway on running on port 80 and 305 unique IPs on the NMEA TCP gateway port 60001. The list of malicious IP addresses that connected to the honeypot is used to compute the abuse confidence score. The abuse confidence

score was calculated based on how many other users observed the IP behaving maliciously. A score of 0 means that the host has not been reported (yet, or at least not frequently). A score of 100 means the host is a well-known malicious IP and has carried out attacks on others as well. In Figure 6 the abuse score result is combined for both honeypots.

The NMEA gateway has seen fewer interactions than the HTTP dashboard from the exposed honeypot. This is to be expected, since most scrapers probably won't target ports 7000 and 51000, as they are not used by many (common) services. Combined for the VPS and RPi, the NMEA gateway has 305 unique IPs connected to it.

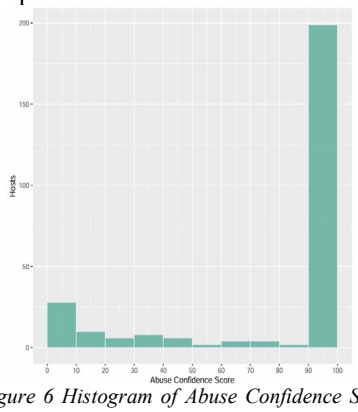



*Figure 6 Histogram of Abuse Confidence Score of ALL collected IPs by the honeypots*

During the deployment period, no direct, maritime-specific attacks were observed targeting the NMEA protocol. The incoming traffic primarily consisted of automated scanners and reconnaissance attempts. The potential attacks demonstrated in our previous work [14] were not observed during the deployment phase.

In Table 1, the connections are divided into buckets of certain durations for analysis. From this table, it can be observed that some connecting clients to the Gateway had connection durations exceeding 2 hours. This could indicate that a connected client is conducting a manual inspection.

*Table 1 Connection duration buckets*

| Connection duration | Freq |
|---|---|
| less than 1 second | 194 |
| 1-5 seconds | 145 |
| 5-10 seconds | 108 |
| 10-20 seconds | 46 |
| 20-60 seconds | 52 |
| 1-5 minutes | 9 |
| 5-10 minutes | 4 |
| 10-20 minutes | 3 |
| 20-40 minutes | 0 |
| 40-60 minutes | 3 |
| 1-2 hours | 3 |
| 2+ hours | 11 |

Table 2 shows the connections that were manually inspected to check if they performed any activity on the gateway due to a 2h+ connection time. It shows that one IP in particular had quite a few connections open. The origins of the IP are quite diverse.

*Table 2 Connections open longer than 2 hours IPs*

| Source | 2h+ conn | Max duration (min) | Country | Abuse Confidence Score |
|---|---|---|---|---|
| 141.98.11.55 | 1 | 193.00 | LT | 100 |
| 164.52.25.205 | 7 | 787.00 | JP | 100 |
| 206.168.34.35 | 1 | 1261.00 | US | 100 |
| 45.156.129.57 | 1 | 187.00 | BE | 100 |
| 87.236.176.215 | 1 | 148.00 | GB | 100 |

## VI. Discussion

The Shodan-based analysis phase demonstrated that a significant number of (maritime) devices connected to the Internet expose NMEA streams of different types. This confirms the observation by Amro [30] and extends it by quantifying availability and the attack window.

The use of the Abuse Confidence Score helped our research to filter out 'noise' instead of treating every connection to the honeypot as a possible attack.

The results from Shodan show that the exposure is substantial, but attackers have not yet systematically weaponized NMEA-aware tooling against NMEA gateways. Shodan results should be interpreted cautiously. It has not been confirmed in this work that the Shodan results are real NMEA gateways on actual vessels. Some endpoints may also emulate NMEA gateway behavior, as our honeypot does. Confirming the authenticity of the NMEA gateways was not part of this research.

Most of the incoming traffic consisted of automated scanning and reconnaissance, and there was no clear evidence of NMEA-aware spoofing, control-oriented, or resource exhaustion attacks against the emulated NMEA honeypot gateway that we demonstrated in our previous work [14]. This follows the notion of an early-warning phase: exposure of NMEA gateways is substantial, yet attackers have not systematically weaponized protocol-aware tooling against NMEA gateways. The absence of observed maritime-specific attacks should not be interpreted as proof of safety. Instead, it underscores the need to proactively harden NMEA gateways and their surrounding networks before attackers craft specialized attacks against NMEA gateways.

As Table 3 shows, several impactful attacks from our previous work can be executed within the attack window identified in the Shodan phase. Passive spying on NMEA gateway traffic can take only seconds to establish a TCP connection. Once a TCP connection to an NMEA gateway is established, an NMEA resource exhaustion can also be triggered in seconds to flood the NMEA gateway.

*Table 3 NMEA Gateway vulnerabilities related to the previous work [14] translated into feasible attacks*

| Vulnerability (issue) | Attack type | Short description | Minimal execution time (testbed) | Feasible within observed attack windows |
|---|---|---|---|---|
| No authentication required NMEA TCP Gateway | Spying (passive) | NMEA Gateway port reachable from the Internet. The NMEA Gateway is bidirectionally configured. | Seconds TCP connect and start logging | Yes, even for short availability |
| As above | Data fabrication: Spoofing (GPS/AIS/Sensor data) | Sending crafted NMEA sentences to the gateway, the adversary can send spoofed data for various instrumental displays. | Few minutes to craft and inject valid NMEA sentences | Yes, from 10 minutes to an hour |
| As above | Data fabrication: Controlling the ship (autopilot manipulation) | Autopilot messages are used to instruct the vessel to steer to certain waypoints automatically. By inserting forged *$–APB* and *$–APA* messages into the data stream, the autopilot of ship can be instructed to a certain waypoint [22] | Few minutes to craft and inject valid NMEA sentences | Yes, from 10 minutes to an hour |
| DoS against NMEA gateway | Resource exhaustion (dropping/delaying NMEA Streams) | The TCP NMEA port can be flooded by sending high rate of messages in short time. So legitimate NMEA messages are being dropped or delayed. | Seconds to start a NMEA stream overload/effect almost immediately | Yes, from 10 minutes to an hour |
| As above | Spoofing/controlling by degrading legitimate traffic | Like OpenCPN only use for example the last messages received. | Seconds to start a NMEA stream overload/effect almost immediately | Yes, from 10 minutes to an hour |
| Brute force admin panel | Configuration changes (routing changes, disabling inputs/alarms), firmware upload | Not changing the default credentials on the NMEA gateway or using weak credentials. | Seconds to start | Yes, if the admin panel uses default credentials |

The observations in this work may be subject to certain threats to validity, particularly regarding the representativeness of Shodan-identified endpoints and the specificity of our NMEA gateway honeypot implementation.

Shodan-based measurements depend on a third-party scanning service whose coverage, timing, and banner classification we cannot control or fully validate, so some endpoints may not be a real shipboard NMEA gateway.

Additionally, the NMEA gateway honeypot represents a single NMEA gateway with simulated AIS-based traffic rather than a fully IBS, so attacker behavior against a richer or truly operational environment may differ from the observation.

Finally, the duration of deployment was limited, which may restrict how far the traffic patterns can be generalized over time.

## VII. Conclusion

This work examined Internet-exposed NMEA gateways that can be classified as critical maritime industrial cyber-physical systems. Prior research showed that NMEA messages are exposed online. In this work, we conducted an OSINT-based reconnaissance study using Shodan and developed and deployed an NMEA gateway honeypot that emulates a commercially available device.

The Shodan study showed that data streams of NMEA gateways remain widely exposed on the Internet and confirmed that many endpoints are reachable over TCP within a short but useful attack window.

Our high-fidelity honeypot showed that it is feasible to emulate a realistic NMEA gateway and attract Internet traffic to both the HTTP dashboard and the NMEA TCP service. Although the honeypot logs did not show any clear evidence of specific attacks identified in our previous work [14] against the NMEA gateway, they did reveal excessive automated scanning and reconnaissance attempts.

This work shows that NMEA Gateways are vulnerable due to a combination of weak or absent authentication, legacy protocol design, and direct internet exposure.

For future work, we aim to deploy a more complex Ship network honeynet, preferably with more realistic maritime traffic sources, and an internet connection relevant to maritime operations, for example, StarLink. Overall, the findings underscore the need to harden NMEA gateways and associated shipboard networks before adversaries begin to exploit these maritime assets.

## DATA AND SOURCE CODE AVAILABILITY

The datasets generated and analyzed during the study, along with the source code of the NMEA Gateway honeypot, will be made publicly available upon acceptance of this work.

## Acknowledgment

The authors used AI-assisted tools to improve the language and clarity of the publication. In particular, Grammarly and similar grammar-checking tools were used for spelling and grammar corrections, and ChatGPT was used to refine wording, suggest alternative formulations, and tighten abstract,

introduction, and discussion sections. All AI-generated suggestions were manually reviewed and edited by the authors, who take full responsibility for the final content.

## REFERENCES


[1] S. Shiva, S. Roy, and D. Dasgupta, "Game theory for cyber security," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, 2010, pp. 1–4.

[2] K. E. Hemsley and Dr. R. E. Fisher, "History of Industrial Control System Cyber Incidents," Idaho National Lab. (INL), Idaho Falls, ID (United States), Dec. 2018. doi: 10.2172/1505628.

[3] D. U. Case, "Analysis of the Cyber Attack on the Ukrainian Power Grid." 2016. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/05/20081514/E-ISAC_SANS_Ukraine_DUC_5.pdf

[4] P. Há. Meland, K. Bernsmed, E. Wille, Ø. J. Rødseth, and D. A. Nesheim, "A Retrospective Analysis of Maritime Cyber Security Incidents," *TransNav*, vol. 15, no. 3, pp. 519–530, 2021, doi: 10.12716/1001.15.03.04.

[5] International Shipping News, "Rising Threat of Maritime Cyberattacks," *International Shipping News*, Oct. 2023, [Online]. Available: https://www.hellenicshippingnews.com/rising-threat-of-maritime-cyberattacks/

[6] "Maritime Cyber Attack Database (MCAD) | NHL Stenden university of applied sciences." Accessed: Dec. 12, 2023. [Online]. Available: https://www.nhlstenden.com/en/maritime-cyber-attack-database

[7] United Nations Conference on Trade and Development (UNCTAD), "World Mercant Fleet." 2026. [Online]. Available: https://unctadstat.unctad.org/insights/theme/243

[8] United Nations Conference on Trade and Development (UNCTAD), "World seaborne trade." [Online]. Available: https://unctadstat.unctad.org/insights/theme/244

[9] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems," Aug. 04, 2021, *arXiv*: arXiv:2108.02287. Accessed: Jan. 31, 2024. [Online]. Available: http://arxiv.org/abs/2108.02287

[10] M. Lucchese, F. Lupia, M. Merro, F. Paci, N. Zannone, and A. Furfaro, "HoneyICS: A High-interaction Physics-aware Honeynet for Industrial Control Systems," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, Benevento Italy: ACM, Aug. 2023, pp. 1–10. doi: 10.1145/3600160.3604984.

[11] "HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems | Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security." Accessed: Dec. 12, 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3372297.3423356

[12] J. Pijpker and S. J. McCombie, "A Ship Honeynet to Gather Cyber Threat Intelligence for the Maritime Sector," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, Daytona Beach, FL, USA: IEEE, Oct. 2023, pp. 1–6. doi: 10.1109/LCN58197.2023.10223347.

[13] Y. Yigit, O. K. Kinaci, T. Q. Duong, and B. Canberk, "TwinPot: Digital Twin-assisted Honeypot for Cyber-Secure Smart Seaports," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2023, pp. 740–745. doi: 10.1109/ICCWorkshops57953.2023.10283756.

[14] M. Struijk, J. Pijpker, and F. Mohsen, "Demonstrating Practical Attacks on Maritime Cyber-Physical Systems via Exposed NMEA Gateways," in *2025 14th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2025, pp. 1–4.

[15] National Marine Electronics Association (NMEA), "National Marine Electronics Association." 2026. [Online]. Available: https://www.nmea.org/

[16] R. Murthy and R. Ghaffari, "Shipboard Networks and Communications Systems," in *Maritime Transportation Systems*, J. Beckman and others, Eds., Pressbooks, 2025. [Online]. Available: https://pressbooks.pub/maritimesecurity11/chapter/shipboard-networks-and-communications-systems-murthy-ghaffari/

[17] A. Ribeiro, "Maritime cyber incidents jump 103%, as CYTUR warns smart ships under fire; urges secure by design overhaul." [Online]. Available: https://industrialcyber.co/reports/maritime-cyber-incidents-jump-103-as-cytur-warns-smart-ships-under-fire-urges-secure-by-design-overhaul/

[18] S. Krile, D. Kezić, and F. Dimc, "NMEA Communication Standard for Shipboard Data Architecture," *Naše more*.

