Data Reliability and Redundancy Optimization of a Secure Multi-Cloud Storage Under Uncertainty of Errors and Falsifications

Andrei Tchernykh
CICESE Research Center, Ensenada, Mexico
South Ural State University, Chelyabinsk, Russia
Ivannikov Institute for System Programming, Moscow, Russia, chernykh@cicese.mx

Mikhail Babenko, Viktor Kuchukov
North-Caucasus Federal University
Stavropol, Russia
{mgbabenko, vkuchukov}@ncfu.ru

Vanessa Miranda-López
CICESE Research Center
Ensenada, Mexico
vmiranda@cicese.edu.mx

Arutyun Avetisyan
Ivannikov Institute for System Programming, RAS, Moscow, Russia Moscow, Russia
arut@ispras.ru

Raul Rivera-Rodriguez
CICESE Research Center, Ensenada, Mexico
rrivera@cicese.mx

Gleb Radchenko
South Ural State University, Chelyabinsk, Russia
gleb.radchenko@susu.ru

***Abstract*—Despite all the benefits a cloud data storages offer to customers, there is a high risk of breach of confidentiality, integrity, and availability related with the uncertainty of errors and falsifications, loss of information, denial of access for a long time, information leakage, conspiracy, and technical failures. In this article, we propose a configurable, reliable, and secure distributed data storage scheme with improved data redundancy, reliability, and encoding/decoding speed. Our system utilizes a Polynomial Residue Number System (PRNS) with a new method of error correction codes and secret sharing schemes. We introduce the concept of an approximate value of a rank (AR) of a polynomial. It reduces the computational complexity of the encoding/decoding and PRNS coefficients size. Based on the properties of the approximate value and PRNS, we introduce the AR-PRNS method for error detection, correction, and controlling computational results with capabilities of scalable parallel computing. We provide a theoretical basis to configure and optimize the redundancy of stored data and encoding/decoding speed to cope with different objective preferences, workloads, and storage properties. Theoretical analysis shows that, by appropriate selection of AR-PRNS parameters, the proposed scheme increases the safety, reliability, and reduces the overhead of data storage.**

Keywords—cloud storage, reliability, polynomial residue number system, secret sharing scheme, security, uncertainty

I. INTRODUCTION

Reliable data storage is a prerequisite for most computer systems. Several uncertain factors can reduce the reliability and safety of the data: unexpected and unauthorized modifications, hardware and software malfunctions, disk errors, integrity violation, loss of data, malicious intrusions, falsifications, denial of access for a long time, information leakage, conspiracy, etc.

Several approaches are proposed in the literature and implemented in real systems to ensure reliability: Data replication (Ghemawat et al., 2003) [1], Secret Sharing Schemes (SSS) (Gomathisankaran et al., 2011) [3], Redundant Residue Number System (RRNS) (Celesti et al., 2016 [4]; Chervyakov et al., 2019 [5]), Erasure Codes (EC) (Lin & Tzeng, 2012) [6], Regenerating Codes (RC) (Dimakis et al., 2010) [7], Homomorphic Encryption (HE) (Gentry, 2010 [8]; Tchernykh et al., 2018 [9]), etc.

Scalable distributed cloud data storage systems are suitable for high load applications that operate with big data. They offer a flexible deployment model so that users can create the right sized storage based on the capacity, cost, and performance needs of the workloads. They are protected against data loss across a zone or region. They can adapt to data volume, velocity, variety, and veracity that are challenging for the traditional data storages.

Multi-cloud storage systems use static (Gomathisankaran et al., 2011 [3]; Tchernykh et al., 2016 [10]; Chervyakov et al., 2019 [5]) and dynamic (Tchernykh et al., 2018) [9] scaling. It permits to distribute load but does not solve the problem of data threats management (Tchernykh et al., 2019) [2].

Chen and Huang, 2013 [11] introduced a technology based on MapReduce and fully homomorphic encryption. The disadvantages of this system include high data redundancy, computational complexity of data encryption algorithms, and low reliability.

Celesti et al., 2016 [4] proposed a cloud storage system based on the Residue Number System (RNS) that overcomes the issues of the above approach. However, Tchernykh et al., 2019 [2] show that SSSs based on RNS have a low speed of data encoding and decoding.

In this paper, we introduce configurable and reliable systems for secure multi-cloud storage with improved data redundancy, reliability, and data encoding speed based on the Polynomial Residue Number System (PRNS). The operation that consumes most resources in PRNS to binary conversion is finding PRNS residue from division by dynamic range. We introduce Approximation of the Rank (AR) that permits to avoid the most resources consuming operation of finding residue from division by a large polynomial. Based on AR and PRNS properties we introduce the AR-PRNS method for error detection, correction, and controlling computational results. Its coding/decoding operations and processing can be performed in parallel as a

series of smaller calculations with horizontal and vertical scalability.

This paper is organized as follows. Section II reviews distributed storage systems for clouds. Section III discusses PRNS properties. Section IV presents the configurable parameters used in our scheme. Section V introduces the rank of PRNS number and its approximation. Section VI focuses on error detection, correction, and control of arithmetic operations, data processing for distributed storage systems. Section VII describes our configurable model. The conclusions and future work are discussed in the last Section VIII.

II. Related Works

A. Secure Cloud Storage

An information system is secure if it prevents information from unauthorized access, use, disclosure, disruption, modification, etc. Research and industry are working on a secure and fault-tolerant multi-cloud environment, where confidentiality, integrity, and availability are preserved even in the presence of failures, deliberate as well as accidental threats (Srisakthi and Shanthi, 2015) [12].

Confidentiality refers to treating the user data as secret, private, and not seen even by the cloud service provider itself. Integrity means that data is not changed during operations such as storage or transmission or by any entity other than the owner. Availability stands for the service being available to the user at any time. Privacy implies that there is no unauthorized access to user data.

To minimize the impact of environmental threats (natural disasters, fires, floods, technological accidents, etc.), geographically distributed data sharing mechanisms, replication, and SSS, where various cloud providers store each piece of information, are used. Data backups distributed geographically and disaster recovery plans increase already high costs of the storages.

Deliberate threats are unauthorized access to information, interception, falsification, forgery, hacker attacks, etc. Cloud Security Alliance announced that, over the last years, unauthorized access to the information in the clouds has dramatically increased in number (Hubbard and Sutton, 2010) [13]. SSSs together with error correction codes reduce this risk.

