# New Error Control Algorithms for Residue Number System Codes

Hanshen Xiao, Hari Krishna Garg, Jianhao Hu, and Guoqiang Xiao

**We propose and describe new error control algorithms for redundant residue number systems (RRNSs) and residue number system product codes. These algorithms employ search techniques for obtaining error values from within a set of values (that contains all possible error values). For a given RRNS, the error control algorithms have a computational complexity of $t \cdot O(\log_2 n + \log_2 \bar{m})$ comparison operations, where $t$ denotes the error correcting capability, $n$ denotes the number of moduli, and $\bar{m}$ denotes the geometric average of moduli. These algorithms avoid most modular operations. We describe a refinement to the proposed algorithms that further avoids the modular operation required in their respective first steps, with an increase of $\lceil \log_2 n \rceil$ to their computational complexity. The new algorithms provide significant computational advantages over existing methods.**

**Keywords: Chinese remainder theorem, residue number systems, redundant residue number systems, RNS product codes, mixed radix system, error control, computational complexity, permutations, CRT, RNS, RRNS, RNS-PC, MRS.**

Manuscript received June 25, 2015; revised Oct. 19, 2015; accepted Oct. 29, 2015.

Hanshen Xiao (xhs13@mails.tsinghua.edu.cn) is with the Department of Mathematics, Tsinghua University, Beijing, China.

Hari Krishna Garg (corresponding author, eleghk@nus.edu.sg) is with the Electrical & Computer Engineering Department, National University of Singapore, Singapore.

Jianhao Hu (jhhu@uestc.edu.cn) is with the National Science and Technology Key Lab of Communications, University of Electronic Science & Technology of China, Chengdu, China.

Guoqiang Xiao (gqxiao@swu.edu.cn) is with the College of Computer and Information Science, Southwest University, Chongqing, China.

## I. Introduction

*Error control techniques* play an important role in the reliability of signal transmission in communication systems. This is due to their capability to enhance the robustness of information transmission in noisy environments.

Residue number systems (RNSs) are used to express a computation in a large integer ring as a direct sum of computations in a number of smaller integer rings. These computations can be carried out in parallel.

The Chinese remainder theorem (CRT) forms a basis for an RNS. Redundant residue number systems (RRNSs) have recently been applied to communication systems, including wireless local area network (WLAN) [1], code division multiple access (CDMA) [2], space–time block codes [3], multicarrier modulation [4], wireless sensor networks [5], and cognitive radio [6]. Application of RRNSs to provide reliability in cloud storage services is described in [7].

In 1976, the concepts of *legitimate range* and *illegitimate range* were introduced [8], an important breakthrough in the field of RRNS-based error control.

One of the earliest schemes for single-burst errors and double errors was developed in 1992. It computes syndromes of the digits in a received vector and compares them with some observations [9], [10].

Knowledge of Hamming weight and minimum distance was used in detecting and correcting error, in [9] and [10]. Yang and Hanzo [11] used base extension (BEX) to achieve error correction in wireless channels suffering from low reliability. Their method requires decoding and encoding operations for each BEX process, and recovers digits by the CRT. In addition, Yang and Hanzo threw new light on the problem with the theory of minimum distance decoding [12].

An algorithm was proposed in [13] to recover directly the original digits from a received vector based on combinations of residues. The method used in [13] is based on continued fractions and Euclid's algorithm. It is a variant of the multiple error correction algorithm in [14]. The work of [13] and other works by the same authors related to [13] can be found in [15] and [16]. Relying on [12], Goldreich and others extended [13] with theories of maximum likelihood decoding (MLD) to determine error-location residues with fewer trials [17]. More recent work on scaling and error correction can also be found in [18].

The major contributions of this work are given below. In what follows, RRNSs and RNS-PCs are collectively termed as RNS codes (RNSCs). First, a unified mathematical framework for RNSCs is developed. Second, new computationally efficient algorithms for error control in an RNSC are described. Two cases are considered: (a) pure error correction and (b) simultaneous error correction and error detection. In both cases, the algorithms decode correctly if the number of errors is less than or equal to the error control capability of the RNSC. These algorithms provide significant computational savings in comparison to existing works. The methods described here are based on search techniques for use within an ordered set. Various refinements are described that simplify step 1 of these algorithms. The computational complexity of the proposed algorithms is $t \cdot \mathrm{O}(\log_2 n + \log_2 \bar{m})$. For a given RNSC, $t$ is the error correcting capability, $n$ is the number of moduli, and $\bar{m}$ is their geometric average. In comparison, the computational complexity of existing works is (i) $\mathrm{O}\left(b\cdot(\log_2 b)^a\right)$, where $a$ is a constant and $b=\sum_{i=1}^{n}\left(1+\lfloor \log_2 m_i \rfloor\right)$ in [13], [15], [16], and (ii) $\mathrm{O}(n^t)$ in [17].

The algorithms presented in this paper require fewer operations by comparison. In addition, they avoid most modular computations. Consequently, the proposed algorithms are expected to be simpler to implement and better suited for high-speed application environments. A preliminary analysis of hardware implementation supports this assertion.

This paper is organized as follows. Section II provides a brief description of the CRT and coding theory for RNSs. Sections III and IV are on the key contributions of this work. Error control algorithms are obtained in Section III and examples are presented to illustrate these algorithms. A refinement to the error control algorithms in Section III is described in Section IV — the refinement avoids modular operations required in the first step of each of the proposed algorithms. The refinement results in a slight increase in complexity. Further refinements are also described that simplify computations, once again, in the first step of each of the proposed algorithms. Computational comparisons with existing algorithms are presented in Section V. Section VI concludes this work.

## II. Mathematical Preliminaries

### 1. RNS

An RNS is a finite integer ring ($Z(M_K)$) defined by $k$ relatively co-prime moduli, $m_1, m_2, \ldots, m_k$, arranged in ascending order (without loss of generality). The range of the RNS is given by $[0, M_K)$, where

$$M_K = \prod_{i=1}^{k} m_i. \tag{1}$$

An integer $X \in [0, M_K)$ in the given RNS is represented by a vector, **x**, of length $k$, via the modular computation

$$X \leftrightarrow \mathbf{x} = (x_1\, x_2, \ldots, x_k), \tag{2}$$

where

$$x_i \equiv X \pmod{m_i} \qquad (i = 1, 2, \ldots, k). \tag{3}$$

RNSs express a computation in a large integer ring as a direct sum of computations in a number of smaller integer rings; the computations can be carried out in parallel. For instance, multiplication of integers $A$ and $B$ in $Z(M_K)$ is computed as $k$ parallel multiplications $a_i \cdot b_i \pmod{m_i}$, $i = 1, \ldots, k$.

### 2. Permutation

Given a set of integers $\mathbf{S} = \{s_1, s_2, \ldots, s_\lambda\}$ of cardinality $\lambda$, we define a permutation of **S** as a rearrangement of the elements in **S** in a different order. When **S** is an RNS $Z(M_K)$, one such permutation is $\mathbf{S} = \{P \cdot s_1, P \cdot s_2, \ldots, P \cdot s_\lambda\}$. Here, $P$ is an integer, and multiplication $P \cdot s_a$ is conducted modulo $M_K$. Not all values of $P$ lead to a permutation. The necessary and sufficient condition is that $P$ and $M_K$ must be co-prime; that is, $\gcd(P, M_K) = 1$. For an RNS $Z(M_K)$, a permutation gives rise to a one-to-one mapping, represented by

$$X' \equiv P \cdot X \pmod{M_K}. \tag{4}$$

As $X$ takes values in the set $\{0, 1, \ldots, M_K - 1\}$, $X'$ also takes values in the same set, though in a different order. Any $P$ such that $\gcd(P, M_K) = 1$ can be used. Such integers always exist. In fact, there are $\phi(M_K)$ such values in the range $(0, M_K)$; $\phi(M)$ being the Euler totient function of $M$. For any given permutation, the following two properties hold:

- $X = 0$ if and only if $X' = 0$.
- $x_i' \equiv X' \pmod{m_i} = p_i \cdot x_i \pmod{m_i}$, $p_i \equiv P \pmod{m_i}$, $i = 1, 2, \ldots, k$.