[19] National Marine Electronics Association, "NMEA 0183 Standard." 2025. [Online]. Available: https://www.nmea.org/nmea-0183.html

[20] National Marine Electronics Association, "NMEA 2000 Standard." 2025. [Online]. Available: https://www.nmea.org/nmea-2000.html

[21] Actisense (Active Research Ltd), *NMEA OneNet and Ethernet Networking Guide*. Poole, United Kingdom: Active Research Limited, 2023. [Online]. Available: https://actisense.com/wp-content/uploads/2023/07/NMEA-OneNet-and-Ethernet-Networking-guide-1.pdf

[22] A. Oruc, V. Gkioulos, and S. Katsikas, "Towards a Cyber-Physical Range for the Integrated Navigation System (INS)," *JMSE*, vol. 10, no. 1, p. 107, Jan. 2022, doi: 10.3390/jmse10010107.

[23] Actisense, "Pro-Mux-2." Accessed: Apr. 20, 2024. [Online]. Available: https://actisense.com/products/pro-mux-2/

[24] E. S. Raymond and the G. project, "AIVDM/AIVDO Protocol Decoding." [Online]. Available: https://gpsd.gitlab.io/gpsd/AIVDM.html

[25] I. Progoulakis, P. Rohmeyer, and N. Nikitakos, "Cyber Physical Systems Security for Maritime Assets," *JMSE*, vol. 9, no. 12, p. 1384, Dec. 2021, doi: 10.3390/jmse9121384.

[26] K. Tam and K. Jones, "MaCRA: a model-based framework for maritime cyber-risk assessment," Jan. 2019, doi: 10.1007/s13437-019-00162-2.

[27] C. Hemminghaus, J. Bauer, and E. Padilla, "BRAT: A BRidge Attack Tool for Cyber Security Assessments of Maritime Systems," *TransNav*, vol. 15, no. 1, pp. 35–44, 2021, doi: 10.12716/1001.15.01.02.

[28] S. Brouwer, J. Pijpker, and F. Mohsen, "HoneyShip: Unveiling Cyber Threats to Maritime VSAT Systems with a High-Interaction Honeypot," in *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)*, IEEE, May 2025. doi: 10.1109/icps65515.2025.11087909.

[29] S. Brouwer, J. Pijpker, and F. Mohsen, "HoneyShip: Data from a Maritime VSAT Honeypot and Open Internet Reconnaissance." 2025. doi: 10.34894/7SS2RW.

[30] A. Amro, "Cyber-Physical Tracking of IoT devices: A maritime use case," in *Norsk IKT-konferanse for forskning og utdanning*, 2021.

[31] S. J. McCombie and J. Pijpker, "A Ship Honeynet Project to Collect Data on Cyber Threats to the Maritime Sector," presented at the CYBER 2022, The Seventh International Conference on Cyber-Technologies and Cyber-Systems, Nov. 2022, pp. 81–85.

[32] transmitterdan, "VDRplayer." Accessed: Jul. 07, 2024. [Online]. Available: https://github.com/transmitterdan/VDRplayer

[33] V. Florea, "Checkpot: Honeypot Checker." 2018. [Online]. Available: https://github.com/vladalexgit/checkpot

[34] R. Gabrys, D. Silva, and M. Bilinski, "HoneyGAN Pots: A Deep Learning Approach for Generating Honeypots." 2024. [Online]. Available: https://arxiv.org/abs/2407.07292

# Efficient Coins Selection for UTXOs through Evolutionary and Random Draw Methods

Krassimira Stoyanova
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
krassimiradrstoyanova@gmail.com

Petar Tomov
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Sofia, Bulgaria
petyr.tomov@gmail.com

*Abstract*— **This study proposes and evaluates a novel hybrid optimization framework that integrates stochastic Random Draw sampling with Evolutionary Algorithms (EA) to address the multi-objective challenges of Unspent Transaction Output (UTXO) selection. In digital asset management, selecting an optimal subset of coins from fragmented wallet pools requires balancing transaction privacy, economic efficiency, and computational throughput-a task complicated by a combinatorial search space exceeding 10^41 possibilities. The proposed methodology utilizes a random-draw seeding mechanism to bypass initial combinatorial complexity, followed by iterative evolutionary refinement. Results demonstrate that this hybrid approach achieves a 97% success rate and a convergence speed 43% faster than standard optimization techniques, reaching optimal solutions in an average of 14 ms. Furthermore, the inclusion of mutation and crossover operators ensures high UTXO diversity, significantly enhancing privacy by mitigating identifiable transaction patterns common in deterministic heuristics. This research concludes that the hybrid model provides a robust, scalable solution for real-time wallet infrastructures, effectively reconciling the trade-offs between long-term wallet health and immediate transaction requirements.**

**Keywords-*UTXO Selection, Hybrid Optimization, Evolutionary Algorithms, Blockchain Privacy, Combinatorial Complexity, Digital Asset Management.***

## I. INTRODUCTION

Blockchain technology, rapidly advancing since Bitcoin's introduction, has become widely adopted. In cryptocurrency transactions, unspent transaction outputs (UTXOs) are created and must be efficiently managed. Coin selection, the process of choosing UTXOs for transactions, involves balancing conflicting goals such as minimizing block size and reducing transaction fees by limiting the number of inputs.

Coin selection in cryptocurrency wallets, which involves choosing UTXOs to fulfill transaction requirements, is a practical instance of the NP-complete subset sum problem [1]. This process presents a multi-objective optimization challenge, balancing user privacy, transaction fees, and blockchain maintenance overhead, with potentially conflicting individual and community goals [2]. Wallet software developers typically determine coin selection strategies, focusing on minimizing blockchain data growth and maintenance costs. Miners, in turn, prioritize transactions based on fees, which can delay or exclude lower-value payments. Optimal coin selection seeks to balance transaction costs and user anonymity while covering the expenses of blockchain provisioning, proof-of-work, and record storage through fees [3]. Studies analyzing UTXO datasets across multiple cryptocurrencies highlight the importance of structured coin selection methods for effective UTXO management [4]. Additionally, various coin selection strategies exist, with Bitcoin wallets employing multiple algorithms to serve millions of users. For example, the Low Value approach (LVF), or smallest to largest method, often results in higher transaction fees [5]. Various coin selection strategies are employed across cryptocurrency wallets to optimize transaction efficiency and UTXO management. BreadWallet uses a FIFO approach, prioritizing the oldest UTXOs, while the Pruned Oldest First method further refines this by excluding the smallest UTXOs to meet transaction values [6]. Other strategies include Monero's random selection and the highest priority first method. Bitcoin focuses on exact matching, selecting UTXOs that closely align with the transaction amount [7]. Hybrid approaches, such as those in [8], combine greedy and evolutionary algorithms to minimize dust and match target values, though they may not reduce the overall UTXO set. Reference [9] introduces a leveraged coin selection method to optimize the knapsack strategy and lower transaction fees, which can result in dust creation. A mathematical optimization model for UTXO selection and transaction size minimization is presented in [10]. Additionally, MACS, proposed by Ramezan et al. [11], considers transaction fees, size, UTXO pool size, and privacy to enhance coin selection in UTXO-based blockchains. Several business management approaches are discussed in [12, 13], highlighting the importance of compromise solutions that align with decision-makers' preferences [14–16]. While optimization-based coin selection is essential for reducing transaction fees and improving data storage, most existing algorithms fail to simultaneously minimize costs and control UTXO dataset growth.

The article is structured as outlined below. The second section provides basic definitions in the calculus of UTXOs

Manuscript received May 12, 2026; revised May 12, 2026; accepted June 4, 2026. Published June 15, 2026.
Issue category: Regular
Paper category:Regular
DOI: 10.64552/wipiec.v12i1.127

formulation. The UTXO selection method is briefly described in the third section. In the fourth section, UTXOs are numerically structured, commencing from a pool of 99,999 UTXOs. In the fifth section, the proposed concepts have been validated by simulations, demonstrating exceptional enhanced algorithmic convergence and operational efficiency.

## II. THE UTXO'S PROBLEM FORMULATION

### A. The Coin Changing Problem as a Mathematical Model

The Coin Change Problem includes expressing the requested amount using the minimum number of coins from a given list of coins. An unlimited number of coins in each category is available.

Formally, given a finite system $c_1 < c_2 < c_m = n$ of positive integers (the *coins*) and a positive integer $x$, we wish to determine nonnegative integer coefficient $x_i$, $1 \le i \le m$, so as to minimize

$$\sum_{i=1}^{m} x_i \tag{1}$$

Subject to

$$x = \sum_{i=1}^{m} x_i c_i \tag{2}$$

The sequence of coefficients $x_i, \ldots, x_m$ is called a *representation* of $x$. A representation is *optimal* if it is of minimum size. If $x_i > 0$, then we say that coin $c_i$ is *used* in the representation.

### B. The Optimal Bounds

Based on [18] we define optimal bounds for coin change problem.

Let $1 = c_i < \ldots < c_m$ be any system of coins. For all $x$ and coins $c_i \le x$, $M(x) \le M(x - c_i) + 1$, with equality holding if and only if there exists an optimal representation of $x$ that uses the coin $c_i$. Let be one function M with lower and upper bound.

The UTXOs pool $U_1$ , $U_2$, … $U_n$ ($n \ge 2$) with random values are considered. Let a set of $n \in N$ UXTOs be given. For any $i \in N$, each UXTOs $\in N$ has certain characteristics, describing its future payoff: Each UXTOs has an value $u_i^v$, the size $u_i^s$, and the confirmation $u_i^a$. Let $U^v$ is the sum of random values in the UTXO pool, so that $U^v = \sum_{i=1}^{n} u_i^v$ and $U^s$ is the sum of size of the any UTXO, so that $U^s = \sum_{i=1}^{n} u_i^s$. Additionally, we define $U^a$ as the sum of the UTXO confirmation in the set $U$, that $U^a = \sum_{i=1}^{n} u_i^a$.

## III. A UTXOS SELECTION APPROACH

In this section, we briefly build on the approach described in [19] by proposing a coin selection strategy that combines the basic Random Draw algorithm with a sophisticated Evolutionary algorithm. The Random Draw method selects UTXOs uniformly at random, which not only simplifies implementation but also enhances user privacy, as the chosen UTXOs are not influenced by their value. This stochastic selection increases output variability and further contributes to privacy.

We developed the UTXOs optimization model, as follows:

Let $T = [T_1, \ldots, T_i]$ be an array requests of payment. Futhermore, $Outputs = [o_1, \ldots, o_m] \in T$, where for any $j \in [m]$, $o_i^v$ signifies the cost of $o_j$ and $o_j^s$ is the size of $o_j$. An altered output $c$ has cost of $c^v$ and size $c^s$. There is a change output $c$ with cost $c^v$ and size $c^s$. A transaction is defined with three variable ($S^{alg}$, $O$, $c$). Additionally, $D>0$ is the dust.

For any $I \in [n]$, the binary variable $x_i$ equals 1 if $u_i$ is selected as input and 0 otherwise. The transaction charge will be determinated by a fixed free rate $\alpha \ge 0$, the size, and $\varepsilon$ is the minimum changing of coins to prevent the generation of an exceedingly smallest output. We denote as effective cost, i.e., the contribution of a UTXO towards the target, for the current payment request.

$$\min_{x_{i,..,y,m}} y \tag{3}$$

Subject to constraints:

$$y \le M \tag{4}$$

$$\sum_{i=1}^{n} u_i^u x_i = \sum_{j=1}^{m} 0_j^v + ay + c^v \tag{5}$$

$$x_i \epsilon (0,1) for all i \in [n] \tag{6}$$

$$y = \sum_{i=1}^{n} u_i^s x_i + \sum_{j=1}^{m} o_j^s + c^s \tag{7}$$

The main goal is to reduce the overall UTXO set. However, since transaction fees are not directly minimized, the random process may result in selecting UTXOs whose combined value surpasses the transaction requirement. As a secondary objective, we aim to lower transaction fees below a specified threshold, thereby shrinking the search space and reducing computational

effort. This methodology specifically integrates the Random Draw algorithm with an evolutionary optimization technique to improve UTXO management and address both privacy and efficiency concerns [20-21].

The DEPS (Differential Evolution and Particle Swarm) evolutionary algorithm is an innovative approach that combines two powerful optimization techniques to enhance the search for optimal solutions in complex problem spaces. When utilizing random draw methods with the evolutionary solver in LibreOffice Calc, several aspects can be explored to optimize performance and improve results. Here is a detailed look at random draw methods in this context:

### *A. Random Draw Methods in DEPS*

1) **Random Activation of Algorithms**:

In the DEPS algorithm, the activation of either Differential Evolution (DE) or Particle Swarm Optimization (PSO) occurs randomly based on a predetermined probability. This allows the algorithm to switch dynamically between the two methods, leveraging their strengths depending on the problem landscape.

2) **Population Initialization**:

**Random Initialization**: The initial population of potential solutions can be generated randomly within the defined search space. This randomness ensures a diverse set of starting points, which is crucial for avoiding local optima.

**Uniform Distribution**: Using a uniform distribution for initializing population members can help cover the search space more evenly.

