IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 6, 2020, accepted October 14, 2020, date of publication October 21, 2020, date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3032655

2Lbp-RRNS: Two-Levels RRNS With Backpropagation for Increased Reliability and Privacy-Preserving of Secure Multi-Clouds Data Storage

VANESSA MIRANDA-LÓPEZ[1], ANDREI TCHERNYKH[1,2,3], (Member, IEEE), MIKHAIL BABENKO[3,4], ARUTYUN AVETISYAN[3], (Senior Member, IEEE), VICTOR TOPORKOV[5], AND ALEXANDER YU DROZDOV[6]

[1]Computer Science Department, CICESE Research Center, Ensenada 22860, México
[2]School of Electronic Engineering and Computer Science, South Ural State University, Chelyabinsk 454080, Russia
[3]Ivannikov Institute for System Programming, Moscow 109004, Russia
[4]Department of Computational Mathematics and Cybernetics, North-Caucasus Federal University, Stavropol 355017, Russia
[5]Department of Computing Technologies, National Research University, MPEI, Moscow 111250, Russia
[6]School of Radio Engineering and Computer Technology, Moscow Institute of Physics and Technology, Moscow 141701, Russia

Corresponding author: Andrei Tchernykh (chernykh@cicese.edu.mx)

This work was supported in part by the Russian Foundation for Basic Research (RFBR), under Project 20-37-70023.

ABSTRACT Cloud storage as service is the mainstream technology used to retain digital data. However, there are significant risks for confidentiality, integrity, and availability violation associated with the loss of information, denial of access, technical failures, etc. In this article, we propose a two-level 2Lbp-RRNS scheme based on a Redundant Residue Number System with a backpropagation and hamming distance mechanisms for increasing reliability of a configurable and secure multi-cloud data storage. We provide a theoretical analysis of the 2Lbp-RRNS solution as an extension of the classical 2L-RRNS and a variant of fully homomorphic encryption for privacy-preserving, parallel processing, and scalability. We formulate, explain, and prove its main properties to extend existing knowledge within the limits of the critical bounding RRNS assumptions. We show that 2Lbp-RRNS can identify and recover more errors than traditional 2L-RRNS. We provide the upper bounds of the traditional threshold 2L-RRNS and our solution to estimate the number of detectable and correctable errors. We study various data access scenarios and show that it detects 1.58x and corrects 3.37x more errors than 2L-RRNS, on average. We also provide efficient implementations of encoding and decoding algorithms MRC8, and MRC16 based on the Mixed-Radix system, Finite Ring Neuronal Network, and signed binary window method. We evaluate encoding/decoding speeds using three algorithms: Mignotte, MRC8, and MRC16. The experimental system includes seven cloud storages: DropBox, GoogleDrive, OneDrive, Sharefile, Box, Egnyte, and Salesforce. To assess the efficiency of the system on real data, we vary scenarios of the first and second levels. The results show that our solution outperforms MRC8 by 2.53x (1.78x), and Mignotte by 4.83x (11.43x) for the encoding (decoding) speed, respectively.

INDEX TERMS Cloud storage, reliability, residue number system, secret sharing scheme, uncertainty, security.

I. INTRODUCTION

A secure and fault-tolerant multi-cloud storage has to prevent information from unauthorized access, use, disclosure, disruption, modification, etc. Confidentiality, integrity, and availability must be preserved even in the presence of failures, deliberate, as well as accidental threats.

To this end, data encryption systems, homomorphic encryption, error correction codes, secured sharing schemes, etc. are widely used.

The associate editor coordinating the review of this manuscript and approving it for publication was Oussama Habachi.

The multi-cloud environment has a dynamic nature with risks of the loss of information, denial of access, information leakage, collusion, and data security breaches that are difficult to predict and anticipate in advance. These types of non-stationarity are one of the main issues in the design of reliable storage capable of mitigating their consequences.

Many potential users are not eager to employ cloud storage services because of the risk of data disclosure. Data encryption mechanisms are not sufficient to deal with security and privacy protection. When the data must be processed, decryption is necessary, which falls on the initial problem of data vulnerability.

Recent developments in the theoretical cryptography address a set of techniques such as Homomorphic Encryption (HE), Fully Homomorphic Encryption (FHE), Some-what Homomorphic Encryption (SHE), and Secure Multi-Party Computation (MPC). These cryptosystems allow applying certain mathematical operations directly to the ciphertext and safely delegate the processing of data to an untrusted remote party. It guarantees that the remote party learns neither the input nor output of the computation.

In the last decade, there is considerable interest in using a widely known and studied number theory system such as the Redundant Residue Number System (RRNS) as a variant of FHE. The study of error correction codes based on RRNS is related to three main issues.

The first goal is to increase the data encoding and decoding speeds from a weighted number system to RRNS and vice versa [14], [16], [39], [46]. The second aim is related to the reduction in the computational complexity of the error correction algorithm [17], [37], [38]. The third objective is associated with an increase in the number of correctable errors in RRNS [23], [37], [38].

In this article, we propose a two-level RRNS backpropagation scheme (2Lbp-RRNS) with increased reliability for a configurable, reliable, and secure multi-clouds data storage. It has a scalable data access structure. In contrast to the classical solutions, it can restore the data with less available shares than the state-of-the-art approaches and provide privacy and security preserving.

This paper is organized as follows. Section II reviews distributed storage systems, error correction codes, and security. Section III discusses one level and two-level residue number systems and their properties. Section IV presents the theoretical support of the 2Lbp-RRNS solution. Section V describes three algorithms for encoding-decoding. Section VI focuses on the access speed of cloud systems and provides performance analysis of encoding/decoding speeds. The conclusions and future work are discussed in the last Section VII.

II. RELATED WORK

In this section, we discuss the distributed data management technologies.

A. DISTRIBUTED STORAGE SYSTEM

A variety of approaches can be used to construct a distributed system for storage and processing. Several of them are based on the cloud and grid computing paradigms [1]. These infrastructures have common characteristics but also principal differences.

The use of clouds for data storing requires several factors, such as security, reliability, and scalability, under limited Internet connection bandwidth [2], [3].

To provide quick access to distributed data and ensure a high degree of reliability, availability, and scalability. Chang *et al.* [4] proposed Bigtable system based on the replication of not encrypted data without providing privacy and data security.

An alternative mechanism is Hadoop and MapReduce based on splitting the data set into independent chunks that are processed in parallel and reducing them [5]. However, as shown in [6], its main drawback is the low efficiency.

Not relational databases (NoSQL) that take into account the heterogeneity of unstructured data become popular [7]. However, the two most popular NoSQL databases, Cassandra and MongoDB, have problems with data security and privacy [8].

Distributed Data Base (DDB) stores data on various sites of a computer network and uses logic to organize the set of data [9]. There are two ways to construct DDBs. The top-down approach takes a database and distributes it over various sites, while the bottom-up approach unites distinct databases with one interface.

The main field of application of DDBs is structured data storage. Therefore, it is not applicable to arbitrary data sets, such as Big data.

Content Delivery Network (CDN) [10] is a set of servers that cache the data, satisfy the client requests to the database, and reduce the workload of origin servers. We can state the following principles of CDN: load balancing, bandwidth conservation, and time efficiency. However, CDNs are not widely used in practice since they are not flexible.

The main principles of P2P Network [11] are scalability and reliability achieved by decentralized structure and redundancy, resource sharing, and anonymity. P2P networks are efficient in providing fast access to files to a group of peers. Nevertheless, most P2P networks do not allow integrated computations and serve as data distribution environment.

B. ERROR CORRECTION CODE

The basis of the error correction codes is Hamming's idea of adding additional data that helps to detect and correct errors [12]. Depending on the areas of application, approaches to building error correction systems are varying.

For storage systems, the balance between reliability and data redundancy is important since data redundancy affects the amount of stored data, and, therefore, costs. The most expensive mechanism for ensuring reliability is data replication.

From another perspective, error correction codes and their modifications, such as erase codes and regeneration codes, can provide greater reliability with the same redundancy than replication [13]. An important issue when choosing an error

correction code is what is the maximum number of errors that it can detect and correct for given data redundancy [14].

From the data processing technology point of view, error correction codes consider blocks [13]. The advantages of block codes are the ability to correct errors located in one block. The disadvantage is the smaller number of free-standing errors that can be corrected compared to ultra-precise ones.

An alternative solution is modular error correction codes constructed using the RRNS [15]. An additional advantage of RRNS for the design of distributed storage systems is that it is a secret sharing scheme that provides data security [16].

However, there are two main problems: the computational and memory complexity of the data decoding algorithm.

For the decoding phase, there are two widely used methods: the projection and syndrome. The projection method is a universal method that allows us to detect and correct an error with any RNS moduli [17]. Its disadvantage is the exponential computational complexity depending on the number of correctable errors [14]. The syndrome method reduces computational complexity to quadratic, but it requires storing large tables of constants in memory [18].

Two approaches are used to reduce the required memory. The first one is the use of auxiliary functions as an error syndrome; for example, the rank of a number [14]. The second approach involves the imposition of additional restrictions on RNS modules when other error correction codes provide the reliability of the storage of individual modules.

The second problem is related to increasing the number of correctable errors with the same encoding parameters [18]. For this problem, the next solutions are applied. The first one is to unbalance RNS moduli when one or more RNS moduli are several times larger than the rest of the moduli [19]. The disadvantage of this approach is related to the case when errors occurred in the largest RNS modulus so that the amount of incorrect data is significant and may exceed the threshold.

An alternative approach is to use 2L-RRNS. Given that the second-level modules act as an independent error correction code, this approach allows you to correct a larger number of errors [20], [21], as well as reduce the computational complexity of decoding [22], [23].

C. DISTRIBUTED STORAGE SECURITY

One of the main goals of cloud technologies is to provide access to data at any time. Users get the opportunity to use cloud services without involving specialists with fairly simple and intuitive interfaces. Classical approaches to ensuring the integrity of data are based on methods of identical or non-identical redundancy (storing copies or storing histories, respectively). Strategies for ensuring reliable data storage are usually chosen for each particular system based on a multiple factor analysis [24].

The task of ensuring data integrity is complex. It includes not only data integrity control but also its maintenance and the data recovery if it is violated for any reason.

There are various ways to solve the problem of monitoring and ensuring data integrity. One of them is calculating the checksums and comparing them with the reference checksums [25], [26]. Other methods are based on the use of cryptographic techniques, key and keyless hashing, and electronic signature [27]–[29]. The disadvantage of these methods is the inability to ensure integrity without extra data for recovery mechanisms.

The redundancy is a widespread solution for ensuring data integrity. Hardware and software implementations of the Redundant Array of Independent Disks (RAID) [30], [31], duplication methods, encoding [32], etc. The disadvantage of these methods is the inability to control data as information-theoretically secure and high redundancy.

Some methods can control integrity by comparing the reference values and calculated hash codes (checksums) when requesting the data. However, the lack of mechanisms for their recovering does not allow ensuring integrity.

On the contrary, other methods ensure data integrity by restoring it, for example, from a backup copy. However, their practical implementation without the possibility of preliminary data integrity control is ineffective.

Separate methods allow monitoring and ensuring data integrity, however, at the cost of high redundancy. One of the solutions is to consistent use of cryptographic transformations and backup technology.

An alternative way is to use RRNS, which on the one hand, is the error correction code, which allows restoring the result when an error occurs, and on the other hand, is a secret sharing scheme that ensures data security [14]. Tchernykh *et al.* [33] show how the security of stored data depends on the RRNS parameters. 2L-RRNS can increase the number of detected and correctable errors compared to 1L-RRNS [23]. Therefore, 2L-RRNS can ensure the reliability and integrity of stored data better than 1L-RRNS (see Sections III, IV).