We are interested in values of $P$ that are associated with CRT computations and RNSCs. In addition, we show that the use of permuted residues simplify decoding algorithms in some cases.

## 3. CRT-I

Given **x**, computation of $X$, $0 \le X < M_K$, can be done via the CRT; this can be stated as follows:

$$X \equiv \sum_{i=1}^{k} a_i \cdot x_i \cdot \left(\frac{M_K}{m_i}\right) (\text{mod}\, M_K). \tag{5}$$

The scalar $a_i$, is computed a-priori by solving the congruence

$$a_i \cdot \left(\frac{M_K}{m_i}\right) \equiv 1\ (\text{mod}\ m_i) \qquad (i = 1, 2, \ldots, k). \tag{6}$$

It is clear from (6) that

$$\gcd(a_i, m_i) = 1, \qquad (i = 1, 2, \ldots, k). \tag{7}$$

The CRT computation in (5) can be performed in two steps:
Step 1: Compute the permuted residues

$$x_i' \equiv a_i \cdot x_i\ (\text{mod}\, m_i) \qquad (i = 1, 2, \ldots, k). \tag{8}$$

Step 2: Compute $X$ as

$$X \equiv \sum_{i=1}^{k} x_i' \cdot \left(\frac{M_K}{m_i}\right) (\text{mod}\ M_K). \tag{9}$$

Computation of $X$ from **x** necessarily involves large integers as the dynamic range of RNS is large. For the integers $a_i$ ($i = 1, 2, \ldots, k$) in (5), consider the integer $T$ ($0 < T < M_K$) such that

$$T \equiv a_i\ (\text{mod}\ m_i) \qquad (i = 1, 2, \ldots, k). \tag{10}$$

The integer $T$ can be obtained using the CRT. Stated explicitly,

$$\begin{aligned} T &\equiv \sum_{i=1}^{k} a_i \cdot a_i \cdot \left(\frac{M_K}{m_i}\right) (\text{mod}\ M_K) \\ &= \sum_{i=1}^{k} a_i^2 \cdot \left(\frac{M_K}{m_i}\right) (\text{mod}\ M_K). \end{aligned} \tag{11}$$

It follows from (7) and (10) that

$$\gcd(T, M_K) = 1. \tag{12}$$

### A. CRT-II: Using Permuted Residues

Based on (8), we may also use permuted residues

$$\mathbf{x}' = \left(x_1' x_2', \ldots, x_k'\right), \tag{13}$$

instead of $\mathbf{x} = (x_1\, x_2 \cdots x_k)$ in residue arithmetic. In such a case, $X$ is obtained from (9) in a direct manner. It is clear from (5), (8), and (12) that $X \leftrightarrow \mathbf{x} = (x_1\ x_2, \ldots, x_k)$ and $X' \leftrightarrow \mathbf{x}' = (x_1' x_2', \ldots, x_k')$ are related via the permutation

$$X' = T \cdot X\ (\text{mod}\ M_K). \tag{14}$$

Given $X'$, $X$ can be recovered as

$$X \equiv T^{-1} \cdot X'\ (\text{mod}\ M_K). \tag{15}$$

A unique $T^{-1}$ (mod $M_K$) always exists due to (12).

### B. CRT-III Computation with Permutation

Given the CRT in (5), we can also write

$$X \equiv T \cdot \sum_{i=1}^{k} x_i \cdot \left(\frac{M_K}{m_i}\right) (\text{mod}\, M_K). \tag{16}$$

This provides an alternative expression and a way to compute $X$ from its residues in the given RNS. The CRT computation based on (16) can be performed in two steps:
Step 1: Compute

$$X'' \equiv \sum_{i=1}^{k} x_i \cdot \left(\frac{M}{m_i}\right). \tag{17}$$

Step 2: Compute $X$ as

$$X \equiv T \cdot X''\ (\text{mod}\ M_K). \tag{18}$$

The three formulations of the CRT, enumerated as CRT-I, CRT-II, and CRT-III, are equivalent and have their own computational features, thus making each one uniquely suitable for a certain computational environment (as shown in refinement 3 in Section IV). In essence, $T$ establishes a permutation between (i) $X'$ and $X$ via (15), and (ii) $X$ and $X''$ via (18). We note that for a given RNS, $T$ is fixed and needs to be computed once only. Finally, a permutation in an RNS can be carried out via any $P$ such that $\gcd(P, M_K) = 1$. A permutation is used to establish decoding algorithms for an RNS-PC.

## 4. Mixed Radix System (MRS)

An MRS can be used to compute $X$ from its residues **x**. In an MRS, $X$ is represented as follows:

$$X = y_1 + y_2 \cdot m_1 + \cdots + y_n \cdot (m_1 \cdot m_2, \ldots, m_{k-1}).$$

The mixed radix digits satisfy $0 \le y_i < m_i$. They are computed from **x** using RNS-to-MRS conversion algorithms [19].

In our work, integer $X$ can be computed from its residues using either the CRT or the MRS. Such a computation

Table 1. Permutations and CRT computations.

| $X$ | $\underline{\mathbf{x}}$ | $X'$ | $\underline{\mathbf{x}'}$ | $X''$ | $\underline{\mathbf{x}''}$ |
|---|---|---|---|---|---|
| 1 | (1 1) | 2 | (2 2) | 8 | (2 3) |
| 2 | (2 2) | 4 | (1 4) | 1 | (1 1) |
| 3 | (0 3) | 6 | (0 1) | 9 | (0 4) |
| 4 | (1 4) | 8 | (2 3) | 2 | (2 2) |
| 13 | (1 3) | 11 | (2 1) | 14 | (2 4) |

constitutes the first step in the new decoding algorithms. We also use the CRT to establish mathematical aspects of the new decoding algorithms. The choice of CRT versus MRS for an actual computation rests with the system designer.

*Example 1.* Consider an RNS defined by residues $m_1 = 3$ and $m_2 = 5$. Then, we have $M_K = m_1 \cdot m_2 = 15$, and $X$ is an integer in the range (0, 15). The values $a_1$ and $a_2$ needed for the CRT are obtained by solving congruences $a_1 \cdot 5 \equiv 1 \pmod 3$ and $a_2 \cdot 3 \equiv 1 \pmod 5$. This results in $a_1 = a_2 = 2$. Using the CRT on $a_1$ and $a_2$, we get $T = 2$ and $T^{-1} = 8 \pmod{15}$. Selected integers $X$, $X'$, and $X''$ for the permutations in (15) and (18) and their residues are given in Table 1.

## 5. RRNS

An RRNS is obtained by appending $(n - k)$ additional relatively co-prime moduli, $m_{k+1}, \ldots, m_n$, to an RNS defined by moduli $m_1, m_2, \ldots, m_k$. We further assume that moduli $m_1, m_2, \ldots, m_k, m_{k+1}, \ldots, m_n$ are arranged in an ascending order. An RRNS is a $(n, k)$ code with redundancy given by

$$M_R = \prod_{i=k+1}^{n} m_i . \tag{19}$$

An integer $X \in [0, M_K)$ in the given RNS is represented by a codeword, $\mathbf{x}$, of length $n$, via modular computations as follows:

$$X \leftrightarrow \mathbf{x} = (x_1 \quad x_2, \ldots, x_k \quad x_{k+1}, \ldots, x_n), \tag{20}$$

where

$$x_i \equiv X \pmod{m_i} \qquad (i = 1, 2, \ldots, n). \tag{21}$$

The $k$ residues $(x_1\ x_2, \ldots, x_k)$ constitute the information digits, and the $(n - k)$ residues $(x_{k+1}, \ldots, x_n)$ the parity digits. Let

$$M_N = M_K \cdot M_R. \tag{22}$$

Given a vector, $\mathbf{a}$, the Hamming weight of $\mathbf{a}$, denoted by H($\mathbf{a}$), is the number of non-zero elements in $\mathbf{a}$. The Hamming distance between two vectors is the number of places in which the two vectors differ. The Hamming distance between two codewords is same as the Hamming weight of another codeword. The Hamming weight of any non-zero codeword in an RRNS is at least $d = n - k + 1$. An RRNS is a maximum distance separable (MDS) code having minimum distance $d$. These properties follow even though RRNSs are nonlinear [19].

