Article

En-AR-PRNS: Entropy-Based Reliability for Configurable and Scalable Distributed Storage Systems

Andrei Tchernykh ^{1,2,3,*} , Mikhail Babenko ^{2,4} , Arutyun Avetisyan ² and Alexander Yu. Drozdov ⁵¹ Computer Science Department, Center for Scientific Research and Higher Education at Ensenada, 22860 Ensenada, Mexico² Ivannikov Institute for System Programming, Moscow 109004, Russia; mgbabenko@ncfu.ru (M.B.); arut@ispras.ru (A.A.)³ School of Electrical Engineering and Computer Science, South Ural State University, 454080 Chelyabinsk, Russia⁴ Department of Computational Mathematics and Cybernetics, North-Caucasus Federal University, 355017 Stavropol, Russia⁵ Laboratory of Design and Modeling of Special-Purpose Computer Systems, Moscow Institute of Physics and Technology, 141701 Moscow, Russia; alexander.y.drozdov@gmail.com

* Correspondence: chernykh@cicese.mx

Abstract: Storage-as-a-service offers cost savings, convenience, mobility, scalability, redundant locations with a backup solution, on-demand with just-in-time capacity, syncing and updating, etc. While this type of cloud service has opened many opportunities, there are important considerations. When one uses a cloud provider, their data are no longer on their controllable local storage. Thus, there are the risks of compromised confidentiality and integrity, lack of availability, and technical failures that are difficult to predict in advance. The contribution of this paper can be summarized as follows: (1) We propose a novel mechanism, En-AR-PRNS, for improving reliability in the configurable, scalable, reliable, and secure distribution of data storage that can be incorporated along with storage-as-a-service applications. (2) We introduce a new error correction method based on the entropy (En) paradigm to correct hardware and software malfunctions, integrity violation, malicious intrusions, unexpected and unauthorized data modifications, etc., applying a polynomial residue number system (PRNS). (3) We use the concept of an approximation of the rank (AR) of a polynomial to reduce the computational complexity of the decoding. En-AR-PRNS combines a secret sharing scheme and error correction codes with an improved multiple failure detection/recovery mechanism. (4) We provide a theoretical analysis supporting the dynamic storage configuration to deal with varied user preferences and storage properties to ensure high-quality solutions in a non-stationary environment. (5) We discuss approaches to efficiently exploit parallel processing for security and reliability optimization. (6) We demonstrate that the reliability of En-AR-PRNS is up to 6.2 times higher than that of the classic PRNS.

Keywords: distributed data storage; security; reliability; error correction code; polynomial residue number system; entropy

Citation: Tchernykh, A.; Babenko, M.; Avetisyan, A.; Drozdov, A.Y. En-AR-PRNS: Entropy-Based Reliability for Configurable and Scalable Distributed Storage Systems. *Mathematics* **2022**, *10*, 84. <https://doi.org/10.3390/math10010084>

Academic Editor: Oliviu Matei

Received: 15 November 2021

Accepted: 24 December 2021

Published: 27 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Reliable storage is key for computer systems. Various factors can decrease its reliability and safety: hardware and software malfunctions, integrity violation, unexpected and unauthorized data modifications, data loss, malicious intrusions, falsifications, denial of access, etc.

Multi-cloud storage systems can use both static and dynamic approaches to scaling (Gomathisankaran et al., 2011) [1], (Chervyakov et al., 2019) [2], (Tchernykh et al., 2018) [3], (Tchernykh et al., 2016) [4], (Tchernykh et al., 2020) [5], and (Varghese and Buyya, 2018) [6]. Distributed cloud storage systems are well suited for high-volume data (Tchernykh et al., 2020) [5],

(Nachiappan et al., 2017) [7], (Tan et al., 2018) [8], (Sharma et al., 2016) [9], (Li et al., 2016) [10], and (Baker et al., 2015) [11]. They provide a flexible deployment environment for creating the right-sized storage based on capacity, cost, and performance needs. Cloud storage can be adapted to data volume, velocity, variety, and veracity, which is a challenge for traditional storage systems. Modern data storage systems are distributed geographically to reduce disruptions to data centers caused by seismic, hydrogeological, electrical, technogenic, and other catastrophes.

Tchernykh et al. (2019) [12] presented heterogeneous multi-cloud storage based on secret separation schemes (SSS) with data recovery. The authors evaluated its performance, examining the overall coding/decoding speed and data access speed using actual cloud storage providers. A data distribution method with fully homomorphic encryption (FHE) was introduced by Chen and Huang (2013) [13]; however, it has high computational complexity, redundancy, and low reliability. Chervyakov et al. (2019) [14] presented cloud storage based on the redundant residual number system (RRNS). It overcomes several issues of the above approach; however, SSS based on RRNS has a low data coding and decoding speed (Tchernykh et al., 2019 [12]).

The advantage of RRNS is that it has properties of error correction codes, homomorphic encryption (HE), and SSS. It is a non-positional system with no dependence between its shares (i.e., residues). Thus, an error in one share is not propagated to other shares. Therefore, isolation of the faulty residues allows for fault tolerance and facilitates error detection and correction.

Chervyakov et al. (2019) [2] used the Asmuth–Bloom algorithm making the SSS asymptotically ideal. It is effective but inefficient for distributed storage, since it requires redundancy as large as the Shamir scheme. Tchernykh et al. (2018) [3] estimated the risks of cloud conspiracy, DDoS attacks, and errors of cloud providers to find the appropriate moduli and the number of controls and working moduli.

Here, we adapted this approach for a polynomial residue number system (PRNS) to control the computational security of distributed storage. By selecting PRNS moduli, we could also minimize the computational costs associated with the coding–decoding operations. Since they are modular, we had to consider moduli that were irreducible polynomials over the binary field. Parallel execution could be performed as a series of smaller calculations broken down from an overall single, large calculation with horizontal and vertical scalability.

To reduce the decoding complexity from RRNS to a binary representation, Chervyakov et al. (2019) [2] introduced the concept of an approximation of the rank of a number for RRNS (AR-RRNS). This reduces the number of calculated projections and replaces computationally complex long integer division by taking the least significant bits. An approximation of the rank for PRNS (AR-PRNS) was studied by Tchernykh et al. (2019) [15] to avoid the most resource-consuming operations of finding residue from division by a large polynomial. The authors compared the speeds of data coding and decoding of PRNS and RRNS. They also provided a theoretical analysis of data redundancy and mechanisms to configure parameters to cope with different objective preferences.

The contribution of this paper is multifold:

- We propose a novel entropy-based mechanism for improving the reliability of distributed data storage;
- An En-AR-PRNS scheme is proposed that combines the SSS and error correction codes with multiple failure detection/recovery mechanisms. It can detect and correct more errors than the state-of-the-art threshold-based PRNS;
- A theoretical analysis is presented providing the basis for the dynamic storage configuration to deal with varied user preferences and storage properties to ensure high-quality solutions in a non-stationary environment;
- The concept of an AR of the polynomial to improve decoding speed is applied;

- Approaches to efficiently exploit parallel processing to provide fast high-quality solutions for security, reliability, and performance optimization needed in the non-stationary cloud environment are classified.

This paper is organized as follows: Section 2 reviews distributed clouds storage systems and reliability mechanisms. Section 3 introduces notations, basic concepts, and the main properties of the PRNS with error detection mechanisms. Section 4 introduces parameters to configure reliability, data redundancy, coding, and decoding speeds. Section 5 describes the rank of the PRNS number, its approximation, and provides proof of their main properties. Section 6 focuses on a novel En-AR-PRNS entropy-based algorithm to localize and correct errors. Section 7 demonstrates the reliability improvement by analyzing the probability of information loss and the number of detected and corrected errors. Section 8 discusses approaches to efficiently exploit parallel processing for security, reliability, and performance optimization needed in a non-stationary environment. Section 9 presents the conclusion and future work.

2. Related Work

2.1. Security and Privacy

The principal objective of information security is to protect the confidentiality, integrity, and availability of a computer system, named the CIA triad, to prevent data from unauthorized access, use, disclosure, disruption, modification, etc., whether in storage, processing, or transit. Another related direction of extensive study is fault-tolerance, which is the ability to continue operating without interruption when one or more of its components fail, whether due to the fact of failures that are deliberate or not as well as accidents and threats (Srisakthi and Shanthi, 2015) [16].

Confidentiality usually refers to keeping important information secret, which cannot be obtained even by the cloud provider. Data integrity means keeping data unchanged during storage and transmission, ensuring its accuracy and consistency throughout the life cycle. Availability stands for the service being available to the user at any time regardless of hardware, software, or user faults. Privacy implies that there is no unauthorized access to user data. It is centered around proper data handling during collecting, storing, processing, managing, and sharing.

In geographically distributed data centers, cloud providers store several copies of the same data to minimize the impact of natural disasters, fires, floods, technological accidents, etc. However, this solution increases the high costs of storage, backups, and disaster recovery plans. Deliberate threats refer to malicious attempts by an insider or unauthorized person to access information, software cracking, interception, falsification, forgery, etc.

The Cloud Security Alliance announced increased unauthorized access to the information in the clouds (Hubbard and Sutton, 2010) [17]. To reduce this risk, SSS, together with error correction codes, are used. Threads can be reduced by protection systems based on the concept of proactive security (Tchernykh et al., 2018) [18] that incorporates weighted SSS. Data are divided into shares of different sizes, dependent on the reliability of the storage. The secret can be reconstructed even when several shares are modified or unavailable.

2.2. Reliability

Over the past decades, many approaches to ensure reliability have been proposed. The most known are data replication, secret separation schemes (SSS), redundant residual number systems (RRNS), erasure codes (ECs), regenerating codes (RCs), etc. [1–3,14,19–23].

Data replication provides for the storage of the same data in multiple locations to improve data availability, accessibility, system resilience, and reliability. This approach becomes expensive with the growing amount of stored data (Ghemawat et al., 2003) [19]. One typical use of data replication is for disaster recovery to ensure that an accurate backup exists in the case of a catastrophe, hardware failure, or a system breach where data are compromised.

Secret Sharing Schemes (SSS) refer to methods for dividing data and distributing shares of the secret between participants in such a way that it can be reconstructed only when a sufficient number of shares are combined (Shamir, Blackly, etc.). They are suitable for securely distributed storage systems (Gomathisankaran et al., 2011) [19]. Asmuth and Bloom (1983) [24] and Mignotte (1982) [25] proposed asymptotically perfect SSS based on RNS.

