Multiple Error-Correcting WOM-Codes

Eitan Yaakobi, *Student Member, IEEE*, Paul H. Siegel, *Fellow, IEEE*, Alexander Vardy, *Fellow, IEEE*, and Jack K. Wolf, *Life Fellow, IEEE*

***Abstract*—A Write Once Memory (WOM) is a storage medium with binary memory elements, called *cells*, that can change from the zero state to the one state only once. Examples of WOMs include punch cards and optical disks. WOM-codes, introduced by Rivest and Shamir, permit the reuse of a WOM by taking into account the location of cells that have already been changed to the one state. The objective in designing WOM-codes is to use the fewest number of cells to store a specified number of information bits in each of several reuses of the memory. An $[n, k, t]$ *WOM-code* $\mathcal{C}$ is a coding scheme for storing k information bits in n cells t times. At each write, the state of each cell can be changed, provided that the cell is changed from the zero state to the one state. The *rate* of $\mathcal{C}$, defined by $\mathcal{R}(\mathcal{C}) = kt/n$, indicates the total amount of information that is possible to store in a cell in t writes. Two WOM-code constructions correcting a single cell-error were presented by Zémor and Cohen. In this paper, we present another construction of a single-error-correcting WOM-code with a better rate. Our construction can be adapted also for single-error-detection, double-error-correction, and triple-error-correction. For the last case, we use triple-error-correcting BCH-like codes, which were presented by Kasami and more recently described again by Bracken and Helleseth. Finally, we show two constructions that can be combined for the correction of an arbitrary number of errors.**

***Index Terms*—Almost perfect nonlinear mapping, coding theory, error-correcting WOM-codes, flash memories, write once memory (WOM)-codes, write-once memories.**

I. Introduction

WRITE Once Memory (WOM) codes were first presented by Rivest and Shamir almost three decades ago [16]. The codes were designed for memories which consist of binary memory elements that can only be changed from a zero state to a one state. Examples of such memories include punch cards and optical disks. Since then, further results have appeared on this topic, e.g., [2], [4], [5], [7], [8], [15], [18], [23], [24]. Recently, such codes have been suggested for application to flash memories [9], [11], [14].

In a flash memory, the atomic memory element is a floating gate cell. The cell is electrically charged with electrons and can have multiple levels corresponding to different numbers of electrons in the cell [6]. Here, we are concerned with cells that take on two levels. The cells are arranged in rectangular arrays called blocks. Starting from an "all-zero" state, information is recorded in the blocks on a row-by-row basis. However, in order to rewrite a row in a previously written block, the entire block must first be erased, returning it to the "all-zero" state [6]. This block-erase operation introduces a significant delay and also has a detrimental effect on the lifetime of the memory. WOM-codes offer a way to reduce the number of such block erasures.

In the WOM model, the problem that has received the most attention is: what is the minimum number of cells n required to store k bits t times? Or, alternatively: what is the maximum number of bits k that can be written t times using n cells? A code that is designed for this problem is called a ***WOM-code*** $\mathcal{C}$. The ***rate*** $\mathcal{R}$ of a WOM-code $\mathcal{C}$ with t writes is the ratio of the total number of bits written to the memory, kt, to the number of cells n, that is, $\mathcal{R} = \frac{kt}{n}$.

The first example of a WOM-code, presented by Rivest and Shamir, could store two bits twice using only three cells [16]. Since then, several more WOM-code constructions have been presented, including tabular WOM-codes and "linear" WOM-codes [16]. Wolf *et al.* [15] provide several extensions of the results in [16], taking into consideration, for example, whether or not the previous state of the memory is known to the encoder and/or decoder. In [2] and [7], a "coset-coding" technique based upon binary linear codes was used to construct WOM-codes. Fiat and Shamir extended the WOM model for multi-level cells and also studied information-theoretic limits and code constructions for constrained sources. The capacity region of a binary WOM was calculated by Heegard [8] and the extension to non-binary cells with arbitrary constraints on the cell-state transitions was calculated by Fu and Han Vinck [5]. Recently, several more WOM-codes constructions were given in [13], [19]–[21]

Even though the problem of adapting WOM-codes to handle memory errors was suggested in the original Rivest-Shamir paper [16], the first construction of codes addressing this problem wasn't published until a few years later by Zémor [24] and Zémor and Cohen [23]. The capacity of a noisy WOM was studied by Heegard [8]. Recently, in [9], Jiang discussed

Manuscript received May 06, 2011; accepted August 29, 2011. Date of publication November 18, 2011; date of current version March 13, 2012. This work was supported in part by the University of California Lab Fees Research Program, Award 09-LR-06-118620-SIEP, the National Science Foundation under Grant CCF-1116739, and the Center for Magnetic Recording Research at the University of California, San Diego. Part of the material in this paper was presented at the 2010 IEEE International Symposium on Information Theory.

E. Yaakobi is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA, and also with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: eyaakobi@ucsd.edu).

P. H. Siegel and J. K. Wolf, deceased, are with the Department of Electrical and Computer Engineering and the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: psiegel@ucsd.edu; jwolf@ucsd.edu).

A. Vardy is with the Department of Electrical and Computer Engineering, the Department of Computer Science and Engineering, and the Department of Mathematics, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: avardy@ucsd.edu).