*Example 2.* Consider a (4, 2) RRNS defined by $(m_1\ m_2\ m_3\ m_4) = (11\ 13\ 14\ 15)$. The legitimate range is (0, 143), $M_R = 210$. This is a minimum distance 3 RRNS. Given an integer $X = 25 \in (0, 143)$, we have $\mathbf{x} = (3\ 12\ 11\ 10)$. If we extend this RRNS with a third redundant modulus, say $m_5 = 17$, then we get a (5, 2) minimum distance 4 RRNS.

## 6. RNS-PC

An RNS-PC is defined by $n$ moduli $(m_1\ m_2, \ldots, m_n)$ such that all codewords when converted to equivalent integer form are divisible by a code-generator integer $G$, gcd($G$, $M_N$) = 1. Thus, for an integer $X$ in the legitimate range $(0, M_K)$, $M_K = \lfloor M_N/G \rfloor + 1$, the corresponding codeword $\mathbf{x}$ is obtained as $G \cdot X \leftrightarrow \mathbf{x} = (g_1{\cdot}x_1\ g_2{\cdot}x_2, \ldots, g_n{\cdot}x_n)$, where multiplications are conducted mod respective moduli. Here, $\lfloor \theta \rfloor$ is the well-known floor function. The minimum distance for an RNS-PC is $d$, if and only if $G$ satisfies $\prod_{i=1}^{d} m_{n-i} > G \geq \prod_{i=1}^{d-1} m_{n-i}$. RNS-PCs can be studied as an equivalent of cyclic codes over finite fields.

*Example 3.* Consider an RNS-PC defined by $(m_1\ m_2\ m_3\ m_4) = (13\ 16\ 17\ 19)$. For a minimum distance 3 RNS-PC, we have $m_4{\cdot}m_3{\cdot}m_2 > G \geq m_4{\cdot}m_3$ or $5{,}168 > G \geq 323$. Choosing $G = 327$, we have (0, 206) as the legitimate range for this RNS-PC.

## III. Unified Framework and Algorithms for Error Control in RNS

Together, RRNSs and RNS-PCs will be referred to as RNSCs. A minimum distance $d$ RNSC can correct up to $t$ residue errors (also called its error-correcting capability), where

$$t = \lfloor (d-1)/2 \rfloor . \tag{23}$$

A code that simultaneously corrects up to $\alpha$ residues and detects up to $\beta$ ($\beta > \alpha$) residues in error has minimum distance

$$d = \alpha + \beta + 1. \tag{24}$$

It is clear that the (4, 2) RNSC in example 2 can correct a single error, while the (5, 2) RNSC can simultaneously correct single errors and detect double errors. In the following, we propose new algorithms for error control for RNSCs.

### 1. Error Phenomenon

For a transmitted codeword $\mathbf{x}$, an additive error, $e_i$, ($e_i \neq 0$) in the $i$th residue results in a received residue vector, $\mathbf{y}$.

$$\mathbf{y} = \mathbf{x} + \mathbf{e}, \tag{25}$$

where

$$y_i \equiv x_i + e_i \pmod{m_i} \qquad (i = 1, 2, \ldots, n). \tag{26}$$

For an error, both its location and value are unknown. The Hamming weight of $\mathbf{e}$, H($\mathbf{e}$), is the number of errors in $\mathbf{y}$. Using a triangular inequality, we have H($\mathbf{y}$) ≥ H($\mathbf{x}$) – H($\mathbf{e}$), or

$$\mathrm{H}(\mathbf{y}) \geq d - \mathrm{H}(\mathbf{e}). \tag{27}$$

The error model in (25) and (26) is adopted all throughout this work for various error control algorithms.

The first step in error control algorithms in RNSCs being described here is as follows. Given **y**, compute $Y$ from **y** and check if it is in legitimate range $[0, M_K)$ for the given RNSC. Thus far, the first step in decoding of an RNS-PC has been computation of $Y$ from **y** and checking if $G \mid Y$ [19]. In the following, we first unify step 1 for RRNS and RNS-PC leading to a unified framework for error control in RNSCs. Error control algorithms for RNSCs is then presented.

2. RRNS

Consider an $(n, k)$ RRNS with $H(\mathbf{x}) \geq d$, $\mathbf{x} \neq \mathbf{0}$, and a received residue vector **y** in (25).

**Lemma 1.** If up to $t$ errors occur, then

$$0 \leq E \leq M_N - \prod_{i=1}^{n-t} m_i , \tag{28}$$

$$0 \leq X + E < M_N, \tag{29}$$

where $E$ is the error integer corresponding to residue vector **e**.

**Proof**. If $H(\mathbf{e}) = a$, then **e** has $a$ non-zero and $(n-a)$ zero residues. Let **e** be non-zero in locations $\mathbf{I} = (i_1, i_2, \ldots, i_a)$ and 0 in all others. If a residue is 0, then $E$ is a multiple of corresponding modulus. Thus, $E$ is a multiple of all moduli $m_i$, $i = 1, \ldots, n$, that do not belong to **I**. The largest value of $E$ such that it is non-zero in locations **I** is given by

$$\begin{aligned} 0 \leq E &\leq \left(\prod_{\substack{i=1 \\ i\notin \mathbf{I}}}^{n} m_i\right)\left(\prod_{\substack{i=1 \\ i\in \mathbf{I}}}^{n} m_i - 1\right), \\ &= \prod_{\substack{i=1 \\ i\notin \mathbf{I}}}^{n} m_i \prod_{\substack{i=1 \\ i\in \mathbf{I}}}^{n} m_i - \prod_{\substack{i=1 \\ i\notin \mathbf{I}}}^{n} m_i, \\ &= \prod_{i=1}^{n} m_i - \prod_{\substack{i=1 \\ i\notin \mathbf{I}}}^{n} m_i = M_N - \prod_{\substack{i=1 \\ i\notin \mathbf{I}}}^{n} m_i. \end{aligned}$$

The largest value of $E$ implies that the moduli "not belonging" to set **I** is the $(n-a)$ smallest moduli. Hence,

$$0 \leq E \leq M_N - \prod_{i=1}^{n-a} m_i .$$

Again, the largest value of $E$ is obtained when $a$ is as large as possible. We note that Lemma 1 is applicable if $a \leq t$. Settin $a = t$ leads to (28). Equation (29) is obtained by adding $M_K$ to both sides of (28) and noting that $M_K = \prod_{i=1}^{k} m_i \leq \prod_{i=1}^{n-t} m_i$. ■

Using Lemma 1, we can express (25) in integer form as

$$Y = X + E \pmod{M_N} = X + E. \tag{30}$$

Here, $X$ has residue vector **x** and $E$ has residue vector **e** having 0's in all locations except for residue positions where errors have occurred. The first step in the error control algorithms for RRNSs being described here is to compute $Y$ from **y**. This can be done using either the CRT or the MRS.

3. RNS-PC

Expressing (25) in integer form for RNS-PC, we have $Y = G \cdot X + E \pmod{M_N}$. To test whether $Y$ is a codeword or not, the existing algorithms for RNS-PC compute $Y$ first and then test if $G \mid Y$ [19]. We now merge the two steps to compute

$$\begin{aligned} Y_1 &\equiv G^{-1} \cdot Y \pmod{M_N} = X + G^{-1} \cdot E \pmod{M_N} \\ &= X + F \pmod{M_N}, \end{aligned} \tag{31}$$

where $F = G^{-1} \cdot E \pmod{M_N}$. Computationally,

$$Y_1 \equiv \sum_{i=1}^{n} y_i' \cdot \left(\frac{M_N}{m_i}\right) \left(\bmod M_N\right), \tag{32}$$

where the permuted residues $y_i'$ are given by

$$\begin{aligned} y_i' &= \left(g_i^{-1} \cdot a_i\right) \cdot y_i \left(\bmod m_i\right) \\ &= a_i \cdot x_i + \left(g_i^{-1} \cdot a_i\right) \cdot e_i \left(\bmod m_i\right). \end{aligned} \tag{33}$$

Thus, $F$ is a permuted version of $E$ via $(G^{-1} \pmod{M_N})$ and,

$$Y_1 = X + F. \tag{34}$$

Two key properties of such a computation are as follows:

- P1: $H(\mathbf{f}) = H(\mathbf{e})$.
- P2: $Y$ is a codeword if and only if $Y_1$ is a legitimate integer; that is, $0 \leq Y_1 < M_K$.

