Concatenated Permutation Block Codes based on Set Partitioning for Substitution and Deletion Error-Control

Reolyn Heymann*, Jos H. Weber†, Theo G. Swart* and Hendrik C. Ferreira*
* University of Johannesburg, Dept. E&E Eng. Science, South Africa
Email: {rheymann, tgswart, hcferreira}@uj.ac.za
† TU Delft, The Netherlands
Email: J.H.Weber@tudelft.nl

***Abstract*—A new class of permutation codes is presented where, instead of considering one permutation as a codeword, codewords consist of a sequence of permutations. The advantage of using permutations, i.e. their favourable symbol diversity properties, is preserved. Additionally, using sequences of permutations as codewords, code rates close to the optimum rate can be achieved. Firstly, the complete set of permutations is divided into subsets by using set partitioning. Binary data is then mapped to permutations from these subsets. These permutations, together with a parity permutation, will form the codeword. Two constructions will be presented: one capable of detecting and correcting substitution errors and the other capable of detecting and correcting either substitution or deletion errors.**

I. Introduction

Permutations are used in error control coding because of their favourable symbol diversity properties. Combining permutations with M-ary FSK modulation can be used to combat different types of noise (e.g. impulse noise, background noise and frequency disturbances) in Power Line Communications (PLC) [1], where every integer in the permutation maps to a specific frequency. A combination of permutations and convolutional decoding has been proposed in [2] to correct errors when used in PLC. Methods to correct synchronization errors, modelled as insertion(s) and deletion(s) of symbols, are presented in [3]–[6]. In fact, the failure of permutations to overcome insertion(s) and deletion(s) is also investigated in [7].

Permutations are also combined with rank modulation for use in flash memories [8], consisting of floating gate cells which have a discrete number of levels. It is difficult to charge a cell to a specific level without causing an overshoot error. Rank modulation, combined with permutations, provide a solution since the charge levels needed are determined relative to the other cells.

Set partitioning is the process of dividing a set into smaller subsets in such a way that the subsets are nonempty and every object of the original set is included in exactly one subset. The use of set partitioning in Trellis Coded Modulation was proposed by Ungerboeck [9]. A process is used in Trellis Coded Modulation where constellation points are partitioned into subsets, and then those subsets are partitioned into smaller subsets and so forth. Every time set partitioning is applied, the Euclidean distance between signals in the same subset is increased. In this paper, set partitioning will be applied to permutations to increase the Hamming distance in subsets.

Let $\mathcal{S}_M$ denote the set of all $M!$ permutations of the integers $1, 2, \ldots, M$. Traditionally, a permutation code $\mathcal{C}$ of length M is a subset of $\mathcal{S}_M$. The subset is chosen in such a way that the codebook has a specific minimum distance enabling it to correct errors. Binary information is mapped to these permutations. The maximum rate obtainable for any permutation code is

$$\frac{\log_2(M!)}{M} \text{ bits/symbol.} \tag{1}$$

Since only a rather small subset of $\mathcal{S}_M$ is used, the code rate of $\mathcal{C}$ is considerably lower than the maximum rate.

The two constructions presented in this paper use a larger subset of $\mathcal{S}_M$ (compared to other constructions in literature, for example [6]) and, in some cases, even the entire set $\mathcal{S}_M$. It is thus possible to obtain code rates close to the maximum rate obtainable. Initially, the set $\mathcal{S}_M$ is partitioned into subsets. Binary data is mapped to permutations in the subsets. These permutations will form subwords of the codeword. Every subword will thus still have the property of a permutation, i.e. every symbol will occur exactly once. The subwords, together with a parity permutation, will form a codeword as shown in Figure 1. Thus, error control capabilities are achieved with very little loss in rate and simple decoding. The number of permutations in every codeword is an important parameter: a larger number will increase the code rate but also the probability of undetected errors.

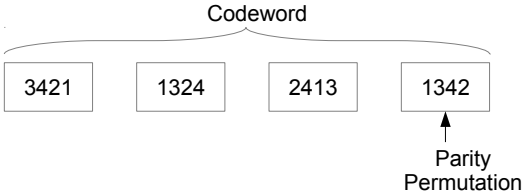

Fig. 1. Concatenation of permutations to form a codeword ($M = 4$)

This paper is organized as follows: in the next section formal definitions and notations are given for the most important concepts used in this paper. The initial step for both constructions is to partition the set $\mathcal{S}_M$ into subsets. This process is described in Section III. A construction capable of correcting and detecting substitution errors is presented in Section IV. This construction is expanded in Section V to correct either substitution or deletion errors. The work is concluded in Section VI.