In RNS, a number is represented by residues—remainders under Euclidean division of the number by several pairwise coprime integers called the moduli. Due to the original number being divided into smaller numbers, operations are carried out independently and concurrently, providing fast computations.

An RRNS is obtained from RNS by adding redundant residues, which bring multiple-error detection and correction capability (Celesti et al., 2016) [14]. With r redundant moduli, it can detect r and correct $r/2$ errors using projection methods. However, the number of necessary projections grows exponentially with increasing r . Significant optimization is required for practical efficiency.

Erasure code (EC) converts p length data and produces s length data ($s > p$) so that the original message can be recovered from a subset of s characters. An efficient implementation of EC is $O(p \cdot \log_2 p)$ complexity (Lin, et al. 2014) [26].

Regenerating codes are designed for providing reliability and efficient repair (reconstruction) of lost coded fragments in distributed storage systems. They overcome the limitations of the traditional codes mentioned above, significantly reducing the traffic (repair-bandwidth) required for repairing. Furthermore, it has reasonable tradeoffs between storage and repair bandwidth (Dimakis et al., 2010) [22]. A special pseudorandom number generator is required to effectively implement RC (Liu et al., 2015) [27].

3. Polynomial Residue Number System

PRNS is used in areas such as digital signal processing (Skavantzios and Taylor, 1991) [28], symmetric block encryption (Chu and Benaissa, 2013) [29], cryptography (Parker and Benaissa, 1995) [30], homomorphic encryption (Chervyakov et al., 2015) [31], wireless sensor networks (Chang et al., 2015) [32], etc. It can support parallel, carry-free, high-speed arithmetic. PRNS over $GF(2^m)$ was introduced by (Halbutogullari and Koc, 2000) [33]. Contrary to RRNS, where each modulo is a coprime number, PRNS moduli are irreducible polynomials. The Chinese remainder theorem (CRT) is also applied.

3.1. Basic Definitions

The original data, A , can be represented by the polynomial $A_P(x)$ over $GF(2^m)$ of the form $A_P(x) = \sum_{i=0}^{m-1} f_i \cdot x^i$, for $f_i \in \{0, 1\}$, where the highest exponent of the variable x is called the degree of the polynomial. For example, $A = 243 = 11110011_2$. Hence, $A_P(x) = x^7 + x^6 + x^5 + x^4 + x + 1$ with a degree of seven.

The PRNS modulo is an n -tuple of irreducible polynomials over the binary field. We denote this n -tuple: $m_1(x), m_2(x), \dots, m_n(x)$, where n is the number of moduli and $k = n - r$ is the number of working moduli. d_i is the degree of the polynomial $m_i(x)$.

Let $\hat{M}(x) = \prod_{i=1}^n m_i(x)$ be the polynomial of degree $\hat{D} = \deg \hat{M}(x) = \sum_{i=1}^n d_i$ that determines the full range $[0, \hat{M}(x))$ of PRNS and $M(x) = \prod_{i=1}^k m_i(x)$ be the polynomial with degree $D = \deg M(x) = \sum_{i=1}^k d_i$ that defines the dynamic range $[0, M(x))$ of PRNS.

For a unique residue representation of an arbitrary element from $GF(2^m)$, D should be greater or equal to m and $\deg A_P(x) < D$. In PRNS, $A_P(x)$ is represented by an n -tuple of residues A_{PRNS} , as follows:

$$A_P(x) \xrightarrow{PRNS} A_{PRNS} = (a_1(x), a_2(x), \dots, a_n(x))$$

where $a_i(x) = |A_P(x)|_{m_i(x)}$ is the residue of the division of the polynomial $A_P(x)$ by $m_i(x)$, for $i = 1, 2, \dots, n$.

To convert the residual representation to a polynomial one, the following CRT extension is used:

$$A_P(x) = \left| \sum_{i=1}^k a_i(x) \cdot O_i(x) \cdot M_i(x) \right|_{M(x)}, \quad (1)$$

where $O_i(x) = |M_i^{-1}(x)|_{m_i(x)}$ and $M_i(x) = M(x)/m_i(x)$. The quantities $O_i(x)$ are known as the multiplicative inverses of $M_i(x)$ by modulo $m_i(x)$ such that $(O_i(x) \cdot M_i(x)) \bmod m_i(x) = 1$.

Throughout the paper, we introduce notations related to main topics such as PRNS, SSS, cloud storage, coding–decoding, error detection and correction, entropy, and reliability. To facilitate understanding, we summarize the main notations in Table 1.

Table 1. Main notations.

A	Original Data
$A_P(x)$	polynomial representation of A
n	number of moduli
r	number of control moduli
k	number of working moduli
$m_i(x)$	i -th polynomial modulo
d_i	degree of $m_i(x)$
$M(x) = \prod_{i=1}^k m_i(x)$	dynamic range $[0, M(x))$
$\hat{M}(x) = \prod_{i=1}^n m_i(x)$	full range $[0, \hat{M}(x))$
$\hat{M}_i(x) = \hat{M}(x)/m_i(x)$	orthogonal basis
$\hat{O}_i(x) = \hat{M}_i^{-1}(x) _{m_i(x)}$	orthogonal basis weight
$D = \deg M(x) = \sum_{i=1}^k d_i$	degree of $M(x)$
$\hat{D} = \deg \hat{M}(x) = \sum_{i=1}^n d_i$	degree of $\hat{M}(x)$
A_{PRNS}	tuple of residues of $A_P(x)$
E_{PRNS}	tuple of errors
$E_P(x)$	polynomial representation of an error
$\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS}$	tuple of residues of $A_P(x)$ with errors
$a_i(x) = A_P(x) _{m_i(x)}$	residue of the division of polynomial $A_P(x)$ by $m_i(x)$, for $i = 1, 2, \dots, n$
$r_A(x)$	rank
$R_A(x)$	approximation of the rank
$H(x)$	entropy
$V_P^l(x)$	l -th candidate of $A_P(x)$ recovered from residues
V_{PRNS}^l	tuple of residues to recover $V_P^l(x)$
$H_W^l = H(V_{PRNS}^l)$	entropy of l -th candidate
H_T^l	hamming distance between V_{PRNS}^l and vector $\bar{A}_{PRNS}$
T_D	total denial of access time
V_D	decoding speed
$V_{D_AR_PRNS}$	AR-PRNS decoding speed
V_C	coding speed
L	data volume
L_T^-	data volume that can be leaked without security violation
Pr_i	probability of error of i -th residue
P_r	probability of information loss
I	tuple of residues without errors
$\bar{I}$	tuple of residues with errors

3.2. Error Detection

PRNS error detection methods are similar to RRNS methods. Adding redundant moduli, we divided the whole representation range into the legitimate (dynamic) range and illegitimate range (Chu and Benaissa, 2013) [29]. An error occurs when the conversion result falls into an illegitimate range.

First, PRNS with the irreducible polynomial k -tuple $m_1(x), m_2(x), \dots, m_k(x)$, which satisfies $\sum_{i=1}^k d_i \geq m$, is defined. Then, one additional polynomial modulo, $m_n(x)$ with the degree $d_n \geq d_i$, for $i = 1, 2, \dots, k$, where $n = k + 1$, is added. With the added modulo, m_n , the whole representation range becomes $\hat{M}(x) = \prod_{i=1}^n m_i(x)$ with a degree $\hat{D} = \deg \hat{M}(x) = \sum_{i=1}^n d_i$. An error is detected if polynomials with a degree greater or equal to D and smaller than $\hat{D}$ have the illegitimate range.

The proof is as follows:

Let $A_P(x)$ be represented in redundant PRNS as $A_{PRNS} = (a_1(x), a_2(x), \dots, a_n(x))$ with the legitimate range of the degree being no higher than D .

Let us assume that an error occurs in the i -th residue:

$$\bar{A}_{PRNS} = (a_1(x), \dots, \bar{a}_i(x), \dots, a_n(x)).$$

The $\bar{A}_{PRNS}$ can be represented as a sum of $A_P(x)$ and the error $E_P(x)$ as follows:

$$\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS}(x),$$

$$\bar{A}_{PRNS} = (a_1(x), \dots, a_i(x), \dots, a_n(x)) + (0, \dots, e_i(x), \dots, 0).$$

Since $A_P(x)$ is an element over $GF(2^m)$, its degree is smaller or equal to $D - 1$; thus, $A_P(x)$ has a legitimate range. Now, let us convert $E_{PRNS}(x)$ from PRNS back to the polynomial representation using (1) with a full range:

$$E_P(x) = |e_i(x) \cdot \hat{O}_i(x) \cdot \hat{M}_i(x)|_{\hat{M}(x)} = |e_i(x) \cdot \hat{O}_i(x)|_{m_i(x)} \cdot \hat{M}_i(x),$$

where $\hat{M}_i(x) = \hat{M}(x)/m_i(x)$ and $\hat{O}_i(x) = |\hat{M}_i^{-1}(x)|_{m_i(x)}$.

The degree of $\hat{M}_i(x)$ is $\hat{D} - d_i$ and the degree of $|e_i(x) \cdot \hat{O}_i(x)|_{m_i(x)}$ can be any number from 0 to $d_i - 1$. Hence, the degree of $E_P(x)$, which we denote as $D_E = \deg E_P(x)$, ranges from $(\hat{D} - d_i)$ to $(\hat{D} - 1)$.

Since $\hat{D} - d_i \geq D$, the following condition is true: if $\deg \bar{A}_P(x) < D$ then $\bar{A}_P(x)$ is correct, otherwise, $\bar{A}_P(x)$ is not correct.

4. Polynomial Residue Number System

4.1. Coding Speed

For performance analysis, we used an VM Intel Xeon® E5-2673 v3, 2 GB of RAM, 16 GB SSD hard drive provided by Microsoft Azure [15]. It had 2^{30} bit operations per second, on average, according to the Geekbench Browser site [34].

The PRNS coding complexity was $O(k \cdot d)$ (Chervyakov et al., 2016) [35]. Hence, the calculation of the remainder of the division required roughly $k \cdot d$ bit operations.

To obtain a polynomial of degree $D - 1$, we performed finding the remainder of the division by PRNS moduli n times. Hence, the total number of the bit operations was $n \cdot k \cdot d$.

The size of the original data was represented by $D = k \cdot d$ bits. The number of D in 1 Mb was $2^{23}/D = 2^{23}/(k \cdot d)$. Therefore, to code 1 Mb of data, $2^{23} \cdot n \cdot k \cdot d / (k \cdot d) = 2^{23} \cdot n$ bit operations were required.