D. PRIVACY-PRESERVING FOR CLOUD COMPUTING

RNS, as a version of Homomorphic Encryption (HE), can be used for privacy-preserving in cloud computing. It allows calculating functions over encrypted data without knowing the moduli set. The owner can restore the real result from the outcome of the calculations on the corresponding encrypted data. This characteristic makes RNS a promising solution for securely delegating cloud computing servers over sensitive client data.

After Rivest *et al*. [49] introduce the concept of HE, cryptographers proposed and analyzed many different homomorphic cryptosystems

Brickell and Yacobi [50] and Paillier [51] proposed a partially HE with one arithmetic operation (addition or multiplication).

Gentry [52] developed the first FHE circuitry based on ideal lattices. It performs arbitrary calculations on encrypted data with an unlimited number of homomorphic multiplications and additions.

All these cryptosystems can be divided into two categories. The first one contains public-key FHE schemes based on noise input [52]–[55]. These cryptosystems are based on the Gentry C technique improving its performance [52].

The *Gentry C Cryptosystem* is safe and resistant to various attacks but has high computational complexity, which does not allow it to be used in practical applications. To reduce the computational complexity, [33] proposed an analog of the *Gentry C* scheme based on the RNS. Tchernykh *et al.* [16] showed that RNS provides the necessary level of security and reduces the computational complexity of encoding and decoding.

The second category includes symmetrical FHE cryptosystems that do not use noise. Examples of such cryptographic algorithms are [56], [57], and others based on diagonal matrices.

The use of diagonal matrices reduces the computational complexity of coding and decoding algorithms but reducing data security. Yagisawa [58], [59] present FHE based on octonion algebra; however, as shown in [60] is unsafe. An alternative mechanism is the use of RNS. As demonstrated by [14], [16], RNS can reduce the computational complexity of encoding and decoding algorithms and provide the necessary security level.

In the next sections, we show how to improve the technical characteristics of RNS through the use of 2Lbp-RRNS and backpropagation.

III. TWO-LEVEL RRNS

Let us introduce the following notations.

Notation	Description
D, S	Original, secret data, binary number, weighted number system
$\bar{S} \overset{RNS}{\longleftarrow} S + E$	Representation of S with error in 2L-RRNS
$\tilde{S}$	Representation of S in 2L-RRNS
$size(D)$	Size of original data D
D_n	n lower bits of the number D
T_E, T_D	Data encryption and decryption time
t_{dow}, t_{up}	Downloading, uploading time from clouds
V_u, V_d	Uploading, downloading velocity $V_u = \frac{size(D)}{T_E + t_{up}}, V_d = \frac{size(D)}{T_D + t_{dow}}$
$\bar{I}$	Tuple of residues with an error
I_D	Subset $\{1, .., n_1\}$, power is equal to k_1
I_E	Subset $\{1, .., n_1\}$, power is equal to $\lfloor (k_1 + n_1)/2 \rfloor$
$\bar{I}_i$	Tuple of residues with an error
N_D^{2L}	Number of detected errors in 2L-RRNS
N_E^{2L}	Number of corrected errors in 2L-RRNS
N_D^{2Lbp}	Number of detected errors in 2Lbp-RRNS
N_E^{2Lbp}	Number of corrected errors in 2Lbp-RRNS
N_{Dl}	Number of detected errors with knowing error localization
N_{El}	Number of corrected errors with knowing error localization
	First Level
n_1	Number of moduli on the first level
$k_1 \leq n_1$	Threshold value on the first level for the secret sharing scheme
$r_1 = n_1 - k_1$	Number of control (redundant) moduli
$p_{1,i}$	i-th RNS moduli on the first level
$P = \prod_{i=1}^{k_1} p_{1,i}$	Legal dynamic range RRNS on the first level and $S \in [0, P)$
$\bar{P} = \prod_{i=1}^{n_1} p_{1,i}$	$[0, \bar{P} - 1]$ full range
$S_i = \lvert S \rvert_{p_{1,i}}$	remainder of the division S by modulo $p_{1,i}$
	Second Level
$n_{2,i}$	Number of moduli $p_{2,i,1}, p_{2,i,2}, \ldots, p_{2,i,n_{2,i}}$ used to representation S_i for each $i = \overline{1, n_1}$.
$k_{2,i} \leq n_{2,i}$	Threshold value for secret sharing scheme used in the representation S_i
$r_{2,i} = n_{2,i} - k_{2,i}$	Number of control (redundant) RRNS moduli used in the representation S_i
$p_{2,i,j}$	RRNS modulo used in the representation S_i for each $i = \overline{1, n_1}$ and $j = \overline{1, n_{2,i}}$
$M_i = \prod_{j=1}^{k_{2,i}} p_{2,i,j} \geq p_{1,i}$	Dynamic range of RRNS definite moduli set $\{p_{2,1}, p_{2,2}, \ldots, p_{2,n_i}\}$ and $S_i \in [0, p_{1,i})$ for each $i = \overline{1, n_1}$
$S_{i,j} = \lvert S_i \rvert_{p_{2,i,j}}$	Remainder of the division S_i by modulo $p_{2,i,j}$ for each $i = \overline{1, n_1}$ and $j = \overline{1, n_{2,i}}$

A. 1L-RRNS

Let $p_{1,1}, p_{1,2}, \ldots, p_{1,n_1}$ are pairwise coprime numbers used as moduli set of 1L-RRNS, $n_1 = k_1 + r_1$. 1L-RRNS legal dynamic range is defined as $P = \prod_{i=1}^{k_1} p_{1,i}$.

Data S is a number in the Binary-Weighted Number System, where $S \in [0, P)$. S is represented in RRNS as a tuple

$$S \xrightarrow{RNS} (S_1, S_2, \ldots, S_n),$$

where $S_i = \lvert S \rvert_{p_{1,i}}$ represents the remainder of the division of S by $p_{1,i}$.

In 1L-RRNS settings (k_1, n_1), if the number of control moduli is r_1, then, according to 1L-RRNS property, the system can detect $r_1 = n_1 - k_1$ and correct $\lfloor r_1/2 \rfloor$ errors.

For error isolation and correction, projection methods are used, where the number of calculated projections grows exponentially depending on the r_1. As a consequence, 1L-RRNS is impractical without significant optimization.

Celesti *et al.* [15] proposed 1L-RRNS for reliable and scalable cloud storage systems. Operations on residues can be accomplished separately and concurrently, which makes the computations simpler and faster. Redundancy of residues allows the building system with multiple error detection and correction.

Since the representation of numbers in 1L-RRNS can be seen as a secret sharing scheme, we can obtain computationally secure data storage.

Gomathisankaran *et al.* [34] studied fully homomorphic cipher systems based on secret sharing in RNS. However, it should be noted that it is not practical to use RNS moduli set as the secret keys. It leads to high redundancy and resource-intensive decoding that can be more complex than the original problem.

Cheon *et al.* [35] offered an alternative way of constructing a homomorphic encryption system in RNS. They proposed a generalization of DGHV (Dijk, Gentry, Halevi, and Vaikuntanathan) algorithm, which improves characteristics of computational complexity and redundancy. This scheme is based on the ideas of the secret sharing scheme in 1L-RRNS [36]. The proposed algorithm has high redundancy compared with schemes in the classic 1L-RRNS.

To determine the problems in the data storage and data processing, we use properties of error detection and correction in 1L-RRNS, considered by [37]. Modification and improvements of detection and error correction in the 1L-RRNS are discussed in [14], [38].

The common issue for the majority of the proposed works is to detect and correct one error. When reliability is provided for a single computer, the detection and correction of a single error are sufficient. However, when we consider big data, it is necessary to have efficient algorithms for detecting and correcting several errors.

The 1L-RRNS scheme for big data storage provides security, reliability, and scalability. It has properties of error correction codes and two cryptographic primitives: secret sharing schemes and homomorphic encryption, which makes it useful for data processing in the encrypted form.

B. 2L-RRNS DATA ENCODING AND DECODING

The constructive version of the Chinese Remainder Theorem (CRT) gives a method to recover S from RRNS representation. In RRNS settings (k_1, n_1), S can be recovered from any k_1 remainders from n_1.

To guarantee the required dynamic range, we can use either a large number of small moduli or several large moduli. For small moduli, converting numbers from RNS to a binary number system is more computationally complex. They should have effective software and hardware implementations of the basic modular operations.

2L-RRNS is a recursive extension of the classical 1L-RRNS. On the first level, n_1 moduli $p_{1,1}, p_{1,2}, \ldots, p_{1,n_1}$ are used to calculate shares $S_1, S_2, \ldots, S_{n_1}$. On the second level, each S_i is transformed into the set of residuals $S_{i,j} = |S_i|_{p_{2,i,j}}$ by its own moduli set $p_{2,i,1}, p_{2,i,2}, \ldots, p_{2,i,n_{2,i}}$ (Fig.1).

$$M_i = \prod_{j=1}^{k_{2,i}} p_{2,i,j} \geq p_{1,i}$$

S_i satisfies the condition $S_i < p_{1,i}$, for all $i = \overline{1, n_1}$.

From the CRT, it follows that for a one-to-one mapping between $S_i \in [0, p_{1,i})$ and $\widetilde{S_i} = (S_{i,1}, S_{i,2}, \ldots, S_{i,n_{2,i}})$, it is necessary and sufficient that $M_i \geq p_{1,i}$ for each $i = \overline{1, n_1}$.

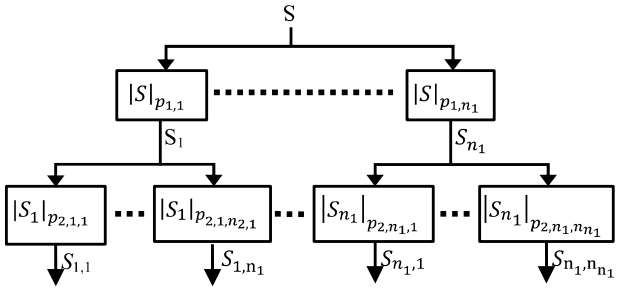

FIGURE 1. 2L-RRNS encoding.

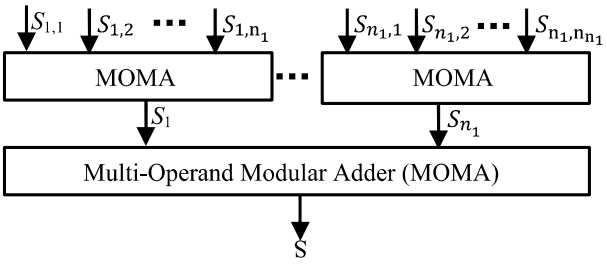

FIGURE 2. 2L-RRNS decoding.

Figure 2 shows a data decoding scheme. S_i for $i = \overline{1, n_1}$ are restored from the corresponding $S_{i,j}$, then S is restored from $S_1, S_2, \ldots, S_{n_1}$.

Multi-Operand Modulo Addition (MOMA) is an algorithmic primitive that accepts n_1 operands $S_1, S_2, \ldots, S_{n_1}$, with $0 \leq S_i < p_{1,i}$ for each $i = \overline{1, n_1}$ and computes the residue of their sum taken modulo P. That is, it computes original data S as

$$S = \left| w_i S_1 + w_2 S_2 + \ldots + w_{n_1} S_{n_1} \right|_P,$$

where $w_i = P_i \cdot \left| P_i^{-1} \right|_{p_{1,i}}$ and $P_i = P/p_{1,i}$, for all $i = \overline{1, n_1}$ [39].

IV. TWO-LEVELS RRNS WITH BACKPROPAGATION

In this section, we present the 2Lbp-RRNS solution as an extension of the classical 2L-RRNS. We provide its theoretical analysis, formulate, explain, and prove its main properties. We show how the reliability and performance of the system depend on the (k, n) parameters on each level.

We discuss the backpropagation and Hamming Distance mechanisms that allow to increase the number of detected and corrected errors.