II. Definitions and Notations

The Hamming distance, d, between two codewords is defined as the number of positions in which the codewords differ. The minimum Hamming distance of a codebook, $d_{\min}$, is defined as the minimum Hamming distance between any two different codewords in the codebook. Since only the Hamming distance will be used in this paper, any reference to distance will refer to Hamming distance.

If a larger integer precedes a smaller one in a permutation, it is defined as an inversion. For example, the total number of inversions in the permutation 613452 is equal to 8. A permutation can be either even or odd. A permutation is even if the total number of inversions is even and odd if the total number of inversions is odd. The set of all even permutations is denoted as $\mathcal{A}_M$, where $|\mathcal{A}_M| = M!/2$ and $d_{\min} = 3$ [10].

A substitution error occurs if one symbol is transmitted but another is received. If a transmitted symbol is not received, then a deletion error occurred.

A permutation can be transformed into another permutation by transposing (or swapping) symbols. For example, the permutation 1423 can be transformed into permutation 3421 by the function $swap(1, 3)$.

Let $\mathcal{B}_v$ be the set consisting of all binary sequences of length v and let K be a positive integer.

III. Set Partitioning

The set $\mathcal{S}_4$ will be divided into subsets. These sets will also form the basis for all the subsets in cases where $M > 4$. The set $\mathcal{S}_4$ has a minimum distance of 2. The set is firstly partitioned into 2 subsets, one containing even permutations and one containing odd permutations, each with a minimum distance of 3 [10]. These two subsets are further partitioned into smaller subsets with distance 4. Thus, the 24 permutations of length 4 are divided into 6 subsets:

$$\begin{aligned}
\mathcal{R}_0 &= \{1234, 2143, 3412, 4321\},\\
\mathcal{R}_1 &= \{1243, 2134, 4312, 3421\},\\
\mathcal{R}_2 &= \{1324, 3142, 2413, 4231\},\\
\mathcal{R}_3 &= \{1342, 3124, 4213, 2431\},\\
\mathcal{R}_4 &= \{1423, 4132, 2314, 3241\},\\
\mathcal{R}_5 &= \{1432, 4123, 3214, 2341\}.
\end{aligned}$$

Hence, the set $\mathcal{S}_4$ has been divided in such a way that two sequences within the same subset will have a distance of 4, while two sequences from different subsets will have a minimum distance of 2.

IV. Code construction I

Construction I will firstly be presented for $M = 4$ and will then be generalized for all values of $M > 4$. Only substitution errors will be considered in this section.

A. $M = 4$

Not all the subsets from Section III will be used. Binary sequences are mapped to the permutation sequences. The number of bits that will be mapped to a permutation is $\lfloor \log_2 4! \rfloor = 4$. The number of subsets that will be used is thus $2^4/4 = 4$. (A generalized equation is given in the next subsection.)

For all permutation sequences $\mathbf{r} \in \mathcal{R}_i$, define $\psi(\mathbf{r}) = i$. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$. Let ϕ be a one-to-one mapping from the set $\mathcal{B}_4$ to $\mathcal{R}$.

Encoding: A source generates a sequence $\mathbf{u}$ of $4K$ bits, which is partitioned into K sequences $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$, all from $\mathcal{B}_4$. Let $\mathbf{x}_k = \phi(\mathbf{u}_k)$ for all k and let $c = -\sum_{k=1}^{K} \psi(\mathbf{x}_k)$, where the summation is done modulo 4. Then the encoder output is the code sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K, \mathbf{x}_{K+1})$, where the check sequence $\mathbf{x}_{K+1}$ is taken from $\mathcal{R}_c$. Note that $\sum_{k=1}^{K+1} \psi(\mathbf{x}_k) \equiv 0 \pmod 4$, which implies, together with the properties of $\mathcal{R}_i$, that any two different code sequences differ in at least 4 positions. Throughout this paper, formal proofs are omitted due to space constraints.

Decoding: Let the received sequence be $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K, \mathbf{y}_{K+1})$, where each $\mathbf{y}_i$ is a sequence of four symbols from $\{1, 2, 3, 4\}$. The decoding procedure consists of the following steps:

1) If all $\mathbf{y}_k$ are in $\mathcal{R}$ and $\sum_{k=1}^{K+1} \psi(\mathbf{y}_k) \equiv 0 \pmod 4$, then set $\mathbf{x}' = \mathbf{y}$ and go to Step 4, else go to Step 2.
2) If there exists an e such that (i) $\mathbf{y}_e$ is not in $\mathcal{R}$, (ii) $\mathbf{y}_k$ is in $\mathcal{R}$ for all $k \neq e$, and (iii) $\mathbf{y}_e$ contains one symbol t from $\{1, 2, 3, 4\}$ twice, another symbol n not at all, and each of the other symbols once, then go to Step 3. Else, go to Step 5.
3) For $j = 1, 2$ set $\mathbf{x}^j$ as $\mathbf{y}$, with the jth symbol t in $\mathbf{y}_e$ replaced by symbol n. If there exists a j such that $\mathbf{x}^j_e \in \mathcal{R}$ and $\sum_{k=1}^{K+1} \psi(\mathbf{x}^j_k) \equiv 0 \pmod 4$, then set $\mathbf{x}' = \mathbf{x}^j$ for this j and go to Step 4. Else, go to Step 5.
4) Set the decoder output as $\mathbf{u}' = (\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_K)$, where $\mathbf{u}'_k = \phi^{-1}(\mathbf{x}'_k)$ for $k = 1, 2, \ldots, K$ and STOP.
5) Detect that at least two substitution errors have occurred and STOP.

Remarks: The code has rate $\frac{4K}{4(K+1)} = \frac{K}{(K+1)} = 1 - \frac{1}{(K+1)}$ bits/symbol. This can be improved to $\frac{4K+2}{4(K+1)} = \frac{2K+1}{(2K+2)} = 1 - \frac{1}{2(K+1)}$ by encoding two extra information bits into the choice of $\mathbf{x}_{K+1}$, which is possible due to the fact that there are four options in the set $\mathcal{R}_c$. The rate is thus close to the optimal rate for large values of K. The proposed decoding procedure will correct any single substitution error and detect the occurrence of two substitution errors in the sequence of $4(K + 1)$ transmitted symbols. The choice of K is a trade-off between efficiency (the higher the value of K, the higher the rate) and reliability (the lower the value of K, the lower the

probability of uncorrected/undetected errors). The effect of K will further be explained in Subsection IV.C.

Mapping: The map ϕ used in the construction could be implemented through a table look-up. Still, it may be good to impose some structure, to allow simple encoding and decoding. The binary sequence $\mathbf{b} = (b_0, b_1, b_2, b_3)$ is uniquely mapped to a member $\mathbf{s} = (s_0, s_1, s_2, s_3)$ from $\mathcal{R}$ as follows: The integer representation p of (b_0, b_1) indicates that a member of $\mathcal{R}_p$ will be chosen. The integer representation q of (b_2, b_3) indicates that within $\mathcal{R}_p$ we choose the sequence with $s_q = 1$. For example, $\mathbf{b} = 1001$ has $p = 2$ and $q = 1$ and is thus mapped to $\mathbf{s} = 3142 \in \mathcal{R}_2$.

Example: Let $K = 3$ and let the information sequence be $\mathbf{u} = (0111, 1000, 1010)$. Then the encoded sequence is $\mathbf{x} = (3421, 1324, 2413, 1342)$. Five cases are considered for the received sequence $\mathbf{y}$ and the corresponding decoding results are given.

- If $\mathbf{y} = \mathbf{x}$ (no errors), then we go from Step 1 to Step 4 and the decoding result is $\mathbf{u}' = \mathbf{u}$.
- If $\mathbf{y} = (3421, 1321, 2413, 1342)$, i.e., an error occurred in the eighth symbol, then we find in Step 1 that $\mathbf{y}_2$ is not in $\mathcal{R}$. Thus, in Step 2 we find that $e = 2$, $t = 1$ (the symbol which appears twice in $\mathbf{y}_2$), and $n = 4$ (the symbol which does not appear in $\mathbf{y}_2$). Hence, in Step 3 we get $\mathbf{x}_2^1 = 4321$ and $\mathbf{x}_2^2 = 1324$ and thus $\Sigma_{k=1}^4 \psi(\mathbf{x}_k^1) = 1 + 0 + 2 + 3 = 6 \equiv 2 \pmod 4$ and $\Sigma_{k=1}^4 \psi(\mathbf{x}_k^2) = 1 + 2 + 2 + 3 = 8 \equiv 0 \pmod 4$. In conclusion, $\mathbf{x}' = \mathbf{x}^2 = (3421, 1324, 2413, 1342)$ and Step 4 gives $\mathbf{u}' = \mathbf{u}$. Hence, the error has been corrected.
- If $\mathbf{y} = (3431, 1324, 2213, 1342)$, i.e., errors occurred in the third and tenth symbol, then we find that $\mathbf{y}_1$ and $\mathbf{y}_3$ are not in $\mathcal{R}$, and thus we go from Step 1 via Step 2 to Step 5, and the decoding result is the detection of (at least) two errors.
- If $\mathbf{y} = (3421, 3124, 2413, 1342)$, errors occurred in the fifth and sixth symbol, then we find in Step 1 that all symbols are in $\mathcal{R}$, but that $\Sigma_{k=1}^4 \psi(\mathbf{y}_k) = 1+3+2+3 = 9 \equiv 1 \pmod 4$. Via Step 2 we end up in Step 5, and the decoding result is the detection of (at least) two errors.
- If $\mathbf{y} = (3421, 1324, 3213, 1342)$, i.e., errors occurred in the ninth and tenth symbol, then we find in Step 1 that $\mathbf{y}_3$ is not in $\mathcal{R}$, and thus $e = 3$, $t = 3$ and $n = 4$ in Step 2. Hence, in Step 3 we get $\mathbf{x}_3^1 = 4213$ and $\mathbf{x}_3^2 = 3214$, and thus $\Sigma_{k=1}^4 \psi(\mathbf{x}_k^1) = 1 + 2 + 3 + 3 = 9 \equiv 1 \pmod 4$, while $\mathbf{x}_3^2 \notin \mathcal{R}$. In conclusion, we end up in Step 5, and the decoding result is the detection of (at least) two errors.