Writing $F = G^{-1} \cdot E \pmod{M_N}$ in residue form, $\mathbf{f} = G^{-1} \cdot \mathbf{e}$ or $f_i = g_i^{-1} \cdot e_i \left(\bmod m_i\right)$, $g_i \equiv G \pmod{m_i}$, $i = 1, 2, \ldots, n$. Since $\gcd(g_i, m_i) = 1$, $f_i = 0$ if and only if $e_i = 0$. This establishes P1, thereby showing that if **e** is a correctable/detectable error vector, then so is **f**, and vice-versa. As P1 is valid, (28) holds for $F$. P2 follows from (28) and (31) through (34). P1 and P2 taken with Lemma 1 further lead to $Y_1 = X + F$, an expression for RNS-PCs that is equivalent to (30) for RRNSs.

In essence, divisibility of $Y$ by $G$ is now replaced by testing for $Y_1$ being in legitimate range $[0, M_K)$. This unifies error control methods for RRNSs and RNS-PCs. In an RRNS, given $\mathbf{y} = \mathbf{x} + \mathbf{e}$, we compute $Y$ from **y** with $Y = X + E$. In an RNS-PC, given $\mathbf{y} = G \cdot \mathbf{x} + \mathbf{e}$, we compute $Y_1$ from **y** with $Y_1 = X + F$ and $H(\mathbf{f}) = H(\mathbf{e})$. Moving forward, we denote $Y_1$ by $Y$ and $F$ by $E$ in our description of error control algorithms within the unified framework for RNSCs. Again, the computation of $Y$ from its residues can be carried out using either the CRT or the MRS.

Given $Y$, the task of any decoding algorithm is to estimate $X$ and $E$, denoted by $\hat{X}$ and $\hat{E}$, respectively, such that $Y = \hat{X} + \hat{E}$. The decoding algorithm decodes "correctly" if $X = \hat{X}$. Correct decoding must take place if the number of errors in **y** does not exceed the error-correcting capability of the RNSC. The following two cases follow from this:

- Case I: When used purely for error correction, correct decoding must take place if H($\mathbf{e}$) $\le t$.

- Case II: When used for simultaneous error correction and detection, correct decoding takes place if H($\mathbf{e}$) $\le \alpha$. Further, the decoding algorithm detects errors if $\alpha <$ H($\mathbf{e}$) $\le \beta$.

Some general properties of an RNSC are as follows. Every legitimate integer $X$ and the corresponding codeword $\mathbf{x}$ satisfy

$$0 \le X < M_K, \tag{35}$$

$$\mathrm{H}(\mathbf{x}) \ge d \qquad (\mathbf{x} \ne \mathbf{0}). \tag{36}$$

Hence an arbitrary integer $R$, such that H($\mathbf{r}$) $< d$, is an illegitimate integer; that is, $M_K \le R < M_N$. It follows that if there are $a$ errors in $\mathbf{y}$, then H($\mathbf{e}$) $= a$ and H($\mathbf{y}$) $\ge d - a$.

**Lemma 2.** If there are at most $(d-1)$ errors in $\mathbf{y}$ (that is, H($\mathbf{e}$) $< d$), then $Y$ is an illegitimate integer.

**Proof**. All codewords differ in at least $d$ places. As $\mathbf{x}$ and $\mathbf{y}$ differ in $(d-1)$ places or less, $\mathbf{y}$ cannot be a codeword. ■

For an RNSC, we construct a set $\mathbf{V}_l$, referred to as an "$l$-error-set," which contains all possible integers $E$ such that H($\mathbf{e}$) $= l$. Thus, we have

$$\mathbf{V}_l = \{E, \forall \mathbf{e}\text{: H}(\mathbf{e}) = l\}. \tag{37}$$

Based on $\mathbf{V}_l$, we construct a set $\mathbf{U}_b$, referred to as an "error-set," which contains all possible integers $E$ such that $1 \le$ H($\mathbf{e}$) $\le b$. Mathematically, we have

$$\mathbf{U}_b = \{E, \forall \mathbf{e}\text{: } 1 \le \mathrm{H}(\mathbf{e}) \le b\} = \mathbf{V}_1 \cup \mathbf{V}_2 \cup, \dots, \cup \mathbf{V}_b. \tag{38}$$

We further assume that the elements in $\mathbf{U}_b$ are listed in an ascending order. It is clear that all integers in $\mathbf{V}_l$ and hence $\mathbf{U}_b$ are illegitimate as long as $b < d$. The error-set $\mathbf{U}_b$ is computed only once and then stored for use in the decoding algorithm. To reinforce the unified framework for error control in RNSCs, we note that given the permutation $F \equiv P \cdot E \pmod{M_N}$, the elements in the set $\mathbf{V}_l$ are the same for both $F$ and $E$ as H($\mathbf{f}$) $=$ H($\mathbf{e}$). Hence, under the condition that the same moduli are used for both, the sets $\mathbf{U}_b$ for an RRNS and an RNS-PC are identical. The only difference is that the error vectors for such an RRNS and RNS-PC are associated with their integer representation via the permutation $P \equiv G^{-1} \pmod{M_N}$.

The cardinality of $\mathbf{U}_b$ (that is, the number of error vectors $\mathbf{e}$ having a Hamming weight in the range $0 <$ H($\mathbf{e}$) $\le b$) can be approximated by a polynomial in $n$ of degree $b$ as follows:

$$\left|\mathbf{U}_b\right| \approx \sum_{l=1}^{b} {}^{n}C_l \left(\bar{m}-1\right)^l . \tag{39}$$

We compute $\bar{m}$ as the geometric mean of all the residues,

$$\bar{m} = \left(\prod_{i=1}^{n} m_i\right)^{1/n} = M_N^{\;1/n}. \tag{40}$$

Here, ${}^{n}C_r = n!/[r!(n-r)!]$ is the combinatorial function. The expression in (39) is exact if moduli are equal. Since they are not, we take geometric mean to estimate the cardinality of $\mathbf{U}_b$.

*Example 4.* Consider a (4, 2) RRNS defined by $(m_1\ m_2\ m_3\ m_4) = (2\ 3\ 5\ 7)$. Let $b = t = 1$. The integers $E$ included in $\mathbf{U}_1$ along with their respective 4-dimensional vectors are 105 ← (1 0 0 0), 70 ← (0 1 0 0), 140 ← (0 2 0 0), 126 ← (0 0 1 0), 42 ← (0 0 2 0), 168 ← (0 0 3 0), 84 ← (0 0 4 0), 120 ← (0 0 0 1), 30 ← (0 0 0 2), 150 ← (0 0 0 3), 60 ← (0 0 0 4), 180 ← (0 0 0 5), and 90 ← (0 0 0 6). The error-set including the aforementioned integers sorted in ascending order is $\mathbf{U}_1$ = {30, 42, 60, 70, 84, 90, 105, 120, 126, 140, 150, 168, 180}. In this case, $|\mathbf{U}_1| = 13$. The approximated values are $\bar{m} = 4$ and $|\mathbf{U}_1| \approx 12$ as per (39). Similarly, $|\mathbf{U}_2| = 69$, while the approximate value is $|\mathbf{U}_2| \approx 67$.

*Example 5.* Consider an RNS-PC defined by the same residues as in example 4 with $G = 37$, $G^{-1} = 193$. Let $b = t = 1$. All the integers $E$ included in $\mathbf{U}_1$ along with their respective 4-dimensional vectors obtained via the permutation $P \equiv G^{-1} \pmod{M_N}$ are 105 ← (1 0 0 0), 70 ← (0 1 0 0), 140 ← (0 2 0 0), 126 ← (0 0 3 0), 42 ← (0 0 1 0), 168 ← (0 0 4 0), 84 ← (0 0 2 0), 120 ← (0 0 0 4), 30 ← (0 0 0 1), 150 ← (0 0 0 5), 60 ← (0 0 0 2), 180 ← (0 0 0 6), and 90 ← (0 0 0 3). The error-set including the aforementioned integers sorted in ascending order is the same as in example 4.