We provide and compare the upper bounds of detectable and correctable errors for both schemes to estimate the benefits of the proposed solution.

A. 2L-RRNS

2L-RRNS uses the error correction code of the classical 1L-RRNS on each level. It can correct the value of S_i, if and only if, the number of errors is less or equal to $\lfloor r_{2,i}/2 \rfloor$. In all other cases, it can detect that S_i is incorrect (similar to 1L-RRNS), but it cannot recover it.

Algorithm 1 2L-RRNS Error Corrections

Input: $(k_1 n_1)\,(k_1, n_1)\,, \left(k_{2,1}, n_{2,1}\right), \cdots, (k_{2,n_1}, n_{2,n_1})$
$S_1 \xrightarrow{RNS} \left(S_{1,1}, S_{1,2}, \ldots, S_{1,n_{2,1}}\right), S_2 \xrightarrow{RNS}$
$(S_{2,1}, S_{2,2}, \ldots, S_{2,n_{2,2}}), \cdots,$
$S_{n_1} \xrightarrow{RNS} (S_{n_1,1}, S_{n_1,2}, \ldots, S_{n_1,,n_{2,n_1}})$
$\left(p_{1,1}, \ldots, p_{1,n_1}\right), \left(p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}\right), \ldots,$
$(p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$
Output: $S, flag, \left(S_1, S_2, \ldots, S_{n_1}\right)$.
$flag = 0$, if there are no errors, $flag = 1$ if errors are detected and corrected, $flag = -1$, if errors are detected but not corrected.

1. flag = 0
2. For $i = 1$ to n_1 do:
   2.1. $S'_i = CRTtoBin((S_{i,1}, S_{i,2}, \ldots, S_{i,n_{2,i}}))$
   2.2. If $S'_i \geq P_{2,i}$ then:
      2.2.1. $temp = ProRRNS\left(S'_i, \left(p_{2,i,1}, p_{2,i,2}, \ldots, p_{2,i,n_{2,i}}\right) k_{2,i}, val\right)$
      2.2.2. $flag = flag + temp$;
      2.2.3. If $temp == 0$ then $S_i = val$ else $S_i = -1$
3. If flag == 0 then:
   3.1. $S = CRTtoBin((S_1 S_2, \ldots, S_{n_1}))$
4. else
   4.1 If $flag \leq \lfloor (n_1 - k_1)/2 \rfloor$ then:
      4.4.1. flag = 1
      4.4.2. $S = CRTtoBin((S_1 S_2, \ldots, S_{n_1}))$
   4.2 else $flag = -1$
5. **return** $S, flag, (S_1, S_2, \ldots, S_{n_1})$

Let us consider Algorithm 1. "2L-RRNS error corrections" that applies the general projection method of the 1L-RRNS on each level. We do not use the syndrome method. The amount of memory required by the syndrome method in 1L-RRNS is increased exponentially, depending on the number of errors to be corrected [14]. Thus, to store in memory a table of constants for the second level requires increasing the memory by n_1 times, which is not reasonable.

The *CRTtoBin* function converts numbers from RRNS to a binary number system using CRT. The *ProRRNS* function calculates the projection value of RRNS.

Let us show the number of detected N_D^{2L} and corrected N_E^{2L} errors of 2L-RRNS.

Theorem 1: 2L-RRNS can detect N_D^{2L} errors and correct N_E^{2L} errors, where

$$N_D^{2L} = \sum_{i=1}^{n_1} (n_{2,i} - k_{2,i}), \quad N_E^{2L} = \sum_{i=1}^{n_1} \left\lfloor \frac{n_{2,i} - k_{2,i}}{2} \right\rfloor.$$

Proof: The theorem is proved in Barati *et al.* [23].

For a better understanding of the 2Lbp-RRNS properties, let us first consider special cases of 1L-RRNS and 2L-RRNS, when we know the localization of errors. We use special subscripts for detection *Dl* and correction *El* cases with error localization.

Lemma 1: For (k_1, n_1) 1L-RRNS, if we know the localization of k_1 correct S_i, 1L-RRNS can restore S.

Proof: Without loss of generality, let the correct values be $S_{i_1}, S_{i_2}, \ldots, S_{i_{k_1}}$, then using the CRT, 1L-RRNS can restore S using the formula: $S = \left| w_1 S_{i_1} + w_2 S_{i_2} + \ldots + w_{k_1} S_{i_{k_1}} \right|_{P_I}$, where $P_I = \prod_j^{k_1} p_{i_j}$ and $w_j = \frac{P_I}{p_{i_j}} \cdot \left| \frac{p_{i_j}}{P_I} \right|_{p_{i_j}}$. The lemma is proved.

Corollary 1: For (k_1, n_1) 1L-RRNS, if we know the localization of k_1 correct S_i, 1L-RRNS can correct $N_{El}^{1L} \leq r_1 = n_1 - k_1$ errors.

Proof:

(a) When we know $S_{i_1}, S_{i_2}, \ldots, S_{i_{k_1}}$ with no errors, then the condition of Lemma 1 is satisfied. Therefore, we can restore the result by correcting $r_1 = n_1 - k_1$ errors.

(b) When there exist $k_1 - 1$ values with no errors, given that (k_1, n_1) is a threshold secret sharing scheme, it follows that we cannot restore the true value of S.

Hence, if there is an algorithm that can determine which of S_i is correct, we can correct no more than $r_1 = n_1 - k_1$ errors. The corollary is proved.

Lemma 2: If there is an algorithm that can determine which of $S_{i,j}$ is correct, then the 2L-RRNS can correct

$$N_{El}^{2L} \leq \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i}$$

errors.

Proof: Without loss of generality, let us assume that $k_{2,1} \leq k_{2,2} \leq \ldots \leq k_{2,n_1}$.

Case 1: Let us assume that $r_{2,i}$ errors occurred in $\widetilde{S^i}$, for all $i \in \overline{1, k_1}$, with total errors $\sum_{i=1}^{k_1} r_{2,i} = \sum_{i=1}^{k_1} (n_{2,i} - k_{2,i})$, and $n_{2,i}$ errors for all $i \in \overline{k_1 + 1, n_1}$, with total errors $\sum_{i=k_1+1}^{n_1} n_{2,i}$ Following Lemma 1, we can restore the true value of S_i for all $i \in \overline{1, k_1}$. Therefore, the condition of Lemma 1 is satisfied, and we can restore the true value of S correcting N_{El}^{2L} errors.

$$\begin{aligned} N_{El}^{2L} &\leq \sum_{i=1}^{k_1} (n_{2,i} - k_{2,i}) + \sum_{i=k_1+1}^{n_1} n_{2,i} \\ &= \sum_{i=1}^{k_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i} + \sum_{i=k_1+1}^{n_1} n_{2,i} \\ &= \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i} \end{aligned}$$

Case 2: If we add one error more to $\widetilde{S^i}$ (without loss of generality, we will consider that the error is added to S_j representations), for $i \in \overline{1, k_1}$, then, according to Corollary 1, we cannot restore the actual value of S_i for all $i \in \{k_1 + 1, k_1 + 2, \ldots, n_1\} \cup \{j\}$, and according to Lemma 1, we can restore the true value of S_i for all $i \in \{1, 2, \ldots, k_1\} \backslash \{j\}$, therefore, we cannot restore the real value of S.

From the first and second cases, it follows that the number of errors that we can correct is

$$\mathrm{N}_{El}^{2L} \leq \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i}$$

Lemma 2 is proved.

B. 2Lbp-RRNS

To increase the number of detected and corrected errors, 2Lbp-RRNS uses the backpropagation and Hamming Distance (HD) mechanisms.

Let us discuss the Algorithm 2. "2Lbp-RRNS error corrections."

Algorithm 2 2Lbp-RRNS Error Corrections

Input: $(k_1, n_1), \left(k_{2,1}, n_{2,1}\right), \cdots, (k_{2,n_1}, n_{2,n_1})$
$S_1 \xrightarrow{RNS} \left(S_{1,1}, S_{1,2}, \ldots, S_{1,n_{2,1}}\right)$,
$S_2 \xrightarrow{RNS} (S_{2,1}, S_{2,2}, \ldots, S_{2,n_{2,2}}), \cdots,$
$S_{n_1} \xrightarrow{RNS} (S_{n_1,1}, S_{n_1,2}, \ldots, S_{n_1,,n_{2,n_1}})$
$\left(p_{1,1}, \ldots, p_{1,n_1}\right), \left(p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}\right), \ldots,$
$(p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$
Output: S, *flag*

Step 1. 2L-RRNS calculates S and *flag*. If $flag \neq -1$, S is recovered, otherwise S is not recovered.

Step 2. Based on $\mathrm{S}_{1,1}, \mathrm{S}_{1,2} \ldots . \mathrm{S}_{\mathrm{n}_{1,},\mathrm{n}_{2,\mathrm{n}_1}}$, choosing an (unordered) subset of $k_{2,i}$ elements from a fixed set of $n_{2,i}$ elements, we calculate possible values S_i^l for each of S_i.

Step 3. Choosing an (unordered) subset of k_1 elements $\mathrm{S}_{\mathrm{i}}^{\mathrm{l}}$ from a fixed set of n_1 elements, we calculate possible values S_i^l and restore S^j by the function *CRTtoBin*.

Step 4. Using the backpropagation concept, we encode each S^{j} to 2L-RRNS representation S^j and compute the HD between $\widetilde{S^j}$ and $\overline{S}$.

Step 5. We choose $\widetilde{S^j}$ for which the HD is minimal. If minimal HD between $\widetilde{S^j}$ and $\overline{S}$ is more than $N_E^{2Lbp} = \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i} - 1$, then return $flag = -1$; otherwise, $S = S^j$ and flag $= 1$.

end.

Consider an example when on the first and the second level the scheme is (2,3). Hence, $k_1 = k_{2,1} = k_{2,2} = k_{2,3} = 2$, $n_1 = n_{2,1} = n_{2,2} = n_{2,3} = 3$.

On the first level, we have a tuple with three elements (S_1, S_2, S_3). On the second level, we have three tuples

$$(S_{1,1}, S_{1,2}, S_{1,3}), \quad (S_{2,1}, S_{2,2}, S_{2,3}), \text{ and } (S_{3,1}, S_{3,2}, S_{3,3}).$$

Let assume errors in $S_{1,3}, S_{2,3}, S_{3,3}$. Then traditional 2L-RRNS detects errors but cannot correct them because we do not know the position of errors.

In the first step, 2Lbp-RRNS trying to restore S_1 uses three possible tuples for recovering S_1^1, S_1^2, and S_1^3.

$S_1^1 \xleftarrow{RNS} \widetilde{S_1^1} = \left(S_{1,1}, S_{1,2}\right)$, $S_1^2 \xleftarrow{RNS} \widetilde{S_1^2} = \left(S_{1,1}, S_{1,3}\right)$, and $S_1^3 \xleftarrow{RNS} \widetilde{S_1^3} = \left(S_{1,2}, S_{1,3}\right)$.

We do the same for S_2 denoting possible tuples as $S_2^1 \xleftarrow{RNS} \widetilde{S_2^1} = \left(S_{2,1}, S_{2,2}\right)$, $S_2^2 \xleftarrow{RNS} \widetilde{S_2^2} = (S_{2,1}, S_{2,3})$, and $S_2^3 \xleftarrow{RNS} \widetilde{S_2^3} = (S_{2,2}, S_{2,3})$.