The coding speed, V_C , in MB/s can be calculated by the formula:

$$V_C = \frac{2^{30} \cdot k}{2^{23} \cdot n} = \frac{k \cdot 2^7}{n}.$$

Figure 1 shows that the coding speed, V_C , versus (k, n) parameters was a saw type. Picks were achieved for the schemes (n, n) , and minimums were achieved for $(2, n)$. The user can select the required parameters to obtain the required speed.

Figure 1. V_C of a sPRNS versus (k, n) .

Figure 2 shows the coding speed, V_C , (Mb/s) of PRNS and RRNS. We can see that the coding speed of PRNS was higher than RRNS for all parameters.

Figure 2. The V_C of a PRNS and an RRNS versus (k, n) .

The PRNS operations on $GF(2^m)$ did not require the transfer of values from the lowest to the highest term. Consequently, the time complexity of the division remainder decreased compared to the arithmetic operations of RRNS performed over $GF(2^m)$.

4.2. PRNS Decoding Speed with Data Errors

To correct the $\lfloor r/2 \rfloor$ errors with r extra moduli, we used the projection method. To compute a projection, CRT with algorithmic complexity $O(D^2)$ required roughly $D^2 = k^2 \cdot d^2$ bit operations. Since the number of projections was $C_n^{k+\lfloor \frac{r}{2} \rfloor}$, to detect and localize an error, we needed $C_n^{k+\lfloor \frac{r}{2} \rfloor} \cdot k^2 \cdot d^2$ bit operations.

Hence

$$2^{23} \cdot \frac{C_n^{k+\lfloor \frac{r}{2} \rfloor} \cdot k^2 \cdot d^2}{k \cdot d} = 2^{23} \cdot C_n^{k+\lfloor \frac{r}{2} \rfloor} \cdot d \cdot k$$

bit operations were required to code 1 Mb with the worst-case decoding speed, V_D .

$$V_D = \frac{2^{30}}{2^{23} \cdot C_n^{k+\lfloor \frac{r}{2} \rfloor} \cdot d \cdot k} = \frac{2^7}{C_n^{k+\lfloor \frac{r}{2} \rfloor} \cdot d \cdot k}.$$

Figure 3 shows the data decoding speed, V_D , varying PRNS parameters for $d = \{8, 16, 32\}$.

Figure 3. V_D of a PRNS (Mb/s) versus (k, n) for $d = \{8, 16, 32\}$.

We can see that the computational complexity of PRNS, when an error was localized, was higher than that of RRNS. Hence, the PRNS decoding speed was less than for RRNS.

Figure 4 shows the PRNS and RRNS decoding speeds, V_D (Mb/s), of the worst-case scenario with the maximum number of errors.

Figure 4. V_D of a PRNS and an RRNS with the maximum number of errors versus (k, n) for $d = 8$.

The approximation of the rank of the PRNS number was used to increase this speed (Section 5).

4.3. Data Redundancy

To prevent system operation disruption in the case of technical failures, disasters, and cyber-attacks by maintaining a continuity of service data redundancy is very important.

The input polynomial is $A_P(x)$, where $\deg A_P(x) < D$. Hence, the total input volume approximately equals D . The number of bits needed for residues is equal to $\hat{D}$ in the worst case. The data redundancy degradation is the ratio of the coded data size to the original minus 1:

$$R = \frac{\hat{D}}{D} - 1.$$

Let us select $m_i(x)$ so that $a_i(x)$ uses at most $d_1 = d_2 = \dots = d_n = d$ bits or one word. Hence, the redundancy is roughly $n/k - 1$.

$$R = \frac{\hat{D}}{D} - 1 = \frac{n \cdot d}{k \cdot d} - 1 = \frac{n - k}{k}.$$

Figure 5 shows the redundancy versus the PRNS parameters. We can see that the minimum values are for (n, n) , where $n = 4, 5, \dots, 9$. These values are less than those of the Bigtable system ($\lceil (n+1)/3 \rceil, n$).

Figure 5. Data redundancy versus PRNS settings (k, n) .

PRNS reduces data redundancy compared to numerical RNS, because the degree of the polynomial (residue) is strictly less than the degree of the divisor. We demonstrate this in the following example.

Example 1. Let us consider a $(4, 4)$ scheme with PRNS. Let the moduli be $m_1(x) = x^8 + x^4 + x^3 + x + 1$, $m_2(x) = x^8 + x^4 + x^3 + x^2 + 1$, $m_3(x) = x^8 + x^5 + x^3 + x + 1$, and $m_4(x) = x^8 + x^5 + x^3 + x^2 + 1$. In this case, the dynamic range is:

$$M(x) = \prod_{i=1}^4 m_i(x) = x^{32} + x^{26} + x^{24} + x^{23} + x^{21} + x^{19} + x^{18} + x^{16} + x^{11} + x^{10} + x^9 + x^5 + x^4 + x^2 + 1$$

Let $A_P(x) = x^{31} + x^{16} + x^{15} + x^{14} + 1$, and it has 32 bits. $A_P(x)$ has the following representation:

$$A_P(x) \xrightarrow{\text{PRNS}} A_{\text{PRNS}} = (a_1(x), a_2(x), a_3(x), a_4(x)),$$

where $a_1(x) = |A(x)|_{m_1(x)} = x^7 + x^4 + x^3$, $a_2(x) = |A(x)|_{m_2(x)} = x^7 + x^5 + x^4 + x^3$, $a_3(x) = |A(x)|_{m_3(x)} = x^7 + x^4 + x^3 + x + 1$, $a_4(x) = |A(x)|_{m_4(x)} = x^7 + x^6 + x^3 + x$, $a_1(x)$, $a_2(x)$, $a_3(x)$, and $a_4(x)$ are seventh-degree polynomials of 8 bits.

Therefore, the redundancy degradation is $4 \times \frac{8}{32} - 1 = 0$.

5. Approximation of the Rank of the PRNS Number

5.1. Rank

Reducing the computational complexity of the decoding algorithm is of the utmost interest. One possible approach is to approximate the value of the AR [15].

In CRT, the rank $r_A(x)$ of A_{PRNS} is determined by Equation (2). It is used to restore $A_P(x)$ from residues:

$$A_P(x) = \sum_{i=1}^k M_i(x) \cdot O_i(x) \cdot a_i(x) - r_A(x) \cdot M(x) \quad (2)$$

Hence, $r_A(x) = \left\lfloor \sum_{i=1}^k \frac{O_i(x)}{m_i(x)} a_i(x) \right\rfloor$, $M(x) = \prod_{i=1}^k m_i(x)$, $M_i(x) = \frac{M(x)}{m_i(x)}$ and $O_i(x) = \left\lfloor M_i^{-1}(x) \right\rfloor_{m_i(x)}$ for all $i = 1, 2, \dots, k$.

$r_A(x)$ is the quotient of dividing $\sum_{i=1}^k M_i(x) \cdot O_i(x) \cdot a_i(x)$ by $M(x)$ in PRNS representation. Its calculation includes an expensive operation of the Euclidean division. Instead of computing the rank $r_A(x)$, we calculate an approximation of the rank, $R_A(x)$, based on the approximate method and modular adder, which decreases the complexity:

$$R_A(x) = \left\lfloor \sum_{i=1}^k b_i(x) a_i(x) / x^N \right\rfloor, \quad (3)$$

where $b_i(x) = \lceil O_i(x) \cdot x^N / m_i(x) \rceil$.

This method reduces the number of projections and replaces the computationally complex operation of the division of polynomials with the remainder by taking the division of a polynomial by the monomial x^N . The complexity is reduced from $O(D^2)$ to $O(D \cdot \log D)$.

Theorem 1 shows that $R_A(x)$ and $r_A(x)$ are equal. It provides the theoretical basis for our approach.

Theorem 1. If $N = \max_{i \in \overline{1, n}} \deg m_i(x)$, then $r_A(x) = R_A(x)$.

The proof is described in Appendix A.

The algorithmic complexity of the rank of $r_A(x)$ based on the Theorem 1 is $O(d \cdot \log k)$. Since the coefficients $b_i(x)$ are of degree d , we can compute A efficiently and reduce the computational complexity of the decoding from $O(D^2)$ down to $O(D \cdot \log D)$. Computation of $\sum_{i=1}^k M_i(x) \cdot O_i(x) \cdot a_i(x)$ can be performed in parallel with the computation of $R_A(x)$.

5.2. AR-PRNS Decoding Speed with Data Errors

In the previous section, we described finding the error using AR to increase the decoding speed.

To detect and localize $\lfloor r/2 \rfloor$ errors using the syndrome method, it is necessary to pre-compute a table consisting of $\lfloor r/2 \rfloor \cdot d$ possible syndromes (rows). The maximum number of bit errors is $\lfloor r/2 \rfloor \cdot d$. Each syndrome indicates the localization of errors. To find the syndrome in the table by binary search, we need $\log_2(\lfloor r/2 \rfloor \cdot d) \cdot r \cdot d$ bit operations. To code 1 Mb, we need

$$2^{23} \left(\log_2(\lfloor r/2 \rfloor \cdot d) \cdot r \cdot d + \lfloor r/2 \rfloor^2 \cdot d^2 \right) / (k \cdot d) = 2^{23} \cdot \left(\log_2(\lfloor r/2 \rfloor \cdot d) \cdot r + \lfloor r/2 \rfloor^2 \cdot d \right) / k$$

bit operations. Therefore, the AR-PRNS decoding speed in the worst case is

$$V_{D_AR_PRNS} = \frac{2^{30} \cdot k}{2^{23} \cdot \left(\log_2(\lfloor \frac{r}{2} \rfloor \cdot d) \cdot r + \lfloor r/2 \rfloor^2 \cdot d \right)} = \frac{2^7 \cdot k}{\log_2(\lfloor \frac{r}{2} \rfloor \cdot d) \cdot r + \lfloor r/2 \rfloor^2 \cdot d}.$$

To detect and correct at least one error, k has to satisfy the inequality $k \leq n - 2$.

Figure 6 shows the decoding speeds of PRNS, AR-RRNS, and AR-PRNS depending on the parameters that satisfy this condition. We can see that AR-PRNS outperformed the others by almost three times.

Figure 6. The decoding speed (Mb/s) of a PRNS, AR-PRNS, and AR-RRNS versus the settings (k, n) in the worst-case scenario with the maximum number of errors for $d = 8$.

6. Entropy Polynomial Error Correction Code

We propose a novel En-AR-PRNS method for data decoding that uses AR for decoding speed improvement and entropy for reliability improvement. We show that the entropy concept in PRNS can correct more errors than a traditional threshold-based PRNS.