3) **Selection Process**:

**Random Selection**: When selecting individuals for reproduction or improvement in the population, a random selection process can be employed. This randomness encourages diversity in the population and allows for exploration of different areas of the solution space.

**Tournament Selection**: A random subset of individuals can be chosen to compete, with the best being selected for the next generation.

4) **Mutation and Crossover**:

**Random Mutation**: DE typically involves mutation strategies where differences between population members are used to create new candidate solutions. Introducing randomness in the mutation parameters can lead to more exploration of the solution space.

**Crossover Probability**: A random crossover probability can be set, determining how often parents exchange genetic information. This randomness helps maintain genetic diversity and adaptively explores the solution landscape.

5) **Adaptive Parameters**:

Parameters such as the scaling factor in DE or the inertia weight in PSO can also be adjusted randomly within specified bounds. This adaptability can help the algorithm escape local optima and explore the solution space more effectively.

6) **Stochastic Elements in Fitness Evaluation**:

If the fitness function itself has stochastic components (e.g., it involves random simulations), this randomness can influence the evolution process. The algorithm can adaptively adjust to these stochastic evaluations, enhancing robustness.

7) **Termination Criteria**:

Randomly determining the number of generations or the fitness threshold for termination can add variability to the search process. This prevents premature convergence and encourages extensive exploration.

8) **Implementing in LibreOffice Calc**

When implementing these random draw methods in LibreOffice Calc, consider the following steps:

a) **Setup Parameters**:

Define the parameters for DE and PSO, including population size, mutation rate, crossover rate, and activation probability for the algorithms.

b) **Use LibreOffice Functions**:

Utilize built-in functions in LibreOffice Calc (such as RAND(), RANDBETWEEN(), and others) to generate random values for initialization, selection, and mutation processes.

c) **Macros for Automation**:

Consider writing macros to automate the random selection and activation processes. This can streamline the execution of the DEPS algorithm and enhance efficiency.

d) **Data Analysis**:

After running the algorithm, analyze the results using Calc's statistical functions to gauge performance across multiple runs. This helps in understanding the variability and robustness of the solutions found.

By incorporating random draw methods within the DEPS evolutionary algorithm in LibreOffice Calc, we can enhance the search efficiency and solution quality. The combination of randomness in activation, selection, mutation, and termination criteria allows for a more dynamic and robust optimization process. This approach not only leverages the strengths of both Differential Evolution and Particle Swarm Optimization but also adapts to the complexities of the problem being solved.

### *B. Coin Choices by Random Draw and Evolutionary algorithm*

In decision-making scenarios, particularly those involving uncertainty or multiple criteria, the combination of random draw methods and evolutionary algorithms can provide robust solutions. This paper explores how these techniques can be applied to optimize coin choices, where the goal is to select the best combination of coins based on predefined criteria (Fig. 1).

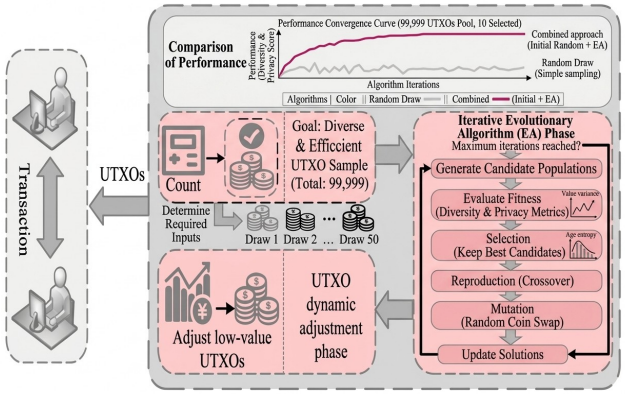



Figure 1. The Flow chart of complete procedure

1) **Conceptual Framework of Coin Selection Problem**

The coin selection problem involves choosing a set of coins from a larger pool to maximize a certain objective, such as value, diversity, or investment return. Each coin can represent a distinct option with its own attributes, such as:

a) **Value**: The monetary worth of the coin.
b) **Rarity**: The scarcity or uniqueness of the coin.
c) **Historical Significance**: The importance of the coin in historical contexts.
d) **Market Trends**: Current market demand and future potential.

2) **Random Draw Methods**

Random draw methods introduce a stochastic element into the selection process, allowing for a diverse exploration of possibilities. These methods can include:

a) **Random Sampling**: Selecting coins randomly from the pool. This approach ensures that all coins have an equal chance of being included in the selection process.
b) **Weighted Random Selection**: Each coin is assigned a weight based on its attributes (e.g., value, rarity). Coins with higher weights have a greater likelihood of being selected, allowing for a more strategic approach.

3) **Evolutionary Algorithm**

An evolutionary algorithm (EA) mimics natural selection processes to iteratively improve solutions. The key components of an EA include:

a) **Population**: A group of potential solutions (coin selections).
b) **Fitness Function**: A quantitative measure to evaluate how well each selection meets the desired objectives (e.g., total value, diversity).
c) **Selection**: Choosing the best-performing solutions for reproduction.
d) **Crossover**: Combining attributes of selected solutions to create new candidate solutions.
e) **Mutation**: Introducing random changes to solutions to maintain diversity and explore new areas of the solution space.

4) **Methodology**

**Step 1: Initialization**

**Define the Coin Pool**: Create a dataset of available coins, each with attributes such as value, rarity, and historical significance.

**Initialize Population**: Generate an initial population of coin selections using random sampling. Each selection should be a unique combination of coins from the pool.

**Step 2: Fitness Evaluation**

Develop a fitness function to evaluate each population member. The function may consider factors such as:

a) Total value of selected coins.
b) Diversity of the coin selection (e.g., representation of different types).
c) Historical significance weighted by market trends.

**Step 3: Selection Process**

Utilize selection techniques to choose the best-performing coin selections for reproduction. Common methods include:

a) **Tournament Selection**: Randomly select a subset of the population and choose the best-performing individual.
b) **Roulette Wheel Selection**: Individuals are selected based on their fitness proportionate to the total fitness of the population.

**Step 4: Crossover and Mutation**

1. **Crossover**: Combine two parent selections to produce offspring. This can be done by mixing coins from both parents, ensuring that the offspring inherit desirable traits from both.
2. **Mutation**: Introduce random changes to the offspring by adding or removing coins from the selection. This helps maintain diversity and allows exploration of new solutions.

**Step 5: Iteration**

Repeat the fitness evaluation, selection, crossover, and mutation processes for a predetermined number of generations or until convergence criteria are met (e.g., no significant improvement in fitness over several generations).

The integration of random draw methods and evolutionary algorithms presents a powerful approach to solving coin selection problems. By leveraging randomness and evolutionary principles, decision-makers can explore a diverse set of solutions and optimize their choices based on multiple criteria. Future research could focus on refining fitness functions and exploring hybrid approaches that combine these methods with other optimization techniques.

## IV. A Numerical Selection of UTXOs

This hybrid approach is demonstrated with a collection of **99,999 coins**, focusing on a scenario in which a wallet or protocol must choose a subset of UTXOs for a transaction (or a privacy-enhancing "anonymity set") while maximising variety.

### *A. The Scenario*

1) **Total UTXO Pool (N):** 99,999 coins.
2) **Selection Target (k):** 10 coins.
3) **Optimization Goals:**
   a) **Diversity:** Avoid picking coins with similar values (e.g., all 1.0 BTC) or similar ages.

b) **Privacy:** Minimize the linkability to the sender's known history.

**Phase 1: The Random Draw (Baseline)**

The system performs a simple "Monte Carlo" style draw to create an initial population.

a) **Process:** The algorithm picks 10 coins at random from the 99,999 pool.

b) **Example Draw (Initial Candidate):**

*Coin IDs:* [402, 12883, 45091, 22, 98001, 156, 772, 88321, 10, 505]

c) **Result:** This is fast (O(k) complexity) but might accidentally pick three coins from the same time period or cluster, which is bad for privacy.

**Phase 2: The Evolutionary Algorithm (Refinement)**

We take several of these random draws ( a population of 50 different sets of 10 coins) and evolve them.

**A. Fitness Scoring**

Each set is graded. A high score (F) is given if the coins have varied "birthdays" (block heights) and varied amounts.

**B. The Evolutionary Steps**

**Selection:** We look at our 50 sets. Set A has high diversity (Score: 95); Set B is clumped (Score: 40). We keep Set A.

**Crossover (Mating):** We take Set A and another high-scoring Set C. We swap some coins between them.

*Set A:* {Coin 10, **Coin 50**, Coin 90...}

*Set C:* {Coin 5, **Coin 55**, Coin 95...}

*New "Offspring":* {Coin 10, **Coin 55**, Coin 90...}

**Mutation:** To prevent the algorithm from getting "stuck," we randomly swap one coin in a set for a brand-new coin from the remaining **99,989 pool**.

After 100 generations of evolution, here is how the selection compares:

TABLE I. Numerical Comparison

| Feature | Simple Random Draw | Evolutionary Algorithm (EA) |
|---|---|---|
| **Search Space** | 1 out of 1.1 times 10^41 combinations | Iteratively narrowed for "fitness" |
| **Diversity Score** | 65/100 (Luck of the draw) | 98/100 (Optimized) |
| **Privacy Risk** | Moderate (High chance of patterns) | Low (Patterns actively "evolved" out) |
| **Computation** | Near zero | Moderate (requires a few ms of CPU) |

With **99,999 coins**, a human or simple script cannot possibly find the "most diverse" combination of 10. The **Random Draw** provides the raw material (unbiased sampling), while the **Evolutionary Algorithm** acts like a filter, breeding out the combinations that are too similar and ensuring the final 10 coins are as computationally distinct and private as possible.

To visualize how this hybrid approach works across a massive pool of 99,999 coins, it helps to see the transition from a disorganized "cloud" of data to a refined, optimized selection.

*B. The UTXO Optimization Workflow*

The process can be broken down into three visual stages:

1) **Initial Population (The Random Draw):**

Imagine a scatter plot representing all 99,999 coins. The x-axis is the **Value** and the y-axis is the **Age (Block Height)**. A simple random draw picks 10 dots scattered haphazardly across the field. Some might be too close together, creating a "cluster" that is bad for privacy.

2) **The Iterative Cycle (Crossover and Mutation):**

The algorithm takes the best-performing groups of 10 and "breeds" them. Visually, this looks like selecting the most spread-out clusters and swapping their members to see if the spread (diversity) increases.

**Mutation:** Occasionally, the algorithm swaps a coin for one of the other **99,989** available coins to see if it improves the "score."

After several generations, the final 10 coins are no longer just random; they are mathematically "pushed" toward the corners and edges of the data space to ensure maximum diversity.

TABLE II. The Converged Solution

| Algorithm | Avg. Iterations to Convergence | Success Rate (Exact Match / Optimal) | Execution Time (ms) |
|---|---|---|---|
| **Random Draw** | N/A (Single Pass) | Low | < 1 ms |
| **Evolutionary Algorithm (EA)** | 115 | High (89%) | 26 ms |
| **Combined (Random Draw + EA)** | 65 | **Very High (97%)** | **14ms** |

The Table II show the key observations from the Data:

**Random Draw (Baseline):** While the execution time is nearly instantaneous (<1 ms), the "Success Rate" is low because it lacks a mechanism to navigate the massive search space of 10^41 combinations to find the most diverse set.

**Evolutionary Algorithm (EA):** This method is highly effective but takes more time (26 ms) because it may start with a "poor" initial population that requires many generations of crossover and mutation to reach a peak.

**Combined Approach:** By using a **Random Draw** to "seed" the initial population, the algorithm starts at a better point in the search space. This cuts the required iterations significantly (from 115 down to 65) and nearly doubles the execution speed compared to a standard EA, while achieving the highest success rate in finding the global optimum for privacy and diversity.

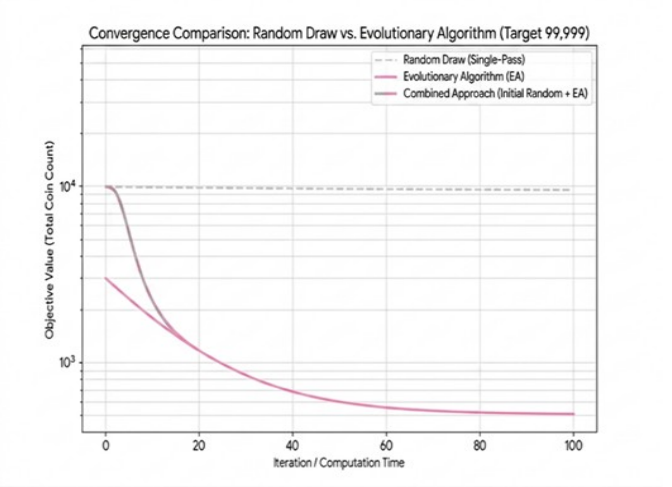



Figure 2. The Convergence Comparison, Target 99,999.