We now describe a decoding algorithm for the two cases, separately, though the analysis is similar.

## 4. Case I

Error correction up to $t$ errors; $0 <$ H($\mathbf{e}$) $\le t$. For correcting up to $t$ errors, we create the error set $\mathbf{U}_t$. Given (19) and (22) and the ensuing discussion, it is clear that $Y$ and $X$ are illegitimate and legitimate integers, respectively, $Y = X + E$ ($E \in \mathbf{U}_t$). In addition, $E < Y$. The following theorem plays a key role in the formulation of new decoding algorithms.

**Theorem 1.** Given $Y = X + E$, $X$ and $E$ are unique for H($\mathbf{e}$) $\le t$.

**Proof**. If these values are not unique, then there are at least two sets $(X_1, E_1)$ and $(X_2, E_2)$ such that

$$Y = X_1 + E_1 = X_2 + E_2. \tag{41}$$

Without any loss in generality, let $X_1 > X_2$. Rearranging (41),

$$X_1 - X_2 = E_2 - E_1. \tag{42}$$

Let $X_3 = X_1 - X_2$ and $E_3 = E_2 - E_1$. Thus, $X_3$ is legitimate with H($\mathbf{x}_3$) $\ge 2t + 1$, while H($\mathbf{e}_3$) $\le$ H($\mathbf{e}_2$) + H($\mathbf{e}_1$) $\le t + t = 2t$. Clearly, (41) is not possible, as the Hamming weight of the left side of (42) can never equal the Hamming weight of the right side of (42). ■

Theorem 1 leads to the following new decoding algorithms.

**Decoding Algorithm 1.** Given $Y$, test if $E = 0$. If not, then find the largest integer $\hat{E}$ in $\mathbf{U}_t$ such that $\hat{E} \le Y$. Thus, the decoding algorithm finds $\hat{E}$ as "the largest integer in $\mathbf{U}_t$ less than or equal to $Y$." The corresponding $\hat{X}$ is computed as

$$\hat{X} = Y - \hat{E}. \tag{43}$$

This analysis also leads us to conclude that the difference between any two arbitrarily selected elements in $\mathbf{U}_t$ is at least $M_K$. Finally, noting that $0 \le X < M_K$, we have

$$Y - M_K < E \le Y. \tag{44}$$

A flow chart for new decoding algorithm 1 is shown in Fig. 1. The bulk of the complexity consists in searching $\mathbf{U}_t$ to determine $\hat{E}$. There are numerous binary tree–based search algorithms known in the literature that have a complexity of O($\log_2 A$), where $A$ is the cardinality of the ordered set $A$. In essence, these algorithms are based on the "divide-and-conquer" technique. Via (39), $|\mathbf{U}_t|$ can be approximated by a polynomial of the kind $|\mathbf{U}_t| \approx a \cdot n^t \cdot \bar{m}^t$. Hence, the complexity of decoding algorithm 1 is $t \cdot$ O($\log_2 n + \log_2 \bar{m}$). The error-set $\mathbf{U}_t$ is straightforward to obtain for a given RNSC, as is shown in examples 4 and 5.

*Example 6.* Consider the same RRNS as in example 4. Let $X = 3$, $\mathbf{x} = (1\ 0\ 3\ 3)$, and an error be introduced in the second residue to obtain $\mathbf{y} = (1\ 1\ 3\ 3)$. Therefore, $Y = 73$, $Y - M_K = 67$. Thus, we need to search in $\mathbf{U}_1$ to find $E$ within (67, 73). There are 13 elements in $\mathbf{U}_1$ as listed in example 4. Let us denote them by $U_1(i)$, $1 \le i \le 13$. First, compare $U_1(7) = 105$ with 73 to get $105 > 73$. Since $105 > Y$, we have $E$ as one of $\{U_1(1), \ldots, U_1(6)\}$. Second, compare $U_1(4) = 70$ with 73 to get $70 < 73$, thus further narrowing $E$ as one of $\{U_1(4), \ldots, U_1(6)\}$. Third, compare $U_1(5) = 84$ with 73 to get $84 > 73$. This eliminates $U_1(5)$ and $U_1(6)$, $\hat{E} = U_1(4) = 70$ with $\hat{X} = Y - \hat{E} = 3$.

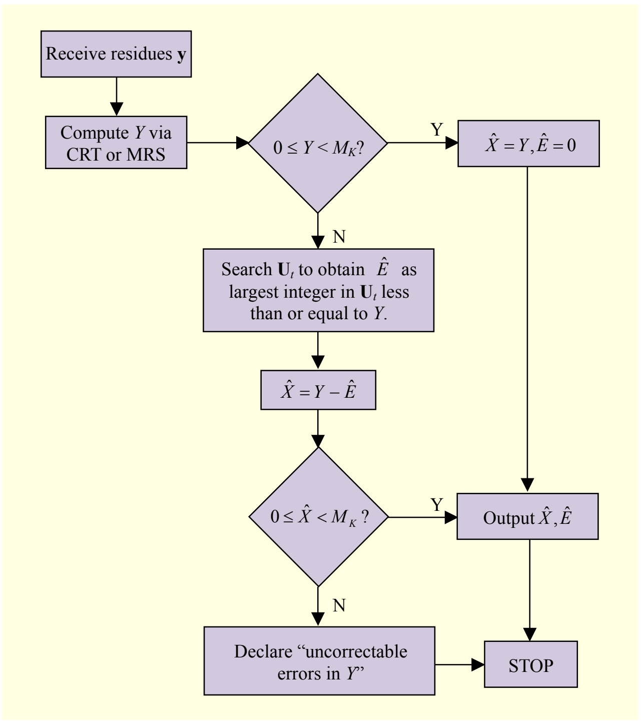
Following RNS-PC in example 5, let $X = 3$ with the codeword $G \cdot X \leftrightarrow (1\ 0\ 1\ 6)$. Let an error be introduced in the second residue to obtain $\mathbf{y} = (1\ 1\ 1\ 6)$. Therefore, in step 1, we compute the permuted version of $\mathbf{y}$ using ($G^{-1}$ (mod $M_N$)) (193 (mod 210)) to obtain $Y = 73$, $Y - M_K = 67$. The rest of the steps are identical to the steps for RRNS.

*Example 7.* Consider a double error correcting (10, 6) RRNS defined by $(m_1, \ldots, m_{10}) = (23\ 25\ 27\ 29\ 31\ 32\ 67\ 71\ 73\ 79)$. In this case, $t = 2$ and $|\mathbf{U}_2| = 87{,}899$.

## 5. Case II

Simultaneous error correction up to $\alpha$ errors and detection up to $\beta$ errors; $0 < \mathrm{H}(\mathbf{e}) \le \alpha$, $\alpha < \beta$. For correcting up to $\alpha$ errors, we create the error-set $\mathbf{U}_\alpha$. Based on a similar analysis as in Case I, the new decoding algorithm for Case II is as follows.

**Decoding algorithm 2.** Given $Y$, test if $E = 0$. If not, then find the largest integer $\hat{E}$ in $\mathbf{U}_\alpha$ such that $\hat{E} \le Y$. The corresponding $\hat{X}$ then is computed as $\hat{X} = Y - \hat{E}$. Theorem 1 also holds in this case. We note that Case II requires fewer comparisons as $|\mathbf{U}_\alpha| < |\mathbf{U}_t|$ for $\alpha < t$. The error control algorithm for simultaneous correction of up to $\alpha$ and detection of up to $\beta$ errors ($d = \alpha + \beta + 1$, $\alpha < \beta$) is almost the same as the one described in Fig. 1 with $\mathbf{U}_t$ replaced by $\mathbf{U}_\alpha$.



Receive residues y
Compute Y via CRT or MRS
$0 \le Y < M_K$?
Y
$\hat{X} = Y, \hat{E} = 0$
N
Search $\mathbf{U}_t$ to obtain $\hat{E}$ as largest integer in $\mathbf{U}_t$ less than or equal to $Y$.
$\hat{X} = Y - \hat{E}$
$0 \le \hat{X} < M_K$?
Y
Output $\hat{X}, \hat{E}$
N
Declare "uncorrectable errors in Y"
STOP

Fig. 1. Flow chart for decoding algorithm for case I.