To recover S, we analyze nine possible candidate tuples of the first level denoted as:

$$S^1 \xleftarrow{RNS} \widetilde{S^1} = (S_1^1, S_2^1), \quad S^2 \xleftarrow{RNS} \widetilde{S^2} = (S_1^1, S_2^2),$$
$$S^3 \xleftarrow{RNS} \widetilde{S^3} = (S_1^1, S_2^3)$$
$$S^4 \xleftarrow{RNS} \widetilde{S^4} = (S_1^2, S_2^1), \quad S^5 \xleftarrow{RNS} \widetilde{S^5} = (S_1^2, S_2^2),$$
$$S^6 \xleftarrow{RNS} \widetilde{S^6} = (S_1^2, S_2^3)$$
$$S^7 \xleftarrow{RNS} \widetilde{S^7} = (S_1^3, S_2^1), \quad S^8 \xleftarrow{RNS} \widetilde{S^8} = (S_1^3, S_2^2),$$
$$S^9 \xleftarrow{RNS} \widetilde{S^9} = (S_1^3, S_2^3).$$

Backpropagation converts each of nine S^j values back to 2L-RRNS denoting $\widetilde{S^j}$.

For each $\widetilde{S^j}$, we calculate the HD between $\overline{S}$ and $\widetilde{S^j}$, for all $j = \overline{1, 9}$. In our example, the HD equals to three, for all $j = \overline{2, 9}$, and equals to two between $\overline{S}$ for $\widetilde{S^1}$. We note that $k = 2$, hence, $S = S^1$. We restore data.

Now, let us calculate the number of errors detected and corrected by 2Lbp-RRNS in the general case.

The basic idea of this method is in the backpropagation of the restored variant that cannot be proved as correct or incorrect. This restored variant is encoded, so the data is moved in the direction reverse to restoring.

This new encoded variant $\widetilde{S^j}$ is compared with the initially encoded value $\overline{S}$ by calculating the HD. If HD is less than a given threshold, $\widetilde{S^j}$ is considered as a correct restored S

Thus, the error correction includes two additional processes: 2L-RRNS encoding and HD calculation.

The number of possible variants $\widetilde{S^j}$ depends on the number of errors in $\overline{S}$. Due to a large number of possible combinations, the time can grow significantly.

Backpropagation, on the one hand, increases the computational complexity of the error detection and correction algorithm, similar to the base extension. On the other hand, it allows to identify and recover more errors.

Here, we discuss the HD properties for the error localization in 2L-RRNS in more detail. Let we have two 2L-RRNS representation of SS: without errors $\widetilde{S}$ and with errors $\overline{S}$.

Property 1: If $HD\left(\widetilde{S}, \overline{S}\right) = 0$ *then* $\overline{S}$ *does not contain errors, i.e.* $\widetilde{S} = \overline{S}$.

Proof: Evidence from the contrary. Assume that $\overline{S}$ contains errors and $HD\left(\widetilde{S}, \overline{S}\right) = 0$. Since $\overline{S}$ contains errors, there exists a representation $\overline{S} \xleftarrow{2L-RRNS} S + E$, where $0 < E < P$, therefore

$$\overline{S} = \left(\left(S'_{1,1}, \ldots, S'_{1,n_{2,1}}\right), \ldots, \left(\left(S'_{n_1,1}, \ldots, S'_{n_1,n_{2,n_1}}\right)\right)\right)$$
$$\overline{E} = \left(\left(E'_{1,1}, \ldots, E'_{1,n_{2,1}}\right), \ldots, \left(\left(E'_{n_1,1}, \ldots, E'_{n_1,n_{2,n_1}}\right)\right)\right),$$
$$\widetilde{S} = \left(\left(S_{1,1}, \ldots, S_{1,n_{2,1}}\right), \ldots, \left(\left(S_{n_1,1}, \ldots, S_{n_1,n_{2,n_1}}\right)\right)\right),$$

where for all i, j:$S'_{i,j} = S_{i,j} + E_{i,j}$.

Considering that $0 < E < P$ then there exists at least one pair (i, j), such that $E_{i,j} \neq 0$, therefore there is at least one value $S_{i,j} \neq S'_{i,j}$, then $HD\left(\widetilde{S}, \overline{S}\right) > 0$. Thus, we have a contradiction. Therefore, if $HD\left(\widetilde{S}, \overline{S}\right) = 0$ then $\overline{S}$ does not contain errors. The property is proven.

Corollary 2: The number of errors that has 2L-RRNS representation of $\overline{S}$ is equal to $HD\left(\widetilde{S}, \overline{S}\right)$.

Property 2: Let $S^1 \neq S^2 \neq \ldots \neq S^t$, for which $HD\left(\widetilde{S^1}, \overline{S}\right) = HD\left(\widetilde{S^2}, \overline{S}\right) = \ldots = HD\left(\widetilde{S^t}, \overline{S}\right)$, then the number of errors in each of the representations $\widetilde{S^j}$ is equal.

Proof: Since the number of errors in the 2L-RRNS representation of S^j by the Corollary 2 is determined by $HD\left(\widetilde{S^j}, \overline{S}\right)$, from the condition $HD\left(\widetilde{S^1}, \overline{S}\right) = HD\left(\widetilde{S^2}, \overline{S}\right) = \ldots = HD\left(\widetilde{S^t}, \overline{S}\right)$, it follows that the number of errors in each $\widetilde{S^j}$ is equal, for all $j = \overline{1, t}$. The property is proven.

Corollary 3: Let $S^1 \neq S^2 \neq \ldots \neq S^t$, for which $HD\left(\widetilde{S^1}, \overline{S}\right) < HD\left(\widetilde{S^2}, \overline{S}\right) < \ldots < HD\left(\widetilde{S^t}, \overline{S}\right)$, then the S^1 representation in 2L-RRNS contains the least errors.

It follows from Corollary 3 and Property 2. If there are at least two S^j values such that $S^1 \neq S^2$, $HD\left(\widetilde{S^1}, \overline{S}\right) = HD\left(\widetilde{S^2}, \overline{S}\right) = d$ and $d \leq HD\left(\widetilde{S^j}, \overline{S}\right)$ for all $j = \overline{1, t}$, then it is impossible to correct S since we cannot determine which value from two S^1, S^2 is true.

Theorem 2: 2Lbp-RRNS can detect N_D^{2Lbp} and correct N_E^{2Lbp} errors, where

$$\mathrm{N}_D^{2Lbp} = \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i},$$

$$\mathrm{N}_E^{2Lbp} \leq \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i} - 1$$

Proof: From Corollary 2, it follows that the maximum number of errors that we can determine using HD is equal to the maximum number of errors that can be corrected if we know them, therefore

$$\mathrm{N}_D^{2Lbp} = \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i}.$$

To estimate the number of correctable errors of 2Lbp-RRNS, we consider their upper bound.

As in traditional threshold 2L-RRNS, no more than $\lfloor r_{2,i}/2 \rfloor$ errors are localized in each $\widetilde{S^i}$. Hence, there are $k_1 + \lceil r_{2,i}/2 \rceil$ values of S_i, the correctness of which can be confirmed. Thus, the correct value can be corrected.

Without loss of generality, we assume that in each of the representations in 1L-RRNS $S_{i_q} \in \{S_{i_1}, S_{i_2}, \ldots, S_{i_l}\}$ contains no more than r_{2,i_q} errors, where $l \geq k_1$. We denote $I = \{i_1, \ldots, i_l\}$ and there exists $u \notin I$ for which the representation in 1L-RRNS S_u contains less than $n_{2,u}$ errors, then using Corollary 3, we can restore S based on backpropagation mechanism. Backpropagation encodes each S^j candidate of S back to RRNS representation denoting $\widetilde{S^j}$.

Consider the number of errors that the upper bound case can correct. The maximum number of errors is less than or equal to

$$\mathrm{N}_E^{2Lbp} \leq \sum_{i=1}^{k_1} (n_{2,i} - k_{2,i}) + \sum_{i=k_1+1}^{n_1} n_{2,i} - 1$$

$$= \sum_{i=1}^{n_1} n_{2,i} - \sum_{i=1}^{k_1} k_{2,i} - 1.$$

The theorem is proved.

C. 2Lbp-RRNS PROPERTIES

In this section, we compare the properties of 2Lbp-RRNS and 2L-RRNS schemes for the example settings described in Table 1.

TABLE 1. Scheme settings.

id	k_1	n_1	$(k_{2,1}, n_{2,1}) - \cdots - (k_{2,n_1}, n_{2,n_1})$
1	2	4	(2,4)-(2,4)-(2,4)-(2,4)
2	3	4	(3,4)-(3,4)-(3,4)-(3,4)
3	4	4	(4,4)-(4,4)-(4,4)-(4,4)
4	2	5	(2,5)-(2,5)-(2,5)-(2,5)-(2,5)
5	3	5	(3,5)-(3,5)-(3,5)-(3,5)-(3,5)
6	4	5	(4,5)-(4,5)-(4,5)-(4,5)-(4,5)
7	5	5	(5,5)-(5,5)-(5,5)-(5,5)-(5,5)
8	2	6	(2,6)-(2,6)-(2,6)-(2,6)-(2,6)-(2,6)
9	3	6	(3,6)-(3,6)-(3,6)-(3,6)-(3,6)-(3,6)
10	4	6	(4,6)-(4,6)-(4,6)-(4,6)-(4,6)-(4,6)
11	5	6	(5,6)-(5,6)-(5,6)-(5,6)-(5,6)-(5,6)
12	6	6	(6,6)-(6,6)-(6,6)-(6,6)-(6,6)-(6,6)
13	2	7	(2,7)-(2,7)-(2,7)-(2,7)-(2,7)-(2,7)-(2,7)
14	3	7	(3,7)-(3,7)-(3,7)-(3,7)-(3,7)-(3,7)-(3,7)
15	4	7	(4,7)-(4,7)-(4,7)-(4,7)-(4,7)-(4,7)-(4,7)
16	5	7	(5,7)-(5,7)-(5,7)-(5,7)-(5,7)-(5,7)-(5,7)
17	6	7	(6,7)-(6,7)-(6,7)-(6,7)-(6,7)-(6,7)-(6,7)
18	7	7	(7,7)-(7,7)-(7,7)-(7,7)-(7,7)-(7,7)-(7,7)
19	2	8	(2,8)-(2,8)-(2,8)-(2,8)-(2,8)-(2,8)-(2,8)-(2,8)
20	3	8	(3,8)-(3,8)-(3,8)-(3,8)-(3,8)-(3,8)-(3,8)-(3,8)
21	4	8	(4,8)-(4,8)-(4,8)-(4,8)-(4,8)-(4,8)-(4,8)-(4,8)
22	5	8	(5,8)-(5,8)-(5,8)-(5,8)-(5,8)-(5,8)-(5,8)-(5,8)
23	6	8	(6,8)-(6,8)-(6,8)-(6,8)-(6,8)-(6,8)-(6,8)-(6,8)
24	7	8	(7,8)-(7,8)-(7,8)-(7,8)-(7,8)-(7,8)-(7,8)-(7,8)
25	8	8	(8,8)-(8,8)-(8,8)-(8,8)-(8,8)-(8,8)-(8,8)-(8,8)

Here, $(k_1, n_1) = (k_{2,i}, n_{2,i})$, hence, each storage has the same number of shares with the same threshold.

Figures 3-4 show the number of detected and corrected errors. We observe that 2Lbp-RRNS can detect and correct more errors than 2L-RRNS for all test cases. For the given experiments, 2bp-RRNS can detect 1.58x (see, Fig. 3) and correct 3.37x (see, Fig.4) times more errors than 2L-RRNS, on average.

Table 2 presents the average encoding/decoding speeds for settings described in Table 1 varying data size.

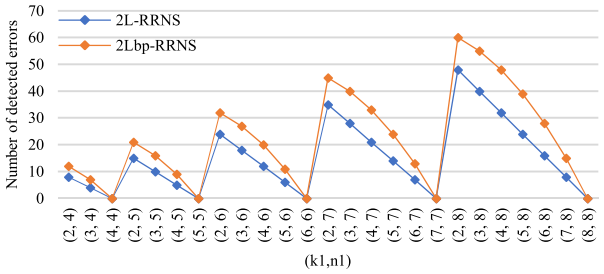

FIGURE 3. Error detection in 2L-RRNS and 2Lbp-RRNS.