There are numerous accidental threats such as user errors, carelessness, curiosity, etc. They can be tackled by information control and protection systems based on the concept of proactive security (Tchernykh et al., 2018) [14]. The proactive concept incorporates weighted SSS based on RNS, encryption keys, and checksums. In this paper, we propose an approach that protects the system from the threats mentioned above and optimizes RNS efficiency.

B. Reliability and Privacy

Now, we examine six basic approaches to ensure the reliability of data storage: data replication, secret sharing schemes, redundant residue number system, erasure codes, regenerating codes, and homomorphic encryption.

Data replication provides high reliability and availability. But this approach is expensive, the amount of stored data and the costs are growing dramatically (Ghemawat et al., 2003) [1].

Secret sharing schemes (SSS) (e.g., Shamir, Blackly, etc.) divide data into shares and distribute them to participants in such a way that only authorized subsets of participants can recover the original data from shares. SSSs permit to build secure distributed storage systems (Gomathisankaran et al., 2011) [3]. Asmuth et al., 1983 [15] and Mignotte, 1982 [16] proposed fully homomorphic SSS based on RNS that is asymptotically perfect and balanced (depending on the problem).

Redundant Residue Number System (RRNS) represents a number with its residues with respect to a moduli set. Computations in RRNS are fast due to the original number is divided into smaller numbers, operations are carried out independently and concurrently. Moreover, redundancy of residues allows multiple-error detection and correction, which permits to build reliable data storage (Celesti et al., 2016[4]; Tchernykh et al., 2019 [2]; Chervyakov et al., 2019 [5]). Let RRNS have r control moduli, then r errors can be detected and $\lfloor r/2 \rfloor$ corrected. In RRNS, errors are isolated and corrected with projection methods. The number of required projections grows exponentially with r. It follows that significant optimization is required for RRNS to be efficient in practice.

Erasure code (EC) is an error correction code. EC takes a message of length k and produces a message of length n (where $k<n$) so that the original message can be recovered from a subset of the n symbols. There is an efficient implementation of EC by Lin, et al. 2014 [17] with $O(n \cdot \log_2 n)$ complexity. In case of an error, the encoded block may be repaired or, alternatively, a new one can be generated. EC can be used to implement a reliable data storage. However, EC is not homomorphic and, hence, does not allow efficient data processing. Dimakis, et al. 2010 [7] has shown sub-optimality of EC.

Regenerating code (RC) is a class of codes designed for reliable data storage and efficient repair (reconstruction) of lost encoded fragments in distributed systems. They can significantly reduce the total traffic required for repairing, a.k.a. repair-bandwidth. Furthermore, RC has reasonable tradeoffs between storage and repair bandwidth (Dimakis et al., 2010) [7]. A special pseudorandom number generator is required for the effective implementation of RC (Liu et al., 2015) [18]. However, high-performance computing is not possible in RC, since it is not homomorphic w.r.t. addition, subtraction, and multiplication (Chen et al., 2014) [19].

Homomorphic encryption (HE). HE allows us to carry out computations over encrypted data. HE systems were proposed by Rivest et al., 1978 [20]. They can be based on exponentiation, in which case it is homomorphic, and RSA function (public-key cryptosystem). Gentry, 2010 [8] introduced a wide range of approaches to build fully homomorphic ciphers and solve underlying performance problems. There is an alternative approach which uses matrix polynomials to construct a fully homomorphic scheme. It was proposed by Trepacheva et al., 2014 [21]. This approach solves the main problems of existing homomorphic ciphers such as low performance and high redundancy. However, it is not widely used in practice, since it

lacks compactness and crypto security (Rivest et al., 1978) [20], (Gomathisankaran et al., 2011) [3].

III. Polynomial Residue Number System

A. Main Properties

PRNS over GF(2^m) was firstly introduced by Halbutogullari & Koc, 2000 [22]. Contrary to RNS, where each modulus is a co-prime number, in PRNS each modulus is an irreducible polynomial. However, the same way as in RNS, the Chinese Remainder Theorem (CRT) can be applied to PRNS. Error detection and correction can be implemented in PRNS by adding redundant residue moduli.

First, a set of irreducible polynomials over the binary field is selected as the field generating polynomials for PRNS moduli. We denote this set: $m_1(x), m_2(x), \dots, m_n(x)$, where n is the number of moduli. d_i is the degree of the polynomial $m_i(x)$. For a unique representation of an arbitrary element from $GF(2^m)$ with its residues, the degree D of the product polynomial $M(x) = \prod_{i=1}^{n} m_i(x)$ should be greater or equal to m, that is $\sum_{i=1}^{n} d_i \geq m$.

A polynomial $A(x)$ is represented with a set of remainders as follows

$$A(x) \xrightarrow{PRNS} (a_1(x), a_2(x), \dots, a_n(x))$$

where $a_i(x) = |A(x)|_{m_i(x)}$ for $i = 1,2, \dots, n$ and $|A(x)|_{m(x)}$ is the remainder of dividing the polynomial $A(x)$ by $m(x)$.

In PRNS over $GF(2^m)$, operations such as addition and multiplication are performed in parallel as follows:

$$A(x) \pm B(x) \xrightarrow{PRNS} \left(|a_1 \text{XOR } b_1|_{m_1}, \dots, |a_n \text{XOR } b_n|_{m_n}\right)$$

$$A(x) \times B(x) \xrightarrow{PRNS} \left(|a_1 \times b_1|_{m_1}, \dots, |a_n \times b_n|_{m_n}\right)$$

Addition and subtraction in PRNS do not require modular reduction since overflow is not possible with bitwise XOR in the binary field. However, the modular reduction is still necessary for multiplication. To prevent overflow, since it is not possible to determine the magnitude directly in PRNS, a conversion back to polynomial representation is necessary before performing the $GF(2^m)$ modular reduction. Meanwhile, error detection methods also require the conversion.

To convert from the PRNS format to weighted polynomial representation, the extension of the CRT to polynomials is used.

$$A(x) = \left| \sum_{i=1}^{n} a_i(x) \cdot I_i(x) \cdot M_i(x) \right|_{M(x)} \quad (1)$$

where $I_i(x) = |M_i^{-1}(x)|_{m_i(x)}$ and $M_i(x) = \frac{M(x)}{m_i(x)}$ for $i = 1,2, \dots, n$.