B. $M > 4$

This subsection will extend and generalize the construction given in the previous subsection to higher values of M. The prefix method from [11] is proposed to construct subsets of permutations of length M from the subsets of permutations of length $(M-1)$. The recursive process starts with the subsets as defined for $M = 4$ in Section III.

Let $\mathcal{Q}$ be a partitioning of $\mathcal{S}_M$ into $M!/4$ subsets $\mathcal{R}_i$ of size 4 each, with the property that sequences within the same subset have distance 4. Similarly, let $\mathcal{Q}'$ be a partitioning of $\mathcal{S}_{M-1}$ into $(M-1)!/4$ subsets $\mathcal{R}'_i$ of size 4 each, with the same distance properties as $\mathcal{R}_i$.

Every subset in $\mathcal{Q}'$ can be extended and used to form M new subsets with 4 sequences of length M. Let $\mathcal{P}_{ij}$, $j = 0, 1, \ldots, M-1$, be the subsets created from $\mathcal{R}'_i$. The construction steps are:

1) Add the symbol M as a prefix to every permutation in $\mathcal{R}'_i$ to form $\mathcal{P}_{i0}$.
2) Let $j = 1, 2, \ldots, (M-1)$. To construct $\mathcal{P}_{ij}$, $swap(M, j)$ in every permutation contained in $\mathcal{P}_{i0}$.
3) All sets, $\mathcal{P}_{ij}$, $j = 0, 1, 2, \ldots, (M-1)$, are added to $\mathcal{Q}$.
4) Repeat the previous steps for all the subsets in $\mathcal{Q}'$.

The number of subsets that will be used during encoding and decoding is $L = \frac{2^v}{4}$, where $v = \lfloor \log_2 M! \rfloor$. $\mathcal{R}$ can then be formed by choosing any L subsets from $\mathcal{Q}$, thus $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_{L-1}$.

Example: Let $M = 5$ and $\mathcal{R}'_0 = \{1234, 2143, 3412, 4321\}$. The extended sets from $\mathcal{R}'_0$ are:

$$\begin{aligned}
\mathcal{P}_{00} &= \{51234, 52143, 53412, 54321\},\\
\mathcal{P}_{01} &= \{15234, 12543, 13452, 14325\},\\
\mathcal{P}_{02} &= \{21534, 25143, 23415, 24351\},\\
\mathcal{P}_{03} &= \{31254, 32145, 35412, 34521\},\\
\mathcal{P}_{04} &= \{41235, 42153, 43512, 45321\}.
\end{aligned}$$

The encoding and decoding procedures are similar to the $M = 4$ case, except that all summations are done modulo L.

Remarks: The code rate is $\frac{vK}{M(K+1)}$ bits/symbol. The mapping can still be done as in the previous subsection, where the first $(v-2)$ binary bits represent the subset index and the last 2 binary bits represent the specific permutation in a subset.

The parity permutation can be any permutation in the subset $\mathcal{R}_c$. Two additional bits can be used to determine which of the permutations in $\mathcal{R}_c$ is used as the parity permutation. The code rate can thus be improved to $\frac{vK+2}{M(K+1)}$ bits/symbol.

C. Effect of K

The choice of K is a trade-off between efficiency and reliability. The optimal code rate for permutation codes is $\frac{\log_2(M!)}{M}$. If K is large, a code rate close to optimal can be obtained. Figure 2 shows the code rates for different values of K and M. The improved code rate of $\frac{vK+2}{M(K+1)}$ bits/symbol, as given in the previous subsection, is used.