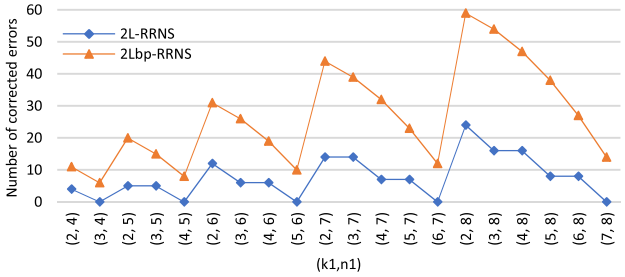

FIGURE 4. Error correction in 2L-RRNS and 2Lbp-RRNS.

TABLE 2. The average data encoding and decoding speed (MB/s).

	Data size	Encoding			Decoding		
		Min	Ave	Max	Min	Ave	Max
Mignotte	1 MB	0.505	0.975	1.379	0.529	0.813	0.783
	10 MB	0.507	0.995	1.399	0.647	0.923	1.137
	100 MB	0.510	1.015	1.419	0.764	1.047	1.491
MRC8	1 MB	0.784	2.174	3.145	5.921	7.371	9.314
	10 MB	0.786	2.194	3.166	6.038	7.926	9.667
	100 MB	0.788	2.221	3.186	6.156	8.483	10.02
MRC16	1 MB	2.405	4.983	6.645	12.50	13.63	15.60
	10 MB	2.408	5.003	6.665	12.62	14.19	15.96
	100 MB	2.410	5.023	6.685	12.74	14.74	16.30

V. ENCODING AND DECODING ALGORITHMS

In this section, we describe three algorithms: Mignotte, Mixed Radix Conversion based MRC8, and MRC16 used for encoding and decoding. Their pseudocodes are presented in Appendix.

A. ENCODING ALGORITHMS

Mignotte [43] is a classical secret sharing scheme with a CRT projection mechanism for error detecting and correcting. Mignotte represents an integer S by a tuple $(S_1, \ldots, S_{n_1})$, where $S_i = |S|_{p_{1,i}}$, $P_i = \frac{P}{p_{1,i}}$ and recovers S with classic CRT:

$$S = \left| \sum_{i=1}^{n} S_i P_i \left| P_i^{-1} \right|_{p_{1,i}} \right|_P$$

MRC8 and MRC16 [16], [41] are based on a weighted system called Mixed-Radix Systems (MRS) - a non-standard positional numeral system in which the numerical base varies from position to position. The basic implementations based

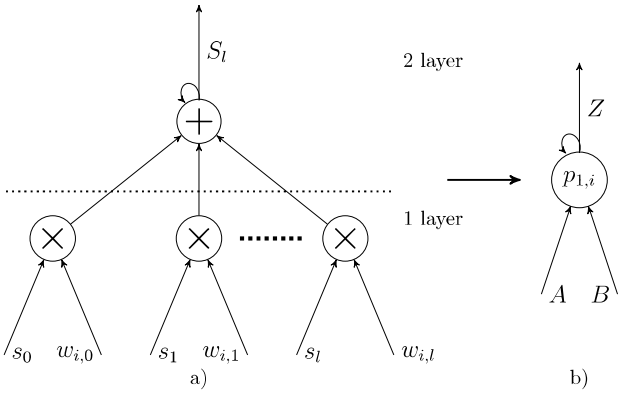

FIGURE 5. FRNN architecture (a) and its symbolic mapping (b).

on a neural-like network architecture named Finite Ring Neuronal Network (FRNN) are presented in [42].

Figure 5 shows the general interpretation of this architecture: a parallel, interconnected network of simple elements. It consists of two important components: 1) the neuronal processing elements capable of performing basic operations and 2) weights, which represent the knowledge of the system. All finite ring arithmetic like addition, multiplication, and their combination can be reduced to this architecture.

The simple FRNN architecture is based on the Pascal method of finding the division remainder and on the signed binary window method [61].

Original data S is represented as $S = s_l |s_{l-1}| .. |s_0$, where "|" is L-bits string concatenation of s_i. L defines the window size $L \in \{8, 16\}$. $w_{i,j} = \left|2^{L \cdot j}\right|_{p_{1,i}}$ are synaptic weights.

FRNN consists of two layers: the first one is the prefabricated layer on which the product s_i is calculated by the synaptic weight $w_{i,j}$. On the second computational layer, the sum of the values and remainder of the division is calculated by modulo $p_{1,i}$. Thus, FRNN is described using the following formula:

$$S_i = \left| \sum_{j=0}^{l} s_j \cdot w_{i,j} \right|_{p_{1,i}}$$

B. DECODING ALGORITHMS

To convert numbers from RNS to a binary, various algorithms are used: CRT [44], Wang method [45], MRS [41], Diagonal function [46], Core Function [47], and Approximate method [48].

The Diagonal function and Core Function are not advisable to use for such conversions [46].

CRT uses the computationally complex operation of finding the remainder of the division by the RNS range to convert numbers.

An approximate method reduces its complexity. It is based on replacing absolute values by relative values and replacing the operation of the division with the remainder of the general form by trial division. However, to obtain the correct value in the approximate method, it is necessary to increase the

size of the coefficients from $\lceil \log_2 P \rceil$ to $\lceil \log_2 (P \cdot \rho) \rceil$, where $\rho = -n_1 + \sum_i^{n_1} p_{1,i}$, which eliminates the resulting gain.

The recursive doubling Wang method is used to reduce its computational complexity. It reduces the size of the divisor from P to $\sqrt{P}$. But at the same time, the number of reminders of the division is increased from one to $\lceil \log_2 n_1 \rceil$.

Alternative solutions for converting from RNS to binary are MRS-based algorithms. The decoding consists of two-stages. In the first stage, the number is converted from RNS to MRS, then from MRS to binary.

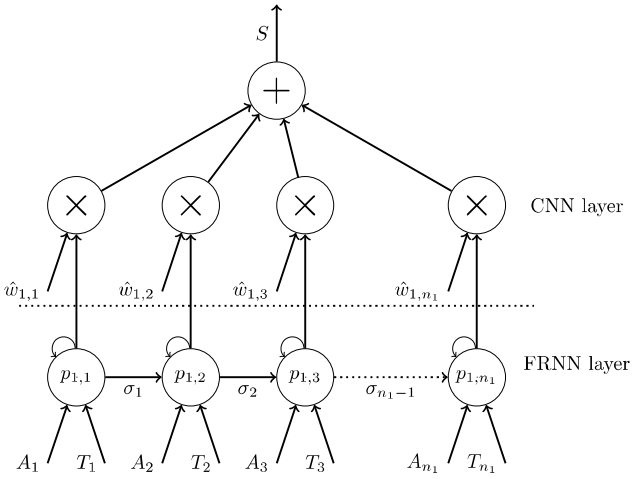

FIGURE 6. DNN architecture for decoding from 1L-RRNS to binary.

To reduce the computational complexity of the first stage, we propose a modification of the converting from RNS to binary by using CRT and the Neural Network of the Finite Ring (FRNN). The second stage of the transition from MRS to binary is implemented using the Convolutional Neural Network (CNN) (Fig. 6).

The RRNS residues are converted to MRS and then to binary $S \overset{MRS}{\longleftarrow} \widehat{S} = [\widehat{s}_1, \ldots, \widehat{s}_{n_1}]$ by:

$$\mathrm{S} = \sum_{i=1}^{n_1} \widehat{s}_i \widehat{w}_{1,i},$$

where $\widehat{w}_{1,i}$ are the radices and $\widehat{s}_{1,i}$ are the MRS digits, $0 \leq \widehat{s}_{1,i} < p_{1,i}$ and $\widehat{w}_{1,i} = \prod_{j=1}^{i-1} p_{1,j}$.

To recover S, the classic MRC is formulated as follows:

$$S = \widehat{s}_1 + \widehat{s}_2 p_{1,1} + \widehat{s}_3 p_{1,1} p_{1,2} + \ldots + \widehat{s}_n p_{1,1} p_{1,2} \ldots p_{1,n_1-1}$$

The MRC digits can be computed as:

$$\begin{aligned}
\widehat{s}_1 &= S_1 \\
\widehat{s}_2 &= \left| (S_2 - \widehat{s}_1) \left| p_{1,1}^{-1} \right|_{p_{1,2}} \right|_{p_{1,2}} \\
\widehat{s}_3 &= \left| \left((S_3 - \widehat{s}_1) \left| p_{1,1}^{-1} \right|_{p_{1,3}} - \widehat{s}_2 \right) \left| p_{1,2}^{-1} \right|_{p_{1,3}} \right|_{p_{1,3}} \\
\widehat{s}_n &= \left| \left((\ldots (S_n - \widehat{s}_1) \left| p_{1,1}^{-1} \right|_{p_{1,n_1}} - \widehat{s}_2 \right) \left| p_{1,2}^{-1} \right|_{p_{1,n}} \right. \\
&\quad \left. - \ldots - \widehat{s}_{1,n_1-1}) \left| p_{1,n_1-1}^{-1} \right|_{p_{1,n_1}} \right|_{p_{1,n_1}}
\end{aligned}$$

TABLE 3. Access speeds of seven clouds (MB/s).

Cloud storage	Upload speed			Download speed		
	Min	Max	Avg	Min	Max	Avg
GoogleDrive	1.79	3.24	2.98	2.15	3.26	3.06
OneDrive	0.91	1.70	1.46	1.21	2.41	2.18
Dropbox	2.59	3.05	2.93	3.07	3.32	3.25
Box	1.91	3.26	2.55	2.01	3.20	2.62
Egnyte	1.24	1.93	1.70	2.17	2.36	2.30
Sharefile	0.11	0.65	0.51	0.72	0.76	0.75
Salesforce	0.52	0.73	0.64	0.68	0.72	0.71

A positive number in the interval $[0, P-1]$ can be uniquely represented.

MRS reduces the computational complexity of 1L-RRNS to binary conversion by eliminating the operation of finding the remainder of division by P, by calculating $\widehat{s}_i$. The computational complexity of the $\widehat{s}_i$ is quadratic of n_1. The simultaneous use of the ideas of MRS and CRT to translate from 1L-RRNS to the binary number system allows speeding up the algorithm.

Let $B_i = \left| P_i^{-1} \right|_{p_{1,i}} \cdot P_i$ is the orthogonal basis of 1L-RRNS, where for all $i = \overline{1, n_1} : P_i = P/p_{1,i}$. For any $i = \overline{1, n_1} : B_i$ is represented in the MRS as $B_i \overset{MRS}{\longleftarrow} \widehat{B}_i = [\widehat{b}_{i,1}, \ldots, \widehat{b}_{i,n_1}]$, $T_i = (\widehat{b}_{1,i}, \ldots, \widehat{b}_{i,i})$ is a tuple of coefficients in MRS representation, $A_i = (\underbrace{S_i, S_i, \ldots, S_i}_{i \text{ times}})$ and $\sigma_i = \left\lfloor \frac{1}{p_{1,i}} \cdot \sum_{j=1}^{i} S_j \cdot b_{j,i} \right\rfloor$ is the bias.

Figure 6 shows the architecture of a Decoding Neural Network (DNN) for conversion from 1L-RRNS to binary consisting of two layers: FRNN and CNN.

VI. PERFORMANCE ANALYSIS

In this section, we evaluate the 2Lbp-RRNS performance in terms of encoding/decoding speeds using three algorithms: Mignotte, MRC8, and MRC16, and uploading and downloading speed to real cloud storages.

Our software platform is based on JMetal 5.6 and JDK 11.0.1 (64-bits). The hardware platform has Dell Precision T3610, Intel Xeon CPU E5-1606 @ 2.80 GHz, 16 GB DDR3 RAM with Windows 10 Enterprise 64-bits.