Communicated by M. Blaum, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2011.2176465

the generalization of error-correcting WOM-codes for the flash/floating codes model [10], [11], [14].

Two constructions of error-correcting WOM-codes were given in [23]. Both constructions correct a single cell-error during the writes. The first construction, based on a double-error-correcting BCH code, enables one to write k bits using $n = 2^k - 1$ cells $t \approx n/15.42$ times, so its rate is roughly $\frac{k}{15.42} \approx \frac{\log_2(15.42t+1)}{15.42}$. The second construction, which uses the same number of cells, is based on a triple-error-correcting BCH code and stores $2k$ bits $t \approx n/26.9$ times. Its rate is approximately $\frac{2k}{26.9} \approx \frac{\log_2(26.9t+1)}{13.45}$. While there are different ways to compare WOM-codes, we find that the appropriate figure of merit is to compare the rates under the assumption of a fixed number of writes. In general, the more writes the WOM-code can support, the better the rate it can achieve. The second construction in [23] is superior to the first one as it achieves a better rate even though its number of writes is smaller.

A simple scheme to construct an e-error-correcting WOM-code is based upon an existing WOM-code $\mathcal{C}$ that stores k bits t times in n cells. In this scheme, each one of the n cells is replicated $2e+1$ times so it is possible to correct any e or fewer cell-errors. If the WOM-code $\mathcal{C}$ has rate $\mathcal{R} = \frac{kt}{n}$, then the generated e-error-correcting WOM-code has rate $\frac{1}{2e+1}\mathcal{R} = \frac{kt}{(2e+1)n}$. For example, in [7], a WOM-code which stores k bits $t = 5 \cdot 2^{k-4} + 1$ times using $n = 2^k - 1$ cells, for $k \geqslant 4$ is presented. If we use this WOM-code to construct a single-error-correcting WOM-code, then its rate, $\frac{1}{3}\frac{k(5\cdot 2^{k-4}+1)}{2^k-1} > \frac{\log_2(3.2(t-1))}{9.6}$, outperforms for t large enough the rate of the two constructions in [23].

In this paper, we present several new constructions of error-correcting WOM-codes. In Section II, we give a precise definition of the problem and the general scheme underlying our codes. In Section III, we show our first construction of WOM-codes that can correct a single cell-erasure. These codes can alternatively detect a single cell-error. In Section IV, WOM-codes correcting a single cell-error are presented. The last construction is modified in Section V in order to construct double-error-correcting WOM-codes. In order to correct three cell errors, in Section VI, we find cyclic binary triple-error-correcting codes that satisfy the following property: each of the three roots of the code, α_1, α_2, α_3, is a primitive element and every pair of roots generates a double-error-correcting code. We show the existence of these codes using almost perfect nonlinear (APN) power functions [1], [12]. Finally, in Section VII, we give a construction that uses a triple-error-correcting WOM-code to construct WOM-codes correcting an arbitrary number of errors. Then, another recursive construction is given which can provide improved block-length for some rates and error-correction capabilities.

II. Preliminaries

In this work, the memory elements, called ***cells***, have two states: zero and one. At the beginning, all the cells are in their zero state. A ***programming operation*** changes the state of a cell from zero to one. This operation is irreversible in the sense that one cannot change the cell state from one to zero unless the entire memory is first erased. The ***memory-state vectors*** are all the binary vectors of length n, $\{0,1\}^n$. The ***data vectors*** are the set of all binary vectors of length k, $\{0,1\}^k$. Any WOM-code $\mathcal{C}$ is specified by its encoding map $\mathcal{E}_\mathcal{C}$ and decoding map $\mathcal{D}_\mathcal{C}$. The ***decoding map*** $\mathcal{D}_\mathcal{C} : \{0,1\}^n \to \{0,1\}^k$ assigns to each memory-state vector $\boldsymbol{c} \in \{0,1\}^n$ its corresponding data vector $\boldsymbol{v} = \mathcal{D}_\mathcal{C}(\boldsymbol{c}) \in \{0,1\}^k$. The ***encoding map*** $\mathcal{E}_\mathcal{C} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n \cup \{\mathsf{E}\}$ indicates for each data vector $\boldsymbol{v} \in \{0,1\}^k$ and previous memory-state vector $\boldsymbol{c} \in \{0,1\}^n$, a new memory-state vector $\boldsymbol{c}' = \mathcal{E}_\mathcal{C}(\boldsymbol{v}, \boldsymbol{c})$ such that $\mathcal{D}_\mathcal{C}(\boldsymbol{c}') = \boldsymbol{v}$, and $c_i \leqslant c'_i$, for all $1 \leqslant i \leqslant n$. In case such a $\boldsymbol{c}' \in \{0,1\}^n$ does not exist, the value of the encoding map is $\mathcal{E}_\mathcal{C}(\boldsymbol{v}, \boldsymbol{c}) = \mathsf{E}$.

Definition: An $[n, k, t]$ ***WOM-code*** $\mathcal{C}(\mathcal{E}_\mathcal{C}, \mathcal{D}_\mathcal{C})$ is a coding scheme which consists of n cells and is defined by its encoding and decoding maps, denoted by $\mathcal{E}_\mathcal{C}$ and $\mathcal{D}_\mathcal{C}$, respectively. The WOM-code $\mathcal{C}$ guarantees any t writes of a k-bit data vector $\boldsymbol{v}$ without producing the block erasure symbol E. The ***rate*** of the WOM-code $\mathcal{C}$ is defined as $\mathcal{R} = \frac{kt}{n}$.