First, $\sum_{i=1}^n \hat{M}_i(x) \cdot \hat{O}_i(x) \cdot a_i(x)$ can be represented in the form:

$$\sum_{i=1}^n \hat{M}_i(x) \cdot \hat{O}_i(x) \cdot a_i(x) = A_P(x) + r_A(x) \cdot \hat{M}(x), \quad (4)$$

Assume that an error has the following form:

$$E_P(x) \xrightarrow{PRNS} E_{PRNS} = (e_1(x), e_2(x), \dots, e_n(x)).$$

Then, instead of A_{PRNS} , we have $\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS}$.

Using (4), we have:

$$\sum_{i=1}^n \hat{M}_i(x) \cdot \hat{O}_i(x) \cdot (a_i(x) + e_i(x)) = A_P(x) + E_P(x) + r_A(x) \cdot \hat{M}(x) + r_E(x) \cdot \hat{M}(x)$$

Without loss of generality, we assume that moduli are in ascending order, i.e., $\deg m_1 \leq \deg m_2 \leq \dots \leq \deg m_n$.

Since k moduli are included in the dynamic range and r is redundant (control moduli), where $k + r = n$, $\deg A_P(x) < D = \deg \prod_{i=1}^k m_i(x) = \sum_{i=1}^k d_i$. Since an error is of the form $E(x) = \beta(x) \hat{M}_I(x)$, where $\hat{M}_I(x) = \prod_{i \in I} m_i(x)$, $\beta(x)$ is a nonzero polynomial, and I is the tuple of residues without errors.

If I is not an empty tuple and $\deg \hat{M}_I(x) \geq \sum_{i=1}^k d_i$, then an error can be detected, since $\deg E_P(x) \geq \sum_{i=1}^k d_i$.

6.1. Entropy in PRNS

According to CRT, $A_P(x)$ is a polynomial over the binary field of degree less than $D = \sum_{i=1}^k d_i$. Therefore, $A_P(x)$ can have 2^D different values. Following Kolmogorov (1965) [36] and Ivanov et al. (2019) [37], we can state that the entropy of $A_P(x)$ is equal to:

$$H(A_P(x)) = \log_2 2^D = D = \sum_{i=1}^k d_i, \quad (5)$$

If $i \in [1, n]$ and $a_i(x) = |A_P(x)|_{m_i(x)}$, then the entropy of $a_i(x)$ is equal to d_i :

$$H(a_i(x)) = d_i \quad (6)$$

Hence, the residue $a_i(x)$ carries some information of A . If the entropy $d_i = 0$, the residue does not carry information about $A_P(x)$. In another extreme, if $d_i = D$, the residue equals to $A_P(x)$. From Equation (5) and Equation (6), it follows that:

$$\sum_{i \in I} d_i \geq \sum_{i=1}^k d_i. \quad (7)$$

If Equation (7) is satisfied, the amount of known information is greater than or equal to the initial information. Hence, we can restore $A_P(x)$, where I is a tuple of residues without error.

From the information theory point of view, the entropy of the residue $a_i(x)$ can be viewed as a measure of how close it is to the minimal entropy case. Hence, it measures the amount of information that $a_i(x)$ carries from $A_P(x)$. The maximal entropy corresponds to a non-coding–non-secure case.

Using an entropy-based approach, we can verify the obtained result for its correctness using the following theorem.

Theorem 2. Let $m_1(x), m_2(x), \dots, m_n(x)$ be the PRNS moduli n -tuple, $\bar{A}_P(x) \xrightarrow{PRNS} \bar{A}_{PRNS} = (\bar{a}_1(x), \bar{a}_2(x), \dots, \bar{a}_n(x))$, where r control moduli, $k = n - r$, and $\bar{I}$ is a tuple of residues with an error. If

$$\sum_{i \in \bar{I}} d_i \leq \sum_{i=1}^r d_{k+i}, \quad (8)$$

an error can be detected.

Proof. Let us consider two cases: when there is an error and when there is no error.

Case 1. If $\bar{I}$ is an empty tuple, then there are no errors and $A_P(x)$ can be calculated using (1).

Case 2. If $\bar{I}$ is not an empty tuple and it satisfies the condition (8), then we show that $\deg \bar{A}_P(x) > \deg A_P(x)$, where $\bar{A}_P(x) = A_P(x) + E_P(x)$. Due to the fact that $E_P(x) = \beta(x) \prod_{i \in I} m_i(x)$, where $\beta(x) \neq 0$, we have:

$$\deg E_P(x) = \deg \beta(x) + \sum_{i \in I} d_i, \quad (9)$$

Given that $\sum_{i \in I} d_i = \sum_{i=1}^n d_i - \sum_{i \in \bar{I}} d_i$, then Equation (9) takes the following form:

$$\deg E_P(x) = \deg \beta(x) + \sum_{i=1}^n d_i - \sum_{i \in \bar{I}} d_i, \quad (10)$$

Substituting the condition of the theorem $\sum_{i \in \bar{I}} d_i \leq \sum_{i=1}^r d_{k+i}$ into Equation (10), we obtain that $\deg E_P(x)$ satisfies the following inequality:

$$\deg E_P(x) \geq \deg \beta(x) + \sum_{i=1}^n d_i - \sum_{i=1}^r d_{k+i}, \quad (11)$$

Because $\sum_{i=1}^n d_i - \sum_{i=1}^r d_{k+i} = \sum_{i=1}^k d_i$, then Equation (11) takes the following form:

$$\deg E_P(x) \geq \deg \beta(x) + \sum_{i=1}^k d_i. \quad (12)$$

Given that $\deg \beta(x) \geq 0$, then from the inequality (12), it follows $\deg E_P(x) \geq \sum_{i=1}^k d_i$. Because $\deg A_P(x) < \sum_{i=1}^k d_i$ and $\deg \bar{A}_P(x) = \deg E_P(x) \geq \sum_{i=1}^k d_i$, an error is detected. The theorem is proved. $\square$

Example 2. (see Appendix B) considers the case when the degrees of control moduli are less than the degrees of working moduli so that we can find an error in the remainder of the division by the working moduli.

Example 3 considers the case when one control modulo is used. We can detect two errors unlike one error in the traditional PRNS.

Example 3. Let PRNS moduli tuple be $m_1(x) = x^2 + x + 1$, $m_2(x) = x^3 + x + 1$, $m_3(x) = x^3 + x^2 + 1$, and $m_4(x) = x^6 + x + 1$, where $k = 3$, $r = 1$, and $n = 3 + 1 = 4$. The dynamic range of PRNS is $M(x) = x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + 1$ and $\hat{M}(x) = x^{14} + x^{12} + x^{11} + x^{10} + x^7 + x^6 + x^2 + x + 1$.

Let $A_P(x) = x^7 \xrightarrow{PRNS} (x, 1, 1, x^2 + x)$. We consider three cases in which errors occurred on two shares.

Case A. If errors occurred in $a_1(x)$ and $a_2(x)$, then the error vector had the following form: $E_{PRNS} = (1, 1, 0, 0)$.

$$\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS} = (x + 1, 0, 1, x^2 + x).$$

Because $\sum_{j \in \bar{I}_A} d_j = d_1 + d_2 = \deg m_1(x) + \deg m_2(x) = 5$ and $\sum_{i=1}^r d_{k+i} = d_4 = \deg m_4(x) = 6$, the condition of Theorem 2 (see Equation (8)) is satisfied, and errors can be detected.

Case B. If errors occurred in $a_1(x)$ and $a_3(x)$, then the error vector is $E_{PRNS} = (1, 0, 1, 0)$; hence, $\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS} = (x + 1, 1, 0, x^2 + x)$. Because $\sum_{j \in \bar{I}_B} d_j = d_1 + d_3 = \deg m_1(x) + \deg m_3(x) = 5$ and $\sum_{i=1}^r d_{k+i} = d_4 = \deg m_4(x) = 6$, the condition of Theorem 2 (see Equation (8)) is satisfied, and errors can be detected.

Case C. If an error occurred in $a_2(x)$ and $a_3(x)$ then the error vector is $E_{PRNS} = (0, 1, 1, 0)$ and $\tilde{A}_{PRNS} = A_{PRNS} + E_{PRNS} = (x, 0, 0, x^2 + x)$. Because $\sum_{j \in \bar{I}_C} d_j = d_2 + d_3 = \deg m_2(x) + \deg m_3(x) = 6$ and $\sum_{i=1}^r d_{k+i} = d_4 = \deg m_4(x) = 6$, the condition of Theorem 2 (see Equation (8)) is satisfied, and an error can be detected.

Let us show how an error is detected.

First, we calculated the PRNS constants $\hat{M}_i(x) = \hat{M}(x)/m_i(x)$.

$$\hat{M}_1(x) = x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1, \hat{M}_2(x) = x^{11} + x^7 + x^5 + x^2 + 1,$$

$$\hat{M}_3(x) = x^{11} + x^{10} + x^4 + x + 1, \hat{M}_4(x) = x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + 1;$$

$$\left| \hat{M}_1^{-1}(x) \right|_{m_1(x)} = x + 1, \left| \hat{M}_2^{-1}(x) \right|_{m_2(x)} = x, \left| \hat{M}_3^{-1}(x) \right|_{m_3(x)} = x,$$

$$\left| \hat{M}_4^{-1}(x) \right|_{m_4(x)} = x^5 + x^2 + x.$$

Third, we compute $\bar{A}_P(x)$, $\bar{\bar{A}}_P(x)$, and $\tilde{A}_P(x)$, using Equation (1).

$$\bar{A}_P(x) = x^{13} + x^{12} + x^3 + 1, \bar{\bar{A}}_P(x) = x^{13} + x^{12} + x^{11} + x^8 + x^6 + x^5 + x^2 + 1,$$

$$\tilde{A}_P(x) = x^{11} + x^8 + x^7 + x^6 + x^5 + x^3 + x^2.$$

Finally, we determine whether $\bar{A}_P(x)$, $\bar{\bar{A}}_P(x)$, and $\tilde{A}_P(x)$ have errors.

Case A. Because $\deg \bar{A}_P(x) = 13 > \deg M(x) = 8$, then $\bar{A}_P(x)$ contains an error.

Case B. Because $\deg \bar{\bar{A}}_P(x) = 13 > \deg M(x) = 8$, then $\bar{\bar{A}}_P(x)$ contains an error.