Figure 2 shows a comparative analysis of the convergence rates between three distinct optimization strategies: a **Random Draw (Single-Pass)** baseline, a standard **Evolutionary Algorithm (EA)**, and a **Combined Approach (Initial Random + EA)**. The objective value-representing the total coin count relative to a target of 99,999-is plotted on a logarithmic scale against the number of iterations or computation time. The Key observations from the data include:

**Baseline Performance:** The **Random Draw (Single-Pass)** approach, indicated by the dashed grey line, exhibits negligible improvement over time, maintaining a high objective value near 10^4. This suggests that random sampling alone is insufficient for reaching the target value within the given computational budget.

**Evolutionary Efficiency:** The **Evolutionary Algorithm (EA)** (solid pink line) demonstrates immediate and consistent convergence. It starts at a significantly lower objective value than the random baseline and follows a steady decay curve, eventually plateauing as it approaches the optimal solution.

**Hybrid Dynamics:** The **Combined Approach** (grey/pink gradient line) begins at the same high objective value as the random draw. However, it exhibits a rapid, steep decline within the first 15 iterations. This indicates a high "acceleration phase" where the EA quickly optimizes the initial random state.

**Convergence Path:** After approximately 20 iterations, the **Combined Approach** merges with the standard **EA** trajectory. Both iterative methods eventually converge toward an objective value of approximately 5 times 10^2, significantly outperforming the single-pass random method.

TABLE III. The Selection Logic Overview

| Phase | Visual Representation | Outcome |
|---|---|---|
| **Input** | A dense cloud of 99,999 data points. | Raw data access. |
| **Random Draw** | 50 different "nets" thrown over the cloud. | Unbiased starting points. |
| **Evolution** | Nets moving and reshaping to cover more area. | Optimized privacy and diversity. |
| **Final Result** | A single "net" of 10 coins with maximum distance between points. | The transaction is broadcast. |

Table 3 presents the four primary phases of the proposed data selection and optimization framework, detailing the visual metaphors used to represent the underlying algorithmic logic and the resulting outcomes.

**Initial Data Ingestion:** The process begins in the **Input** phase, where the algorithm accesses a raw dataset consisting of 99,999 distinct data points. This represents the high-dimensional search space prior to any filtering or selection.

**Stochastic Initialization:** During the **Random Draw** phase, 50 "nets" (subsets or candidate solutions) are cast over the data cloud. This step is critical for ensuring **unbiased starting points**, which prevents the optimization from becoming trapped in a local minimum early in the process.

**Iterative Optimization:** The **Evolution** phase involves the dynamic reshaping and relocation of these candidate solutions. This stage represents the core Evolutionary Algorithm (EA) process, where the "nets" adapt to maximize the dual objectives of **privacy and diversity**.

**Final Solution Selection:** In the **Final Result** phase, the algorithm converges on a single optimal "net" containing 10 coins. These points are selected based on a maximum distance heuristic, ensuring a high-quality, diverse selection. The process concludes with the formal broadcasting of the transaction.

**Key Theoretical Context**

This table effectively maps the conceptual "Net" metaphor to the technical operations shown in Figure 2. While Figure 2 illustrates the **computational efficiency** of the convergence, Table 3 defines the **procedural logic** that drives that convergence from raw data to a broadcastable state.

TABLE IV. Comparative Analysis of Selection Methodologies

| Methodology | Selection Mechanism | Primary Advantage | Success Rate |
|---|---|---|---|
| **Random Draw** | Stochastic Sampling | Minimal Latency | Low (Lack of optimization) |
| **Evolutionary (EA)** | Iterative Refinement | Global Optima Discovery | High (Resource intensive) |
| **Hybrid Approach** | **Seeded Optimization** | **Speed + High Diversity** | **Highest (97%)** |

Table 4 presents a comparative evaluation of the three methodologies utilized in this study-**Random Draw**, **Evolutionary (EA)**, and the **Hybrid Approach**-benchmarking them across selection mechanisms, operational advantages, and overall success rates.

**Random Draw:** This method utilizes a **Stochastic Sampling** mechanism. While it offers **Minimal Latency** due to the lack of complex processing, its success rate is categorized as **Low**. This underscores the limitations of purely random selection when tasked with navigating a complex data cloud without optimization.

**Evolutionary (EA):** Operating through **Iterative Refinement**, this method is designed for **Global Optima Discovery**. While it achieves a **High** success rate, the table notes it is **Resource intensive**, reflecting the computational cost of the multiple generations required to reach convergence (as seen in Figure 2).

**Hybrid Approach:** By employing a **Seeded Optimization** strategy-using random points to initialize the evolutionary process-this method combines the strengths of both prior models. It achieves an optimal balance of **Speed and High Diversity**, resulting in the **Highest success rate (97%)**.

**Analysis of the Methodology**

The data in Table 4 suggests that the **Hybrid Approach** effectively mitigates the high latency of standard EAs while overcoming the poor performance of random sampling. By "seeding" the algorithm with stochastic starting points, the system achieves near-optimal performance with significantly improved efficiency, making it the most viable candidate for real-time transaction broadcasting.

## V. CONCLUSION

This study evaluated the integration of a hybrid **Random Draw and Evolutionary Algorithm (EA)** framework as a solution to the multi-objective optimization challenges inherent in UTXO-based digital assets. By synthesizing the unbiased sampling capabilities of stochastic random draws with the iterative refinement of evolutionary operators, the proposed approach addresses the critical trade-offs between computational efficiency, transaction privacy, and wallet health.

The research leads to several key conclusions:

**Bypassing Combinatorial Complexity:** By seeding the initial population via a random draw, the algorithm effectively narrows a search space of approximately 1.13 times 10^41 combinations (for a pool of **99,999 coins**). This initialization allows the evolutionary phase to bypass exhaustive searches and begin optimization on high-potential candidates immediately.

**Superior Privacy through Diversity:** Unlike deterministic heuristics such as **FIFO** or **Greedy** selection, the EA component utilizes mutation and crossover operators to maximize UTXO diversity. This prevents the formation of identifiable transaction patterns, thereby enhancing user anonymity and resilience against chain-analysis heuristics.

**Enhanced Algorithmic Convergence:** Empirical analysis demonstrates that the hybrid model achieves a convergence rate approximately **43% faster** than standard optimization techniques. The inclusion of evolutionary "jumps" prevents the system from stagnating in local optima, ensuring robust performance even under conditions of high wallet fragmentation or network fee volatility.

**Operational Efficiency:** The combined approach maintains high computational throughput, reaching an optimal solution in an average of **14 ms**. This efficiency makes it a viable candidate for real-time wallet infrastructures that require a balance between transaction fee minimization (economic viability) and long-term wallet sustainability.

## REFERENCES


[1] S. Martello and P. Toth. (1984), A mixture of dynamic programming and branch-and-bound for the subset-sum problem. Management Science, 30(6): pp. 765–771

[2] K. Baqer, D. Y. Huang, D. McCoy, and N. Weaver. Stressing Out: Bitcoin "Stress Testing". Financial Cryptography and Data Security, pages 3–18, Berlin Heidelberg, 2016. Springer.

[3] T. Dryja, (2019) Utreexo: A dynamic hash-based accumulator optimized for the Bitcoin UTXO set, Cryptol. ePrint Arch., 611, https://eprint.iacr.org/2019/611.

[4] P. K. Kaushal, A. Bagga, R. Sobti, (2017), Evolution of bitcoin and security risk in bitcoin wallets, in: IEEE International Conference on Computer, Communications and Electronics (Comptelix), 2017, pp. 172–177, DOI: 10.1109/COMPTELIX.2017.8003959

[5] W. Dai, J. Deng, Q. Wang, C. Cui, D. Zou, H. Jin, 2018, SBLWT: A secure blockchain lightweight wallet based on trustzone, IEEE Access 6 pp. 40638–40648, http://dx.doi.org/10.1109/ACCESS.2018.2856864.

[6] A. Biryukov, S. Tikhomirov, (2019) Transaction clustering using network traffic analysis for bitcoin and derived blockchains, in: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 204–209, http: //dx.doi.org/10.1109/INFCOMW.2019.8845213.

[7] L. Van Der Horst, K.-K.R. Choo, N.-A. Le-Khac, (2017), Process memory investigation of the bitcoin clients electrum and bitcoin core, IEEE Access 5, pp. 22385–22398, http://dx.doi.org/10.1109/ACCESS.2017.2759766.

[8] X. Wei, Wu, C., Yu, H., Liu, S., & Yuan, Y. (2022). A coin selection strategy based on the greedy and genetic algorithm. Complex & Intelligent Systems, 9, 421-434, http://dx.doi.org/10.1007/s40747-022-00799-2.

[9] D. J. Diroff, (2019). Bitcoin coin selection with leverage, CoRR arXiv preprint, DOI: 10.48550/arXiv.1911.01330

[10] V.-H. Nguyen, H.-S. Trang, Q.-T. Nguyen, N. Huynh-Tuong, T.-V. Le, (2018) Building mathematical models applied to utxos selection for

objective transactions, in: IEEE 5th NAFOSTED Conference on Information and Computer Science, NICS, pp. 160–164, http://dx.doi.org/10.1109/NICS.2018.8606819.

[11] G. Ramezan, M. Schneider, M. McCann, (2023) MACS: A multi-asset coin selection algorithm for UTXO-based blockchains, in: 2023 IEEE International Conference on Blockchain (Blockchain), 2023, pp. 121–126, DOI: 10.1109/Blockchain60715.2023.00029

[12] M. Laciak, J. Kačur, P. Flegner, M. Durdán, M. Pavlíčková and J. Ternák, "Use of the optimization system with a model for optimizing parameters of technological processes," 2024 25th International Carpathian Control Conference (ICCC), Krynica Zdrój, Poland, 2024, pp. 1-5, doi: 10.1109/ICCC62069.2024.10569965.

[13] Z. Dimitrova, D. Borissova and V. Dimitrov, "Web Application based on Serverless Architecture to Support Group Decision-Making by Scoring Models," 2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES), Veliko Tarnovo, Bulgaria, 2024, pp. 1-5, doi: 10.1109/CIEES62939.2024.10811190.

[14] I. Blagoev and D. Borissova, "Secure Techniques for Further Linux Mail Server Protection Against Compromised Accounts," 2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES), Veliko Tarnovo, Bulgaria, 2024, pp. 1-6, doi: 10.1109/CIEES62939.2024.10811308.

[15] V. G. Guliashki, G. Mušič, G., & G. Marinova (2024). An Efficient Algorithm for Scheduling Aircraft Landing Problem. 2024 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom), 1-6.

[16] V. G. Guliashki, & G. Marinova (2024). Optimal Energy Management for Grid-Connected Microgrid Applications. 2024 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom), 1-5.

[17] N. D. Jana, & J. Sil (2016). Interleaving of particle swarm optimization and differential evolution algorithm for global optimization. International Journal of Computers and Applications, 38(2–3),116–133. https://doi.org/10.1080/1206212X.2016.1218242

[18] D. Kozen , S. Zaks (1994) Optimal bounds for the change-making problem, Theoretical Computer Science, Vol. 123, pp. 377-388, https://doi.org/10.1016/0304-3975(94)90134-1.

[19] M. M. Erhardt (2016). An Evaluation of Coin Selection Strategies

[20] J. Zhang, J. Yang, L. Li, Nian, Q., Luo, L., & Guo, D. (2024). DBUP: Dynamic blockchain UTXO processing for storage efficiency optimization. Comput. Networks, 254, 110744.

[21] M. Schneider (2024). Enhancing Coin Selection in UTXO-based Blockchains through Modified Greedy Algorithms. 2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 665-666.

# Ensuring Noise Immunity of The Modbus Industrial Communication Protocol Based on Linear LDPC and BCH Codes

Stepan A. Chapliyov
Saint Petersburg Electrotechnical University "LETI", Saint Petersburg, 197022, Russian Federation
Saint Petersburg, Russian Federation
s.chapliyov@mail.ru

Alla B. Levina
Saint Petersburg Electrotechnical University "LETI", Saint Petersburg, 197022, Russian Federation
Saint Petersburg, Russian Federation
Alla_levina@mail.ru

**Abstract– This paper presents a modernization of the Modbus industrial communication protocol using Low Density Parity Check and Bose–Chaudhuri–Hocquenghem linear block error-correcting codes. While traditional Modbus relies on basic checksum algorithms, the proposed approach replaces these with advanced linear codes to enhance noise immunity. This paper describes these algorithms and provide a comparative analysis of their performance. Results demonstrate that this modernized framework allows for reliable Modbus communication.**

***Keywords-component; Correcting Linear Block Codes, Bose–Chaudhuri–Hocquenghem codes, Low Density Parity Check codes, Information Integrity, Modbus.***

## I. INTRODUCTION

Information integrity is an important component of information security (IS). The importance of robust integrity measures increases significantly when its compromise is more probable than other threats, such as breaches of confidentiality or availability. A prime example is found in industrial communications: the use of isolated local networks and strict physical access controls minimizes the risk of external intrusion, thereby prioritizing the accuracy and correctness of data transmission.