The experimental scenario includes seven cloud storages: DropBox, OneDrive, Box, Salesforce, GoogleDrive, Sharefile, and Egnyte. To access the public REST API of the CSPs, we used a Java wrapper for Google Drive, Dropbox, Box, and Sharefile. For OneDrive, Egnyte, and Salesforce, we used the Apache HttpClient library [40].

Table 3 shows low, high, and average access speeds of seven CSPs. We see that, in most cases, the access speeds are less than encoding/decoding speeds (Table 2).

For MRC16, the average uploading speed is $5.023/2.98 = 1.67$ times less than average encoding speed. The average downloading speed is $14.74/\ 3.25\ = 4.53$ times less than decoding speed (Table 2).

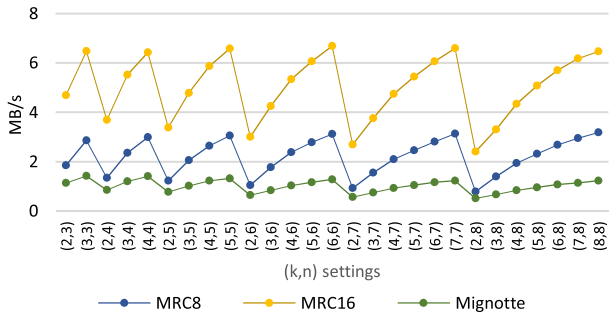

FIGURE 7. Encoding speeds for setting (3,4) on Level 1.

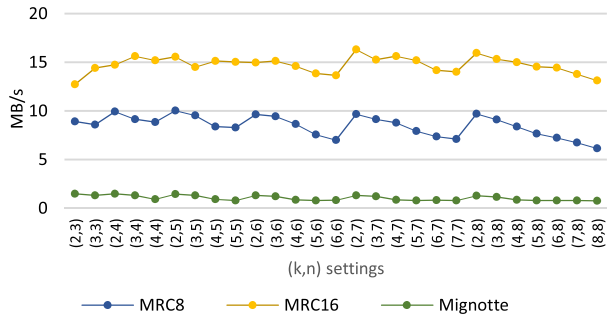

FIGURE 8. Decoding speeds for settings (3,4) on Level 1.

A. 2Lbp-RRNS ENCODING AND DECODING SPEED

To draw all aspects of the proposed system, we run all settings (k_1, n_1) of Level 1 and all settings $(k_{2,i}, n_{2,i})$ of Level 2, where $1 \leq i \leq n_1$.

Figures 7-8 show examples of the encoding/decoding speeds, respectively, for up to eight clouds. On the first level, the access structure is limited to $(k_1, n_1) = (3, 4)$. On the second level, we consider 27 variants of $(k_{2,i}, n_{2,i})$, from $n_{2,i} = 3$ to $n_{2,i} = 8$.

Due to the share size is decreasing while k_2 is increasing, the highest encoding speeds are obtained when $k_2 = n_2$ (Fig. 7).

On the other hand, the lowest decoding speeds are obtained for $k_2 = n_2$ (Fig. 8) because it is necessary to decode all n_2 shares to recover the original data.

Obtained data shows that encoding speed is in the range 0.505-6.685 MB/s, decoding speed is in the range 0.529-16.30 MB/s. The access speed is in the range of 0.11-3.32 MB/s.

Mignotte is the slowest of the considered algorithms with speeds no higher than 1.23 MB/s for encoding and 1.3 MB/s for decoding.

MRC8 has a maximum decoding speed of 10.02 MB/s in (2.5) of Level 2 and encoding speed of 3 MB/s for settings where $k_2 = n_2$ of Level 2.

We can see that MRC16 outperforms two other algorithms in all the experiments. For example, MRC16 achieves a maximum encoding speed of 6.68 MB/s for a setting (3,4) in Level 1 and (6,6) in Level 2 (Fig. 7).

Finally, MRC16 is 2.53 times faster than MRC8, and 4.83 times faster than Mignotte in the encoding phase. In the decoding phase, MRC16 is 1.78 times faster than MRC8, and 11.43 times faster than Mignotte.

MRC16 decoding speed is 15.04 MB/s, and Mignotte is 0.8019 MB/s (Fig.8).

We focus on the performance of a serial and parallel execution of 2Lbp-RRNS.

Figures 9-10 show the boxplots for encoding/decoding speed, respectively for both versions.

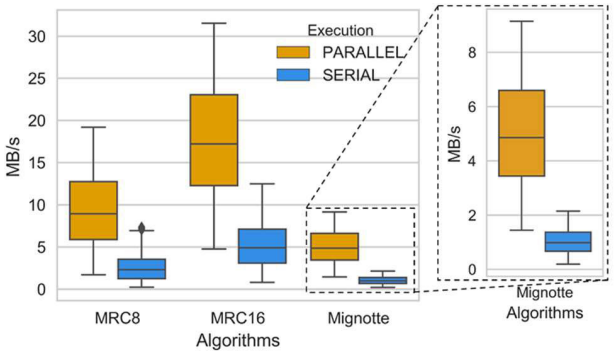

FIGURE 9. Boxplot encoding speeds for all combinations of settings.

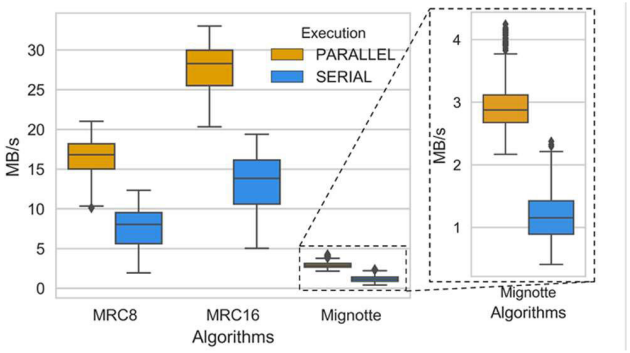

FIGURE 10. Boxplot decoding speeds for all combinations of settings.

Boxplot conveniently displays the median, lower, and upper quartiles, minimum and maximum sample values, and outliers.

For each setting of Level 1, from $n_1 = 3$ to $n_1 = 8$, we execute all combinations of Level 2 settings.

We run the experiments 30 times to have statistically significant results.

MRC8 algorithm has outliers that show that there is a large gap between the minimum and maximum speeds. Most of the values are in the vicinity of the low-speed highlands.

MRC16 algorithm is more balanced, and it is faster than MRC8 and Mignotte reaching a maximum speed higher than 30 MB/s for parallel execution.

The decoding speed of MRC16 is higher than MRC8 and Mignotte (Fig. 10). MRC16 and MRC8 have no outliers. Therefore, we can conclude that there are no speeds that stand out from the general sample.

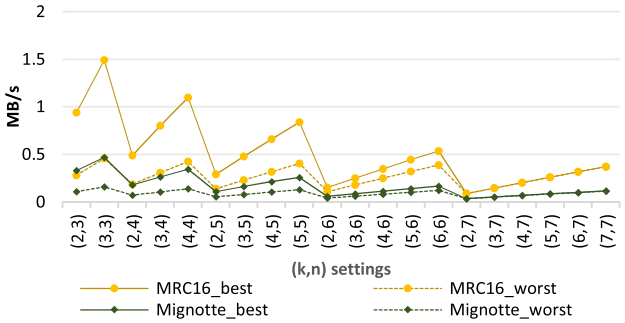

FIGURE 11. Uploading velocity for Level 1 setting (3,4).

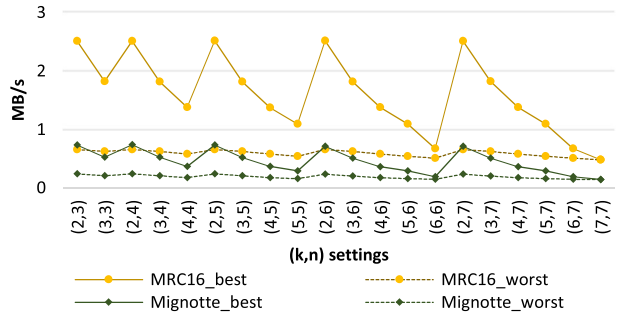

FIGURE 12. Downloading velocity for Level 1 setting (3,4).

Mignotte has outliers, i.e., values that exceed the possible values of the total sample, given that they are more than the median. Overall, MRC16 shows the best performance for all combinations of settings of the 2L-RRNS.

B. VELOCITY

Since encoding and decoding speeds are similar order to the downloading and uploading speeds, we consider the complete cycle of the data: encoding plus uploading and downloading plus decoding. We name them uploading and downloading velocities.

The uploading velocity $V_u = \frac{size(D)}{T_E + t_{up}}$ includes encoding from binary to RNS and uploading time. The downloading velocity $V_d = \frac{size(D)}{T_D + t_{dow}}$ includes downloading time from the cloud storage and decoding from RNS to binary (see Sec III.).

We are focusing on a multi-cloud environment. It has a dynamic nature with parameters changing over time and difficult to predict and anticipate in advance. These types of non-stationarity are one of the main issues in the design of efficient algorithms capable of mitigating their consequences.

To determine the practical applicability of the proposed scheme and study its properties, we consider the best and worst scenarios. In the best one, clouds with the best access speeds are selected for data storage. In the worst one, the slowest clouds are chosen.

We calculate the upload time $t_{up} = \sum_{j=1}^{n_{2,i}} size\left(S_{i,j}\right)/up\,(j)$ and download time $t_{dow} = \sum_{j=1}^{k_{2,i}} size(S_{i,j})/down(j)$, where $i \leq n_1$, $up(j)$ and $down(j)$ are the upload and download access speeds to/from j-th cloud.

Let us consider Level 1 setting (3,4) and Level 2 with (5,5) setting and the access speeds of Table 3 .

Figure 11 shows that MRC16 has an uploading velocity $V_u = 0.837$ MB/s, in the best case, and $V_u = 0.406$ MB/s, in the worst-case. Mignotte has an uploading velocity $V_u = 0.257$ MB/s, in the best case, and $V_u = 0.130$ MB/s, in the worst case.

We note that for the same settings, MRC16 encoding speed is 6.583 MB/s (Fig. 7) and Mignotte encoding speed is 1.317 MB/s.

Figure 12 shows that MRC16 has the downloading velocity $V_d = 1.093$ MB/s, in the best case, and $V_d = 0.540$ MB/s, in the worst case. Mignotte has $V_d = 0.292$ MB/s, in the best case, and $V_d = 0.160$ MB/s, in the worst case.

VII. CONCLUSION

We present a two-level secret sharing scheme based on a Redundant Residue Number System named 2Lbp-RRNS designed as a configurable, reliable, and secure multi-clouds data storage mechanism, and a variant of fully homomorphic encryption for privacy-preserving, parallel processing, and scalability. It is based on backpropagation and Hamming distance mechanisms.

Our contribution is multi-fold.

- We provide a theoretical analysis of the 2Lbp-RRNS solution as an extension of the classical 2L-RRNS. We formulate, explain, and prove its main properties to extend existing knowledge within the limits of RRNS assumptions.
- We show how the reliability and performance of the system depend on the 2Lbp-RRNS parameters.
- We show how the backpropagation and Hamming Distance mechanisms allow to increase the number of detected and corrected errors.
- We provide the upper bounds of traditional threshold 2L-RRNS and 2Lbp-RRNS to estimate the number of detectable and correctable errors.
- We study various data access scenarios exploring the properties of 2Lbp-RRNS and show that it restores data in more cases than traditional 2L-RRNS. It detects errors up to 1.58 and corrects 3.37 times more than 2L-RRNS, on average.
- We provide an efficient implementation of two data encoding and decoding algorithms based on the Mixed-Radix system, Finite Ring Neuronal Network (FRNN), and Pascal method named MRC8 and MRC16.
- We evaluate the 2Lbp-RRNS performance considering the complete cycle of the data storage: encoding-uploading and downloading-decoding with real cloud storage parameters.
- We compare 2Lbp-RRNS performance with three algorithms: Mignotte, MRC8, and MRC16. Results show that MRC16 is a more balanced and faster algorithm outperforming MRC8 and Mignotte. We also evaluate the performance of their serial and parallel execution for up to eight real cloud storages.