However, as K increases, the probability that the error correction capability of the code will be exceeded, also increases. Errors will not be corrected if more than 1 error occurs in a codeword. Let P_e be the probability of a substitution error occurring and let $n = M(K+1)$ be the length of a codeword. Using Bernoulli trials, the probability that an error will not be corrected, P_2, is given by

$$P_2 = 1 - (1 - P_e)^n - nP_e(1 - P_e)^{n-1}. \tag{2}$$

These probabilities are given in Figure 3 for $M = 4$ and different values of K.

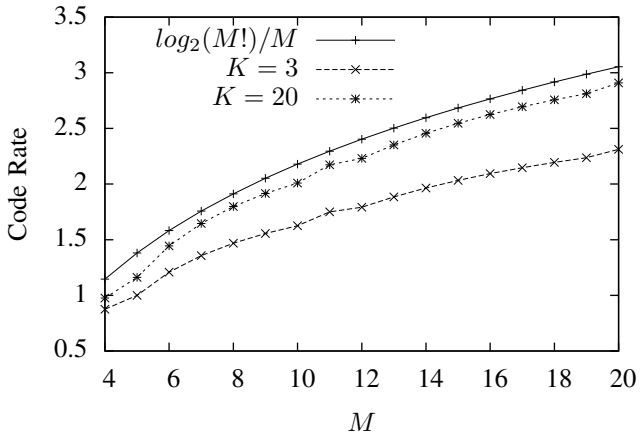

Fig. 2. Code rates for different values of M and K

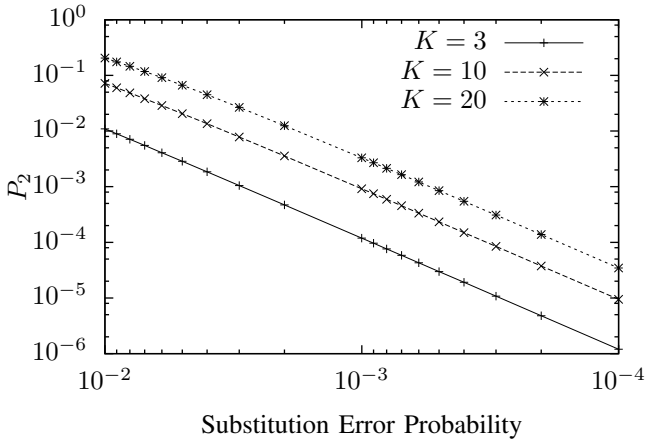

Fig. 3. Probability that an error will not be corrected, $M = 4$

V. Code construction II

Construction I is adapted in this section to provide deletion error correcting capabilities. If a deletion does not occur, the construction will be able to detect and correct substitution errors. Only the $M = 4$ case is presented in this paper.

In Construction II, the sequence of permutation words will be divided into two substrings. Each substring will use different subsets. The set $\mathcal{S}_4$ is divided into subsets as explained in Section III. The two subsets which were not used in Construction I, will now form a different set that will be used exclusively by the second substring. Referring back to the subsets given in Section III, let $\mathcal{T}_0 = \mathcal{R}_4$ and $\mathcal{T}_1 = \mathcal{R}_5$.

For all permutation sequences $\mathbf{r} \in \mathcal{R}_i$, define $\psi(\mathbf{r}) = i$, and for all permutation sequences $\mathbf{t} \in \mathcal{T}_j$, define $\xi(\mathbf{t}) = j$. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$, and $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$. The construction method will exploit both $\mathcal{R}$ and $\mathcal{T}$, and all 24 permutations will be used. Using all 24 permutations gives a communications diversity benefit in some applications, e.g. the harsh PLC channel.

Let ϕ be a one-to-one mapping from the set $\mathcal{B}_4$ to $\mathcal{R}$, and let χ be a one-to-one mapping from the set $\mathcal{B}_3$ to $\mathcal{T}$.

Encoding: A source generates a sequence $\mathbf{u}$ of $7K$ bits, which is partitioned in K sequences $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K$, all from $\mathcal{B}_7$. Each $\mathbf{u}_k$ is partitioned as $\mathbf{u}_k = (\mathbf{v}_k, \mathbf{w}_k)$, where $\mathbf{v}_k \in \mathcal{B}_4$ and $\mathbf{w}_k \in \mathcal{B}_3$. Let $\mathbf{x}_{2k-1} = \phi(\mathbf{v}_k)$ for all k and let $c_{\text{odd}} = -\Sigma_{k=1}^{K} \psi(\mathbf{x}_{2k-1})$, where the summation is done modulo 4. Similarly, let $\mathbf{x}_{2k} = \chi(\mathbf{w}_k)$ for all k and let $c_{\text{even}} = -\Sigma_{k=1}^{K} \xi(\mathbf{x}_{2k})$, where the summation is done modulo 2.