Case C. Because $\deg \tilde{A}_P(x) = 11 > \deg M(x) = 8$, then $\tilde{A}_P(x)$ contains an error.

To detect an error, we use Theorem 2. To localize and correct the error, we modify the maximum likelihood decoding (MLD) method from Goh and Siddiqi (2008) [38].

6.2. MLD Modification

To correct errors, we need tuple I residues, where $\sum_{i \in I} d_i > D$. One of the ways to select them from all correct residues is the MLD method.

In the process of localization and correction of errors, we calculate the tuple V of possible candidates, $\bar{A}_P(x)$, satisfying the condition $\deg \bar{A}_P(x) < D$. Each of the possible $\bar{A}_P(x)$ is denoted by $V_P^l(x)$, that is:

$$V = \left\{ V_P^1(x), V_P^2(x), \dots, V_P^l(x), \dots, V_P^\lambda(x) \right\}$$

where λ is the total number of candidate $\bar{A}_P(x)$ that fall within the legitimate range, from which $A_P(x)$ is selected.

If their entropies equal $d_1 = d_2 = \dots = d_n$, then we can use the Hamming distance, H_T^l (i.e., Hamming distance between V_{PRNS}^l and vector $\bar{A}_{PRNS}$), of residues. H_T^l is defined as the number of elements in which two vectors, V_{PRNS}^l and $\bar{A}_{PRNS}$, differ.

But if we consider weighted error correction codes in which $d_1 \neq d_2 \neq \dots \neq d_n$, then the Hamming distance does not provide correct measurement, since residues carry a different amount of information about $A_P(x)$.

Let $m_1(x), m_2(x), \dots, m_n(x)$ be the PRNS moduli tuple:

$$V_P^l(x) \xrightarrow{PRNS} V_{PRNS}^l = (v_{l,1}(x), v_{l,2}(x), \dots, v_{l,n}(x)).$$

Let us calculate H_W^l as the Hamming weight of V_{PRNS}^l and $\bar{A}_{PRNS}$ using the following Algorithm 1.

Algorithm 1. Calculation H_W^l .

Input:

$$\bar{A}_{PRNS} = (\bar{a}_1(x), \bar{a}_2(x), \dots, \bar{a}_n(x))$$

$$V_{PRNS}^l = (v_{j,1}(x), v_{j,2}(x), \dots, v_{j,n}(x))$$

$$h_E = (H(a'_1(x)), H(a'_2(x)), \dots, H(a'_n(x)))$$

Output: H_W^l .

$$h \leftarrow (h_1, h_2, \dots, h_n)$$

// Hamming vector

for $i \leftarrow 1$ **to** n

if $v_{j,i}(x) = \bar{a}_i(x)$ **then**

$$h_i \leftarrow 0$$

else

$$h_i \leftarrow 1$$

end if

end for

$$\bar{h} \leftarrow (\bar{h}_1, \bar{h}_2, \dots, \bar{h}_n)$$

// Inverse of vector h

for $i \leftarrow 1$ **to** n

if $h_i = 0$ **then**

$$\bar{h}_i \leftarrow 1$$

// $a_i(x)$ does not contain an error

else

$$\bar{h}_i \leftarrow 0$$

end if

end for

$$h_E \leftarrow (H(\bar{a}_1(x)), H(\bar{a}_2(x)), \dots, H(\bar{a}_n(x))) \leftarrow (d_1, d_2, \dots, d_n)$$

$$H_W^l \leftarrow H(V_{PRNS}^l) \leftarrow \bar{h} \cdot h_E$$

Return H_W^l

According to Algorithm 1, three steps are used to calculate H_W^l : First, similar to MLD, calculate the Hamming vector, $h = (h_1, h_2, \dots, h_n)$, where $h_i \in \{0, 1\}$. Second, calculate the inverse of the vector h , $\bar{h} = (\bar{h}_1, \bar{h}_2, \dots, \bar{h}_n)$, where $\bar{h}_i$ is equal to one if the remainder $a_i(x)$ does not contain an error, otherwise $\bar{h}_i$ is equal to zero. Third, calculate the amount of entropy, H_W^l , as a dot product of two vectors, $\bar{h} = (\bar{h}_1, \bar{h}_2, \dots, \bar{h}_n)$, and vectors consisting of the entropies of the remainders of the division, $a'_i(x)$.

The idea is to calculate the amount of entropy, $H_W^l = H(V_{PRNS}^l)$, but not the number of errors, H_T^l , as in MLD. If the volume of correct data is greater than D , then according to Theorem 2, we can restore the value of $A_P(x)$.

Now, we can select $A_P(x)$ from all corrected $V_P^l(x)$. To this end, traditional MLD picks the candidate with the minimal Hamming distance. In our approach, the best candidate is the $V_P^l(x)$ with the maximal entropy, H_W^l .

Example 4. Let PRNS moduli 4-tuple be $m_1(x) = x^2 + x + 1$, $m_2(x) = x^3 + x + 1$, $m_3(x) = x^3 + x^2 + 1$, and $m_4(x) = x^6 + x + 1$, where $k = 3$, $r = 1$, and $n = k + r = 4$.

Let the dynamic range $M(x)$ and the full range of $\hat{M}(x)$ of PRNS be:

$$M(x) = x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + 1$$

and

$$\hat{M}(x) = x^{14} + x^{12} + x^{11} + x^{10} + x^7 + x^6 + x^2 + x + 1$$

Let $A_P(x) = x^7 \xrightarrow{PRNS} A_{PRNS} = (x, 1, 1, x^2 + x)$ and $E_{PRNS} = (1, 0, 0, 0)$, then $\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS} = (x + 1, 1, 1, x^2 + x)$. There are three possible values of V_P^l that satisfy the condition $\deg V_P^l(x) < D = \sum_{i=1}^k d_i$.

We present the calculation results in Table 2.

Table 2. V with three candidates of $V_P^l(x)$.

j	$V_P^l(x) \xrightarrow{PRNS} V_{PRNS}^l$	H_W^l, H_T^l
1	$x^7 \xrightarrow{PRNS} (x, 1, 1, x^2 + x)$	$H_W^1 = 0 + 3 + 3 + 6 = 12$ $H_T^1 = 1 + 0 + 0 + 0 = 1$
2	$x^6 + x^2 + 1 \xrightarrow{PRNS}$ $(x + 1, 0, x + 1, x^2 + x)$	$H_W^2 = 2 + 0 + 0 + 6 = 8$ $H_T^2 = 0 + 1 + 1 + 0 = 2$
3	$x^7 + x^6 + x^5 + x^4 + x^3 + x^2 +$ $x + 1 \xrightarrow{PRNS}$ $(x + 1, 1, 1, x^5 + x^4 + x^3 + x)$	$H_W^3 = 2 + 3 + 3 + 0 = 8$ $H_T^3 = 0 + 0 + 0 + 1 = 1$

If we use the MLD from Kolmogorov (1965) [36], then the tuple of possible recovering candidates $A_P(x)$ consists of two values $V_P^1(x) = x^7$ and $V_P^3(x) = \sum_{i=0}^7 x^i$ with the minimal Hamming distance. The proposed MLD modification for the error correction codes selects for maximal entropy. As we showed above, it allows us to determine the true result, $A_P(x) = V_P^1(x) = x^7 \xrightarrow{PRNS} (x, 1, 1, x^2 + x)$, unambiguously and correct the error. In classical PRNS, an error can be detected but not corrected.

7. Reliability

The detection of errors and failures in distributed storage and communication is mainly based on error correction codes, ECs, RCs, and modifications. In contrast to the existing methods, the PRNS method is simple and fast. However, it has one significant drawback: the limitation of detection and localization of errors. To overcome this limitation and increase the reliability, we propose an entropy-based approach to correct errors (see Section 6.1.).

Let $\Pr_i$ be the probability of error of data stored in the i -th cloud. It can be seen as the probability of unexpected and unauthorized modifications, falsifications, hardware and software failures, disk errors, integrity violations, denial of access or data loss, etc.

To correct errors, we have to take enough residues without errors. The data access structure is defined as a set of residues without errors allowing correct errors. It is defined as follows: $\Gamma_{En-AR-PRNS} = \{I | \sum_{i \in I} d_i \geq D\}$.

Hence, the probability of information loss of an En-AR-PRNS is determined by (13).

$$\Pr_{En-AR-PRNS} = 1 - \sum_{I \in \Gamma} \prod_{i \in I} (1 - \Pr_i) \prod_{i \notin I} \Pr_i \quad (13)$$

To estimate the probability of i -th storage error (i.e., failure), $\Pr_i$, we used data from an analysis of the downtime of public cloud providers presented by CloudHarmony [12]. It monitors the health status of service providers by spinning up workload instances in public clouds and constantly pinging them. It does not present a complete insight into the types of failures due to the limited information obtained by monitoring; however, it serves as a valuable point of reliability analysis.

Table 3 describes the downtime, T_D (in minutes per year), of eight cloud storage providers [39]. We calculated the $\Pr_i$ by the geometric probability definition (ratio of

measures) $\Pr_i = T_D/525,600$, where $525,600 = 365 \times 24 \times 60$ is the number of minutes in a year. It included upload and download times (Tchernykh et al., 2018) [18].

Table 3. Characteristics of the clouds.

	Cloud	$T_D(\text{min})$	$\Pr_i$	Speed (MB/s)		Latency (ms)
				Upload	Download	
1	CenturyLink	1889	0.003594	3.04	4.56	277.25
2	IBM's Cloud	1020	0.001941	3.51	4.05	80.42
3	Rackspace	770	0.001465	2.02	2.44	288.33
4	Digital Ocean	764	0.001454	5.08	5.82	87.50
5	Google	694	0.001320	2.68	4.23	201.58
6	Azure	649	0.001235	4.64	4.03	374.25
7	AWS	150	0.000285	2.63	2.9	281.17
8	Joyent	34	0.000065	2.13	3.11	390.83

Based on these probabilities, we can roughly estimate reliability by calculating the probability of information loss. In threshold PRNS, we lose data when at least $r + 1$ storages have errors when we are not able to correct errors. In En-AR-PRNS, as we showed in Section 6, the entropy-based approach allows for correcting more errors.

Let us consider both approaches. Table 4 shows an example of data sharing in eight storages. Column $m_i(x)$ shows the eight moduli we used for coding, $H(a_i(x))$ is the entropy of obtained residues, and the Cloud column is the used storage.