2L-RRNS Encoding
Input: settings $= (k_1, n_1), (k_{2,1}, n_{2,1}), \cdots, (k_{2,n_1}, n_{2,n_1})$
S−Input data;
$\hat{p} = (p_{1,1}, \ldots, p_{1,n_1}), (p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}), \cdots, (p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$. $W_{1,i} = (w_{1,i,0}, \ldots, w_{1,i,l})$ and $W_{2,i,j} = (w_{2,i,j,0}, \ldots, w_{2,i,j,l})$ is synaptic weights FRNN, algo – Mignotte, MRC8, MRC16
Output: $\tilde{S}$
1. Case *algo* == *Mignotte*:
 1.1 $\tilde{S} = Mignotte$(settings, $S, \hat{p}$, encoding)
2. Case *algo* == *MRC8*:
 2.1 $\tilde{S} = MRC8$ (settings, $S, \hat{p}, W_{1,i}, W_{2,i,j}$, encoding)
3. Case *algo* == *MRC16*:
 3.1 $\tilde{S} = MRC16$ (settings,$S, \hat{p}, W_{1,i}, W_{2,i,j}$, encoding)
4. **return** $\tilde{S}$

2L-RRNS decoding
Input: settings $= (k_1, n_1), (k_{2,1}, n_{2,1}), \cdots, (k_{2,n_1}, n_{2,n_1})$
S−Representation of S in 2L-RRNS;
$\hat{p} = (p_{1,1}, \ldots, p_{1,n_1}), (p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}), \ldots, (p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$
;I_D; $\hat{W}_1 = (\hat{w}_{1,1}, \ldots, \hat{w}_{1,n_1})$ and $\hat{W}_{2,i} = (\hat{w}_{2,i,1}, \ldots, \hat{w}_{2,i,n_{2,i}})$ is synaptic weights DNN; algo – Mignotte, MRC8, MRC16
Output: S
1 Case *algo* == *Mignotte*:
 1.1 $S = Mignotte$ (settings, $I_D, \tilde{S}, \hat{p}$, decoding)
2 Case *algo* == *MRC8*:
 2.1 $S = MRC8$ (settings, $I_D, \tilde{S}, \hat{p}, \hat{W}_1, \hat{W}_{2,i}$, decoding)
3 Case *algo* == *MRC16*:
 3.1 $S = MRC16$ (settings, $I_D, \tilde{S}, \hat{p}, \hat{W}_1, \hat{W}_{2,i}$, decoding)
4 **return** S

Mignotte Encoding
Input: settings $= (k_1, n_1), (k_{2,1}, n_{2,1}), \cdots, (k_{2,n_1}, n_{2,n_1})$
S−Input data;
$\hat{p} = (p_{1,1}, \ldots, p_{1,n_1}), (p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}), \cdots, (p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$.
Output: $S_{i,j}$ - RNS encoded shares
1. For i = 1 to n_1 do:
 1.1 $S_i = |S|_{p_{1,i}}$
2. For i = 1 to n_1 do:
 2.1 For j = 1 to $n_{2,i}$ do:
  2.1.2 $S_2 = |S_i|_{p_{2,i,j}}$
3. **return** $\tilde{S}$

Mignotte Decoding
Input: settings $= (k_1, n_1), (k_{2,1}, n_{2,1}), \cdots, (k_{2,n_1}, n_{2,n_1})$
S− Representation of S in 2L-RRNS;
$\hat{p} = (p_{1,1}, \ldots, p_{1,n_1}), (p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}), \cdots, (p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$.
Output: S
1. $S_{list} = []$; $p_{list} = []$; // auxiliary lists
2. For $i = 1$ to k_1 do:
 2.1. $j = I_D[i]$;
 2.2. $S_j = CRTtoBin((S_{j,1}, \ldots, S_{j,n_{2,i}}), (p_{2,j,1}, \ldots, p_{2,j,n_{2,i}}))$;
 2.3. $S_{list}.append(S_j)$; $p_{list}.append(p_{1,j})$;
3. $S = CRTtoBin\,(S_{list}, p_{list})$
4. **return** S

MRC Encoding
Input: settings $= (k_1, n_1), (k_{2,1}, n_{2,1}), \cdots, (k_{2,n_1}, n_{2,n_1})$
S−Input data;
$\hat{p} = (p_{1,1}, \ldots, p_{1,n_1}), (p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}), \ldots, (p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$
$W_{1,i} = (w_{1,i,0}, \ldots, w_{1,i,l})$ and $W_{2,i,j} = (w_{2,i,j,0}, \ldots, w_{2,i,j,l})$ is synaptic weights FRNN,
Output: $S_{i,j}$
1. For $i = 1$ to n_1 do:
 1.1 $S_i = FRNN(S, p_{1,i}, W_{1,i})$
2. For i = 1 to n_1 do:
 2.1 For j = 1 to $n_{2,i}$ do:
  2.1.2 $S_{i,j} = FRNN\,(S_i, p_{2,i,j}, W_{2,i,j})$
3. **return** $S_{i,j}$

MRC Decoding
Input: settings $= (k_1, n_1), (k_{2,1}, n_{2,1}), \cdots, (k_{2,n_1}, n_{2,n_1})$
$\tilde{S}$ - Representation of S in 2L-RRNS;
$\hat{p} = (p_{1,1}, \ldots, p_{1,n_1}), (p_{2,1,1}, \ldots, p_{2,1,n_{2,1}}), \ldots, (p_{2,n_1,1}, \ldots, p_{2,n_1,n_{2,n_1}})$; I_D; $\hat{W}_1 = (\hat{w}_{1,1}, \ldots, \hat{w}_{1,n_1})$ and $\hat{W}_{2,i} = (\hat{w}_{2,i,1}, \ldots, \hat{w}_{2,i,n_{2,i}})$ is synaptic weights DNN,
Output: S
1. $S_{list} = []$; $p_{list} = []$; / auxiliary lists
2. For i = 1 to k_1 do:
 2.1. $c = DNN\,(S_{i,j}, p_{2,I_D[i],j}, \hat{W}_{2,I_D[i]})$
 2.2. $S_{list}.append(S_i)$; $p_{list}.append(p_{1,I_D})$
3. $S = DNN(S_{list}, p_{list}, \hat{W}_1)$
4. **return** S

It is important to provide a multi-objective comparison with known approaches based on erasure codes, regeneration codes, and secret sharing schemes. It is the subject of future work for complex improving the technical characteristics of security, redundancy, and reliability of the cloud storage.

There are several open research challenges. We want to evaluate our solution with dynamic variations of the cloud characteristics, failures, and attacks to study how an adaptive approach can mitigate the non-stationary uncertainty and provide a good compromise over the criteria. The number of shares and their distribution on the second level can be dynamically adjusted to cope with the situation in each stor-

age and different user requirements. To this end, past technical characteristics and failures statistics of cloud providers have to be analyzed for a certain time interval.

APPENDIX

We describe the algorithms used for the analysis of a 2L-RRNS model of Section V. For simplicity, in the algorithms, the variable $\widetilde{S}$ represents an array with all the shares of level 2.

Algorithms *Mignotte Encoding*, *MRC Encoding*, and the functions of algorithm 2L-RRNS encoding obtain the total shares $\tilde{S}$ from input data S. The three algorithms first convert S into n_1 shares, next take each share S_i at a time and convert it into $n_{2,i}$ shares for each combination of settings of Level 2.

Mignotte Encoding uses mod operation $S_i = |S|_{p_i}$ to convert each input. *MRC* [8,16] *Encoding* represents the input into a residue representation, lines 1.1 and 2.1.2 using FRNN.

To recover S, we use *2L-RRNS Decoding* algorithm and functions *Mignotte Decoding* that uses classic CRT, and *MRC Decoding* that uses classic MRC decoding conversion [16] and FRNN modification based on FRNN [42]. The three functions use $k_{2,i}$ shares of Level 2 to retrieve the S_i shares of Level 1 for each combination of settings of Level 2. Next, we take $k_1$shares $S_i$and, finally, retrieve S.

For *MRC Decoding*, we use a DNN. First, we calculate the coefficients (weights) of the neuronal architecture in lines from 2.1 to 3. Then, we perform the calculation of each part of the S_i for an FRNN.

REFERENCES

[1] M. A. Vouk, "Cloud computing–issues, research and implementations," *J. Comput. Inf. Technol.*, vol. 16, no. 4, p. 235, 2008, doi: 10.2498/cit.1001391.

[2] A. C. Mora, Y. Chen, A. Fuchs, A. Lane, R. Lu, and P. Manadhata. (2012). *Top Ten Big Data Security and Privacy Challenges*. Cloud Security Alliance. Accessed: Apr. 29, 2020. [Online]. Available: https://www.isaca.org/Groups/Professional-English/big-data/GroupDocuments/Big_Data_Top_Ten_v1.pdf

[3] D. Hubbard and M. Sutton (2010), *Top Threats to Cloud Comput. V1.0*. Cloud Security Alliance. Accessed: Apr. 29, 2020. [Online]. Available: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf

[4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26, Jun. 2008, doi: 10.1145/1365815.1365816.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

[6] H. Herodotou, H. Lim, G. Luo, N. Borisov, and L. Dong, "Starfish: A self-tuning system for big data analytics," in *Proc. 5th Biennial Conf. Innovative Data Syst. Res. (CIDR)*, 2011, pp. 261–272.

[7] N. Leavitt, "Will NoSQL databases live up to their promise?" *Computer*, vol. 43, no. 2, pp. 12–14, Feb. 2010, doi: 10.1109/MC.2010.58.

[8] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, "Security issues in NoSQL databases," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 541–547, doi: 10.1109/TrustCom.2011.70.

[9] M. T. Ozsu and P. Valduriez, "Distributed database systems: Where are we now?" *Computer*, vol. 24, no. 8, pp. 68–78, Aug. 1991, doi: 10.1109/2.84879.

[10] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep. 2002, doi: 10.1109/MIC.2002.1036038.

[11] A. Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, 1st ed. Newton, MA, USA: O'Reilly, 2001.

[12] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.

[13] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006, p. 263.

[14] N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-López, and J. M. Cortés-Mendoza, "AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security," *Future Gener. Comput. Syst.*, vol. 92, pp. 1080–1092, Mar. 2019, doi: 10.1016/j.future.2017.09.061.

[15] A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems," *J. Netw. Comput. Appl.*, vol. 59, pp. 208–218, Jan. 2016, doi: 10.1016/j.jnca.2014.09.021.

[16] A. Tchernykh, M. Babenko, N. Chervyakov, V. Miranda-Lopez, A. Avetisyan, A. Y. Drozdov, R. Rivera-Rodriguez, G. Radchenko, and Z. Du, "Scalable data storage design for non-stationary IoT environment with adaptive security and reliability," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10171–10188, Oct. 2020, doi: 10.1109/JIOT.2020.2981276.

[17] N. I. Chervyakov, P. A. Lyakhov, M. G. Babenko, I. N. Lavrinenko, A. V. Lavrinenko, and A. S. Nazarov, "The architecture of a fault-tolerant modular neurocomputer based on modular number projections," *Neurocomputing*, vol. 272, pp. 96–107, Jan. 2018, doi: 10.1016/j.neucom.2017.06.063.

[18] T. F. Tay and C.-H. Chang, "A non-iterative multiple residue digit error detection and correction algorithm in RRNS," *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 396–408, Feb. 2016, doi: 10.1109/TC.2015.2435773.