Then the encoder output is the code sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{2K}, \mathbf{x}_{2K+1}, \mathbf{x}_{2K+2})$, where the check sequence $\mathbf{x}_{2K+1}$ is taken from $\mathcal{R}_{c_{\text{odd}}}$ and the check sequence $\mathbf{x}_{2K+2}$ is taken from $\mathcal{T}_{c_{\text{even}}}$. Note that $\Sigma_{k=1}^{K+1} \psi(\mathbf{x}_{2k-1}) \equiv 0 \pmod 4$ and $\Sigma_{k=1}^{K+1} \xi(\mathbf{x}_{2k}) \equiv 0 \pmod 2$, which imply, together with the properties of the $\mathcal{R}_i$ and $\mathcal{T}_j$ subsets, that any two different code sequences differ in at least four positions.

Decoding: The occurrence of deletions can be observed from the length of the received sequence $\mathbf{y}$. If no deletions occur, we receive the sequence $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{2K}, \mathbf{y}_{2K+1}, \mathbf{y}_{2K+2})$, where each $\mathbf{y}_j$ is a sequence of four symbols from $\{1, 2, 3, 4\}$. By applying the procedure from the previous section on the odd substring $(\mathbf{y}_1, \mathbf{y}_3, \ldots, \mathbf{y}_{2K-1}, \mathbf{y}_{2K+1})$ and the even substring $(\mathbf{y}_2, \mathbf{y}_4, \ldots, \mathbf{y}_{2K}, \mathbf{y}_{2K+2})$, one substitution error can be corrected and two substitution errors can be detected in each of the substrings. In the procedure for the even substring, the roles of $\mathcal{R}$, ϕ, and ψ are substituted by $\mathcal{T}$, χ, and ξ, respectively, in a straightforward way.

If one deletion occurs (and no substitution errors), it can be corrected by the following procedure, which exploits the fact that all sequences in $\mathcal{T}$ starts with 14, 41, 23 or 32, while none of the sequences in $\mathcal{R}$ starts with such a combination.

1) Partition the received sequence $\mathbf{y}$ as $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{2K}, \mathbf{y}_{2K+1}, \mathbf{y}_{2K+2})$, where $\mathbf{y}_j$ are sequences of symbols from $\{1, 2, 3, 4\}$, which are of length four, except the last one, which is of length three. Partition each $\mathbf{y}_j$ into two subsequences, i.e., $\mathbf{y}_j = (\mathbf{y}_j^{\text{head}}, \mathbf{y}_j^{\text{tail}})$, where each subsequence is of length two, except $\mathbf{y}_{2K+1}^{\text{tail}}$, which is of length one. Let $\mathbf{y}^{s,p,k}$ denote the sequences which are obtained from $\mathbf{y}$ by inserting the symbols $s \in \{1, 2, 3, 4\}$ at position $p \in \{0, 1, 2, 3\}$ of $\mathbf{y}_k$.
2) Find the smallest value of $e \in \{1, 2, \ldots, K+1\}$ such that $\mathbf{y}_{2e}^{\text{head}}$ is not equal to 14, 41, 23 or 32. If no violated $\mathbf{y}_{2e}^{\text{head}}$ is found, then set $e = K + 2$.
3) If (i) $\mathbf{y}_{2e-2} \in \mathcal{T}$, (ii) $\mathbf{y}_{2e-1} \in \mathcal{R}$, and (iii) $\Sigma_{k=1}^{K+1} \psi(\mathbf{y}_{2k-1}^{1,0,2e}) \equiv 0 \pmod 4$, then set $\mathbf{x}' = \mathbf{y}^{s,p,2e}$, where s and p are chosen such that $\mathbf{y}_{2e}^{s,p,2e} \in \mathcal{T}$ and $\Sigma_{k=1}^{K=1} \xi(\mathbf{y}_{2k}^{s,p,2e}) \equiv 0 \pmod 2$, and go to Step 6.
4) If (i) $\mathbf{y}_{2e-2} \in \mathcal{T}$, (ii) $\Sigma_{k=1}^{K+1} \xi(\mathbf{y}_{2k}^{1,0,2e-1}) \equiv 0 \pmod 2$, and (iii) $\mathbf{y}_{2e-1} \notin \mathcal{R}$ or $\Sigma_{k=1}^{K+1} \psi(\mathbf{y}_{2k-1}^{1,0,2e}) \equiv 1, 2, 3 \pmod 4$, then set $\mathbf{x}' = \mathbf{y}^{s,p,2e-1}$, where s and p are chosen such that $\mathbf{y}_{2e-1}^{s,p,2e-1} \in \mathcal{R}$ and $\Sigma_{k=1}^{K+1} \psi(\mathbf{y}_{2k-1}^{s,p,2e-1}) \equiv 0 \pmod 4$, and go to Step 6.
5) Set $\mathbf{x}' = \mathbf{y}^{s,p,2e-2}$, where s and p are chosen such that $\mathbf{y}_{2e-2}^{s,p,2e-2} \in \mathcal{T}$ and $\Sigma_{k=1}^{K+1} \xi(\mathbf{y}_{2k}^{s,p,2e-2}) \equiv 0 \pmod 2$.
6) Set the decoder output as $\mathbf{u}' = (\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_K)$, where