Table 4. Data allocation based on PRNS.

i	$m_i(x)$	$H(a_i(x))$	Cloud
1	$x^8 + x^4 + x^3 + x + 1$	8	CenturyLink
2	$x^8 + x^4 + x^3 + x^2 + 1$	8	IBM's Cloud
3	$x^8 + x^5 + x^3 + x + 1$	8	Rackspace
4	$x^8 + x^5 + x^3 + x^2 + 1$	8	Digital Ocean
5	$x^8 + x^5 + x^4 + x^3 + 1$	8	Google
6	$x^8 + x^6 + x^3 + x^2 + 1$	8	Azure
7	$x^8 + x^6 + x^5 + x + 1$	8	AWS
8	$x^8 + x^6 + x^5 + x^2 + 1$	8	Joyent

Due to the entropies and polynomial degrees being equal, the residues could be allocated based on the access speed, trustiness, etc., of the storages. In this case, the reliabilities of both the En-AR-PRNS and PRNS methods had no difference.

Figure 7 shows the probabilities of information loss. Figure 7a shows that this probability varied ranging from 24 (four storages) to 64 (eight storages) versus the threshold (dynamic range). Figure 7b shows this probability varying threshold from 24 to 64 versus the range.

Figure 7. Probability of information loss of an En-AR-PRNS (a) and a PRNS (b).

We observed that:

- By reducing thresholds (i.e., increasing data redundancy), we reduced the probability of information loss (Figure 7a);
- By increasing the number of clouds (i.e., increasing the range), we reduced the probability of information loss (Figure 7a);
- Reduction in the information loss probability by increasing the range was approximately equal for all thresholds (see Figure 7b).

Example 5. Let five cloud providers (i.e., Joyent, Azure, Google, Rackspace, and CenturyLink) be used for storage. The probabilities of errors and information loss were calculated using the data presented in [39] (see Table 5).

Table 5. Data allocation based on En-AR-PRNS.

i	$m_i(x)$	$H(a_i(x))$	Pr_i	Cloud
1	$x^2 + x + 1$	2	0.003594	CenturyLink
2	$x^4 + x + 1$	4	0.001465	Rackspace
3	$x^4 + x^3 + 1$	4	0.001320	Google
4	$x^4 + x^3 + x^2 + x + 1$	4	0.001235	Azure
5	$x^6 + x + 1$	6	0.000065	Joyent

Table 5 shows an example of the data sharing in these storage systems. The $m_i(x)$ column shows the five used moduli, the $H(a_i(x))$ column shows the entropy of residues, and the Cloud column shows the storage provider.

Due to the fact that the entropies were different, we allocated the residue with the highest entropy, $H(a_i(x)) = 6$, on storage with the lowest error probability, $i = 5$. Then,

we allocated three shares with entropy four on storages with the next smaller probabilities, $i = 4, 3, 2$. Finally, we allocated the last share with entropy two on storage $i = 1$.

For such an allocation, we used a PRNS with $k = 3$ and $r = 2$. The access structures $\Gamma_{En-AR-PRNS}$ and Γ_{PRNS} included combinations of the residues (recovery cases) that could be used to recover original data.

$\Gamma_{PRNS} = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}, \{2, 4, 5\}, \{3, 4, 5\}, \{1, 2, 3, 4\}, \{1, 2, 3, 5\}, \{1, 2, 4, 5\}, \{1, 3, 4, 5\}, \{2, 3, 4, 5\}, \{1, 2, 3, 4, 5\}\}$.

$\Gamma_{En-AR-PRNS} = \{\{2, 5\}, \{3, 5\}, \{4, 5\}\} \cup \Gamma_{PRNS}$.

$\Gamma_{En-AR-PRNS}$ had nineteen recovery cases and Γ_{PRNS} had sixteen cases. As we showed before, to restore data in a PRNS, we can have errors in two residues at most. In an En-AR-PRNS, we showed that, in some cases, we could restore data even with three residues have errors.

The probability of information loss calculated by (13) was $\Pr_{PRNS} = 2.3 \times 10^{-8}$ and $\Pr_{En-AR-PRNS} = 3.7 \times 10^{-9}$. Therefore, En-AR-PRNS improved the reliability of the storage by about $\frac{2.3 \cdot 10^{-8}}{3.7 \cdot 10^{-9}} \approx 6.2$ times using the same number of clouds with the same probability of errors for each provider.

Now, let us compare the reliability of data storages based on PRNS and En-AR-PRNS with various (k, n) settings. We defined the degree of residues as follows: $d_1 = d_2 = \dots = d_k = 8$ and $d_{k+1} = d_{k+2} = \dots = d_n = 16$. The probabilities of information loss of threshold PRNS and entropy-based PRNS on a logarithmic scale are presented in Figure 8.

Figure 8. The probability of information loss of a threshold PRNS and an entropy-based PRNS.

To show the advantage of an entropy-based approach, Figure 9 presents the reliability improvement of En-AR-PRNS over PRNS. We can see that the largest improvement was approximately eighty times for $(5, 8)$ settings. The top improvements were for $(3, 4)$, $(3, 5)$, $(3, 6)$, $(4, 7)$, and $(5, 8)$.

Figure 9. Reliability improvement of En-AR-PRNS over PRNS.

We can conclude that for the same number of cloud providers, the probability of information loss for En-AR-PRNS was much lower than for PRNS.

Now, we show how many recovery cases can be detected and corrected by both approaches.

Figure 10 shows the number of recovery cases found by an En-AR-PRNS and a PRNS. For instance, for (3, 4) settings, the PRNS could detect only one error in shares $i = 1, 2, 3, 4$. The En-AR-PRNS could detect the same errors and three more cases when detecting errors in two shares, for a total of seven cases.

Figure 10. Number of recovery cases detected by the En-AR-PRNS and the PRNS.

Figure 11 shows how many times the En-AR-PRNS outperformed the PRNS in error detection. We can see that for (7, 8) settings, the En-AR-PRNS outperformed the PRNS by 3.6 times.

Figure 11. Improvement of number of recovery cases detected by En-AR-PRNS over PRNS.

An En-AR-PRNS detects and corrects more errors than classical methods in a PRNS. Table 6 shows the comparative analysis of the methods.

Table 6. Comparison of multiple residue digit error detection and correction algorithms.

Algorithm	Number of Errors Correctable	Number of Errors Detected	Output Domain	Method Locate and Corrected Error
[27]	$\leq \lfloor r/2 \rfloor$	$\leq r$	Polynomial	Syndrome
[36]	$\leq \lfloor r/2 \rfloor$	$\leq r$	Polynomial	Projection
new	$\geq \lfloor r/2 \rfloor$	$\geq r$	Polynomial	En-AR-PRNS

Figure 12 shows the number of recovery cases corrected by En-AR-PRNS and PRNS. We see that for (3, 8) settings, the En-AR-PRNS corrected errors in 227 cases and PRNS in 36 cases.

Figure 12. Number of recovery cases corrected by En-AR-PRNS and PRNS.

Example 6 explains the results shown in Figure 12.

Example 6. Let PRNS moduli 4-tuple be $m_1(x) = x^6 + x + 1$, $m_2(x) = x^6 + x^3 + 1$, $m_3(x) = x^6 + x^5 + 1$, and $m_4(x) = x^{12} + x^3 + 1$, where $k = 3$, $r = 1$, and $n = k + r = 4$. The number of recovery cases found by PRNS is zero and by En-AR-PRNS it was three: $a_1(x)$, $a_2(x)$, and $a_3(x)$.

8. Discussion

The complexity of the presented algorithm motivates the usage of parallel computing. The inherent properties of PRNS attract much attention from both the scientific community and industry. Special interest is paid to the parallelization of the following algorithms:

- Coding: data conversion from conventional representation to PRNS;
- Decoding: data conversion from PRNS to conventional representation;
- Homomorphic computing: processing encrypted information without decryption;
- PRNS-based communication: transmitting shares;
- Computational intelligence: Estimating and classifying the security and reliability levels and finding adequate settings.

Coding. Since the computation of the residue for each modulo is independent, polynomial-to-PRNS conversion can be efficiently parallelized: independent calculation of residues $a_i(x)$; parallel calculation of each residue $a_i(x)$ based on a neural-like network approach to finite ring computations.

Decoding. Despite En-AR-PRNS performance improvement for PRNS-residues-to-polynomial conversion, presented in this paper, this conversion still significantly impacts the performance limitation of the whole system. It is one of the most difficult PRNS operations, and parallel computing can significantly improve it.

The following parallelization options are exploited:

- Parallel syndrome table search;
- Independent generation of the candidates, $V_P^l(x)$, with data recovery from all possible tuples of residues;
- Parallel calculation of $\sum_{i=1}^k M_i(x) \cdot O_i(x) \cdot a_i(x)$ and $R_A(x) \cdot M(x)$ to generate the candidate $V_P^l(x)$;
- Independent calculation of the entropy, H_W^l , for each candidate. PRNS offers an important source of parallelism for addition, subtraction, and multiplication operations;
- Data are converted to many smaller polynomials (residues). Operations on initial data are substituted by faster operations on residuals executed in parallel. The complexity of operations is reduced;
- These features are used by the cryptography (RSA) community (Yen et al., 2003) [40], (Bajard and Imbert, 2004) [41], homomorphic encryption (Cheon et al., 2019) [42], and Microsoft SEAL (Laine, 2017) [43];
- Errors in a faulty computational logic element are localized in the corresponding residue without impacting other residues. This property is used in the algorithm for

detecting errors in the AES cipher (Chu and Benaissa, 2013) [29] and control calculation results with encrypted data (Chervyakov et al., 2019) [2].

- Operations are based on fractional representations. Regardless of word sizes, they can be performed on bits without carrying in a single clock period. This improves computational efficiency, which has been proven to be especially useful in digital signal processing (Chang et al., 2015) [32], cryptography: RSA (Bajard and Imbert, 2004) [41], elliptic curve cryptography (Schinianakis et al., 2009 [44]; Guillermin, 2010) [45]), homomorphic encryption (Cheon et al., 2019 [42]; Laine, 2017 [43]), etc. This property is a way to approach a famous bound of speed at which addition and multiplication can be performed (Flynn, 2003) [46]. This bound, called Winograd's bound, determines a minimum time for arithmetic operations and is an important basis for determining the comparative value of the various implementations of algorithms.

Other optimization criteria can also be considered, for instance, power consumption. Using small arithmetic units for the PRNS processor reduces the switching activities in each channel and dynamic power (Wang et al., 2000) [47]. The enhanced speed and low power consumption make the PRNS very encouraging in applications with intensive operations.