[19] A. Tchernykh, V. Miranda-López, M. Babenko, F. Armenta-Cano, G. Radchenko, A. Y. Drozdov, and A. Avetisyan, "Performance evaluation of secret sharing schemes with data recovery in secured and reliable heterogeneous multi-cloud storage," *Cluster Comput.*, vol. 22, no. 4, pp. 1173–1185, Dec. 2019, doi: 10.1007/s10586-018-02896-9.

[20] A. Skavantzos and M. Abdallah, "Implementation issues of the two-level residue number system with pairs of conjugate moduli," *IEEE Trans. Signal Process.*, vol. 47, no. 3, pp. 826–838, Mar. 1999, doi: 10.1109/78.747787.

[21] P. Ali, M. Kambiz, S. S. Mohammad, T. E. Shiva, and R. Mehdi, "Fault-tolerant and information security in networks using multi-level redundant residue number system," *Res. J. Recent Sci.*, vol. 3, no. 3, pp. 89–92, 2014.

[22] S. Timarchi and K. Navi, "Efficient class of redundant residue number system," in *Proc. IEEE Int. Symp. Intell. Signal Process.*, Oct. 2007, pp. 1–6, doi: 10.1109/WISP.2007.4447506.

[23] A. Barati, M. Dehghan, A. Movaghar, and H. Barati, "Improving fault tolerance in ad-hoc networks by using residue number system," *J. Appl. Sci.*, vol. 8, no. 18, pp. 3273–3278, Dec. 2008.

[24] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York, NY, USA: Wiley, 2007.

[25] A. J. Menezes, J. Katz, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.

[26] M. P. Mohite and S. B. Ardhapurkar, "Design and implementation of a cloud based computer forensic tool," in *Proc. 5th Int. Conf. Commun. Syst. Netw. Technol.*, Apr. 2015, pp. 1005–1009, doi: 10.1109/CSNT.2015.180.

[27] T. V. Lillard, C. P. Garrison, C. A. Schiller, J. Steele, and J. Murray, *Digital Forensics for Network, Internet, and Cloud Computing*, T. V. Lillard, C. P. Garrison, C. A. Schiller, and J. Steele, Eds. Syngress, 2010, doi: 10.1016/B978-1-59749-537-0.00015-6.

[28] D. Attas and O. Batrafi, "Efficient integrity checking technique for securing client data in cloud computing," *IJECS*, vol. 8282, no. 6105, p. 11, 2011.

[29] X. Wang and H. Yu, "How to break MD5 and other hash functions," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2005, pp. 19–35, doi: 10.1007/11426639_2.

[30] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," *ACM Comput. Surv.*, vol. 26, no. 2, pp. 145–185, Jun. 1994, doi: 10.1145/176979.176981.

[31] Z. Sun, Q. Zhang, Y. Li, and Y.-A. Tan, "DPPDL: A dynamic partial-parallel data layout for green video surveillance storage," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28,no. 1, pp. 193–205, Jan. 2018, doi: 10.1109/TCSVT.2016.2605045.

[32] A. Tchernykh, U. Schwiegelsohn, E.-G. Talbi, and M. Babenko, "Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability," *J. Comput. Sci.*, vol. 36, Sep. 2019, Art. no. 100581, doi: 10.1016/j.jocs.2016.11.011.
[33] A. Tchernykh, M. Babenko, N. Chervyakov, V. Miranda-López, V. Kuchukov, J. M. Cortés-Mendoza, M. Deryabin, N. Kucherov, G. Radchenko, and A. Avetisyan, "AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage," *Int. J. Approx. Reasoning*, vol. 102, pp. 60–73, Nov. 2018, doi: 10.1016/j.ijar.2018.07.010.
[34] M. Gomathisankaran, A. Tyagi, and K. Namuduri, "HORNS: A homomorphic encryption scheme for cloud computing using residue number system," in *Proc. 45th Annu. Conf. Inf. Sci. Syst.*, Mar. 2011, pp. 1–5, doi: 10.1109/CISS.2011.5766176.
[35] J. H. Cheon, J. Kim, M. S. Lee, and A. Yun, "CRT-based fully homomorphic encryption over the integers," *Inf. Sci.*, vol. 310, pp. 149–162, Jul. 2015, doi: 10.1016/j.ins.2015.03.019.
[36] C. Asmuth and J. Bloom, "A modular approach to key safeguarding," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 2, pp. 208–210, Mar. 1983.
[37] F. Barsi and P. Maestrini, "Error detection and correction by product codes in residue number systems," *IEEE Trans. Comput.*, vol. C-23, no. 9, pp. 915–924, Sep. 1974, doi: 10.1109/T-C.1974.224055.
[38] V. T. Goh and M. U. Siddiqi, "Multiple error detection and correction based on redundant residue number systems," *IEEE Trans. Commun.*, vol. 56, no. 3, pp. 325–330, Mar. 2008, doi: 10.1109/TCOMM.2008.050401.
[39] S. J. Piestrak, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Trans. Comput.*, vol. 43, no. 1, pp. 68–77, Jan. 1994, doi: 10.1109/12.250610.
[40] *HttpClient*. Accessed: Oct. 30, 2020. [Online]. Available: https://github.com/amcewen/HttpClient
[41] C. H. Huang, "A fully parallel mixed-radix conversion algorithm for residue number applications," *IEEE Trans. Comput.*, vol. C–2, no. 4, pp. 398–402, Apr. 1983, doi: 10.1109/TC.1983.1676242.
[42] D. Zhang, G. A. Jullien, and W. C. Miller, "A neural-like network approach to finite ring computations," *IEEE Trans. Circuits Syst.*, vol. 37, no. 8, pp. 1048–1052, Aug. 1990, doi: 10.1109/31.56084.
[43] M. Mignotte, "How to share a secret," in *Proc. Workshop Cryptogr.*, 1982, pp. 371–375.
[44] N. Szabo and R. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York, NY, USA: McGraw-Hill, 1967.
[45] Y. Wang, "Residue-to-binary converters based on new chinese remainder theorems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 3, pp. 197–205, Mar. 2000, doi: 10.1109/82.826745.
[46] P. V. A. Mohan, "RNS to binary conversion using diagonal function and Pirlo and Impedovo monotonic function," *Circuits, Syst., Signal Process.*, vol. 35, no. 3, pp. 1063–1076, Mar. 2016, doi: 10.1007/s00034-015-0090-9.
[47] I. J. Akushskii, V. M. Burcev, and I. T. Pak, "A new positional characteristic of non-positional codes and its application," in *Coding Theory Optim. Complex Syst.*, SSR, Alm-Ata'Nauka'Kazah, 1977.
[48] N. I. Chervyakov, M. G. Babenko, P. A. Lyakhov, and I. N. Lavrinenko, "An approximate method for comparing modular numbers and its application to the division of numbers in residue number systems," *Cybern. Syst. Anal.*, vol. 50, no. 6, pp. 977–984, Nov. 2014, doi: 10.1007/s10559-014-9689-2.
[49] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secur. Comput.*, vol. 4, no. 11, pp. 169–180, 1978.
[50] E. F. Brickell and Y. Yacobi, "On privacy homomorphisms (extended abstract)," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1987, pp. 117–125, doi: 10.1007/3-540-39118-5_12.
[51] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 1999, pp. 223–238, doi: 10.1007/3-540-48910-X_16.
[52] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.
[53] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, "Batch fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2013, pp. 315–335, doi: 10.1007/978-3-642-38348-9_20.
[54] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 1–36, Jul. 2014, doi: 10.1145/2633600.
[55] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 10031, J. Cheon and T. Takagi, Eds. Berlin, Germany: Springer, 2016, pp. 3–33, doi: 10.1007/978-3-662-53887-6_1.
[56] A. Kipnis and E. Hibshoosh, "Efficient methods for practical fully homomorphic symmetric-key encryption, randomization, and verification," *IACR Cryptol. EPrint Arch.*, vol. 2012, p. 637, 2012.
[57] L. Xiao, O. Bastani, and I.-L. Yen, "An efficient homomorphic encryption protocol for multi-user systems," *IACR Cryptol. EPrint Arch.*, vol. 2012, p. 193, 2012.
[58] M. Yagisawa, "Fully homomorphic encryption without bootstrapping," *IACR Cryptol. EPrint Arch.*, vol. 2015, p. 474, 2015.
[59] M. Yagisawa, "Improved fully homomorphic encryption with composite number modulus," *IACR Cryptol. EPrint Arch.*, vol. 2016, p. 50, 2016.
[60] F. Wang, K. Wang, and B. Li, "LWE-based FHE with better parameters," in *Proc. Int. Workshop Secur.*, 2015, pp. 175–192, doi: 10.1007/978-3-319-22425-1_11.
[61] K. Koyama and Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method," in *Advances in Cryptology—CRYPTO'92*. Berlin, Germany: Springer, 1992, pp. 345–357.

VANESSA MIRANDA-LÓPEZ received the bachelor's degree in electronics engineering from the Technological Institute of Sonora, Mexico, in 2006, and the master's degree in computer sciences from the CICESE Research Center, in 2010, where she is currently pursuing the Ph.D. degree in computer science in the area of data security for cloud environments. Her interests include cloud computing, grid scheduling, big data, security, and electronic design.

ANDREI TCHERNYKH (Member, IEEE) received the Ph.D. degree from the Institute of Precise Mechanics and Computer Technology, Russian Academy of Sciences, Russia, in 1986. He is currently a Full Professor with the Computer Science Department, CICESE Research Center, Ensenada, Baja California, Mexico, and an Adjunct Professor with the Institute for System Programming, RAS, Russia. He is also the Head of the Parallel Computing Laboratory, CICESE, and the Laboratory of Problem-Oriented Cloud Computing, South Ural State University, Russia. His main research interests include resource optimization technique, adaptive resource provisioning, multi-objective optimization, computational intelligence, incomplete information processing, cloud computing, and security. He is a member of the National System of Researchers of Mexico (SNI), Level II, and also leads several national and international research projects.

MIKHAIL BABENKO received the degree in mathematics and the Ph.D. degree in mathematics from Stavropol State University (SSU), in 2007 and 2011, respectively. He is currently the Head of the Computational Mathematics and Cybernetics Department, North-Caucasus Federal University. He is an author of more than 63 publications and five patents. His research interests include cloud computing, high-performance computing, residue number systems, neural networks, and cryptography.

ARUTYUN AVETISYAN (Senior Member, IEEE) received the master's degree in applied mathematics from Yerevan State University, in 1993, and the Ph.D. degree in computer science from the Institute for System Programming, RAS, in 2012. Since 2015, he has been the Director of the Institute for System Programming, Russian Academy of Sciences. He is currently the Head of Samsung Laboratory, Institute for System Programming, RAS, where he is also the Head of the NVIDIA Research Center. His research interests include program analysis and transformation, software security, parallel, and distributed computing.

VICTOR TOPORKOV received the D.Sc. degree in computer science from the Moscow Power Engineering Institute (MPEI), in 2000. He is currently the Head of the Computing Technologies Department, National Research University, MPEI, where he also holds a Full Professor position. From 1991 to 2000, he was the Head of the Advanced Computer Control in Avionics Laboratory (Ministry of Science and Education of Russia). He is also a Visiting Professor of the Wroclaw University of Science and Technology (Gold Medal Reward in IT). His primary research interests include resource management and scheduling in grid and cloud computing.

ALEXANDER YU DROZDOV received the M.Sc. degree in mathematics from Moscow State University, Russia, in 1988. He is currently a Full Professor with the Moscow Institute of Physics and Technology, Russia, and the Head of the Laboratory of Design and Modeling of Special-Purpose Computer Systems. His research interests include research and development of new high-performance architectures and embedded computing systems, embedded control systems, together with the development of tools, embedded, and system software.