$\mathbf{u}'_k = (\phi^{-1}(\mathbf{x}'_{2k-1}), \chi^{-1}(\mathbf{x}'_{2k}))$ for $k = 1, 2, \ldots, K$, and STOP.

In Step 2, the deletion in the transmitted code sequence $\mathbf{y}$ is determined to have occurred in $\mathbf{y}^{\text{tail}}_{2e-2}$, $\mathbf{y}_{2e-1}$ or $\mathbf{y}^{\text{head}}_{2e}$. Then, in Steps 3-5, it is determined whether the deletion occurred in $\mathbf{y}^{\text{head}}_{2e}$ (if the three conditions in Step 3 are all true), in $\mathbf{y}_{2e-1}$ (if the three conditions in Step 4 are all true), or in $\mathbf{y}^{\text{tail}}_{2e-2}$ (otherwise). Insertions are made accordingly. Finally, the binary information sequence is retrieved in Step 6.

Remarks: The code has rate $\frac{7K}{8(K+1)}$ bits/symbol. This can be improved to $\frac{7K+4}{8(K+1)}$ by encoding two extra information bits into the choice of $\mathbf{x}_{2K+1}$ and two extra information bits into the choice of $\mathbf{x}_{2K+2}$. The rate is close to (7/8) for large values of K. If no deletions occur, the proposed decoding procedure will correct any single substitution error and detect the occurrence of two substitution errors in each of the two substrings of length $4(K+1)$. If a single deletion error occurs and no substitution errors, then the deletion will be corrected by the proposed procedure. Again, the choice of K is a trade-off between efficiency (the higher the value of K, the higher the rate) and reliability (the lower the value of K, the lower the probability of uncorrected/undetected errors).

Mapping: The maps ϕ and χ used in the construction could be implemented through a table look-up. Still, some structure can be imposed. For ϕ, this can be done as described in the previous section. For χ, this can be done as follows: The binary sequence $\mathbf{b} = (b_0, b_1, b_2)$ is uniquely mapped to a member $\mathbf{t} = (t_0, t_1, t_2, t_3)$ from $\mathcal{T}$ as follows: The value of $b = b_0$ indicates that a member of $\mathcal{T}_b$ will be chosen. The integer representation q of (b_1, b_2) indicates that within $\mathcal{T}_b$ we choose the sequence with $t_q = 1$. For example, $\mathbf{b} = 110$ has $b = 1$ and $q = 2$ and is thus mapped to $\mathbf{t} = 3214 \in \mathcal{T}_1$.

Example: Let K=3 and let the information sequence be $\mathbf{u} = (0111, 000, 1000, 111, 1010, 101)$. Then the code sequence is $\mathbf{x} = (3421, 1423, 1324, 2341, 2413, 4123, 1342, 1423)$. We consider two cases for the received sequence $\mathbf{y}$ and give the corresponding decoding results.

- If $\mathbf{y} = (3423, 1423, 1324, 2241, 2413, 4123, 1342, 1423)$, i.e., substitution errors occurred in the fourth and fourteenth symbol, then the decoder observes that $\mathbf{y}_1$ is not in $\mathcal{R}$ and $\mathbf{y}_4$ is not in $\mathcal{T}$. From $\psi(\mathbf{y}_3)+\psi(\mathbf{y}_5)+\psi(\mathbf{y}_7) = 2+2+3 = 7 \equiv 3 \pmod 4$, the decoder finds that the first substrings should be in $\mathcal{R}_1$, and thus $\mathbf{x}'_1 = 3421$. Similarly, from $\xi(\mathbf{y}_2) + \xi(\mathbf{y}_6) + \xi(\mathbf{y}_8) = 0+1+0 \equiv 1 \pmod 2$, the decoder finds that the fourth substring should be in $\mathcal{T}_1$, and thus $\mathbf{x}'_4 = 2341$. Hence, both errors are corrected.
- If $\mathbf{y} = (3421, 1423, 1342, 3412, 4134, 1231, 3421, 423)$, i.e., a deletion occurred in the eleventh symbol, then the decoder detects that a deletion occurred from the length of the sequence $\mathbf{y}$. Since $\mathbf{y}^{\text{head}}_2 = 14$ and $\mathbf{y}^{\text{head}}_4 = 34$, the deletion is determined to have occurred in $\mathbf{y}^{\text{tail}}_2$, $\mathbf{y}_3$ or $\mathbf{y}^{\text{head}}_4$, i.e., $e = 2$ in Step 2 of the described decoding algorithm. Since $\Sigma_{k=1}^{4} \psi(\mathbf{y}^{1,0,2e}_{2k-1}) \equiv 1 \pmod 4$, it follows from Step 4 that the symbol $s = 2$ should be inserted at position $p = 2$ of $\mathbf{y}_3$. The resulting sequence $\mathbf{x}'$ is correctly decoded in Step 6.