9. Conclusions

In this paper, we studied data reliability based on a polynomial residual number system and proposed a configurable, reliable, and secure distributed storage scheme, named En-AR-PRNS. We provided a theoretical analysis of the dynamic storage configurations.

Our main contributions were multi-fold:

- We proposed a novel decoding technique based on entropy to increase reliability. We showed that it can detect and correct more errors than the classical threshold-based PRNS;
- We provided a theoretical analysis of the reliability, redundancy, and coding/decoding speed, depending on configurable parameters to dynamically adapt the current configuration to various changes in the storage that are difficult to predict in advance;
- We reduced the computational complexity of the decoding from $O(D^2)$ down to $O(D \cdot \log D)$ using the concept of an approximation of the rank of a polynomial.

The main idea of adaptive optimization is to set (k, n) PRNS parameters, moduli, and storages and dynamically change them to cope with different objective preferences and current properties. To this end, the past characteristics can be analyzed for a certain time interval to determine appropriate parameters. This interval should be set according to the dynamism of the characteristics and storage configurations.

This reactive approach deals with the uncertainty and non-stationarity associated with unauthorized data modifications, hardware and software malfunctions, disk errors, loss of data, malicious intrusions, denial of access for a long time, data transmission failures, etc. To detect changes, estimate, and classify the security and reliability levels, and violations of confidentiality, integrity, and availability, multi-objective techniques using computational intelligence and artificial intelligence can be applied. Future work will center on a comprehensive experimental study to assess the proposed mechanism's efficiency and effectiveness in real dynamic systems under different types of errors and malicious attacks.

Author Contributions: Conceptualization, A.T., M.B., A.A. and A.Y.D.; Data curation, A.Y.D.; Formal analysis, A.T. and M.B.; Investigation, M.B., A.A. and A.Y.D.; Methodology, A.T., A.A. and A.Y.D.; Project administration, A.A.; Writing—Original draft, A.T. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Education and Science of the Russian Federation (Project: 075-15-2020-915).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The proof of Theorem 1. If $N = \max_i \deg m_i(x)$, then $r_A(x) = R_A(x)$.

Proof . Let $b_i(x) = \lceil O_i(x) \cdot x^N / m_i(x) \rceil$, then $b_i(x)$ can be represented as $b_i(x) = O_i(x) \cdot x^N / m_i(x) + \theta_i(x)$, where $0 \leq \deg m_i(x) \cdot \theta_i(x) \leq \deg m_i(x)$.

Let us compute the value of $\sum_{i=1}^k b_i(x) a_i(x)$:

$$\begin{aligned} \sum_{i=1}^k b_i(x) \cdot a_i(x) &= \sum_{i=1}^k \left(\frac{O_i(x) \cdot x^N}{m_i(x)} + \theta_i(x) \right) \cdot a_i(x) \\ &= \sum_{i=1}^k \frac{O_i(x) \cdot x^N}{m_i(x)} a_i(x) + \sum_{i=1}^k \theta_i(x) \cdot a_i(x) \\ &= x^N \sum_{i=1}^k \frac{O_i(x)}{m_i(x)} a_i(x) + \sum_{i=1}^k \theta_i(x) \cdot a_i(x) \end{aligned} \quad (A1)$$

Computing $R_A(x)$ by substituting Equation (A1) into Equation (3), we obtain:

$$R_A(x) = \left\lfloor \sum_{i=1}^k \frac{O_i(x)}{m_i(x)} a_i(x) + \frac{\sum_{i=1}^k \theta_i(x) \cdot a_i(x)}{x^N} \right\rfloor \quad (A2)$$

From Equation (A2) and Equation (2), it follows that:

$$r_A(x) = R_A(x), \text{ if } \deg \left\lfloor \sum_{i=1}^k \theta_i(x) \cdot a_i(x) \right\rfloor < N.$$

The sufficient condition is:

$$\deg \left\lfloor \sum_{i=1}^k \theta_i(x) \cdot a_i(x) \right\rfloor < N.$$

Since:

$$\deg \left\lfloor \sum_{i=1}^k \theta_i(x) \cdot a_i(x) \right\rfloor \leq \deg \sum_{i=1}^k m_i(x) \leq \max_i \deg m_i(x),$$

then $N = \max_i \deg m_i(x)$ is sufficient to hold the inequality $\deg \left\lfloor \sum_{i=1}^k \theta_i(x) \cdot a_i(x) \right\rfloor < N$. The theorem is proved. $\square$

Appendix B

Example A1. Let PRNS moduli tuple be $m_1(x) = x^6 + x + 1$, $m_2(x) = x^6 + x^3 + 1$, $m_3(x) = x^3 + x + 1$, and $m_4(x) = x^3 + x^2 + 1$, where $k = 2$, $r = 2$, and $n = k + r = 4$. The dynamic range of PRNS is:

$$M(x) = m_1(x) \cdot m_2(x) = x^{12} + x^9 + x^7 + x^4 + x^3 + x + 1$$

and

$$\hat{M}(x) = m_1(x) \cdot m_2(x) \cdot m_3(x) \cdot m_4(x) = x^{18} + x^{17} + x^{16} + x^{13} + x^{12} + x^{10} + x^8 + x^3 + 1$$

$$A_P(x) = x^{11} \xrightarrow{\text{PRNS}} (x^5 + x + 1, x^2, x^2 + x, x^2 + x + 1)$$

$E_{PRNS} = (0, 1, 0, 0)$ and $\bar{A}_{PRNS} = A_{PRNS} + E_{PRNS}$, then:

$$\bar{A}_{PRNS}(x) \xrightarrow{PRNS} (x^5 + x + 1, x^2 + 1, x^2 + x, x^2 + x + 1).$$

Because $\sum_{j \in \bar{I}} d_j = d_2 = \deg m_2(x) = 6$ and $\sum_{i=1}^r d_{k+i} = d_3 + d_4 = \deg m_3(x) + \deg m_4(x) = 6$, the condition of Theorem 2 is satisfied (see Equation (8)) and an error is detected.

In the following, we show how the error is detected.

First, we calculate the PRNS constants b_1, b_2, b_3 , and b_4 .

$$\hat{M}_1(x) = \frac{\hat{M}(x)}{m_1(x)} = x^{12} + x^{11} + x^{10} + x^6 + x^2 + x + 1,$$

$$\hat{M}_2(x) = \frac{\hat{M}(x)}{m_2(x)} = x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1,$$

$$\hat{M}_3(x) = \frac{\hat{M}(x)}{m_3(x)} = x^{15} + x^{14} + x^{11} + x^{10} + x^5 + x^3 + x^2 + x + 1,$$

$$\hat{M}_4(x) = \frac{\hat{M}(x)}{m_4(x)} = x^{15} + x^{13} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + 1,$$

$$\left| \hat{M}_1^{-1}(x) \right|_{m_1(x)} = x^5 + x^4 + x + 1, \left| \hat{M}_2^{-1}(x) \right|_{m_2(x)} = x^3 + x^2 + 1, \left| \hat{M}_3^{-1}(x) \right|_{m_3(x)} = x,$$

$$\left| \hat{M}_4^{-1}(x) \right|_{m_4(x)} = x^2 + x + 1, N = \max_i \deg m_i(x) = 6;$$

$$b_1 = \left[\left| \hat{M}_1^{-1}(x) \right|_{m_1(x)} x^6 / m_1(x) \right] = x^5 + x^4 + x + 1,$$

$$b_2 = \left[\left| \hat{M}_2^{-1}(x) \right|_{m_2(x)} x^6 / m_2(x) \right] = x^3 + x^2 + 1,$$

$$b_3 = \left[\left| \hat{M}_3^{-1}(x) \right|_{m_3(x)} x^6 / m_3(x) \right] = x^4 + x^2 + x.$$

$$b_4 = \left[\left| \hat{M}_4^{-1}(x) \right|_{m_4(x)} x^6 / m_4(x) \right] = x^5 + x^3.$$

Second, to calculate $\bar{A}_P(x)$, we compute the sum:

$$\sum_{i=1}^4 b_i(x) \cdot \bar{a}_i(x) = (x^5 + x^4 + x + 1)(x^5 + x + 1) + (x^3 + x^2 + 1)(x^2 + 1) + (x^4 + x^2 + x)(x^2 + x) + (x^5 + x^3)(x^2 + x + 1) = x^{10} + x^9 + x^7 + x^5.$$

Using Equation (3), we obtain:

$$R_A(x) = \frac{\sum_{i=1}^4 b_i(x) \cdot \bar{a}_i(x)}{x^6} = \frac{x^{10} + x^9 + x^7 + x^5}{x^6} = x^4 + x^3 + x.$$

Finally, we compute $\bar{A}(x)$ using Equation (2) and Theorem 1.

$$\bar{A}_P(x) = \sum_{i=1}^4 \hat{M}_i(x) \left| \hat{M}_i^{-1}(x) \right|_{m_i(x)} \bar{a}_i(x) - R_A(x) \hat{M}(x) = x^{15} + x^{12} + x^6 + x^3 + x^2 + 1.$$

Because $\deg \bar{A}(x) = 15 > \deg M(x) = 12$, $\bar{A}(x)$ contains an error.