Remark 1: It is possible to generalize the definition of WOM-codes to allow an arbitrary number of bits or symbols to be stored at each write. In this paper, we focus only on the case where the same number of bits is written at each write. However, we note that it is possible to change the constructions to support the case where a different number of bits is written on each write.

The following definitions are also used in our work.

1) An $[n, k, t]$ WOM-code that can correct e errors is called an $[n, k, t]$ ***e-error-correcting WOM-code***.
2) An $[n, k, t]$ WOM-code that can detect e errors is called an $[n, k, t]$ ***e-error-detecting WOM-code***.

The definition of the decoding map in the second case is extended to be $\mathcal{D}_\mathcal{C} : \{0,1\}^n \to \{0,1\}^k \cup \{\mathtt{F}\}$, where the symbol $\mathtt{F}$ indicates an error detection flag.

Remark 2: If after decoding on the ith write, a cell which is in state zero is erroneous, this error can be corrected (at least theoretically) prior to the next write by changing the state of this cell to a one. However, if after decoding on the ith write, a cell which is in state one is erroneous, the state of this cell cannot be changed prior to the next write. In this case, however, it is assumed that on the $(i+1)$th write the encoder knows that the cell's true state is a zero. There is no problem if the encoder wants to write a one in this cell. However, if the encoder wants to write a zero in this cell, then the error which was corrected on the ith write will also occur on the $(i+1)$th write because in this case it is not possible to physically change the cell's state. When we say that a WOM-code is an e-error-correcting code we mean that the code will correct e or fewer errors on each write but we realize that some of the errors which were corrected on one write could appear on subsequent writes. This information could be used in decoding but the decoder we consider here does not do so. We also assume here that there are no reading errors; that is, the correct state of a cell is always read.

In this paper, we present error-detecting and error-correcting WOM-codes, which have the following generic structure:

1) We assume that there exists an $[n, k, t]$ WOM-code $\mathcal{C}(\mathcal{E}_{\mathcal{C}}, \mathcal{D}_{\mathcal{C}})$. Its n cells are denoted by $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$ and called the ***information cells***. Note that this original code $\mathcal{C}$ *cannot* correct errors.
2) The constructed code consists of the n information cells $\boldsymbol{c}$, and r additional cells, called the ***redundancy cells***, and denoted by $\boldsymbol{p} = (p_0, \ldots, p_{r-1})$. The redundancy cells enable the decoder to correct cell-errors. That is, we get an $[n + r, k, t]$ WOM-code with some error correction/detection capabilities.

III. Single-Error-Detecting WOM-Codes

In this section, we present single-error-detecting WOM-codes. As described in Section II, we let $\mathcal{C}(\mathcal{E}_{\mathcal{C}}, \mathcal{D}_{\mathcal{C}})$ be an $[n, k, t]$ WOM-code, and its cells, called the information cells, are denoted by $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$. We construct an $[n + t, k, t]$ single-error-detecting WOM-code, denoted by $\mathcal{C}_{\mathrm{SED}}(\mathcal{E}_{\mathcal{C}_{\mathrm{SED}}}, \mathcal{D}_{\mathcal{C}_{\mathrm{SED}}})$.

In this construction there are t redundancy cells, denoted by $\boldsymbol{p} = (p_0, p_1, \ldots, p_{t-1})$, i.e., the value of r in the general structure is t. The code $\mathcal{C}_{\mathrm{SED}}$ satisfies the following property: at each write, the parity of the t redundancy cells, $\sum_{i=0}^{t-1} p_i$, and the parity of the n information cells, $\sum_{i=0}^{n-1} c_i$, are the same.

Theorem 1: If $\mathcal{C}$ is an $[n, k, t]$ WOM-code, then $\mathcal{C}_{\mathrm{SED}}$ is an $[n + t, k, t]$ single-error-detecting WOM-code.

Proof: We prove this theorem by showing the correctness of the encoding and decoding maps. In the encoding map $\mathcal{E}_{\mathcal{C}_{\mathrm{SED}}}$, the new data vector $\boldsymbol{v}$ is encoded in the n information cells by the encoding map $\mathcal{E}_{\mathcal{C}}(\boldsymbol{c}, \boldsymbol{v})$. If the parity of the information cells is changed, then one of the t redundancy cells is programmed. Since there are initially t redundancy cells in state zero and each time at most one of them is programmed, there is at least one unprogrammed cell at each write.

In the decoding map $\mathcal{D}_{\mathcal{C}_{\mathrm{SED}}}$, at most one of the cells is in error. If the information cell's parity is different than the redundancy cell's parity, then the flag $\mathtt{F}$ is returned to indicate a single error detection. Otherwise, the data vector $\boldsymbol{v}$ is simply decoded by the decoding map $\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(\boldsymbol{c})$. ∎

This scheme can be applied to all known WOM-codes. In particular, the next example shows how to adapt the scheme to WOM-codes which are based on Hamming codes [2], [7].