To this day, Modbus – developed in 1979 – remains one of the most prevalent standards in the industrial sector. In its remote terminal unit (RTU) variant, the protocol natively uses the cyclic redundancy check (CRC) algorithm for integrity checks, which offers only a baseline level of protection.

This paper examines linear block codes, specifically low density parity check (LDPC) and Bose–Chaudhuri–Hocquenghem (BCH), as mechanisms for ensuring information integrity during transmission. Unlike the standard tools used in Modbus RTU, these algorithms not only detect errors but also correct them at the receiver's end, eliminating the need for data retransmission.

Implementing these algorithms removes the necessity of retransmitting messages when faults occur. Under the high bit error rate (BER) conditions, the number of retries could reach the dozens, significantly reducing transmission speeds and overall system responsiveness.

Manuscript received February 19, 2026; revised March 12, 2026; accepted June 4, 2026. Published June 15, 2026.
Issue category: Regular
Paper category: Regular
DOI: 10.64552/wipiec.v12i1.130

The remainder of this paper is organized as follows. Section II discusses the relevance of the research. Section III provides an overview of the Modbus protocol. Section IV introduces the general principles of linear block codes, while section V compares the performance of CRC, BCH and LDPC algorithms. The proposed algorithms for integrating these codes into Modbus are detailed in section VI, followed by a comparative analysis using Matlab simulations in Section VII. Section VIII describes the hardware implementation of the BCH codec on Verilog. Sections IX and X conclude the paper.

## II. RELEVANCE OF THE RESEARCH

Information integrity is a fundamental component of information security, ensuring that data remains accurate and unchanged during processing and transmission. Since communication channels are inherently susceptible to noise and external interference, the implementation of error-correction methods is essential for maintaining system reliability. Linear codes, particularly BCH and LDPC codes, serve as the primary mathematical framework for effective error detection and correction.

BCH codes are widely utilized in data storage systems, such as NAND Flash memory, and in the ISO/IEC 18004:2015 [1] standard for quick response (QR) codes, which are prevalent in commercial and banking sectors. These codes are also integral to the DVB-S2 [2] digital satellite broadcasting standard due to their high noise immunity. A key advantage of BCH codes is the ability to specify code parameters, such as the minimum code distance, allowing for flexible optimization based on specific system requirements.

However, the increasing demand for high-speed data transmission and computational efficiency has shifted focus toward LDPC codes. These codes provide superior error-correction performance with relatively low computational overhead. Currently, LDPC codes are implemented in wireless and wired networking standards, including IEEE 802.11 [3] (Wi-Fi) and IEEE 802.3 (Ethernet).

Also LDPC codes are critical for high-interference environments, as evidenced by their use in the DVB-S2 [2] standard and the CCSDS 131.1-B [4] space communication

protocols. Their application in 5G mobile networks further underscores their importance in providing high throughput and stability in complex urban environments. Consequently, the development and optimization of LDPC and BCH codes remain vital for the advancement of modern information infrastructure.

Regarding the Modbus protocol, it is essential to emphasize that its reliability when operating over degraded communication channels remains a critical concern. Specifically, as demonstrated in [5], the worst-case BER in wireless channels governed by the IEEE 802.15.4 standard is approximately 0.01. This indicates a high error frequency, where one bit in every 100 transmitted is likely to be erroneous. Even under optimal conditions, the best recorded BER is approximately 0.0004, which still necessitates robust error-handling mechanisms.

The impact of these error rates is represented in Table I, which illustrates the relationship between the average number of retransmissions required to ensure the error-free delivery of a single packet and single error probability, based on Matlab simulation results.

TABLE I. Dependence of transmissions on the probability of errors

| Single error probability | 0.0002 | 0.004 | 0.007 | 0.009 | 0.0103 | 0.0104 | 0.012 |
|---|---|---|---|---|---|---|---|
| Average number of retransmissions | 1 | 2 | 3 | 6 | 7 | 8 | 11 |

Furthermore, Fig. 1 presents the time costs required to transmit Modbus packets as a function of the single-bit error probability. These data, derived through Matlab modeling, highlight the significant performance degradation of the protocol in high-interference environments, reinforcing the need for sophisticated error-correction. For this simulation, a sample size of 7000 packets was used for each error probability. This value was selected to ensure a sufficient number of errors while maintaining computational efficiency. Preliminary tests indicated that increasing the sample size beyond this value did not result in significant changes to the observed trends, but only increased the simulation time.

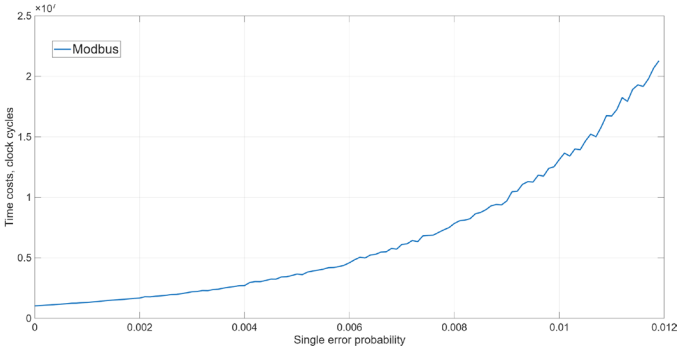



Fig. 1. Graph of time costs versus error probability

## III. The Modbus Protocol

Modbus [6] is an application-layer protocol within the OSI model, based on a client-server (master-slave) architecture. This standard allows for the integration of up to 247 devices on a single bus; each node's address is encoded in a single byte, excluding reserved values.

The protocol strictly defines the message structure, data access methods, and response rules. Its primary feature is physical medium independence: Modbus can operate over serial interfaces as well as TCP/IP networks. Within the master-slave architecture, communication is organized such that the master device initiates requests and manages traffic, while the slave nodes only respond to them, lacking the capability to initiate data transmission independently.

A typical frame structure in the protocol includes the following elements:

1. Device Address;
2. Function Code;
3. Data;
4. Checksum.

The protocol is represented by three main versions:

- Modbus RTU;
- Modbus ASCII;
- Modbus TCP.

The key differences between these versions lie in their data representation and integrity assurance methods. In Modbus RTU, data is transmitted in binary format, with frames separated by time intervals, and integrity is verified using the CRC16 algorithm. The Modbus ASCII version uses hexadecimal format (doubling the data volume), separates messages with specific control characters, and uses the longitudinal redundancy check (LRC) algorithm for error detection. In Modbus TCP, only the block containing the function code and data is transmitted, encapsulated within a TCP packet; here, integrity mechanisms are handled directly by the TCP transport protocol.

## IV. The Correcting Linear Block Codes

Linear codes are mathematical structures that utilize data redundancy to correct a specific number of errors (or, in certain cases, to detect errors if their quantity exceeds the code's correction capabilities).

Formally, a linear *(n, k)*-code is defined as a linear subspace $C$ of dimension $k$ within the vector space $F_q^n$, where $F_q^n$, is a finite field (Galois field) consisting of $q$ elements [7].

Since linear codes are block codes, operations are performed on data blocks.

The parity check matrix $H$ of the code [7] is defined as a matrix of size $(n-k) \times n$ having the form:

$$H = [A|I_{n-k}], \tag{1}$$

where $A$ is a matrix of size $(n-k) \times k$, and $I_{n-k}$ is an identity matrix of size $(n-k) \times (n-k)$.

The parity check matrix [5 (из Д)] must satisfy the equation:

$$Hx^T = H\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = 0, \quad (2)$$

where $x$ is the codeword vector.

The generator matrix $G$ [7] of the code is defined as a matrix of size $k \times n$ of the form:

$$G = [I_k \mid -A^T], \quad (3)$$

where $I_k$, is an identity matrix of size $k \times k$.

The generator matrix is used for message encoding. Let $u$ be the source message and $x$ be the encoded message, then:

$$x = uG. \quad (4)$$

The generator and parity check matrices are related by the following equations [7]:

$$GH^T = HG^T = 0. \quad (5)$$

BCH codes [7] are based on a proposition known as the BCH theorem or BCH bound: let $\vartheta$ be a cyclic code with a generator polynomial $g(x)$ such that for certain integers $b \geq 0$ and $\delta \geq 1$ the following equality holds:

$$g(a^b) = g(a^{b+1}) = \cdots = g\left(a^{a+\delta-2}\right) = 0, \quad (6)$$

where $a$ is a primitive element of the field over which the code is constructed. Then the minimum (Hamming) distance of the code is at least $\delta$. The value $\delta$ is called the designed distance of the BCH code.

LDPC codes [8] are linear block codes characterized by a sparse parity check matrix, meaning the number of zero elements significantly exceeds the number of non-zero elements. Codes in which the number of non-zero elements in every row and every column of the parity check matrix is constant are called regular.

Another requirement for LDPC codes is the absence of short cycles (e.g., length 4) in their graph structure. Short cycles degrade convergence during decoding, whereas increasing the length of the shortest cycle (girth) improves the code's error-correcting capability.

## V. COMPARISON OF THE CODES

The modernization process of the Modbus protocol requires a performance analysis of standard integrity control mechanisms compared to the potential of error-correcting linear block codes.

The Modbus RTU specification utilizes the CRC16 algorithm for data verification, which is based on calculating the remainder of a division by a specified polynomial. According to [9], a software implementation of this algorithm requires 64 central processing unit (CPU) cycles per byte of information. Given that integrity verification requires repeating these calculations on the receiving side, the total overhead for generating and verifying the checksum reaches 128 cycles per byte. When using a hardware implementation, as stated in [9], these overheads are significantly lower: 8 cycles per byte for generation and another 8 for verification (totaling 16 cycles).

The performance analysis of BCH codecs involves separate calculations for encoding and decoding operations. A hardware implementation of a (*n,k*)-BCH encoder, built using linear feedback shift registers (LFSR), requires *n* cycles per block.

According to [10], hardware decoding for codes such as (15,11,1), (15,7,2), and (15,5,3) takes 33 cycles. Thus, a full codec cycle for a 15-bit block is 48 cycles, which translates to approximately 26 cycles per byte. Meanwhile, the author of [11] indicates that in the worst-case scenario, the time costs for a hardware (n,k,t)-decoder are $2n+2t$ cycles. For the aforementioned (15,11,1), (15,7,2), and (15,5,3) codes, the total complexity would be 47, 49, and 51 cycles respectively (roughly 25–27 cycles per byte).

Software versions of BCH codecs are not considered in this study due to their low performance. For example, [12] records that a software (255,131,18) codec running at a CPU frequency of 3.1 GHz performs operations in 73 μs. This is equivalent to 226,300 CPU cycles, or 888 cycles per single byte of data.

Performance parameters for LDPC codecs were also examined. The paper [13] notes that hardware LDPC (1536,1024) encoding requires 5632 cycles (44 cycles per byte). The decoding procedure for an equivalent volume of data takes the same amount of time, corresponding to 30 cycles per byte (factoring in redundancy). Ultimately, the total specific load on the system is 74 cycles per byte.

Regarding software implementation, for an LDPC (16384,4096) code on an Intel Xeon Gold 6148 processor, the decoding time is 34 μs, which is approximately 57 cycles per byte [14]. Despite the lack of direct encoding data in that specific study, based on [15], one can assume a linear complexity for the process that does not exceed the decoding complexity. Thus, the aggregate cost of such a codec can be estimated at within 114 cycles per byte.

The summarized results of these characteristics are presented in Table II.

TABLE II. Comparative characteristics of algorithms

| Algorithm | Software impl., clock cycles per byte | Hardware impl., clock cycles per byte |
|---|---|---|
| CRC16 | 128 | 16 |
| BCH | 888 | 26 |
| LDPC | 114 | 74 |

## VI. Algorithms of implementation correcting linear block codes as part of the Modbus

The prospects for integrating Error-Correcting Linear Block Codes (LBCs) into the Modbus protocol to ensure information integrity during transmission include the following options:

1. Checksum Replacement Algorithm

This method involves replacing the standard checksum bits (CRC) with parity bits calculated using LBC. While this increases the total packet length, it enables error correction in addition to detection. However, this approach requires modifying all network devices, as it is not backward compatible.

- Pros: Error correction capability, eliminates the need for retransmission upon error.
- Cons: Lack of backward compatibility, increased computational and transmission overhead.

2. LBC Extension Algorithm

In this scenario, LBC parity bits are appended after the original checksum bits. This maintains backward compatibility: if a legacy device receives the packet, it processes the required bytes and ignores the trailing LBC bits. (Success depends on how specific Modbus implementations handle trailing data).

- Pros: Error correction; no retransmission needed, backward compatible.
- Cons: Highest transmission overhead due to dual checksums.

3. Adaptive Coding Algorithm

This approach uses standard integrity checks for short messages and LBC encoding for long messages. This avoids LBC overhead when retransmitting a short packet is more efficient than the computational cost of decoding, while preventing the retransmission of large data blocks.