The $M > 4$ case is not presented in this paper due to space constraints. Expanding the $\mathcal{R}$ and $\mathcal{T}$ sets in such a way that the sequences in $\mathcal{T}$ still start with specific combinations, while none of the sequences in $\mathcal{R}$ starts with those combinations, is the most important step. Once that has been done, the encoding and decoding will be similar to the $M = 4$ case.

VI. Conclusion

Sequences of permutations, rather than single permutations, are used as codewords to achieve error control capabilities. Two constructions are presented in this paper: one to correct substitution errors and the other to correct either deletion or substitution errors. Construction I is generalized for all values of M. Simple decoding algorithms are presented as well as methods to map from the binary data to permutations.

For any $M \geq 4$ and large values of K, the codes obtained by Construction I have a code rate close to $\log_2(M!)/M$, which is the maximum value for the rate of any permutation code. Thus, the error control capabilities are achieved with very little loss in rate, especially when K is large.

Future work includes researching different methods to construct the $\mathcal{R}$ and $\mathcal{T}$ sets, since these sets are the foundations for the two constructions. Increasing the error-correcting capabilities of the constructions should also be investigated. For higher values of M, M permutations can be included in every subset with a distance of M. The increased distance properties should lead to higher error-correcting capabilities. Adapting the algorithms to correct insertion errors or a combination of substitution and deletion errors should also be investigated.

References

[1] A. J. H. Vinck, "Coded modulation for powerline communications," *Proc. Int. J. Elec. Commun.*, vol. 54, no. 1, pp. 45–49, Jan. 2000.

[2] H. C. Ferreira, A. J. H. Vinck, T. G. Swart and I. de Beer, "Permutation trellis codes," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1782–1789, Nov. 2005.

[3] V. I. Levenshtein, "On perfect codes in deletion and insertion metric," *Discrete Math. Appl.*, vol. 2, no. 3, pp. 241–258, Jan. 1992.

[4] L. Cheng, T. G. Swart and H. C. Ferreira, "Synchronization using insertion/deletion correcting permutation codes," *Proc. IEEE Int. Symp. on Powerline Commun. and its Applic.*, Jeju Island, Korea, pp. 135–140, Apr. 2008.

[5] L. Cheng, T. G. Swart and H. C. Ferreira, "Re-synchronization of permutation codes with Viterbi-like decoding," *Proc. IEEE Int. Symp. on Powerline Commun. and its Applic.*, Dresden, Germany, pp. 36–40, Mar. 2009.

[6] R. Heymann and H. C. Ferreira, "Using tree structures to resynchronize permutation codes," *Proc. IEEE Int. Symp. on Powerline Commun. and its Applic.*, Rio de Janeiro, Brazil, pp. 108–113, Mar. 2010.

[7] T. Shongwe, T. G. Swart, H. C. Ferreira and T. van Trung, "Good synchronization sequences for permutation codes," *IEEE Trans. Commun.*, vol. 60, no. 5, pp. 1204–1208, May 2012.

[8] A. Jiang, R. Mateescu, M. Schwartz and J. Bruck, "Rank modulation for flash memories," *Proc. IEEE Int. Symp. Inform. Theory*, Toronto, Canada, pp. 1731–1735, Jul. 2008.

[9] G. Ungerboeck "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, vol. 28, no. 1, pp. 55–67, Jan 1982.

[10] M. Deza and S. A. Vanstone, "Bounds on permutation arrays," *Journal of Statistical Planning and Inference*, vol. 2, no. 2, pp. 197–209, 1978.

[11] H. C. Ferreira and A. J. H. Vinck, "Interference cancellation with permutation Trellis Codes," *Proc. 2000 IEEE Vehicular Technology Conference*, Boston, MA, USA, pp. 2401–2407, Sept. 2000.