Example 1: In [2], a construction of WOM-codes, based on Hamming codes, is presented. For $k \geqslant 4$, the construction gives a $[2^k - 1, k, 2^{k-2} + 2]$ WOM-code, and for $k = 2, 3$ a $[2^k - 1, k, 2^{k-2} + 1]$ WOM-code. In particular, the $[3, 2, 2]$ WOM-code, presented by Rivest and Shamir [16], is a special case of this construction for $k = 2$. Later, in [7] the case $k \geqslant 4$ was improved and $[2^k - 1, k, 5 \cdot 2^{k-4} + 1]$ WOM-codes were presented.

For $k \geqslant 4$, Zémor showed that it is possible to change the construction such that, excluding the first write, the number of programmed cells at each write is even [24]. Therefore, the parity bit changes its values at most once. Thus, one redundancy cell is sufficient for the construction and we get a $[2^k, k, 5 \cdot 2^{k-4} + 1]$ single-error-detecting code. In fact, a similar construction to this code with the same parameters was presented by Zémor in [24].

TABLE I
A Construction for the $[4, 2, 2]$ Single-Error-Detecting WOM-Code

Data Vector	First Memory State Vector	Alternative Memory State Vector
00	000**1**	111**0**
01	001**0**	110**1**
10	010**0**	101**1**
11	100**0**	011**1**

For $k = 2, 3$, the construction is slightly modified. At each write, the redundancy cells' parity is the complement of the information cells' parity. Then, at most $2^{k-2} = t - 1$ cells are sufficient and thus a $[2^k + 2^{k-2} - 1, k, 2^{k-2} + 1]$ single-error-detecting code exists. Table I demonstrates the construction for the $[4, 2, 2]$ single-error-detecting WOM-code. The bold font represents the bit in the redundancy cell. A similar table can be built for the $[9, 3, 3]$ single-error-detecting WOM-code.

IV. Single-Error-Correcting WOM-Codes

In order to construct single-error-correcting WOM-codes, we start as in Section III with an $[n, k, t]$ WOM-code, $\mathcal{C}(\mathcal{E}_{\mathcal{C}}, \mathcal{D}_{\mathcal{C}})$. Its information cells are $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$ and we add r redundancy cells, $\boldsymbol{p} = (p_0, \ldots, p_{r-1})$, that form a word in $\mathcal{C}_{\mathcal{D}}(\mathcal{E}_{\mathcal{C}_{\mathcal{D}}}, \mathcal{D}_{\mathcal{C}_{\mathcal{D}}})$, an $[r, \lceil \log_2(n + 1) \rceil, t]$ single-error-detecting WOM-code. Then, we construct an $[n+r, k, t]$ single-error-correcting WOM-code, denoted by $\mathcal{C}_{\mathrm{SEC}}(\mathcal{E}_{\mathcal{C}_{\mathrm{SEC}}}, \mathcal{D}_{\mathcal{C}_{\mathrm{SEC}}})$, as follows.

At each write we generate a $\lceil \log_2(n + 1) \rceil$-bit vector, called the ***syndrome*** and denoted by $\boldsymbol{s}$. The syndrome will correspond to the redundancy bits of a Hamming code (or a shortened Hamming code) of length n, and will make it possible to locate an information cell in error.

Next, and in Sections V and VI, we provide the exact specification of the given error-correcting WOM-codes by their encoding and decoding maps. These maps are described algorithmically using a pseudo-code notation. In this specification we will use the encoding and decoding maps $\mathcal{E}_{\mathcal{C}}$, $\mathcal{D}_{\mathcal{C}}$ of the WOM-code $\mathcal{C}$ and the encoding and decoding maps $\mathcal{E}_{\mathcal{C}_{\mathcal{D}}}, \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}$ of the single-error-detecting WOM-code $\mathcal{C}_{\mathcal{D}}$. We let α be a primitive element in the extension field $GF(2^{\lceil \log_2(n+1) \rceil})$.

Encoding Map $\mathcal{E}_{\mathcal{C}_{\mathrm{SEC}}}$: The input is the memory-state vector $(\boldsymbol{c}, \boldsymbol{p})$ and the new k-bit data vector $\boldsymbol{v}$. The output is either a new memory-state vector $(\boldsymbol{c}', \boldsymbol{p}')$ or the erasure symbol $\mathtt{E}$.

1. $\boldsymbol{c}' = \mathcal{E}_{\mathcal{C}}(\boldsymbol{c}, \boldsymbol{v})$;
2. `if` $(\boldsymbol{c}' == \mathtt{E})$ `return` $\mathtt{E}$;
3. $\boldsymbol{s} = \sum_{i=0}^{n-1} c'_i \alpha^i$;
4. $\boldsymbol{p}' = \mathcal{E}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}, \boldsymbol{s})$;
5. `if` $(\boldsymbol{p}' == \mathtt{E})$ `return` $\mathtt{E}$;
6. `return` $(\boldsymbol{c}', \boldsymbol{p}')$;

Note that since the encoding map $\mathcal{E}_{\mathcal{C}}$ can write t messages of k-bits each and the encoding map $\mathcal{E}_{\mathcal{C}_{\mathcal{D}}}$ can write t times the

$\lceil \log_2(n+1) \rceil$-bit syndrome $\boldsymbol{s}$, the encoding map $\mathcal{E}_{\mathcal{C}_{\rm SEC}}$ also can write k-bits t times.