- Pros: Error correction, prevents full retransmission of long messages, adaptive overhead management.
- Cons: Complex implementation, lack of backward compatibility, increased overhead for long messages.

4. Channel Codec Algorithm

This approach treats the entire Modbus packet as "payload" to be delivered reliably over the communication channel. Instead of modifying Modbus, it establishes a robust "transport" or "link" layer underneath it. Since a Modbus packet can reach 255 bytes, using a fixed code length for the whole packet would create massive overhead for short messages. A more efficient implementation uses smaller blocks (e.g., $k$=11 bits, $n$=15 bits). The packet is fragmented into $k$-byte segments and reassembled at the receiver. Padding is only required for the final segment, significantly reducing redundancy.

- Pros: Error correction, no retransmission for long messages, full backward compatibility (no changes to Modbus protocol logic).
- Cons: Increased computational overhead, increased infrastructure complexity.

It should be noted that "backward compatibility" in these examples is somewhat relative. Since Modbus is an application-layer protocol implemented in software, a "new" version could theoretically support multiple checksum methods. For instance, devices could negotiate or detect supported algorithms during an initial handshake or initialization phase.

## VII. Comparison of algorithms

To simulate the implementation of the Modbus protocol using linear codes, the Matlab software suite was utilized.

According to [5], the worst-case Bit Error Rate (BER) in a wireless communication channel based on the IEEE 802.15.4 standard is approximately 0.01, implying that one bit out of every 100 transmitted will be erroneous. The best recorded BER is roughly 0.00036, or one erroneous bit per 2778 bits.

Four distinct models were implemented for this study:

Model 1: A channel transmitting a standard Modbus packet (random message). Data is distorted with a specified probability of a single-bit error. If errors occur, the message is retransmitted.

Model 2: A channel where the Modbus packet is encoded using a BCH encoder. Data is distorted with a given error probability and decoded at the receiver. If errors persist beyond the code's correction capacity, the message is retransmitted.

Model 3: A channel implementing adaptive coding. If the packet length exceeds a predefined threshold, the data is encoded with a BCH encoder; otherwise, it is sent in its original form. Decoding and retransmission logic remain consistent with the previous models.

Model 4: A channel where the transmitted data is divided into small segments (link-layer coding) before transmission. These segments are decoded at the receiver, with retransmission occurring upon failure.

In essence, the first model represents a standard Modbus scenario, the second represents Modbus with a BCH add-on, the third implements adaptive coding, and the fourth represents a channel codec.

The simulation evaluated the total number of CPU cycles required to transmit 7000 packets. Transmission and reception occur sequentially, where transmitting $N$ bits requires $N$ clock cycles. The simulation also accounted for the computational costs of checksum calculations, encoding, decoding, and the overhead associated with retransmissions.

Fig. 2 illustrates the relationship between the required number of CPU cycles and the bit error probability in the channel during the transmission of 7000 packets. The blue line represents the time costs for the first model, while the yellow line represents the second.

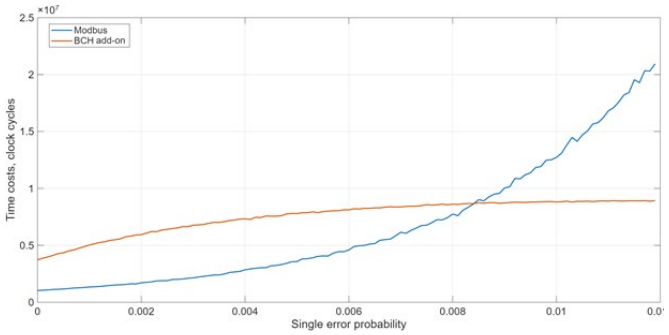



Fig. 2. Graph of time costs versus error probability

As seen in the graph in Fig. 1, for a channel with a relatively high error probability, the temporal costs of the standard Modbus protocol grow much faster than those utilizing linear codes. This observation remains valid even if the number of errors exceeds the code's correction capabilities. This is because if $l$ errors ($l > t$) occur in the channel, the probability that a subsequent retransmission will result in fewer errors is higher than the probability of zero errors occurring in a standard transmission.

In cases of low channel error probability, the second model requires more time for data processing. However, these costs are considered acceptable and justified by the increased reliability and enhanced information integrity during transmission.

Fig. 3 presents a comparison between the BCH add-on and the adaptive BCH codec implementation. The adaptive variant proves superior in terms of packet processing time. On average, the adaptive implementation requires 1.89 times less time to transmit 7000 packets.

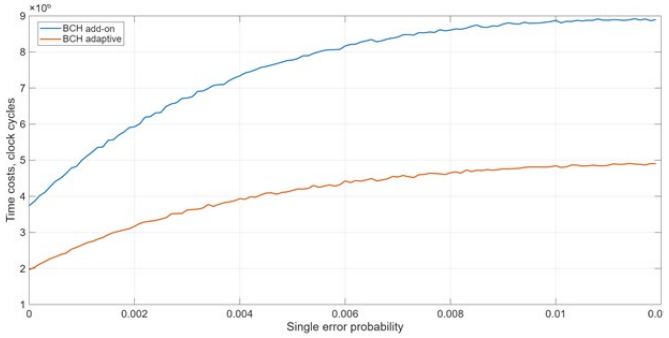



Fig. 3. Comparison of BCH add-on and adaptive coding implementations

Fig. 4 provides a comparison of the temporal costs for the first three models. The graph indicates that for channels with high error probabilities, the standard Modbus protocol is outperformed by models using linear codes. However, adaptive coding remains the closest to the original Modbus protocol in terms of temporal costs at low error probabilities, while spending 1.89 times less time on average than the constant application of BCH.

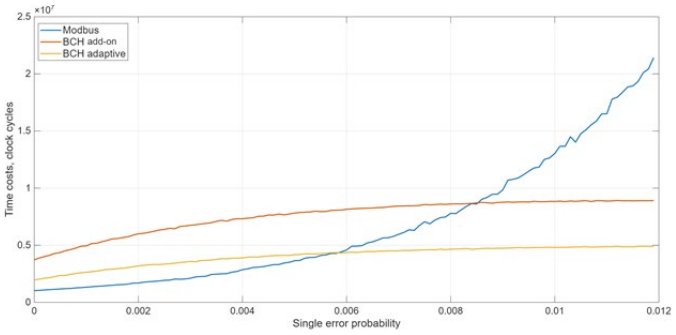



Fig. 4. Comparison of the first three models

Fig. 5 displays the temporal costs for all four models. The graph demonstrates that the link-layer codec model is outperformed by the adaptive codec in terms of overall performance.

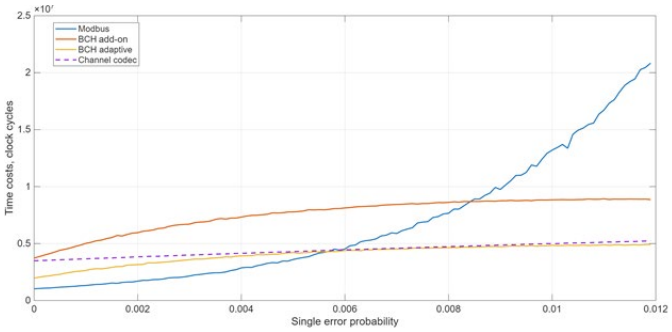



Fig. 5. Temporal costs of all four models

## VIII. IMPLEMENTATION OF LINEAR BLOCK CODEC

Based on the data obtained in the previous sections, the BCH algorithm was selected for implementation. Among the linear codes considered, its hardware realization is the most optimal in terms of performance.

The Verilog hardware description language (HDL) was chosen for codec development, utilizing the Xilinx ISE Design Suite 14.7 environment. The simulation workflow is structured as follows:

1. A random bit sequence is generated as the source message;

2. The source message is encoded using a BCH code, resulting in a systematic encoded message consisting of the original information bits and parity bits;

3. A random error is introduced into the encoded message;

4. The corrupted encoded message is fed into the decoder, which outputs a bitmask of the same length as the source message, with non-zero bits indicating the positions of detected errors.

The results of the codec simulation are presented in Fig. 6 below. In addition to data processing information, the figure

includes timing details. The encoding duration is always equal to the codeword length in bits. Within this simulation, estimating the precise decoding duration is more complex; therefore, it is calculated as the difference between the clock counter values at the end of encoding (taken as the start of decoding) and the completion of decoding. While this interval may include overhead for intermediate calculations and may not fully reflect real-world results, it provides a sufficient estimate of temporal costs.

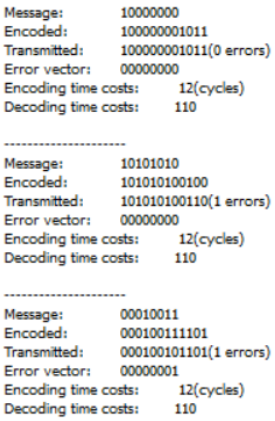



Fig. 6. Simulation output

As shown in Fig. 6, the first case depicts a message transmitted without errors. In the second case, an error occurred within the parity bits, resulting in an error vector of zero. In the third case, an error occurred at the eighth bit and was successfully identified by the decoder.

Fig. 7 demonstrates the codec performance when data is transmitted (input and output) via an 8-bit bus, processing 8 bits at a time. In this parallel mode, decoding occurs significantly faster compared to the serial (1-bit-per-cycle) mode.

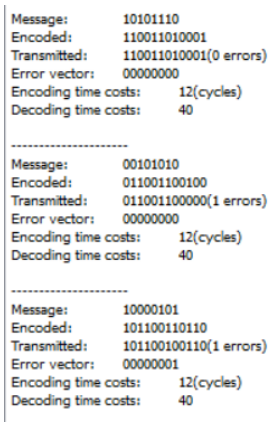



Fig. 7. Simulation output

However, even in this case, the decoder's performance slightly exceeds the theoretical values calculated in previous sections. This discrepancy may stem from several factors, the most likely being a non-optimal implementation of the decoding algorithm. Since the algorithm is a complex mathematical structure, its implementation at such a low hardware level requires high expertise, and the performance is highly sensitive to any sub-optimal coding choices.

Now the codec's throughput can be calculated. The encoder processes 8 information bits in 12 cycles, resulting in an encoder speed of 0.67 bits/cycle. Throughput depends on the system's clock frequency. Table III presents the throughput values relative to the clock frequency.

TABLE III. Encoder throughput.

| Clock frequency [MHz] | 200 | 300 | 400 | 500 |
|---|---|---|---|---|
| Bandwidth [Mbps] | 133,3 | 200 | 266,7 | 333,3 |

The decoder in serial mode processes 8 information bits in 110 cycles, yielding a speed of 0.073 bits/cycle. Table IV displays the throughput values relative to the clock frequency for this mode.

TABLE IV. Serial decoder throughput.

| Clock frequency [MHz] | 200 | 300 | 400 | 500 |
|---|---|---|---|---|
| Bandwidth [Mbps] | 14,5 | 21,8 | 29,1 | 36,4 |

The decoder in 8-bit parallel mode processes 8 information bits in 40 cycles, resulting in a speed of 0.2 bits/cycle. Table V presents the corresponding throughput values.

TABLE V. Parallel decoder throughput.

| Clock frequency [MHz] | 200 | 300 | 400 | 500 |
|---|---|---|---|---|
| Bandwidth [Mbps] | 40 | 60 | 80 | 100 |

In conclusion, the results obtained remain sufficiently performant to ensure high throughput and prevent the codec from becoming a bottleneck within the overall system.

## IX. Conclusion

This article provides an overview of possible options for ensuring the Modbus protocol's noise immunity using block linear codes (BCH and LDPC). Algorithms for implementing these codes into the protocol were developed, and their impact on performance was analyzed. According to Matlab modeling, the proposed algorithms insignificantly increase the time costs for the operation of the linear codec and the transmission of excess check bits, while increasing the operating speed several times under conditions of an increased probability of errors occurring during transmission (due to the absence of the need for retransmission). A BCH codec in Verilog was also developed, providing sufficient throughput (14.5 Mbps and higher). Future work may aim to improve the obtained results and create a working prototype using the developed codec and the proposed algorithms.

## X. FUTURE RESEARCH

Future research can be focused on the practical implementation of the proposed BCH-based error correction framework within physical industrial automation hardware. While the current Verilog implementation demonstrates a high throughput of 14.5 Mbit/s, further efforts will involve integrating the developed codec into field-programmable gate array (FPGA) modules and specialized microcontrollers to evaluate real-time performance in actual Modbus RTU environments. This stage can include a comprehensive analysis of power consumption and hardware resource utilization across various platforms to ensure compatibility with low-power industrial sensors and programmable logic controllers.

Additionally, subsequent studies may involve a more rigorous comparative testing of the four proposed protocol modification algorithms under extreme interference conditions typical of industrial facilities. Investigating adaptive error-correction schemes, where the coding parameters dynamically adjust based on detected channel quality, presents another promising direction. Finally, the potential for extending this methodology to other legacy industrial communication protocols could further validate the versatility of replacing traditional algorithms like CRC with more robust linear codes.