*Example 8.* Consider a (16, 10) RRNS defined by $(m_1, \ldots, m_{16}) = (23\ 29\ 31\ 32\ 35\ 37\ 39\ 41\ 43\ 47\ 53\ 59\ 61\ 67\ 71\ 73)$. In this case, we can use $\mathbf{U}_3$ for correcting three errors, $|\mathbf{U}_3| = 51{,}159{,}743$, or $\mathbf{U}_2$ for simultaneous correction of up to two and detection of up to four errors, $|\mathbf{U}_2| = 245{,}231$.

*Example 9.* Consider the same RRNS as in example 7. Let us use it for simultaneous correction of one error and detection of up to three errors. Here, $M_K = 446{,}623{,}200$, $|\mathbf{U}_1| = 447$. Let $X = 15$, $\mathbf{x} = (15, \ldots, 15)$, and two errors be introduced to obtain $\mathbf{y} = (16\ 15, \ldots, 15\ 75)$. First, $Y = 2{,}285{,}962{,}767{,}813{,}615$. Ten comparisons are needed in the decoding algorithm.

- 1: $U_1(224) = 6{,}203{,}792{,}762{,}208{,}000 > Y$,
- 2: $U_1(112) = 3{,}161{,}933{,}085{,}254{,}400 > Y$,
- 3: $U_1(56) = 1{,}645{,}856{,}960{,}421{,}600 < Y$,
- 4: $U_1(84) = 2{,}377{,}348{,}942{,}831{,}200 < Y$,
- 5: $U_1(70) = 2{,}014{,}108{,}061{,}155{,}200 < Y$,
- 6: $U_1(77) = 2{,}194{,}475{,}947{,}228{,}800 < Y$,
- 7: $U_1(81) = 2{,}326{,}422{,}285{,}828{,}000 > Y$,
- 8: $U_1(79) = 2{,}268{,}979{,}760{,}252{,}000 < Y$,
- 9: $U_1(80) = 229{,}734{,}2007{,}255{,}150 > Y$,
- 10: $\hat{E} = U_1(79) = 2{,}268{,}979{,}760{,}252{,}000$, $\hat{X} = Y - \hat{E} = 16{,}983{,}007{,}561{,}615 > M_K$.

Thus, "uncorrectable errors in $Y$" is declared.

## IV. Further Refinements

### 1. Refinement 1

One may reason that computation of $Y$ in the first step of the new decoding algorithms described here still requires a modular operation (mod $M_N$) if the CRT is used to compute $Y$. We now describe a refinement, whereby such an operation may be avoided at an increase of $\lceil \log_2 n \rceil$ to the number of comparisons. The first step in the decoding algorithms is the computation of $Y$, which can be carried out using either the CRT or the MRS. Mathematically, $Y$ can be represented by

$$Y \equiv \sum_{i=1}^{n} y_i' \cdot \left( \frac{M_N}{m_i} \right) (\bmod M_N), \tag{45}$$

where $y_i' \equiv y_i \cdot a_i \ (\bmod m_i)$, $i$ = 1, 2, … , $n$. Given $\mathbf{y}$, the computation in (45) can equivalently be expressed as

$$Y' \equiv \sum_{i=1}^{n} y_i' \cdot \left( \frac{M_N}{m_i} \right), \tag{46}$$

$$Y \equiv Y' \ (\bmod M_N). \tag{47}$$

To avoid the modular operation in (47), one may compute $Y'$ in (46) instead of $Y$ in (47). Recalling (30), we may write $Y'$ as

$$Y' = Y + \Delta \cdot M_N = X + E + \Delta \cdot M_N, \tag{48}$$

where $\Delta$ is an unknown that satisfies $0 \le \Delta < n$. Based on (48), we construct $n$ sets, $\mathbf{A}_i$, $i = 0, \ldots, n-1$, as follows:

$$\mathbf{A}_i = \{\text{all values of } X + E + i \cdot M_N;\ 0 \le X < M_K,\ \mathrm{H}(\mathbf{e}) \le t\}. \tag{49}$$

Since $0 < X + E < M_N$ for $\mathrm{H}(\mathbf{x}) \ge 2t + 1$ and $\mathrm{H}(\mathbf{e}) \le t$, these sets are disjoint. We create a super error-set $\mathbf{U}_b^{\mathrm{S}}$, an $n$-fold replica of the previously described error-set $\mathbf{U}_b$, as follows:

$$\mathbf{V}_l^{\mathrm{S}} = \{E, E + M_N, \ldots, E + (n-1) \cdot M_N; \forall \mathbf{e} : H(\mathbf{e}) = l\}, \tag{50}$$

$$\mathbf{U}_b^{\mathrm{S}} = \mathbf{V}_1^{\mathrm{S}} \cup \mathbf{V}_2^{\mathrm{S}} \cup, \ldots, \cup \mathbf{V}_b^{\mathrm{S}}. \tag{51}$$

The elements of $\mathbf{U}_b^{\mathrm{S}}$ are listed in an ascending order. We also have $|\mathbf{U}_b^{\mathrm{S}}| = n \cdot |\mathbf{U}_b|$. Based on this formulation of $\mathbf{U}_b^{\mathrm{S}}$, the decoding algorithms can now be described as follows:

- Step 1: Compute $Y'$ using (46). When $E = 0$, $X \in$ one of $n$ intervals, $i$th interval = $[i \cdot M_N, i \cdot M_N + M_K]$, $i = 0, \ldots, n-1$.
- Decoding algorithm 1: Given $Y'$, test if $E = 0$. If not, find the largest integer $\hat{E}$ in $\mathbf{U}_t^{\mathrm{S}}$ such that $\hat{E} \le Y'$.
- Decoding algorithm 2: Given $Y'$, test if $E = 0$. If not, find the largest integer $\hat{E}$ in $\mathbf{U}_\alpha^{\mathrm{S}}$ such that $\hat{E} \le Y'$.

Each iteration in the search process narrows the number of possible values of $\hat{E}$ by a factor of two. Here, $|\mathbf{U}_b^{\mathrm{S}}| = n \cdot |\mathbf{U}_b|$. Hence, $\lceil \log_2 n \rceil$ additional comparisons reduce the search to its original size of $|\mathbf{U}_b|$.

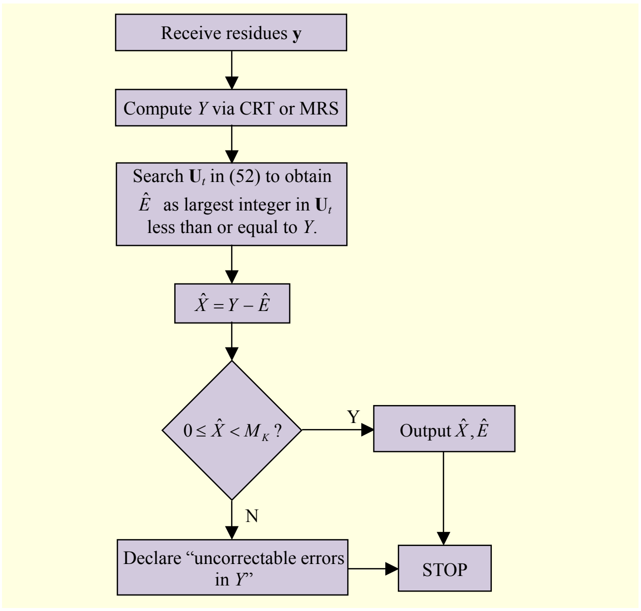

Receive residues **y**
Compute $Y$ via CRT or MRS
Search $\mathbf{U}_t$ in (52) to obtain $\hat{E}$ as largest integer in $\mathbf{U}_t$ less than or equal to $Y$.
$\hat{X} = Y - \hat{E}$
$0 \le \hat{X} < M_K$ ?
Y
N
Output $\hat{X}, \hat{E}$
Declare "uncorrectable errors in $Y$"
STOP

Fig. 2. Modified flow chart for decoding algorithm for case I.