Decoding Map $\mathcal{D}_{\mathcal{C}_{\rm SEC}}$: The input is the memory-state vector $(\boldsymbol{c}', \boldsymbol{p}')$. The output is the decoded k-bit data vector $\boldsymbol{v}$.

1. $\boldsymbol{s}'' = \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}')$;
2. `if` $(\boldsymbol{s}'' == \mathtt{F})$
3. $\{\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(\boldsymbol{c}')$; `return` $\boldsymbol{v}; \}$
4. $\boldsymbol{s}' = \sum_{i=0}^{n-1} c'_i \alpha^i$;
5. `if` $(\boldsymbol{s}' == \boldsymbol{s}'')$
6. $\{\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(\boldsymbol{c}')$; `return` $\boldsymbol{v}; \}$
7. $i = \log_\alpha(\boldsymbol{s}' + \boldsymbol{s}'')$;
8. $\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(c'_0, \ldots, c'_{i-1}, 1 - c'_i, c'_{i+1}, \ldots, c'_{n1})$;
9. `return` $\boldsymbol{v}$;

The syndrome $\boldsymbol{s}''$ is decoded by applying the decoding map $\mathcal{D}_{\mathcal{C}_{\mathcal{D}}}$ on the redundancy cells $\boldsymbol{p}'$ (line 1). The code $\mathcal{C}_{\mathcal{D}}$ is a single-error-detecting WOM-code and hence by its decoding map $\mathcal{D}_{\mathcal{C}_{\mathcal{D}}}$ it is possible to determine if there is an error in one of the r redundancy cells (line 2). We distinguish between the following two cases:

1) If one of the redundancy cells is in error, i.e., the condition in line 2 holds, then there is no error in the information cells and $\boldsymbol{v}$ is decoded by the decoding map $\mathcal{D}_{\mathcal{C}}$ (line 3).
2) If there is no error in the redundancy cells, then $\boldsymbol{s}''$ is the correct value of the syndrome $\boldsymbol{s}$. The received syndrome $\boldsymbol{s}'$ from the received n information cells is $\boldsymbol{s}' = \sum_{i=0}^{n-1} c'_i \alpha^i$ (line 4). If $\boldsymbol{s}' = \boldsymbol{s}''$ (line 5), then there is no error in the n information cells and it is possible to decode the correct value of the data vector $\boldsymbol{v}$ (line 6). Otherwise, if the ith cell is in error, then $\boldsymbol{s}' + \boldsymbol{s}'' = \alpha^i$. The calculation of $\log_\alpha(\boldsymbol{s}' + \boldsymbol{s}'')$ returns the value i such that $\alpha^i = \boldsymbol{s}' + \boldsymbol{s}''$ (line 7). This identifies the erroneous cell and again we can decode the data vector $\boldsymbol{v}$ (line 8).

Thus we have proved the following theorem.

Theorem 2: If $\mathcal{C}$ is an $[n, k, t]$ WOM-code, $\mathcal{C}_{\mathcal{D}}$ is an $[r, \lceil \log_2(n+1) \rceil, t]$ single-error-detecting WOM-code, then $\mathcal{C}_{\rm SEC}$ is an $[n+r, k, t]$ single-error-correcting WOM-code.

The next example demonstrates how to use this construction to build specific single-error-correcting WOM-codes.

Example 2: As in Example 1, the code $\mathcal{C}$ is chosen to be the $[2^k - 1, k, 5 \cdot 2^{k-4} + 1]$ WOM-code for $k \geqslant 4$ from [7]. Therefore, $n = 2^k - 1$, and $\lceil \log_2(n+1) \rceil = k$, so we can use the $[2^k, k, 5 \cdot 2^{k-4} + 1]$ single-error-detecting WOM-code from Example 1. The resulting $[2 \cdot 2^k - 1, k, 5 \cdot 2^{k-4} + 1]$ single-error-correcting WOM-code has rate

$$\mathcal{R} = \frac{k(5 \cdot 2^{k-4} + 1)}{2 \cdot 2^k - 1} > \frac{\log_2(3.2(t-1))}{6.4}$$

which is an improvement upon the constructions in [23] and the simple construction presented in the Introduction.

V. Double-Error-Correcting WOM-Codes

The double-error-correcting WOM-codes construction is very similar to the single-error-correcting case in Section IV, where the same WOM-codes $\mathcal{C}$, $\mathcal{C}_{\mathcal{D}}$ are used. There are $2r$ redundancy cells, partitioned into two r-cell groups, $\boldsymbol{p}_1 = (p_0, p_1, \ldots, p_{r-1})$ and $\boldsymbol{p}_2 = (p_r, p_1, \ldots, p_{2r-1})$. The redundancy groups $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ store $\lceil \log_2(n+1) \rceil$-bit syndrome vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, respectively. The two syndromes correspond to the two roots α, α^3 of a double-error-correcting BCH code, denoted by $\mathcal{C}_{2\text{-BCH}}$, where α is a primitive element in the field $GF(2^{\lceil \log_2(n+1) \rceil})$. In this construction, $\lceil \log_2(n+1) \rceil$ is assumed to be an odd integer. The code is denoted by $\mathcal{C}_{\rm DEC}(\mathcal{E}_{\mathcal{C}_{\rm DEC}}, \mathcal{D}_{\mathcal{C}_{\rm DEC}})$.