References

- Gomathisankaran, M.; Tyagi, A.; Namuduri, K. HORNS: A homomorphic encryption scheme for Cloud Computing using Residue Number System. In Proceedings of the 2011 45th Annual Conference on Information Sciences and Systems, Baltimore, MD, USA, 23–25 March 2011; pp. 1–5. [\[CrossRef\]](#)
- Chervyakov, N.; Babenko, M.; Tchernykh, A.; Kuchеров, N.; Miranda-López, V.; Cortés-Mendoza, J.M. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. *Futur. Gener. Comput. Syst.* **2019**, *92*, 1080–1092. [\[CrossRef\]](#)
- Tchernykh, A.; Babenko, M.; Chervyakov, N.; Miranda-López, V.; Kuchukov, V.; Cortés-Mendoza, J.M.; Deryabin, M.; Kuchеров, N.; Radchenko, G.; Avetisyan, A. AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage. *Int. J. Approx. Reason.* **2018**, *102*, 60–73. [\[CrossRef\]](#)
- Tchernykh, A.; Schwiegelshohn, U.; Talbi, E.-G.; Babenko, M. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. *J. Comput. Sci.* **2019**, *36*, 100581. [\[CrossRef\]](#)
- Tchernykh, A.; Babenko, M.; Chervyakov, N.; Miranda-Lopez, V.; Avetisyan, A.; Drozdov, A.Y.; Rivera-Rodriguez, R.; Radchenko, G.; Du, Z. Scalable Data Storage Design for Nonstationary IoT Environment with Adaptive Security and Reliability. *IEEE Internet Things J.* **2020**, *7*, 10171–10188. [\[CrossRef\]](#)
- Varghese, B.; Buyya, R. Next generation cloud computing: New trends and research directions. *Future Gener. Comput. Syst.* **2018**, *79*, 849–861. [\[CrossRef\]](#)
- Nachiappan, R.; Javadi, B.; Calheiros, R.N.; Matawie, K.M. Cloud storage reliability for Big Data applications: A state of the art survey. *J. Netw. Comput. Appl.* **2017**, *97*, 35–47. [\[CrossRef\]](#)
- Tan, C.B.; Hijazi, M.H.A.; Lim, Y.; Gani, A. A survey on Proof of Retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends. *J. Netw. Comput. Appl.* **2018**, *110*, 75–86. [\[CrossRef\]](#)
- Sharma, Y.; Javadi, B.; Si, W. Sun, Reliability and energy efficiency in cloud computing systems: Survey and taxonomy. *J. Netw. Comput. Appl.* **2016**, *74*, 66–85. [\[CrossRef\]](#)
- Li, S.; Cao, Q.; Wan, S.; Qian, L.; Xie, C. HRSPC: A hybrid redundancy scheme via exploring computational locality to support fast recovery and high reliability in distributed storage systems. *J. Netw. Comput. Appl.* **2016**, *66*, 52–63. [\[CrossRef\]](#)
- Baker, T.; Mackay, M.; Shaheed, A.; Aldawsari, B. Security-Oriented Cloud Platform for SOA-Based SCADA. In Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, 4–7 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 961–970. [\[CrossRef\]](#)
- Tchernykh, A.; Miranda-López, V.; Babenko, M.; Armenta-Cano, F.; Radchenko, G.; Drozdov, A.Y.; Avetisyan, A. Performance evaluation of secret sharing schemes with data recovery in secured and reliable heterogeneous multi-cloud storage. *Clust. Comput.* **2019**, *22*, 1173–1185. [\[CrossRef\]](#)
- Chen, X.; Qiming, H. The data protection of mapreduce using homomorphic encryption. In Proceedings of the 2013 IEEE 4th International Conference on Software Engineering and Service Science, Beijing, China, 23–25 May 2013; pp. 419–421. [\[CrossRef\]](#)
- Celesti, A.; Fazio, M.; Villari, M.; Puliafito, A. Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. *J. Netw. Comput. Appl.* **2016**, *59*, 208–218. [\[CrossRef\]](#)
- Tchernykh, A.; Babenko, M.; Kuchukov, V.; Miranda-Lopez, V.; Avetisyan, A.; Rivera-Rodriguez, R.; Radchenko, G. Data Reliability and Redundancy Optimization of a Secure Multi-cloud Storage Under Uncertainty of Errors and Falsifications. In Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Rio de Janeiro, Brazil, 20–24 May 2019; pp. 565–572. [\[CrossRef\]](#)
- Srisakthi, S.; Shanthi, A.P. Towards the Design of a Secure and Fault Tolerant Cloud Storage in a Multi-Cloud Environment. *Inf. Secur. J. Glob. Perspect.* **2015**, *24*, 109–117. [\[CrossRef\]](#)
- Hubbard, D.; Sutton, M. Top Threats to Cloud Computing V1.0, Cloud Security Alliance, 2010, (n.d.). Available online: <https://ioactive.com/wp-content/uploads/2018/05/csathreats.v1.0-1.pdf>. (accessed on 24 April 2021).
- Tchernykh, A.; Babenko, M.; Miranda-Lopez, V.; Drozdov, A.Y.; Avetisyan, A. WA-RRNS: Reliable Data Storage System Based on Multi-cloud. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, Canada, 21–25 May 2018; pp. 666–673. [\[CrossRef\]](#)
- Ghemawat, S.; Gobioff, H.; Leung, S.-T. The Google file system. *ACM SIGOPS Oper. Syst. Rev.* **2003**, *37*, 29. [\[CrossRef\]](#)
- Miranda-Lopez, V.; Tchernykh, A.; Babenko, M.; Avetisyan, A.; Toporkov, V.; Drozdov, A.Y. 2Lbp-RRNS: Two-Levels RRNS With Backpropagation for Increased Reliability and Privacy-Preserving of Secure Multi-Clouds Data Storage. *IEEE Access* **2020**, *8*, 199424–199439. [\[CrossRef\]](#)
- Lin, H.-Y.; Tzeng, W.-G. A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 995–1003. [\[CrossRef\]](#)
- Dimakis, A.G.; Godfrey, P.B.; Wu, Y.; Wainwright, M.J.; Ramchandran, K. Network Coding for Distributed Storage Systems. *IEEE Trans. Inf. Theory* **2010**, *56*, 4539–4551. [\[CrossRef\]](#)
- Gentry, C. Computing arbitrary functions of encrypted data. *Commun. ACM* **2010**, *53*, 97. [\[CrossRef\]](#)
- Asmuth, C.; Bloom, J. A modular approach to key safeguarding. *IEEE Trans. Inf. Theory* **1983**, *29*, 208–210. [\[CrossRef\]](#)
- Mignotte, M. How to Share a Secret. In *Cryptography. EUROCRYPT 1982; Lecture Notes in Computer Science*; Beth, T., Ed.; Springer: Berlin/Heidelberg, Germany, 1982; Volume 149. [\[CrossRef\]](#)

26. Lin, S.-J.; Chung, W.-H.; Han, Y.S. Novel Polynomial Basis and Its Application to Reed-Solomon Erasure Codes. In Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, Philadelphia, PA, USA, 18–21 October 2014; pp. 316–325. [\[CrossRef\]](#)
27. Liu, J.; Huang, K.; Rong, H.; Wang, H.; Xian, M. Privacy-Preserving Public Auditing for Regenerating-Code-Based Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1513–1528. [\[CrossRef\]](#)
28. Skavantzios, A.; Taylor, F.J. On the polynomial residue number system (digital signal processing). *IEEE Trans. Signal Process.* **1991**, *39*, 376–382. [\[CrossRef\]](#)
29. Chu, J.; Benaissa, M. Error detecting AES using polynomial residue number systems. *Microprocess. Microsyst.* **2013**, *37*, 228–234. [\[CrossRef\]](#)
30. Parker, M.G.; Benaissa, M. $GF(p^m)$ multiplication using polynomial residue number systems. *IEEE Trans. Circuits Syst. II Analog. Digit. Signal Process* **1995**, *42*, 718–721. [\[CrossRef\]](#)
31. Chervyakov, N.I.; Babenko, M.G.; Kuchero, N.N. Development of Homomorphic Encryption Scheme Based on Polynomial Residue Number System. *Sib. Electron. Math. Rep.-Sib. Elektron. Mat. Izv.* **2015**, *12*, 33–41.
32. Chang, C.-H.; Molahosseini, A.S.; Zarandi, A.A.E.; Tay, T.F. Residue Number Systems: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications. *IEEE Circuits Syst. Mag.* **2015**, *15*, 26–44. [\[CrossRef\]](#)
33. Halbutogullari, A.; Koc, C.K. Parallel multiplication in $GF(2^k)$ using polynomial residue arithmetic. *Des. Codes Cryptogr.* **2000**, *20*, 155–173. [\[CrossRef\]](#)
34. Geekbench Browser. Available online: <https://browser.geekbench.com> (accessed on 24 April 2021).
35. Chervyakov, N.I.; Lyakhov, P.A.; Babenko, M.G.; Garyanina, A.I.; Lavrinenko, I.N.; Lavrinenko, A.V.; Deryabin, M.A. An efficient method of error correction in fault-tolerant modular neurocomputers. *Neurocomputing* **2016**, *205*, 32–44. [\[CrossRef\]](#)
36. Kolmogorov, A.N. Three approaches to the definition of the concept “quantity of information”. *Probl. Peredači Inf.* **1965**, *1*, 3–11.
37. Ivanov, M.; Sergiyenko, O.; Mercorelli, P.; Hernandez, W.; Tyrsa, V.; Hernandez-Balbuena, D.; Rodriguez Quinonez, J.C.; Kartashov, V.; Kolendovska, M.; Iryna, T. Effective informational entropy reduction in multi-robot systems based on real-time TVS. In Proceedings of the 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada, 12–14 June 2019; LNCS; Volume 11349, pp. 1162–1167. [\[CrossRef\]](#)
38. Goh, V.T.; Siddiqi, M.U. Multiple error detection and correction based on redundant residue number systems. *IEEE Trans. Commun.* **2008**, *56*, 325–330. [\[CrossRef\]](#)
39. Research and Compare Cloud Providers and Services. Available online: <https://cloudharmony.com/status> (accessed on 26 November 2021).
40. Yen, S.-M.; Kim, S.; Lim, S.; Moon, S.-J. RSA speedup with chinese remainder theorem immune against hardware fault cryptanalysis. *IEEE Trans. Comput.* **2003**, *52*, 461–472. [\[CrossRef\]](#)
41. Bajard, J.C.; Imbert, L. a full RNS implementation of RSA. *IEEE Trans. Comput.* **2004**, *53*, 769–774. [\[CrossRef\]](#)
42. Cheon, J.H.; Han, K.; Kim, A.; Kim, M.; Song, Y. *A Full RNS Variant of Approximate Homomorphic Encryption*, in LNCS; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11349, pp. 347–368. [\[CrossRef\]](#)
43. Laine, K. Simple Encrypted Arithmetic Library 2.3.1., Microsoft Res. 2017. Available online: <https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf> (accessed on 17 April 2021).
44. Schinianakis, D.M.; Fournaris, A.P.; Michail, H.E.; Kakarountas, A.P.; Stouraitis, T. An RNS Implementation of an F_p Elliptic Curve Point Multiplier. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2009**, *56*, 1202–1213. [\[CrossRef\]](#)
45. Guillermin, N. A High Speed Coprocessor for Elliptic Curve Scalar Multiplications over F_p . In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 48–64. [\[CrossRef\]](#)
46. Flynn, M.J.; Oberman, S. *Advanced Computer Arithmetic Design*; Wiley: Hoboken, NJ, USA; p. 344.
47. Wang, W.; Swamy, M.N.S.; Ahmad, M.O. An area-time-efficient residue-to-binary converter. In Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Cat.No.CH37144), Lansing, MI, USA, 8–11 August 2000; Volume 2, pp. 904–907. [\[CrossRef\]](#)