B. Error Detection

Approach to error detection in PRNS over $GF(2^m)$ is similar to error detection in RNS: redundant polynomial residue moduli is added, and the whole representation range is divided into the legitimate range and illegitimate range (Chu & Benaissa, 2013) [23]. Then, the fact that the conversion result falls into a illegitimate range means that an error occurred in a modulus.

To define redundant PRNS, we first define PRNS with the irreducible polynomial set $m_1, m_2, \dots, m_n$, which satisfies $\sum_{i=1}^{n} d_i \geq m$. Then, we add one additional polynomial modulus m_{n+1} with the degree $d_{n+1} \geq d_i$, for $i = 1,2, \dots, n$.

The legitimate range is represented by the product polynomial $M(x) = \prod_{i=1}^{n} m_i(x)$ and the highest possible degree is $D = \sum_{i=1}^{n} d_i$. With the new added modulus m_{n+1}, the whole representation range becomes $\bar{M}(x) = \prod_{i=1}^{n+1} m_i(x)$, where the highest possible degree of $\bar{M}(x)$ is $\bar{D} = \sum_{i=1}^{n+1} d_i$. All polynomials with the highest degree greater or equal to D and smaller than $\bar{D}$ fall into the illegitimate range and indicate an error. The proof is given below.

Let an element $A(x)$ from $GF(2^m)$ be represented in redundant PRNS as $(a_1(x), a_2(x), \dots, a_n(x), a_{n+1}(x))$. Let this representation belongs to the legitimate range with the degree no higher than D. Then, we assume that an error occurs in the i-th modulus during multiplication, the result yields a faulty PRNS representation of $A'(x) \xrightarrow{PRNS} (a_1(x), \dots, a_i'(x), \dots, a_{n+1}(x))$. The $A'(x)$ can be represented as a sum of its correct value $A(x)$ and error value $E(x)$ as follows: $A'(x) = A(x) + E(x)$ or in PRNS format:

$$A'(x) = \left(a_1(x), \dots, a_i(x), \dots, a_{n+1}(x)\right) + (0, \dots, e_i(x), \dots, 0).$$

Since $A(x)$ is an element over $GF(2^m)$, its degree is smaller or equal to $m-1$, so $A(x)$ falls into the legitimate range. Now we convert $E(x)$from PRNS back to the weighted polynomial representation using Eq. (1). It yields:

$$\begin{aligned} E(x) &= |e_i(x) \cdot \bar{\mathrm{I}}_i(x) \cdot \bar{\mathrm{M}}_i(x)|_{\bar{M}(x)} \\ &= |e_i(x) \cdot \bar{I}_i(x)|_{m_i(x)} \cdot \bar{\mathrm{M}}_i(x) \end{aligned}$$

where $\bar{\mathrm{M}}_i(x) = \frac{\bar{M}(x)}{m_i(x)}$ and $\bar{\mathrm{I}}_i(x) = \left|\frac{1}{\bar{\mathrm{M}}_i(x)}\right|_{m_i(x)}$.

The highest degree of $\bar{\mathrm{M}}_i(x)$ is $\sum_{j=1}^{n+1} d_j - d_i$ and the degree of $|e_i(x) \cdot \bar{I}_i(x)|_{m_i(x)}$ can be any number from 0 to $d_i - 1$. So the possible highest degree of $E(x)$, which we denote D_E, ranges from $(\sum_{j=1}^{n+1} d_j - d_i)$ to $(\sum_{j=1}^{n+1} d_j - 1)$.

Since $\sum_{j=1}^{n+1} d_j - d_i \geq D$, the following condition is true: if $\deg(A'(x)) < D$ then $A'(x)$ is correct, otherwise $A'(x)$ is not correct.

IV. Configurable Parameters

A. Data Redundancy Degradation

Redundancy in a big data storage is an important issue. Since, in the worst case, the number of bits that need to be stored $(d_1, d_2, \dots, d_n)$ is equal to $\sum_{i=1}^{n} d_i$. The input data is the L-bit number. It approximately equals to D. We calculate redundancy degradation as the ratio of the stored encoded data to the original data size minus 1:

$$\frac{\sum_{i=1}^{n} d_i}{D} - 1 .$$

In PRNS settings (k, n), using data from any k remainders from n, we can recover $r = n - k$ data.

For efficiency reasons, we select each $m_i(x)$ so that $a_i(x)$ is represented using at most $d_1 = d_2 = \cdots = d_n = d$ bits or 1 word. Then the redundancy degradation satisfies the inequality:

$$\frac{\sum_{i=1}^{n} d_i}{D} - 1 = \frac{n \cdot d}{k \cdot d} - 1 = \frac{n-k}{k}$$

Consequently, redundancy degradation is roughly $(n-k)/k$.

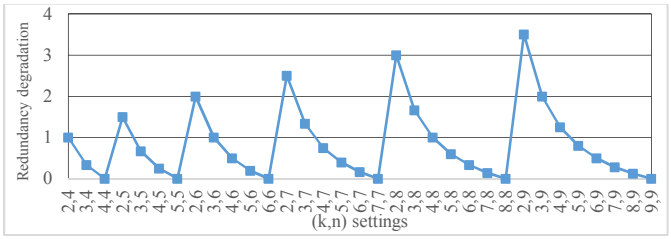

Fig. 1. Data redundancy degradation versus PRNS settings (k, n)

Redundancy degradation depending on PRNS parameters is shown in Fig. 1. We see that the redundancy has minimal values with PRNS settings (n, n), where $n = 4, 5, \ldots, 9$, and less than those of the Bigtable system $(\lceil (n+1)/3 \rceil, n)$.

Example 1. Let us consider an example of (4,4) scheme with PRNS. Let moduli set are $m_1(x) = x^8 + x^4 + x^3 + x + 1$, $m_2(x) = x^8 + x^4 + x^3 + x^2 + 1$, $m_3(x) = x^8 + x^5 + x^3 + x + 1$ and $m_4(x) = x^8 + x^5 + x^3 + x^2 + 1$. In this case, the dynamic range is $M(x) = \prod_{i=1}^{3} m_i(x) = x^{32} + x^{26} + x^{24} + x^{23} + x^{21} + x^{19} + x^{18} + x^{16} + x^{11} + x^{10} + x^9 + x^5 + x^4 + x^2 + 1$.