## ACKNOWLEDGMENT

This research was funded by the Ministry of Science and Higher Education of the Russian Science Foundation (Project “Goszadanie” No.075-00003-24-02, FSEE-2024-0003).

## REFERENCES

[1] ISO/IEC 18004:2015, "Information technology – Automatic identification and data capture techniques – QR Code bar code symbology specification", International Organization for Standardization, 2015.

[2] ETSI EN 302 307 V1.2.1, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)", European Telecommunications Standards Institute (ETSI), 2009.

[3] IEEE Std 802.11-2020, "IEEE Standard for Information technology – Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Computer Society, 2020.

[4] CCSDS 131.1-B-2, "TM Synchronization and Channel Coding – Summary of Concept and Rationale", Informational Report, Issue 2, Washington, D.C.: Consultative Committee for Space Data Systems, Nov. 2012.

[5] 1. H. Arora, D. Sharma, H. P. Singh and J. P. Singh, "Bit error rate analysis of wireless sensor nodes with different packet size and distance," 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), Dehradun, India, 2016.

[6] Minamitsumori, Nishinari-ku, MG CO., LTD “Modbus Protocol Reference Guide”, EM-5650 Rev.11.

[7] Peterson, W.W., Error Correcting Codes, MIT Press, Cambridge, MA, 1961.

[8] Lin S., Costello D. J. Error Control Coding: Fundamentals and applications. – Prentice-Hall, 1983.

[9] Sudhir Bommena, standard ISO/TS 16949:2002 AN1148 “Cyclic Redundancy Check (CRC)”, Microchip Technology Inc.

[10] Yathiraj H U, Mahasiddayya R Hiremath “Implementation of BCH Code (n, k) Encoder and Decoder for Multiple Error Correction Control”, International Journal of Computer Science and Mobile Applications, Vol.2 Issue. 5, May- 2014, pg. 45-54.

[11] Y. -M. Lin, H. -C. Chang and C. -Y. Lee, "Improved High Code-Rate Soft BCH Decoder Architectures With One Extra Error Compensation," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 11, pp. 2160-2164, Nov. 2013.

[12] Dejan Azinović, Klaus Tittelbach-Helmrich, Zoran Stamenković, “Performance investigation on BCH codec implementations”, in IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), 27 March 2017.

[13] Liao, Xiong & Guo, Junxiong & Luo, Zhenghua & Xu, Yanghui & Chu, Yingjun. (2023). Research and Implementation of High-Efficiency and Low-Complexity LDPC Coding Algorithm. Electronics. 12. 3696. 10.3390/electronics12173696.

[14] V. Pignoly, B. Le Gal, C. Jego, B. Gadat and L. Barthe, "Fair comparison of hardware and software LDPC decoder implementations for SDR space links," 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, UK, 2020.

[15] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," in IEEE Transactions on Information Theory, vol. 47, no. 2, pp. 638-656, Feb 2001.

# A Method of Increasing the Integrity of Stored Information at the RAID 6 Level Using the Reed-Solomon Code and Combination of Syndrome and Probabilistic Decoding

Alina Lanina
Faculty of Computer Science and Technology
Saint Petersburg Electrotechnical University "LETI", 197022
St. Petersburg, Russia
lanina-20019@yandex.ru

Alla Levina
Faculty of Computer Science and Technology
Saint Petersburg Electrotechnical University "LETI", 197022
St. Petersburg, Russia
alla_levina@mail.ru

*Abstract*—**Currently, the volumes of information that need to be stored are growing. To store such data, specialized storage systems are used — they are capable of accommodating ever-increasing amounts of data. During storage and transmission, errors may occur, which can lead to data corruption or loss. The technology of adding redundant disks to a RAID disk array is widely known — it is used to subsequently recover lost information. This article proposes a method that combines the Reed-Solomon code (used at the RAID 6 level) with a decoding technique that is a combination of syndrome-based and probabilistic decoding methods.**

***Keywords: coding theory, syndrome decoding, soft decoding, storage systems, redundant array of independent disks***

## I. INTRODUCTION

In modern conditions, large volumes of data are processed and stored. The information to be stored may be housed in data centers (DCs — data processing centers). According to statistics presented in the article Uptime Institute Data Center Outage Report 2025: Trends, Causes, and Recommendations, 2025 [1], which is based on data from the Annual Outage Analysis 2025 report by the Uptime Institute [2], the issue of outages is highly relevant. Specifically, 60 % of surveyed organizations reported experiencing outages caused by failures at their own facilities or with third-party service providers. Among the most frequently cited causes were network/connectivity issues, power supply problems, IT systems/software issues.

The financial damage caused to organizations by the outage and the wait for restoration of the data processing center's functionality was also assessed. In 2020, among the surveyed organizations: 60 % reported that losses amounted to less than $100 thousand; 28 % — between $100 thousand and $1 million; 12 % — more than $1 million. By 2025, the share of organizations with losses between $100 thousand and $1 million had risen to 53 %.

As follows from the statistical data presented above, preserving and enhancing the integrity of stored information is an urgent task. There are various methods and technologies capable of recovering data after equipment failures. One of them is RAID technology, which allows restoring information on failed disks through the use of error-correcting coding.

This article will present a method for combining Reed-Solomon codes and a technique for enhancing the integrity of information stored in storage systems (SS), proposed in [3] for the RAID 6 level.

## II. RAID TECHNOLOGY

### A. Redundant Arrays of Inexpensive Dicsks

Redundant disk arrays, known as RAID (Redundant Arrays of Inexpensive Disks), were introduced by researchers Peterson, Gibson, and Katz from the University of California, Berkeley, in 1987 [4].

The RAID technology distinguishes various levels. The key differences between them lie in the methods of data formation and placement, as well as in the algorithms for distributing data across disks. The RAID levels will be discussed in greater detail below.

### B. RAID 5 u RAID 6 levels

RAID 5 level uses parity checksums and data striping. In RAID 5, parity checksums are distributed across the entire array, which improves write speed thanks to parallel operations. The system requires at least three disks to operate, and the storage space allocated for parity checksums equals the capacity of one disk (Figure 1).

Manuscript received April 20, 2026; revised May 26, 2026; accepted June 4, 2026. Published June 15, 2026.
Issue category: Regular
Paper category: Regular
DOI: 10.64552/wipiec.v12i1.131

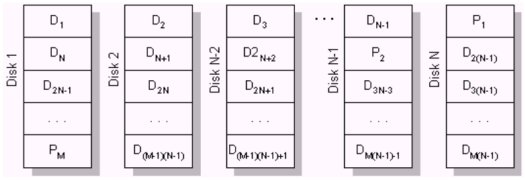



Figure 1 - RAID 5

- In RAID 6, information is divided at the block level in the same way as in RAID 5; however, this architecture incorporates a second, additional mechanism to improve fault tolerance. Reed-Solomon encoding is commonly used for this purpose. The resulting structure can withstand dual disk failures (Figure 2).

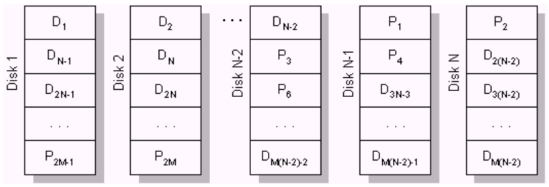



Figure 2 - RAID 6

A pressing issue for every RAID level is errors that occur when reading from or writing to a specific disk sector. Sectors that generate errors upon access are called bad blocks. The causes of their appearance may include external physical impact on the storage device, natural wear and tear of the disk surface, power surges or unstable power supply to the device, malfunctions in the software or hardware components during data transmission.

## III. Error-Correction Codes

### A. Linear Block Codes

Let there be a message consisting of *n* symbols, of the form $m = (m_1, m_2, \dots, m_n)$. The encoded message is a word $c = (c_1, c_2, \dots, c_n)$; such words are called codewords and together they form a code.

One of the encoding methods is linear coding [5]. In linear coding, the first part of the codeword coincides with the symbols of the original message (Formula 1):

$$x_1 = u_1, x_2 = u_2, \dots, x_k = u_k \quad (1)$$

The following are the $n-k$ check (parity) symbols: $x_{k+1}, \dots, x_n$. Codewords must satisfy Equation 2:

$$H \cdot \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = Hx^T = 0 \quad (2)$$

where $H$ is a parity-check matrix of size $\big((n-k) \times n\big)$, of the following form (Formula 3):

$$H = [A|I_{n-k}] \quad (3)$$

By the matrix $H$, we mean a fixed matrix consisting of 0 and 1, and $I_{n-k}$ is the identity matrix of size $\big((n-k) \times (n-k)\big)$ — as given in Formula 4:

$$I_{n-k} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix} \quad (4)$$

### B. Reed-Solomon Codes

These belong to linear block codes [6] and are also cyclic. They are specified as $RS(n,k)$ $s$ −bit symbols, where: $k$ is the number of information symbols of $s$ bits each; $n$ is the number of symbols in a codeword, obtained as the information symbols plus the added parity symbols. A Reed-Solomon decoder can correct up to *t* symbols containing errors in a codeword, where $2t = n - k$.

The decoder can correct any 16 erroneous symbols in a codeword — but no more than that. With a symbol size of $s$ bits, the maximum codeword length is $n = 2^s - 1$. Thus, the maximum code length with 8-bit symbols is 255 bytes.

Reed-Solomon codes operate using Galois field arithmetic (GF), also known as finite field arithmetic.

To form a Reed-Solomon codeword, an external (generator) polynomial is used. Any valid codeword must be divisible by all factors of the generator polynomial. In general form, the generator polynomial is presented in Formula 5:

$$g(x) = (x - a^i)(x - a^{i+1}) \dots (x - a^{i+2t}) \quad (5)$$

The codeword is formed using the operation $c(x) = g(x)i(x)$ where: $g(x)$ is the generator polynomial; $i(x)$ is the information block (message polynomial); $c(x)$ is the codeword (an element of the field).

**Encoder architecture**

To construct a codeword, $2t$ parity symbols are required.

There are two possible encoding variants: systematic and nonsystematic. In nonsystematic encoding, the information vector $i(x)$ is multiplied by the generator polynomial vector $g(x)$ as shown in Formula 6:

$$c(x) = i(x)g(x) \quad (6)$$

In systematic encoding, the $2t$ parity symbols are found as the remainder of dividing the information vector $i(x)$ by the generator polynomial $g(x)$ as shown in Formula 7:

$$R(x) = \frac{i(x)}{g(x)} mod\ g(x) \quad (7)$$

Then, the information vector is shifted by a certain number of positions toward the higher-order bits (multiplication by $x^{n-k}$ ), and the computed remainder is appended to the right as shown in Formula 8:

$$c(x) = i(x) \cdot x^{n-k} + R(x) \quad (8)$$

Each of the 6 registers holds a symbol (8 bits). Arithmetic operators perform addition or multiplication on the symbol as an element of a finite field.

## IV. Method of Increasing Integrity

Data integrity is enhanced by encoding the stored information using a Hamming code. Subsequently, decoding is performed using a combination of syndrome decoding and soft decoding methods [3, 7, 8]. The decoding device decides on the decoding algorithm for each input word: syndrome decoding is applied if the word contains up to *t* errors; soft decoding is used if there are more than *t* errors. This decision-making process provides the following benefits: an improvement in the number of correctable errors compared to using syndrome decoding alone; a gain in decoding time and resource consumption

compared to using soft decoding alone, since probabilistic symbols do not need to be processed for all words.

Figure 3 shows the block diagram of the proposed data integrity increasing method:

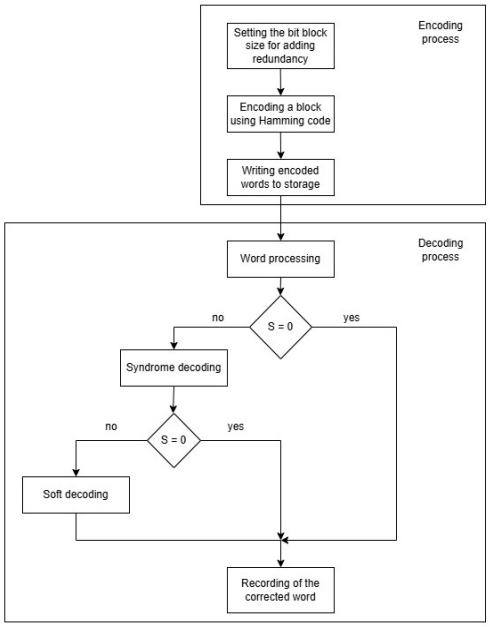



Figure 3 – Block diagram of the data integrity increasing method

The process of applying checksum methods for integrity control of information stored using RAID 5 technology was also simulated. To simulate checksum [9] and proposed method a simulation model was created. Information words were recorded in files simulating disks, the information in which is distributed using RAID technology.