The search scheme may be further improved in hardware implementation by arranging elements in $\mathbf{U}_b^{\mathrm{S}}$ into different subsets, where the arranged elements have the same bit-length in each subset. In this case, the decoding algorithms carry out a bit-length check of $Y'$, and then implement the search scheme in the subset in which numbers have the same bit-length as $Y'$.

### 2. Refinement 2

If it is not required to isolate the case of no error ($Y = X$, $E = 0$; $\mathbf{y} = \mathbf{x}$, $\mathbf{e} = \mathbf{0}$) from the case of one or more errors ($Y \ne X$, $E \ne 0$; $\mathbf{y} \ne \mathbf{x}$, $\mathbf{e} \ne \mathbf{0}$) at the receiver, then $\mathbf{U}_b$ is obtained as

$$\mathbf{U}_b = \{E, \forall \mathbf{e}: 0 \le \mathrm{H}(\mathbf{e}) \le b\} = \mathbf{V}_0 \cup \mathbf{V}_1 \cup, \ldots, \cup \mathbf{V}_b. \tag{52}$$

Similarly, $\mathbf{U}_b^{\mathrm{S}}$ is constructed as:

$$\mathbf{U}_b^{\mathrm{S}} = \mathbf{V}_0^{\mathrm{S}} \cup \mathbf{V}_1^{\mathrm{S}} \cup, \ldots, \cup \mathbf{V}_b^{\mathrm{S}}. \tag{53}$$

In this case, no comparisons are required once $Y$ in (30) or $Y'$ in (46) is computed; furthermore, a search is performed even when $E = 0$ ($\mathbf{e} = \mathbf{0}$). A flow chart for decoding algorithm 1 incorporating these refinements is shown in Fig. 2. In this flow chart, $\mathbf{U}_t$ can be replaced by $\mathbf{U}_\alpha$ to obtain the decoding algorithm for simultaneously correcting $\alpha$ and detecting $\beta$ ($\alpha < \beta$) errors. Finally, $Y$ and $\mathbf{U}_t$ in Fig. 2 can be replaced by $Y'$ and $\mathbf{U}_t^{\mathrm{S}}$ in (53), respectively, to obtain the decoding algorithm for correcting up to $t$ errors. In Fig. 2, $\mathbf{U}_t$ is replaced by $\mathbf{U}_\alpha^{\mathrm{S}}$ to obtain the decoding algorithm for simultaneously correcting $\alpha$ and detecting $\beta$ ($\alpha < \beta$) errors. In these cases, $Y'$ in Fig. 2 is computed using (46) instead of the CRT or the MRS. As a result, decoding algorithms incorporating these refinements avoid all modular operations in Step 1.

3. Refinement 3

The CRT computation of $Y$ in RRNS can be simplified by computing and transmitting permuted residues $\mathbf{x}'$ instead of $\mathbf{x}$, $x_i' \equiv a_i \cdot x_i \pmod{m_i}$, $i = 1, 2, \ldots, n$. The arithmetic remains the same. In this case, $\mathbf{y} = \mathbf{x}' + \mathbf{e}$, and $Y$ is calculated as

$$Y \equiv \sum_{i=1}^{n} x_i' \cdot \left(\frac{M_N}{m_i}\right) + \sum_{i=1}^{n} e_i \cdot \left(\frac{M_N}{m_i}\right) \pmod{M_N}, \qquad (54)$$

leading to $Y = X + T^{-1} \cdot E \pmod{M_N}$. There is no other change in the error control algorithms. This saves $n$ modular multiplications required to compute $\mathbf{x}'$ from $\mathbf{x}$ at the receiving end.

The framework described here can also be used for error control in other scenarios. For instance, if we are interested in controlling burst errors, then $\mathbf{U}_b$ is the error-set corresponding to all correctable burst errors of up to length $b$. Finally, the error correction/list decoding problem mentioned in page 1,332 of [13] requires computation of all codewords that differ from $\mathbf{y}$ in up to $e$ places, $e > t$. This can also be done by using the methodology described here. In such a case, the decoding algorithm uses error-set $\mathbf{U}_e$ instead of $\mathbf{U}_t$ and finds codewords within $M_K$ of $Y$. There may be multiple solutions as $e > t$.

## V. Computational Complexity Comparisons

The work of [10] deals with double-error and single burst error correction. However, it doesn't perform multiple error correction in general. To date, the algorithms in [13], [15], [16], and [17] have been the most efficient error correction schemes for RRNS. In this section, a computational complexity comparison is given for our algorithm, a variant of the algorithm in [13], and the algorithm in [17]. The works of [15] and [16] are closely related to [13]. The error-correction capabilities of the methods in [14] and [13] are restricted by the values of moduli in RRNS. In [13], correct decoding takes place in the presence of $e$ errors if $e < \log m_1 (n-k) / (\log m_1 + \log m_n)$, thereby requiring that the moduli be close to each other. This is in contrast to our work where correct decoding takes place in the presence of $e$ errors if $e \le t = \lfloor (n-k)/2 \rfloor$. The method in [17] aims at finding a tuple of correct residues to recover the original integer. The algorithms in [13], [15], and [16] calculate and subtract the error value using Euclid's algorithm and continued fractions involving large integers. This can be computationally intensive. An idea intended to achieve error correction, analogous to similar ideas appearing in [13], [15], and [16], is shown in [17] — it is equivalent to recovering and checking all possible combinations of $(n-a)$ residues in the received vector $\mathbf{y}$ assuming that $a$ errors have occurred. The decoding algorithm in [17], a variant of the decoding algorithm in [13], uses modular arithmetic. It is simpler to implement.

1) Recover $X$ from the received vector with the CRT.
2) If $Y$ is in the legitimate range, then stop and output $X = Y$. Otherwise, go to step 3.

Table 2. Complexity comparison of decoding algorithms.

| | Variant of paper [13] | Paper [17] | Algorithm (Section III) | Algorithm (Section IV) |
|---|---|---|---|---|
| Computation of $Y$ via CRT or MRS | √ | √ | √ | N/A |
| Modular operation | ${}^nC_t$ | $f \cdot (r+1)$ | 0 | 0 |
| Comparison operation | ${}^nC_t$ | $f \cdot (r+2)$ | $\lceil \log_2 \lvert\mathbf{U}_t\rvert \rceil + 2$ | $\lceil \log_2 (n \cdot \lvert\mathbf{U}_t\rvert) \rceil + 2$ |
| Subtraction operation | 0 | 0 | 2 | 2 |
| Total complexity | $2 \cdot {}^nC_t$ | $f \cdot (2 \cdot r + 1)$ | $\lceil \log_2 \lvert\mathbf{U}_t\rvert \rceil + 4$ | $\lceil \log_2 (n \cdot \lvert\mathbf{U}_t\rvert) \rceil + 4$ |

Note: $f$ is an experimental number such that $f \ge \left\lceil \frac{{}^nC_t}{{}^rC_t} \right\rceil$.

Table 3. Complexity comparison of decoding algorithms for (10, 6) & (16, 10) RRNS.

| | Variant of paper [13] | Paper [17] | Algorithm (Section III) | Algorithm (Section IV) |
|---|---|---|---|---|
| Computation of $Y$ via CRT or MRS | √ | √ | √ | N/A |
| Modular operation | 45; 560 | 50; 392 | 0; 0 | 0; 0 |
| Comparison operation | 45; 560 | 60; 448 | 19; 28 | 23; 32 |
| Subtraction operation | 0; 0 | 0; 0 | 2; 2 | 2; 2 |
| Total complexity | 90; 1,120 | 110; 840 | 21; 30 | 25; 34 |

Note: The first & second entry in each box corresponds to (10, 6) & (16, 10) RRNS.

3) Assume $a = 1$.
4) Compute $X_i = \langle Y \rangle_{Z_i}$, where $Z_i$, $i = 1, 2, \ldots, {}^nC_{n-a}$ is a product of any $(n-a)$-dimensional combination of moduli.
5) If $X_i$ is within legitimate range, then output $\hat{X} = X_i$. Or, if $i = {}^nC_{n-a}$, then increment $a$ by 1. If $a > t$, then go to step 6. Otherwise, go to step 4.
6) Declare more than $t$ errors and stop.