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

$$A(x) \xrightarrow{PRNS} \big(a_1(x), a_2(x), a_3(x), a_4(x)\big),$$

where

$$a_1(x) = |A(x)|_{m_1(x)} = x^7 + x^4 + x^3,$$

$$a_2(x) = |A(x)|_{m_2(x)} = x^7 + x^5 + x^4 + x^3,$$

$$a_3(x) = |A(x)|_{m_3(x)} = x^7 + x^4 + x^3 + x + 1,$$

$$a_4(x) = |A(x)|_{m_4(x)} = x^7 + x^6 + x^3 + x,$$

$a_1(x), a_2(x), a_3(x), a_4(x)$ and it has 8 bits. Therefore, the redundancy degradation is $4 \cdot \frac{8}{32} - 1 = 0$.

B. Data Encoding Speed

To analyze the speed of data encoding, we use the minimum technical characteristics of VM provided by Microsoft Azure. It is Intel Xeon® E5-2673 v.3, 2GB of RAM, 16GB SSD hard drive. It has an average speed 2^{30} bit operations per second according to tests from (Geekbench Browser site) [24].

The modification from work (Chervyakov et al., 2016) [25] allows achieving $O(d \cdot k)$ algorithmic complexity. Hence, we assume that calculation of the remainder of division requires roughly $d \cdot k$ bit operations. To represent a L-bit number in RNS, we have to perform an operation of finding the remainder of the division by RNS modulus n times. Thus, a total number of the bit operations is $n \cdot d \cdot k$. Since the size of the input block is $L \approx k \cdot d$ bits, the number of blocks in 1 Mb is $2^{23}/L \approx 2^{23}/(k \cdot d)$. Therefore, in order to encode 1 Mb of data, $2^{23} \cdot n \cdot d \cdot k/(k \cdot d) = 2^{23} \cdot n$ bit operations are required. The speed of data encoding in MB/s can be calculated by the formula:

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

The dependence of the data encoding speed on the parameters of the scheme is shown in Fig. 2.

We see that the graph is a saw type, where the maximum values of the coding rate are achieved in the schemes (n, n), and minimum coding rates are achieved in the schemes $(2, n)$. The user can select the required parameters to provide the required value of the data coding rate.

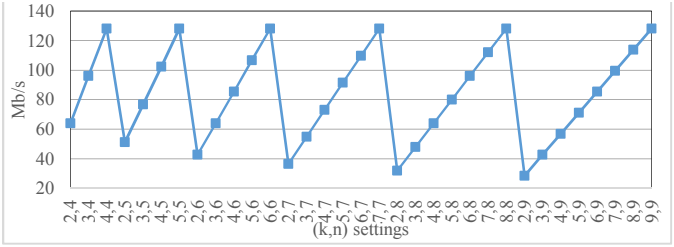

Fig. 2. The speed of data encoding (Mb/s) versus PRNS settings (k, n)

C. Data Decoding Speed

When the data is decoded with no errors, according to Chinese Remainder Theorem (CRT) with algorithmic complexity $O(L^2)$, it requires roughly $L^2 \approx k^2 \cdot d^2$ bit operations. In the case of r extra moduli, we can detect and correct $\left\lfloor \frac{r}{2} \right\rfloor$ errors.

We use the algorithm based on projections. To compute a projection, we use CRT. Therefore, one projection is computed in roughly $L^2 \approx k^2 \cdot d^2$ bit operations. Since the number of projections is $C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor}$, to detect and localize an error, we need $C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor} \cdot k^2 \cdot d^2$ bit operations. To encode 1 Mb of data, $2^{23} \cdot C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor} \cdot k^2 \cdot d^2/(k \cdot d) = 2^{23} \cdot C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor} \cdot d \cdot k$ bit operations are required. Therefore, the speed of decoding is $V_D = \frac{2^{30}}{2^{23} \cdot C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor} \cdot d \cdot k} = \frac{2^7}{C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor} \cdot d \cdot k}$.

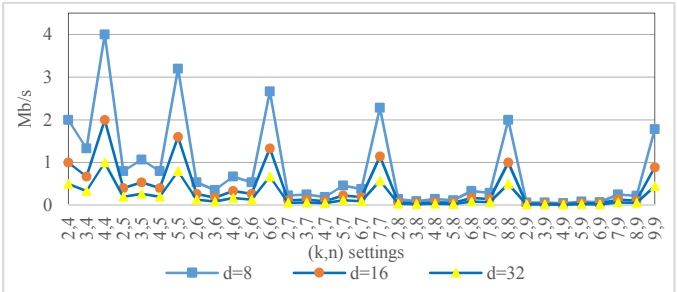

Fig. 3. The speed of data decoding (Mb/s) versus PRNS settings (k, n) for $d = \{8, 16, 32\}$

The dependence of the data decoding speed from the parameters of the scheme is shown in Fig. 3 for d $= \{8, 16, 32\}$.

D. Comparative Analysis

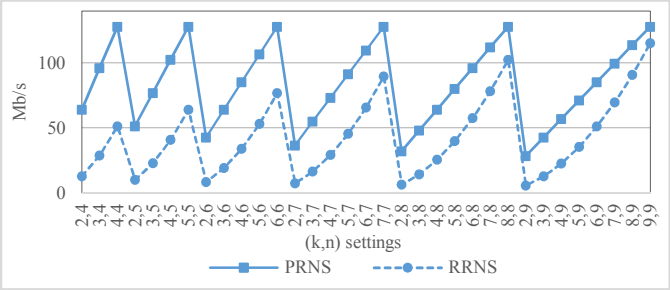

Fig. 4. The speed of data encoding (Mb/s) based on PRNS and RRNS settings (k, n)

Since arithmetic operations of PRNS over $GF(2^m)$ do not require transferring values from the lowest to the highest degree, the computational complexity of the algorithm for finding the remainder of the division is reduced compared to the arithmetic operations RRNS performed over Z_n (Fig. 4).

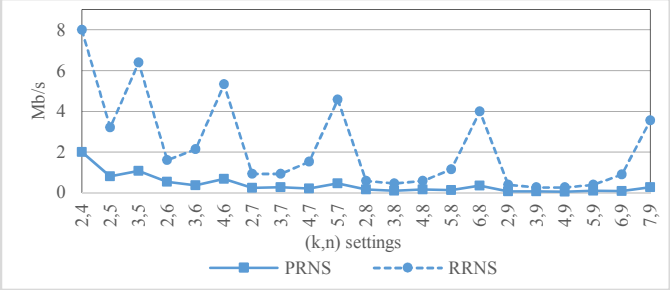

Fig. 5. The speed of decoding data (Mb/s) based on PRNS and RRNS on the worst scenario of data recovery with the maximum number of errors versus PRNS settings (k, n) for $d = 8$ and $b = 8$.