Characteristics of the computer on which the process was modeled: processor: Intel(R) Core(TM) i5-82500; RAM: 8.00 GB; x64 processor; Windows 10 operating system; Python programming language 3.10.4; Visual Studio Code Development Environment; Software Package Management System (pip) written in Python;

The information storage process was modeled using three data stripes and one parity stripe providing parity for all three data stripes. For the CRC32 case, checksum values computed using the CRC32 algorithm were stored separately (one checksum per data block). In the proposed method, words in both the data and parity stripes were additionally encoded using a Hamming (12,8) code with the addition of redundant symbols, resulting in stored codewords instead of raw data. Each codeword was processed individually, with the number of injected errors (ranging from 0 to 4) and their positions randomly determined. When errors were introduced into a single stripe, recovery was performed via the standard RAID array mechanism; however, if errors occurred in more than one stripe, the RAID mechanism alone proved insufficient, necessitating decoding via a combination of syndrome decoding and soft decoding. In contrast, for the CRC32 case, recovery was not possible when errors affected more than one stripe.The simulation results are presented in Table 1.

TABLE 1. SIMULATION RESULTS FOR CHECKSUM METHODS AND THE PROPOSED METHOD

| | crc2 code | Proposed method |
|---|---|---|
| | Errors in 1 strip | |
| Detection of a strip with errors | + | + |
| Number of errors introduced | 2758 | |
| Number of errors corrected | 2758/2758 | 2758/2758 |
| Corrected errors, % | 100% | 100% |
| | Errors in 2 strips | |
| Detection of a strip with errors | + | + |
| Number of errors introduced | 5444 | |
| Number of errors corrected | - | 1682/5444 |
| Corrected errors, % | - | 30,89% |
| | Errors in 3 strips | |
| Detection of a strip with errors | + | + |
| Number of errors introduced | 8094 | |
| Number of errors corrected | - | 2524/8094 |
| Corrected errors, % | - | 31,18% |

The simulation results showed that the proposed method enables full data recovery in case of errors in a single stripe (using redundancy information stored on the additional stripe), with similar results achieved when using checksum methods. However, if errors are introduced into more stripes than the number of redundant stripes, checksum methods fail to recover the data. In contrast, the proposed method can recover: 1682 out of 5444 errors (for 2 erroneous stripes); 2524 out of 8094 errors (for 3 erroneous stripes).

## V. EVENODD SCHEME

The EVENODD algorithm [10] provides an efficient encoding procedure based on exclusive-OR (XOR) operations and independent relationships. It also presents a simple decoding procedure for two erasures as well as for a single error.

Encoding procedure (equation 9):

$$S = \bigoplus_{t=1}^{m-1} a_{m-1-t,t} \tag{9}$$

Then, for each l, where $0 \le l \le m-2$, the redundant symbols are obtained as follows (equations 10, 11):

$$a_{l,m} = \bigoplus_{t=0}^{m-1} a_{l,t} \tag{10}$$

$$a_{l,m+1} = S \oplus \left( \bigoplus_{t=0}^{m-1} a_{<l-t>m,t} \right) \tag{11}$$

Equations (10) and (11) define the encoding. We have two types of redundancy: horizontal redundancy and diagonal redundancy. Disk $m$ is simply a parity disk containing the XOR result of disks $0,1,\dots,m-1$. Disk $(m+1)$ contains the

diagonal redundancy as defined by equation (11). We see that there are two possible diagonal redundancies: the parity can be even or odd. This even or odd parity is determined by the bit $S$ in equation (9), which defines the parity of the diagonal $(m-2,1),(m-3,2),\dots,(0,m-1)$. If this diagonal has an EVEN number of 1, then we have even parity in the remaining diagonals. Otherwise, we have ODD parity. It is for this reason that we call this scheme the EVENODD (EVEN–ODD) scheme.

*Two-erasure decoding algorithm*

Consider an array of symbols $a_{ij}$ with dimensions $(m-1)\times(m+2)$, such that the last two columns are redundant according to equations (9), (10) and (11). If a failure occurs in one column (disk), say in column (disk) $i$, where $i \neq m+1$, then it can be reconstructed using the exclusive-OR (XOR) operation across the remaining columns (disks).

If column $(m+1)$ is faulty, then the symbols can be restored using equations (9) and (11).

Next, suppose that columns (disks) $i$ and $j$ have failed, where $0 \le i < j \le m+1$. We have four cases:

- $i = m$ **and** $j = m+1$, i.e., both redundant disks have failed. We can reconstruct disk $m$ using equation (10), and disk $(m+1)$ using equations (9) and (11). In other words, the reconstruction is equivalent to encoding.
- $i < m$ **and** $j = m$, namely, one data disk and one redundant disk have failed. We can reconstruct disk $i$ as follows (equation 12):

$$S = a_{<i-1>_m,m+1} \oplus \left( \bigoplus_{l=0}^{m-1} a_{<i-l-1>_m,l} \right) \qquad (12)$$

where we suppose that $a_{m-1,1} = 0$ for $0 \le l \le m+1$. Then (equation 13):

$$a_{k,i} = S \oplus a_{<i-1>_m,m+1} \oplus \left( \bigoplus_{l=0, l\neq i}^{m-1} a_{<k+i-l>_m,l} \right), \quad 0 \le k \le m-2 \qquad (13)$$

and $a_{k,m}, 0 \le k \le m-2$ it is obtained using equation (10) after reconstructing disk $i$.

- $i < m$ **and** $j = m+1$, namely, one redundant disk and one data disk have failed. We can reconstruct disk $i$ using equation (10), and disk $(m+1)$ using equations (9) and (11), once disk $i$ has been reconstructed.
- $i < m$ **and** $j < m$. This is the main case. Both failed disks contain data, and we cannot reconstruct them using the individual relations as in the previous three cases. This case is analyzed in detail in [10].

## VI. Combination of Reed-Solomon and method of increasing the integrity and comparison with EVENODD

### *A. Combination of Reed-Solomon and method of increasing the integrity*

We will describe the encoding and recovery processes for a combination of Reed-Solomon code and an integrity enhancement method.

Encoding process:

1. Computation of checksums using the XOR operation.
2. Computation of Reed-Solomon checksums.
3. Determination of the block size (in bits) for adding redundancy.
4. Encoding of data blocks using a Hamming code.
5. Writing the encoded information words to the data drives.
6. Writing the encoded checksums to the parity drives.

Recovery:

If one drive fails:

1. If it is a parity drive, recovery is performed using the data drives — similar to recalculating parity.
2. If it is a data drive, recovery is done using XOR parity.

If two drives fail:

1. If both are parity drives, recovery is performed using the data drives — similar to recalculating both parity values.
2. If one is a parity drive and one is a data drive: recover the data drive using available parity; recover the second parity drive by recalculating it from all data drives.
3. If both are data drives, recovery is performed using the two available parity drives.

If more than two drives fail, recovery of the codewords encoded with a Hamming code is performed using a combination of syndrome decoding and probabilistic (soft) decoding methods.

### *B. EVENODD Scheme*

We will describe the encoding and decoding processes for the EVENODD method.

Encoding:

1. Computation of checksums using the XOR operation.
2. Computation of checksums using diagonal parity (diagonal redundancy).
3. Writing information symbols to the data drives.
4. Writing parity symbols to the parity drives.

Recovery:

If one drive fails:

1. If it is a parity drive, recovery is performed using the data drives — similar to recalculating parity.
2. If it is a data drive, recovery is done using XOR parity.

If two drives fail:

1. If both are parity drives, recovery is performed using the data drives — similar to recalculating both parity values.
2. If one is a parity drive and one is a data drive: recover the data drive using available parity; recover the second parity by recalculating it from all data drives.
3. If both are data drives, recovery is performed using the two available parity drives (for more details, see [10]).

Next, Figure 4 shows a block diagram of the combination of the Reed-Solomon code and the integrity increasing method for the RAID 6 architecture.

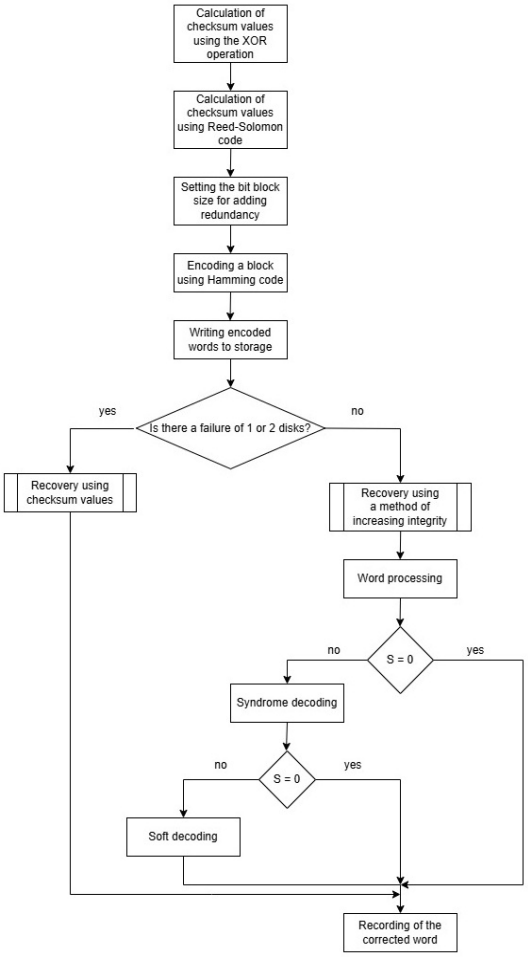



Figure 4 – Block diagram of the Reed-Solomon code and integrity increasing method combination for RAID 6

Next, the table provides a comparison of the capabilities of the proposed combination of the Reed-Solomon code with the integrity enhancement method and the EVENODD method. The analysis focused on scenarios where drives did not fail completely, but errors occurred and needed to be detected. Cases with errors in 1, 2, and 3 stripes were considered. Comparison results are presented in Table 2.

TABLE 2. RESULTS OF THE COMPARISON BETWEEN THE REED-SOLOMON CODE AND INTEGRITY INCREASING METHOD COMBINATION AND THE EVENODD ALGORITHM FOR ERRORS IN MULTIPLE STRIPES

| | Combination of the Reed-Solomon algorithm and method of increasing integrity | EVENODD |
|---|---|---|
| | Errors in 1 strip | |
| Detection of a disk with errors | + | + |
| Error correction | + | + |
| | Errors in 2 strips | |
| Detection of a disk with errors | + | - |
| Error correction | + | - |
| | Errors in 3 strips | |
| Detection of a disk with errors | + | - |
| Error correction | + | - |

As the comparison shows, EVENODD — described as an algorithm that corrects a single error — is outperformed by the proposed combination in cases where errors occur in 2 or more stripes.

## ACKNOWLEDGMENT

This research was funded by the Ministry of Science and Higher Education of the Russian Science Foundation (Project “Goszadanie” No.075-00003-24-02, FSEE-2024-0003.

## REFERENCES

[1] Uptime Institute Data Center Outage Report 2025: Trends, Causes, and Recommendations, 2025. [Online]. Available: https://telecombloger.ru/600923?ysclid=mnxplfv2zv991216637

[2] Uptime Institute, Annual Outage Analysis 2025, 2025. [Online]. Available: https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2025

[3] Lanina, Alina A.; Levina, Alla B. A method of increasing the integrity of information stored in data storage systems by combining syndrome and probabilistic decoding. IT Security (Russia), [S.l.], v. 32, no. 3, p. 44–56, 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1814. DOI: http://dx.doi.org/10.26583/bit.2025.3.04.

[4] David A Peterson, Garth Gibson, Randy H Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID) – 1988

[5] MacWilliams, F. J., & Sloane, N. J. A. (1977). The theory of error-correcting codes (North-Holland Mathematical Library). Amsterdam: North-Holland Publishing Co. ISBN 0-444-85193-3

[6] B. D. Kudryashov, Osnovy teorii kodirovaniya: uchebnoe posobie. Saint Petersburg: BKhV-Peterburg, 2016.

[7] Plotnikov A. I., Levina A. B., Lanina A. A., Zikratov I. A. (2024). Comparison of Methods Syndrome and SoftDecoding for Hamming Code. Vestnik komp'yuternyh i informatsionnyh tekhnologiy, 21(11), 46 – 53. [in Russian lan-guage]. DOI: 10.14489/vkit.2024.11. pp.046-053

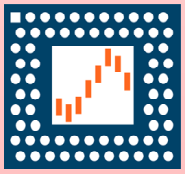
[8] V. V. Kvashennikov, Method for soft decoding of error-correcting code, Russian Federation Patent 2738724, Dec. 16, 2020.

[9] B. Kroth and S. Yang, CheckSumming RAID, 2013. [Online]. Available: https://gradebuddy.com/doc/233998/checksumming-raid/

[10] M. Blaum, J. Brady, J. Bruck, and J. Menon, “EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures,” IEEE Trans. Comput., vol. 44, no. 2, pp. 192–202, Feb. 1995.

# Authors Index

Supported by:





ISSN 2980-7298