Encoding Map $\mathcal{E}_{\mathcal{C}_{\rm DEC}}$: The input is the memory-state vector $(\boldsymbol{c}, \boldsymbol{p}_1, \boldsymbol{p}_2)$ and the new k-bit data vector $\boldsymbol{v}$. The output is either a new memory-state vector $(\boldsymbol{c}', \boldsymbol{p}'_1, \boldsymbol{p}'_2)$ or the erasure symbol $\mathtt{E}$.

1. $\boldsymbol{c}' = \mathcal{E}_{\mathcal{C}}(\boldsymbol{c}, \boldsymbol{v})$;
2. `if` $(\boldsymbol{c}' == \mathtt{E})$ `return` $\mathtt{E}$;
3. $\boldsymbol{s}_1 = \sum_{i=0}^{n-1} c'_i \alpha^i$; $\boldsymbol{s}_2 = \sum_{i=0}^{n-1} c'_i \alpha^{3i}$;
4. $\boldsymbol{p}'_1 = \mathcal{E}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_1, \boldsymbol{s}_1)$; $\boldsymbol{p}'_2 = \mathcal{E}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_2, \boldsymbol{s}_2)$;
5. `if` $((\boldsymbol{p}'_1 == \mathtt{E})\mathtt{OR}(\boldsymbol{p}'_2 == \mathtt{E}))$ `return` $\mathtt{E}$;
6. `return` $(\boldsymbol{c}', \boldsymbol{p}'_1, \boldsymbol{p}'_2)$.

For the decoding map $\mathcal{D}_{\mathcal{C}_{\rm DEC}}$, we use the single-error-correcting WOM-code decoding map $\mathcal{D}_{\mathcal{C}_{\rm SEC}}$, which receives as its input n information cells and r redundancy cells. Note that while the code $\mathcal{C}_{\rm SEC}$ uses a fixed primitive element $\alpha \in GF(2^{\lceil \log_2(n+1) \rceil})$, it is possible to use any other primitive element in the field $GF(2^{\lceil \log_2(n+1) \rceil})$. We slightly modify the input arguments of the decoding map $\mathcal{D}_{\mathcal{C}_{\rm SEC}}$ such that the primitive element is its first parameter. The modified decoding map is denoted by $\mathcal{D}'_{\mathcal{C}_{\rm SEC}}$. We use the decoding map $\mathcal{D}_{\mathcal{C}_{2-\rm BCH}}$ of the double-error-correcting BCH code. Its input is the $2\lceil \log_2(n+1) \rceil$ syndrome bits; its output is the error vector.

Decoding map $\mathcal{D}_{\mathcal{C}_{\rm DEC}}$: The input is the memory-state vector $(\boldsymbol{c}', \boldsymbol{p}'_1, \boldsymbol{p}'_2)$. The output is the decoded k-bit data vector $\boldsymbol{v}$.

1. $\boldsymbol{s}''_1 = \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}'_1)$; $\boldsymbol{s}''_2 = \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}'_2)$;
2. `if` $(\boldsymbol{s}''_1 == \mathtt{F})$
3. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\rm SEC}}(\alpha^3, \boldsymbol{c}', \boldsymbol{p}'_2)$; `return` $\boldsymbol{v}$; $\}$
4. `if` $(\boldsymbol{s}''_2 == \mathtt{F})$
5. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\rm SEC}}(\alpha, \boldsymbol{c}', \boldsymbol{p}'_1)$; `return` $\boldsymbol{v}; \}$
6. $\boldsymbol{s}'_1 = \sum_{i=0}^{n-1} c'_i \alpha^i$; $\boldsymbol{s}'_2 = \sum_{i=0}^{n-1} c'_i \alpha^{3i}$;
7. `if` $((\boldsymbol{s}'_1 == \boldsymbol{s}''_1)\mathtt{OR}(\boldsymbol{s}'_2 == \boldsymbol{s}''_2))$
8. $\{\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(\boldsymbol{c}')$; `return` $\boldsymbol{v}; \}$
9. $\boldsymbol{e}' = \mathcal{D}_{\mathcal{C}_{2\text{-BCH}}}(\boldsymbol{s}'_1 + \boldsymbol{s}''_1, \boldsymbol{s}'_2 + \boldsymbol{s}''_2)$;
10. $\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(\boldsymbol{c}' + \boldsymbol{e}')$;
11. `return` $\boldsymbol{v}$;

The two syndromes $\boldsymbol{s}_1''$, $\boldsymbol{s}_2''$ are decoded using the redundancy cells and the decoding map $\mathcal{D}_{\mathcal{C}_\mathcal{D}}$ (line 1). If $\boldsymbol{s}_1'' = \mathtt{F}$ (line 2) then there is at least one error in the redundancy cells of group $\boldsymbol{p}_1'$, and at most one error in the information cells $\boldsymbol{c}'$ and the second redundancy group $\boldsymbol{p}_2'$. Therefore, it possible to decode the data vector $\boldsymbol{v}$ by applying the decoding map $\mathcal{D}'_{\mathcal{C}_{\mathrm{SEC}}}$ to the cells in $\boldsymbol{c}'$ and $\boldsymbol{p}_2'$ while taking α^3 to be the primitive element (line 3). Note that since $\lceil \log_2(n+1) \rceil$ is an odd integer, α^3 is also a primitive element in $GF(2^{\lceil \log_2(n+1) \rceil})$. Similarly, if $\boldsymbol{s}_2'' = \mathtt{F}$ (line 4), then we decode by applying the decoding map $\mathcal{D}'_{\mathcal{C}_{\mathrm{SEC}}}$ to the cells $\boldsymbol{c}'$ and $\boldsymbol{p}_1'$, while α is the primitive element (line 5).