The computational complexity of calculating projections when an error is localized by the projection method is greater in PRNS than in RRNS. Hence, the decoding speed of PRNS is lower than RRNS, in the worst case (Fig. 5).

V. Approximation of the Rank of PRNS Number

In this section, we propose the method of PRNS to polynomial conversion based on AR of PRNS.

The suggested approach reduces the number of calculated projections and replaces the computationally complex operation of the division of long integers by taking the least significant bits. It reduces the complexity from $O(L \cdot \log L \cdot \log\log L)$ to $O(L)$. The rank of the polynomial in PRNS is determined according to the CRT, the value $A(x)$ can be calculated by the Eq. (2):

$$A(x) = \sum_{i=1}^{n} M_i(x) \cdot I_i(x) \cdot a_i(x) - r_A(x) \cdot M(x), \qquad (2)$$

where $r_A = \left\lfloor \sum_{i=1}^{n} \frac{I_i(x)}{m_i(x)} a_i(x) \right\rfloor$, $M(x) = \prod_{i=1}^{n} m_i(x)$, $M_i(x) = M(x)/m_i(x)$, $I_i(x) = |M_i^{-1}(x)|_{m_i(x)}$, for all $i = 1,2,\dots,n$.

$r_A(x)$ is a rank of $A(x)$, which is a polynomial that shows how many times the dynamic range of the PRNS can be increased.

From Eq. (2), it follows that to compute r_A, we need to perform an expensive operation of Euclidean division or use approximate value with N degree.

For efficient computing of the rank, we use an approach based on the approximate method and modular adder, which decreases the demanded accuracy of computations:

$$R_A(x) = \left\lfloor \sum_{i=1}^{n} k_i(x) a_i(x) / x^N \right\rfloor, \qquad (3)$$

where $k_i(x) = \lceil I_i(x) \cdot x^N / m_i(x) \rceil$.

Now, we derive Theorem 1 that shows how values N, $R_A(x)$, and $r_A(x)$ are related. It provides a theoretical basis of our approach.

Theorem 1. If $N = \max_i \deg m_i$, then $r_A = R_A$.

Proof. Let $k_i(x) = \lceil I_i(x) \cdot x^N / m_i(x) \rceil$, then $k_i(x)$ can be represented as: $k_i(x) = I_i(x) \cdot x^N / m_i(x) + \theta_i(x)$, where $0 \le \deg(m_i(x) \cdot \theta_i(x)) < \deg(m_i(x))$.

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

$$\sum_{i=1}^{n} k_i(x) \cdot a_i(x) = \sum_{i=1}^{n} \left(\frac{I_i(x) \cdot x^N}{m_i(x)} + \theta_i \right) \cdot a_i(x)$$

$$= \sum_{i=1}^{n} \frac{I_i(x) \cdot x^N}{m_i(x)} a_i(x) + \sum_{i=1}^{n} \theta_i(x) \cdot a_i(x)$$

$$= x^N \sum_{i=1}^{n} \frac{I_i(x)}{m_i(x)} a_i(x) + \sum_{i=1}^{n} \theta_i(x) \cdot a_i(x) \qquad (4)$$

Computing $R_A(x)$ by substituting Eq. (4) in Eq. (3), we obtain:

$$R_X(x) = \left\lfloor \sum_{i=1}^{n} \frac{I_i(x)}{m_i(x)} a_i(x) + \frac{\sum_{i=1}^{n} \theta_i(x) \cdot a_i(x)}{x^N} \right\rfloor \qquad (5)$$

From Eq. (5) and Eq. (2), it follows that

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

The sufficient condition is: $\deg(\lfloor \sum_{i=1}^{n} \theta_i(x) \cdot a_i(x) \rfloor) < N$. Since $\deg(\lfloor \sum_{i=1}^{n} \theta_i(x) \cdot a_i(x) \rfloor) \le \deg(\sum_{i=1}^{n} m_i(x)) \le \max_i \deg m_i(x)$, then $N = \max_i \deg m_i(x)$ is sufficient to hold the inequality $\deg(\lfloor \sum_{i=1}^{n} \theta_i(x) \cdot a_i(x) \rfloor) < N$. Theorem is proved. □

The algorithm complexity to calculate the rank of $r_A(x)$ based on the Theorem 1 is $O(d \cdot \log k)$. Since, the coefficients $k_i(x)$ are of degree d, we are able to compute the value of $A(x)$ efficiently.

Example 2. Let PRNS moduli set be $m_1(x) = x^4 + x + 1$, $m_2(x) = x^4 + x^3 + 1$ and $m_3(x) = x^4 + x^3 + x^2 + x + 1$. Dynamic range of PRNS is $M(x) = m_1(x) \cdot m_2(x) \cdot m_3(x) = x^{12} + x^9 + x^6 + x^3 + 1$.

We convert

$A(x) = x^4 \xrightarrow{PRNS} (x+1,\ x^3+1, x^3+x^2+x+1)$,
$B(x) = x^{11} \xrightarrow{PRNS} (x^3+x^2+x, x^3+x^2+1,\ x)$ from PRNS to polynomials using Theorem 1.

1. PRNS constants are computed once and kept in memory.

$$M_1(x) = \frac{M(x)}{m_1(x)} = x^8+x^4+x^2+x+1,$$
$$M_2(x) = \frac{M(x)}{m_2(x)} = x^8+x^7+x^6+x^4+1,$$
$$M_3(x) = \frac{M(x)}{m_3(x)} = x^8+x^7+x^5+x^4+x^3+x+1;$$
$$|M_1^{-1}(x)|_{m_1(x)} = 1\,, |M_2^{-1}(x)|_{m_2(x)} = x^2+x+1,$$
$$|M_3^{-1}(x)|_{m_3(x)} = x^2+x+1, N = 4;$$
$$k_1 = \left\lceil |M_1^{-1}(x)|_{m_1(x)} x^4/m_1(x) \right\rceil = 1,$$
$$k_2 = \left\lceil |M_2^{-1}(x)|_{m_2(x)} x^4/m_2(x) \right\rceil = x^2,$$
$$k_3 = \left\lceil |M_3^{-1}(x)|_{m_3(x)} x^4/m_3(x) \right\rceil = x^2+1$$

2. To calculate $A(x)$ and $B(x)$, we compute the sums:

$\sum_{i=1}^{3} k_i(x)\cdot a_i(x) = 1\cdot(x+1) + x^2\cdot(x^3+1) + (x^2+1)\cdot(x^3+x^2+x+1) = x^4+x^2$.
$\sum_{i=1}^{3} k_i(x)\cdot b_i(x) = 1\cdot(x^3+x^2+x) + x^2\cdot(x^3+x^2+1) + (x^2+1)\cdot x = x^5+x^4$.