Thus, the problem of multiple error correction can be solved in polynomial time in the aforementioned scheme. To remedy the weakness that a large number of iterations are required to select correct residues via traversal, MLD is introduced to reduce nonessential operations in [17]. We omit the details.

The computational complexity comparison between the proposed algorithms and the algorithms in [13] and [17] is listed in Table 2. With the exception of the algorithms in Section IV, these algorithms require computation of integer Y from the received residues **y**. This count is not included in Table 2 explicitly. Due to low complexity requirements for the dichotomy-based search algorithms, the proposed methods perform error control up to the error control capability of RNSCs in a computationally efficient manner. Table 3 lists the numeric values of the various entries in Table 2 for the (10, 6) double error–correcting RRNS and (16, 10) triple error–correcting RRNS described in examples 4 and 5, respectively. We observe that our algorithms are based on "comparison operation." They avoid modular operations or processing of large integers. This is in contrast to the algorithms in [13] and [17]. The algorithms presented here are expected to be simpler and better suited for high-speed, application environments.

## VI. Conclusion

In this work, new computationally efficient error-control algorithms for RRNSs and RNS-PCs, collectively termed RNSCs, are presented under a unified framework. These algorithms require "comparison operations" as opposed to modular operations. This aspect is expected to lead to improvements in processing time when these algorithms are implemented in hardware. The proposed algorithms have a computational complexity of $t \cdot \mathrm{O}(\log_2 n + \log_2 \overline{m})$ operations. This is significantly lower than the computational complexity of existing algorithms. A refinement is described that avoids modular operations in the first step of decoding algorithms, at a slight increase in computational complexity of $\lceil \log_2 n \rceil$ comparisons. Two other refinements are described that further simplify the computations associated with the first step. The techniques can be extended to non–RNSCs and other scenarios such as correction of burst errors.

## Acknowledgement

The work of H.K. Garg is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media Programme Office at the NUS-ZJU Sensor-Enhanced Social Media (SeSaMe) Centre. The work of H. Xiao and J. Hu was supported by Tsinghua University Initiative Scientific Research Program (20141081231), the National Natural Science Foundation of China (under Grant 61371104), and National High Technology Project of China (under Grant 2011AA010201).

## References

[1] A.S. Madhukumar, F. Chin, and A.B. Premkumar, "Incremental Redundancy and Link Adaptation in Wireless Local Area Networks Using Residue Number Systems," *Wireless Pers. Commun.*, vol. 27, no. 4, Dec. 2003, pp. 321–336.

[2] A.S. Madhukumar and F. Chin, "Enhanced Architecture for Residue Number System-Based CDMA for High-Rate Data Transmission," *IEEE Trans. Wireless Commun.*, vol. 3, no. 5, Oct. 2004, pp. 1363–1368.

[3] A. Sengupta and B. Natarajan, "Performance of Systematic RRNS Based Space-Time Block Codes with Probability-Aware Adaptive Demapping," *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, May 2013, pp. 2458–2469.

[4] T. Keller, T.H. Liew, and L. Hanzo, "Adaptive Redundant Residue Number System Coded Multicarrier Modulation," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 11, Nov. 2000, pp. 2292–2301.

[5] B. Zarei, V. Muthukkumarasay, and X.-W. Wu, "A Residual Error Control Scheme in Single-Hop Wireless Sensor Networks," *IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Barcelona, Spain, Mar. 25–28, 2013, pp. 197–204.

[6] S. Zhang, Y. Zhang, and L.-L. Yang, "Redundant Residue Number System Based Multicarrier DS-CDMA for Dynamic Multiple-access in Cognitive Radios," *IEEE Veh. Technol. Conf.*, Yokohama, Japan, May 15–18, 2011, pp. 1–5.

[7] M. Villari et al., "Data Reliability in Multi-provider Cloud Storage Service with RRNS," in *Adv. Service-Oriented Cloud Comput.*, Berlin, Germany: Springer-Verlag, 2013, pp. 83–93.

[8] F. Barsi and P. Maestrini, "Error Correcting Properties of Redundant Residue Number Systems," *IEEE Trans. Comput.*, vol. C-22, no. 3, Mar. 1973, pp. 307–315.

[9] H. Krishna, K.-Y. Lin, and J.-D. Sun, "A Coding Theory Approach to Error Control in Redundant Residue Number Systems, Part I: Theory and Single Error Correction," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 39, no. 1, Jan. 1992, pp. 8–17.

[10] J.-D. Sun and H. Krishna, "A Coding Theory Approach to Error Control in Redundant Residue Number Systems, Part II: Multiple Error Detection and Correction," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 39, no. 1, Jan. 1992, pp. 18–34.

[11] L.-L. Yang and L. Hanzo, "Redundant Residue Number System Based Error Correction Codes," *IEEE Veh. Technol. Conf.*, Atlantic City, NJ, USA, Oct. 2001, pp. 1472–1476.

[12] L.-L. Yang and L. Hanzo, "Minimum-Distance Decoding of Redundant Residue Number System Codes," *IEEE Int. Conf. Commun.*, Helsinki, Finland, June 2001, pp. 2975–2979.

[13] O. Goldreich, D. Ron, and M. Sudan, "Chinese Remaindering with Errors," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, July 2000, pp. 1330–1338.

[14] D.M. Mandelbaum, "On a Class of Arithmetic Codes and a Decoding Algorithm," *IEEE Trans. Inf. Theory*, vol. 22, no. 1, Jan. 1976, pp. 85–88.

[15] O. Goldreich, D. Ron, and M. Sudan, "Chinese Remaindering with Errors," MIT, Cambridge, USA, Tech. Rep. TR98–062 (revised), Aug. 1999.

[16] O. Goldreich, D. Ron, and M. Sudan, "Chinese Remaindering with Errors," *ACM Symp. Theory Comput.*, Atlanta, NJ, USA, May 1999, pp. 225–234.

[17] 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, Mar. 2008, pp. 325–330.

[18] H. Lo and T. Lin, "Parallel Algorithms for Residue Scaling and Error Correction in Residue Arithmetic," *Wireless Eng. Technol.*, vol. 4, no. 4, Oct. 2013, pp. 198–213.

[19] H. Krishna et al., "Computational Number Theory and Digital Signal Processing: Fast Algorithms and Error Control Techniques," Boca Raton, FL, USA: CRC Press, 1994.

**Hanshen Xiao** is currently pursuing his BS degree in mathematics at Tsinghua University, Beijing, China. In 2012, he worked as a research intern for the National Key Lab of Science and Technology on Communications, Chengdu, China. In 2015, he was awarded the Tsinghua Highest Spark Research Fellowship and was a visiting student at the Department of Computer Science, Yale University, CT, USA. His research interests include cryptography; coding theory; and mathematical modeling in signal processing and numerical calculation.

**Hari Krishna Garg** received his BS degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 1981 and his MS and PhD degrees in electrical engineering from Concordia University, Canada, in 1983 and 1985, respectively. He received his MBA degree in international finance from Syracuse University, NY, USA, in 1995. From 1985 to 1995, he was with the Department of Electrical & Computer Engineering, Syracuse University. He joined the Department of Electrical & Computer Engineering, National University of Singapore, in 1995. His research interests include wireless communications from physical to application layers. He also engages in enterprise activity as his passion. Thus far, he has founded or co-founded four companies.

**Jianhao Hu** received his BE and PhD degrees in communication systems from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1993 and 1999, respectively. He joined the City University of Hong Kong, Kowloon Tong, China, in 1999 as a postdoctoral researcher. From 2000 to 2004, he served as a senior system engineer at the 3G Research Center of the University of Hong Kong. He has been a professor of the National Key Lab., UESTC, since 2005. His research interests include high-speed DSP technology with VLSI, NoC, and software radio.

**Guoqiang Xiao** received his PhD degree in signal and information processing from the University of Electronic Science and Technology of China (UESTC), Chengdu, China and his BS degree in radio technology from Chongqing University, China, in 1999 and 1986, respectively. Since 1986, he has been with the College of Computer and Information Science, Southwest University, Chongqing, China, where he is currently a professor. From 2001 to 2004, he was with the Department of Electrical & Electronic Engineering, University of Hong Kong, as a postdoctoral researcher. His research interests include image processing, pattern recognition, neural networks, and wireless network communication.