Using Eq. (3), we obtain:

$$R_A(x) = \left\lfloor \frac{\sum_{i=1}^{3} k_i(x)\cdot a_i(x)}{x^4} \right\rfloor = \left\lfloor \frac{x^4+x^2}{x^4} \right\rfloor = 1,$$

$$R_B(x) = \left\lfloor \frac{\sum_{i=1}^{3} k_i(x)\cdot b_i(x)}{x^4} \right\rfloor = \left\lfloor \frac{x^5+x^4}{x^4} \right\rfloor = x+1.$$

Finally, we compute $A(x)$ and $B(x)$ using Eq. (2), and Theorem 1.

$$A(x) = \sum_{i=1}^{3} M_i(x)|M_i^{-1}(x)|_{m_i(x)} a_i(x) - R_A(x)\cdot M(x) = x^4$$
$$B(x) = \sum_{i=1}^{3} M_i(x)|M_i^{-1}(x)|_{m_i(x)} b_i(x) - R_B(x)\cdot M(x) = x^{11}$$

As shown in Example 2, the ranks of $A(x)$ and $B(x)$ are equal to the true value of the function. Computation of $\sum_{i=1}^{3} M_i(x)|M_i^{-1}(x)|_{m_i(x)} a_i(x)$ can be done in parallel with the computation of AR. Since their computational complexity are equal to the complexity of $R_A(x)\cdot M(x)$, they take approximately the same time to be performed. Therefore, the correct implementation of the decoding algorithm increases the speed of the algorithm.

VI. Method of Error Correction

Using Theorem 1, we propose a new method for data decoding based on AR and EC. From Eq. (2), it follows that $\sum_{i=1}^{n} k_i(x)\cdot a_i(x)$ can be represented in the form:

$$\sum_{i=1}^{n} k_i(x)\cdot a_i(x) = A(x) + r_A(x)\cdot M(x) \qquad (6)$$

Assume that an error $E(x) \xrightarrow{PRNS} \big(e_1(x), e_2(x), \dots, e_n(x)\big)$ occurred during computations, and the user (storage or communication) obtained the value $A(x)+E(x)$ instead of $A(x)$. Then by Eq. (6), we have:

$$\sum_{i=1}^{n} k_i(x)(a_i(x)+e_i(x)) = A(x)+E(x)+r_A(x)\cdot M(x)+r_E(x)\cdot M(x)$$

Without loss of generality, we assume that PRNS moduli set are in ascending order i.e. $\deg(m_1) \le \deg(m_2) \le \cdots \le \deg(m_n)$. Since in the proposed PRNS, k moduli are included in the dynamic range, and r is redundant (control moduli), where $k+r=n$, then

$$\deg(A(x)) < \deg\left(\prod_{i=1}^{k} m_i(x)\right) = \sum_{(i=1)}^{k} d_i$$

$$\prod_{i=1}^{k} m_i(x) = R(x)$$

The value of $\lfloor E(x)/R(x) \rfloor$ is $\lfloor (A(x)+E(x))/R(x) \rfloor$ or $\lfloor (A(x)+E(x))/R(x) \rfloor - 1$. If $\lfloor (A(x)+E(x))/R(x) \rfloor = 0$, then $E(x)=0$. Therefore, we can use the value $\lfloor (E(x)+A(x))/R(x) \rfloor$ to determine if the result is correct, if there is or there is no error.

Since an error is of the form: $E(x) = \beta(x) M_I(x)$, where $M_I(x) = \prod_{i\in I} m_i(x)$, $\beta(x)$ is polynomial, and I is a set of PRNS moduli that do not have an error. The value $\lfloor E(x)/R(x) \rfloor$ can be used as an error syndrome, where each $\lfloor E(x)/R(x) \rfloor$ is unambiguously defined by $E(x)$ and I.

We do some precomputations: sort the values of all possible errors $\lfloor E(x)/R(x) \rfloor$ in ascending order and map to $E(x)$. If we use a binary search in a sorted array of values $\lfloor E(x)/R(x) \rfloor$, we find $E(x)$ and set I in logarithmic of array size time.

Let $A'(x) = A(x)+E(x)$ and $A'(x) \xrightarrow{PRNS} (a(x)'_1, a(x)'_2, \dots, a(x)'_n)$. Using Eq. (2), we compute $\left\lfloor \frac{A'(x)}{R(x)} \right\rfloor =$

$$= \left\lfloor \frac{\sum_{i=1}^{n} |M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x)\cdot a'_i(x) - r_{A'}(x)\mathrm{M}(x)}{R(x)} \right\rfloor \qquad (7)$$

Since $M(x)/R(x)$ is polynomial then according to the property of Euclidean division, it can be taken out as a common factor, and the Eq. (7) can be rewritten:

$$\left\lfloor \frac{A'(x)}{R(x)} \right\rfloor = \left\lfloor \frac{\sum_{i=1}^{n} |M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x)\cdot a'_i(x)}{R} \right\rfloor - \frac{r_{A'}(x)M(x)}{R(x)} \qquad (8)$$

Let $M(x)/R(x) = F(x)$. Due to the fact that $0 \le \deg\left(\left\lfloor \frac{A'(x)}{R(x)} \right\rfloor\right) < \deg\big(F(x)\big)$, Eq. (8) is equivalent to

$$\left\lfloor \frac{A'(x)}{R(x)} \right\rfloor = \left\lfloor \left\lfloor \frac{\sum_{i=1}^{n} |M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x)\cdot a'_i(x)}{R} \right\rfloor - \frac{r_{A'}(x)M(x)}{R(x)} \right\rfloor_{F(x)} \qquad (9)$$

$$= \left|\left| \frac{\sum_{i=1}^{n} |M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x) \cdot a_i'(x)}{R(x)} \right|\right|_{F(x)}$$

From the definition of $M_i(x)$, it follows that for all $i = \overline{k+1, n}$, the value $M_i(x)/R(x)$ is polynomials. Then, according to the property of Euclidean division, polynomial can be taken out as a common factor, and the Eq. (9) can be rewritten:

$$\left\lfloor \frac{A'(x)}{R(x)} \right\rfloor = \left|\left| \frac{\sum_{i=1}^{k} |M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x) \cdot a_i'(x)}{R(x)} \right\rfloor + \sum_{i=k+1}^{n} \frac{|M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x)}{R(x)} \cdot a_i'(x) \right|_{F(x)} \quad (10)$$

In Eq. (10), the value of $\left\lfloor \sum_{i=1}^{k} |M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x) \cdot a_i'(x)/R(x) \right\rfloor$ can be computed according to the Theorem 1. For all $i = \overline{k+1, n}$, the value $|M_i^{-1}(x)|_{m_i(x)} \cdot M_i(x)/R(x)$ are precomputed constants.

VII. Configurable Model

Methods for detecting failures in distributed data storage and communication media are typically based on error correction codes, erasure codes, regeneration codes, and their modifications. However, if they do not allow to control computations, and do not have the property of homomorphism of arithmetic operations, they require extreme computational power for data processing and analysis.

In contrast to the existing methods, error correction codes in PRNS can effectively detect, correct errors, and control computations. They are fully homomorphic that makes them applicable for big data storage and processing. However, the error correction codes in PRNS have one significant drawback: the high complexity of detection and localization of errors. To solve this problem, we propose the approximate method to compute ranks of numbers, sort the array of relative values, and use the binary search. Our AR-PRNS approach reduces complexity from linear to the logarithmic of the power of the set of all possible numbers projections in PRNS. The model is configurable. To determine its parameters, the following optimization criteria should be taken into account: accuracy, scalability, reliability, confidentiality, security, and performance.

Scalability and reliability. If an error occurs, we use the cloud scalability property to maintain a certain level of reliability. We restore and correct the lost data chunks using AR-PRNS. We determine the parameters of the storage system for desirable reliability using the approach that determines the size of store data based on the redundancy coefficient from Section IV.A.

Confidentiality and security. To make the scheme asymptotically ideal, Chervyakov et al., 2019 [5] proposed to use the Asmuth-Bloom algorithm that provides a high degree of data security. However, this algorithm is not efficient for distributed storage, since it requires redundancy of data as big as in the Shamir scheme. In PRNS, the computational security of the system depends on the parameters k, n, m_i. It can be estimated using the formula $V/V_{\bar{I}}$, where V is the volume of data and $V_{\bar{I}}$ is the maximal amount of data that can be leaked to an unauthorized user. To estimate and adapt to the security level, we use the approach from Tchernykh et al., 2019 [2]. It estimates the risks of cloud conspiracy and DDoS attacks on cloud providers according to the chosen level of security.

Performance. An important issue is to minimize the computational costs associated with the implementation of arithmetic operations. Since they are modular, to select PRNS moduli, we have to take into account that moduli are irreducible polynomials over $GF(2)$. The approach from Sections IV.B and IV.C estimates the speed of data encoding and decoding. If there are no errors in the data storage system, then the speed of decoding increases significantly, because there is no need to perform the expensive operation of the error finding.

In Section V, we describe how to increase the speed of data decoding due to the optimization of the method of finding the error using AR. To compute Eq. (10), we use CRT. It takes, roughly, $\deg(F^2(x)) \approx \lfloor r/2 \rfloor^2 \cdot d^2$ bit operations, where $r = n - k$. Since the value of Eq. (10) is $F(x)$, to detect and localize an error, we need $\log_2 \deg(F) \cdot r \cdot d$ bit operations. In order to encode 1 Mb of data, $2^{23}(\log_2 \deg(F) \cdot r \cdot d + \lfloor r/2 \rfloor^2 \cdot d^2)/(k \cdot d) = 2^{23} \cdot (\log_2 \left(\left\lfloor \frac{r}{2} \right\rfloor \cdot d\right) \cdot r + \lfloor r/2 \rfloor^2 \cdot d)/k$ bit operations are required. Therefore, the speed of decoding is

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

This method increases the speed of data decoding from

$$V_D = \frac{2^7}{C_n^{k+\left\lfloor \frac{r}{2} \right\rfloor} \cdot d \cdot k} \text{ to } V_D = \frac{2^7 \cdot k}{\log_2 \left(\left\lfloor \frac{r}{2} \right\rfloor \cdot d\right) \cdot r + \lfloor r/2 \rfloor^2 \cdot d}$$

To detect and correct at least one error, k has to satisfy the inequality $k \leq n - 2$. Fig. 6 shows the data decoding speed depending on parameters that satisfy this condition. AR-PRNS shows better performance.

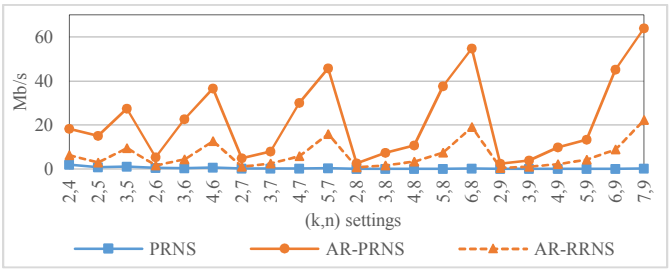

Fig. 6. The decoding speed (Mb/s) of PRNS, AR-PRNS and PRNS worst scenario of data recovery with the maximum number of errors versus settings (k, n), for $d = 8$ and $b = 8$.

To further increase of the data encoding and decoding speed, we can use multi-core processors or several VMs. If we choose the values d_i as multiples of the machine word, the effective implementation of modular arithmetic with distributed

operations and neural network of the finite ring can be used. It decreases the complexity of finding the remainder of the division from quadratic to linear.

VIII. CONCLUSIONS

In this paper, we introduce a configurable, reliable, and confidential distributed data storage scheme AR-PRNS with improved data redundancy, reliability, and encoding speed based on PRNS with a new method of error correction codes and secret sharing schemes. By changing the degree of PRNS polynomials and their number, these characteristics can be adapted to cope with different objective preferences, workloads, and storage properties. Our contribution is multi-fold.

Firstly, we introduce the concept of an AR of a polynomial. It reduces the computational complexity of the decoding from RNS to polynomial and its coefficients size. Based on the proposed approximation rank, we design a new method of data decoding. AR-PRNS reduces complexity from $O(L^2)$ down to $O(L \cdot \log L)$. Using properties of the approximate value and arithmetic properties of RNS, we introduce a method for error detection, correction, and controlling computational results.

Secondly, we provide a theoretical basis to calculate data redundancy, the speed of encoding/decoding, and configure parameters to cope with different objective preferences, workloads, and cloud properties. We show how to configure security, reliability, and reduce overhead of data storage by appropriate selection of PRNS parameters.

However, further study is required to assess its actual efficiency and effectiveness in real systems. This will be the subject of future work providing a comprehensive experimental study of multi-objective optimization with real cloud providers.

ACKNOWLEDGEMENTS

The work is partially supported by Russian Foundation for Basic Research (RFBR) 18-07-01224, Russian Federation President Grants MK-341.2019.9 and SP-2236.2018.5.

REFERENCES

[1] S. Ghemawat, H. Gobioff, S.-T. Leung, "The Google file system.," ACM SIGOPS Oper. Syst. Rev., pp. 29-43, 2003.

[2] A. Tchernykh, V. Miranda-López, M. Babenko, F. Armenta-Cano, G. Radchenko, A. Y. Drozdov, & A. Avetisyan, "Performance evaluation of secret sharing schemes with data recovery in secured and reliable heterogeneous multi-cloud storage," Cluster Comput., pp. 1-13, 2019.

[3] M. Gomathisankaran, A. Tyagi, K Namuduri, "HORNS: A homomorphic encryption scheme for Cloud Computing using Residue Number System," 45st Annual Conference on Information Sciences and Systems, IEEE, pp. 1–5, 2011.

[4] A. Celesti, M. Fazio, M. Villari, A. Puliafito, "Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems," J. Netw. Comput. Appl., vol. 59, pp. 208–218, 2016.

[5] N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-López, J. Cortés-Mendoza, "AR-RRNS: Configurable Reliable Distributed Data Storage Systems for Internet of Things to Ensure Security," Futur. Gener. Comput. Syst., vol. 92, pp. 1080-1092, 2019.

[6] H. Y. Lin, W. G. Tzeng, "A secure erasure code-based cloud storage system with secure data forwarding," IEEE T. Parall. Distr., vol. 23, no. 6, pp. 995-1003, 2012.

[7] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright, K. Ramchandran, "Network coding for distributed storage systems," IEEE T. Inform. Theory, vol. 56, no. 9, pp. 4539-4551, 2010.

[8] C. Gentry, "Computing arbitrary functions of encrypted data," Commun. ACM, vol. 53, no. 3, pp. 97-105, 2010.

[9] A. Tchernykh, M. Babenko, N. Chervyakov, V. Miranda-López, V. Kuchukov, J. M. Cortés-Mendoza, M. Deryabin, N. Kucherov, G. Radchenkoc, & A. Avetisyan, "AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage," Int. J. Approx. Reason., vol. 102, pp. 60-73, 2018

[10] A. Tchernykh, U. Schwiegelsohn, E. Talbi, and M. Babenko, "Towards Understanding Uncertainty in Cloud Computing with risks of Confidentiality, Integrity, and Availability," J. Comput. Sci., 2016.

[11] X. Chen, Q. Huang, "The data protection of MapReduce using homomorphic encryption," Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, IEEE, pp. 419–421, 2013.

[12] S. Srisakthi, A.P. Shanthi, "Towards the Design of a Secure and Fault Tolerant Cloud Storage in a Multi-Cloud Environment," Information Security Journal: A Global Perspective, vol. 24, no. 4-6, pp. 109-117, 2015.

[13] D. Hubbard, M. Sutton, "Top threats to cloud computing v1. 0. Cloud Security Alliance," https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, 2010 (accessed 01.02.19)

[14] A. Tchernykh, M. Babenko, V. Miranda-López, A. Y. Drozdov, & A. Avetisyan, "WA-RRNS: Reliable Data Storage System Based on Multi-cloud," 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, pp. 666-673, 2018.

[15] C. Asmuth, J. Bloom, "A modular approach to key safeguarding," IEEE T. Inform. Theory, vol. 29, no. 2, pp. 208–210, 1983.

[16] M. Mignotte, "How to share a secret," Workshop on Cryptography, pp. 371-375, 1982.

[17] J. Lin, W.H. Chung, Y.S. Han, "Novel polynomial basis and its application to Reed-Solomon erasure codes," An S. Fdn. Co., pp. 316-325, 2014.

[18] J. Liu, K. Huang, H. Rong, H. Wang, M. Xian, "Privacy-preserving public auditing for regenerating-code-based cloud storage," IEEE T. Inf. Foren. Sec., vol. 10, no. 7, pp. 1513-1528, 2015.

[19] H.C. Chen, P.P. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation," IEEE T. Parall. Distr., vol. 25, no. 2, pp. 407–416, 2014.

[20] R.L. Rivest, L. Adleman, M.L. Dertouzos, "On data banks and privacy homomorphisms," Nato. Adv. Sci. I F-Com, vol. 4, no. 11, pp. 169-180, 1978.

[21] A. Trepacheva, L. Babenko, "Known plaintexts attack on polynomial based homomorphic encryption," Proceedings of the 7th International Conference on Security of Information and Networks, ACM, p. 157, 2014,

[22] A. Halbutogullari & C. K. Koc, "Parallel multiplication in GF (2 k) using polynomial residue arithmetic," Design. Code. Cryptogr., vol. 20, no. 2, pp. 155-173, 2000.

[23] J. Chu & M. Benaissa, "Error detecting AES using polynomial residue number systems," Microprocess Microsy., vol. 37, no. 2, pp. 228-234, 2013.

[24] Geekbench Browser. https://browser.primatelabs.com (accessed 01.02.19)

[25] N. I. Chervyakov, P. A. Lyakhov, M. G. Babenko, A. I. Garyanina, I. N. Lavrinenko, A. V. Lavrinenko, M. A. Deryabin, "An efficient method of error correction in fault-tolerant modular neurocomputers," Neurocomputing, vol. 205, pp. 32–44, 2016.