If according to the decoding map $\mathcal{D}_{\mathcal{C}_\mathcal{D}}$, no error is decoded in both the redundancy cell groups, then either there is no error in all the redundancy cells or there are exactly two errors in one of the two redundancy cell groups. First, the syndromes $\boldsymbol{s}_1'$, $\boldsymbol{s}_2'$ from the received n information cells are calculated (line 6). Then, we consider the following two cases:

1) If $\boldsymbol{s}_1' = \boldsymbol{s}_1''$ or $\boldsymbol{s}_2' = \boldsymbol{s}_2''$ (line 7), then necessarily there is no error in the n information cells and the k-bit data vector is calculated and returned (line 8). (This is true since if there is at least one error in the information cells then there is no error in the redundancy cells and neither of these equalities holds, which is a contradiction.)
2) If $\boldsymbol{s}_1' \neq \boldsymbol{s}_1''$ and $\boldsymbol{s}_2' \neq \boldsymbol{s}_2''$ (line 9) then at least one error occurred in the n information cells and no errors in the redundancy cells. The error vector is found by applying the decoding algorithm of the two-error-correcting BCH code, $\mathcal{D}_{\mathcal{C}_{2\text{-BCH}}}$, to $\boldsymbol{s}_1' + \boldsymbol{s}_1''$ and $\boldsymbol{s}_2' + \boldsymbol{s}_2''$ (line 9). Then, we know the correct value of the n information cells and it is again possible to successfully decode the data vector $\boldsymbol{v}$ (line 10).

We summarize this construction in the following theorem.

Theorem 3: If $\mathcal{C}$ is an $[n, k, t]$ WOM-code, $\mathcal{C}_\mathcal{D}$ is an $[r, \lceil \log_2(n+1) \rceil, t]$ single-error-detecting WOM-code, and $\lceil \log_2(n+1) \rceil$ is an odd integer, then $\mathcal{C}_{\mathrm{DEC}}$ is an $[n+2r, k, t]$ double-error-correcting WOM-code.

The construction does not work if $\lceil \log_2(n+1) \rceil$ is an even integer since α^3 is no longer a primitive element in the field $GF(2^{\lceil \log_2(n+1) \rceil})$, and thus the decoding map in line 3 cannot succeed. Clearly, it is possible to modify it by working over the field $GF(2^{1+\lceil \log_2(n+1) \rceil})$ and storing syndromes of $1 + \lceil \log_2(n+1) \rceil$ bits. However, we can also modify the construction to handle this case, without changing the field size, by adding t more cells, as we now describe.

Assume that $\lceil \log_2(n+2) \rceil$ is an even integer. The required modifications to the encoding and decoding maps of the previous construction are as follows:

1) Instead of using the $[n, k, t]$ WOM-code $\mathcal{C}$, an $[n+t, k, t]$ single-error-detecting WOM-code is used and we denote it by $\mathcal{C}'(\mathcal{E}_{\mathcal{C}'}, \mathcal{D}_{\mathcal{C}'})$. The t additional redundancy cells are denoted by $\boldsymbol{q} = (q_0, \ldots, q_{t-1})$.
2) Instead of using the root α^3 we use the root α^{-1}.
3) The syndromes $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are calculated according to the new roots applied to the information cells $\boldsymbol{c}$ and their parity value, which is stored in the new redundancy cells $\boldsymbol{q}$.

The input and output to the encoding map are changed accordingly where the memory-state vector is $(\boldsymbol{c}, \boldsymbol{q}, \boldsymbol{p}_1, \boldsymbol{p}_2)$. In the first and second lines, we use the encoding map $\mathcal{E}_{\mathcal{C}'}$ instead of $\mathcal{E}_\mathcal{C}$ on the cells $(\boldsymbol{c}, \boldsymbol{q})$. The syndrome values in line 3 and the returned new memory-state vector in line 6 are also changed accordingly.

1. $(\boldsymbol{c}', \boldsymbol{q}') = \mathcal{E}_{\mathcal{C}'}((\boldsymbol{c}, \boldsymbol{q}), \boldsymbol{v})$;
2. `if` $((\boldsymbol{c}', \boldsymbol{q}') == \mathtt{E})$ `return` E;
3. $\boldsymbol{s}_1 = \sum_{i=0}^{n-1} c_i' \alpha^i + (\sum_{i=0}^{t-1} q_i') \alpha^n$;
   $\boldsymbol{s}_2 = \sum_{i=0}^{n-1} c_i' \alpha^{-i} + (\sum_{i=0}^{t-1} q_i') \alpha^{-n}$;
6. `return` $(\boldsymbol{c}', \boldsymbol{q}', \boldsymbol{p}_1', \boldsymbol{p}_2')$;

The decoding algorithm is also very similar. Since we use the root α^{-1} and also the value of the t new redundancy cells, lines 3 and 6 are changed as follows. Note that α^{-1} is also a primitive element and therefore the decoding map in line 3 succeeds.

3. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\mathrm{SEC}}}(\alpha^{-1}, \boldsymbol{c}', \boldsymbol{p}_2')$; `return` $\boldsymbol{v}$; $\}$
6. $\boldsymbol{s}_1' = \sum_{i=0}^{n-1} c_i' \alpha^i + (\sum_{i=0}^{t-1} q_i') \alpha^n$;
   $\boldsymbol{s}_2' = \sum_{i=0}^{n-1} c_i' \alpha^{-i} + (\sum_{i=0}^{t-1} q_i') \alpha^{-n}$.

If the decoder reaches line 9, then there is at least one error in the n information cells $\boldsymbol{c}'$ and t redundancy cells $\boldsymbol{q}'$. The main difference in the decoding is that at this line we need to know if there are one or two cells in error among the n information cells $\boldsymbol{c}'$ and t redundancy cells $\boldsymbol{q}'$. If there is a single error, that is, the parity of the n information cells and the parity of the t additional redundancy cells are not the same (line 9), then we can decode the data vector using the decoding map $\mathcal{D}'_{\mathcal{C}_{\mathrm{SEC}}}$ with the root α since there is at most one error in the information cells and no error in the redundancy cells $\boldsymbol{p}_1'$ (line 10). Otherwise, there are exactly two errors in the n information cells and t redundancy cells. The values of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ which are calculated in line 11 are of the form

$$\boldsymbol{e}_1 = \alpha^i + \alpha^j, \ \boldsymbol{e}_2 = \alpha^{-i} + \alpha^{-j},$$

for some $0 \leqslant i, j \leqslant n$, $i \neq j$, and

$$\begin{aligned} &\boldsymbol{e}_1(\boldsymbol{e}_1^2 + \boldsymbol{e}_1 \boldsymbol{e}_2^{-1}) \\ &= (\alpha^i + \alpha^j)\left((\alpha^i + \alpha^j)^2 + (\alpha^i + \alpha^j)\frac{\alpha^{i+j}}{\alpha^i + \alpha^j}\right) \\ &= (\alpha^i + \alpha^j)\left(\alpha^{2i} + \alpha^{2j} + \alpha^{i+j}\right) = \alpha^{3i} + \alpha^{3j}. \end{aligned}$$

Therefore, the values of i and j, i.e., the error vector, can be found by applying the decoding procedure $\mathcal{D}_{\mathcal{C}_{2\text{-BCH}}}$ to $\boldsymbol{e}_1$ and $\boldsymbol{e}_1(\boldsymbol{e}_1^2 + \boldsymbol{e}_1 \boldsymbol{e}_2^{-1})$ (line 12). Next, the data vector can be successfully decoded (line 13). Note that the error vector in line 12 consists of $n+1$ bits while for the decoding map in line 13 we need only its first n bits.

9. `if` $((\sum_{i=0}^{n-1} c_i') \; ! = (\sum_{i=0}^{t-1} q_i'))$
10. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\mathrm{SEC}}}(\alpha, \boldsymbol{c}', \boldsymbol{p}_1')$; `return` $\boldsymbol{v}$; $\}$
11. $\boldsymbol{e}_1 = \boldsymbol{s}_1' + \boldsymbol{s}_1''$; $\boldsymbol{e}_2 = \boldsymbol{s}_2' + \boldsymbol{s}_2''$;
12. $\boldsymbol{e}' = \mathcal{D}_{\mathcal{C}_{2\text{-BCH}}}(\boldsymbol{e}_1, \boldsymbol{e}_1(\boldsymbol{e}_1^2 + \boldsymbol{e}_1 \boldsymbol{e}_2^{-1}))$;
13. $\boldsymbol{v} = \mathcal{D}_\mathcal{C}(\boldsymbol{c}' + \boldsymbol{e}')$;
14. `return` $\boldsymbol{v}$;

To conclude, we state the following theorem.

Theorem 4: Let $\mathcal{C}$ be an $[n, k, t]$ WOM-code and $\mathcal{C}_{\mathcal{D}}$ be an $[r, \lceil \log_2(n+2) \rceil, t]$ single-error-detecting WOM-code. Suppose $\lceil \log_2(n+2) \rceil$ is an even integer. Then there exists an $[n+2r+t, k, t]$ double-error-correcting WOM-code.

VI. Triple-Error Correcting WOM-Codes

From the previous sections we might think that a general scheme to construct an e-error correcting WOM-code is to combine an existing WOM-code and a cyclic e-error-correcting code, where the latter code is defined by e roots $\alpha_1, \ldots, \alpha_e$. However, not every e-error-correcting code would work in this scheme. For example, in the double-error-correcting construction in Section V, the BCH code with roots α and α^3 cannot work if $\lceil \log_2(n+1) \rceil$ is an even integer. This results from the fact that α^3 is not a primitive element and hence the code generated only by α^3 is not a single-error-correcting code. For arbitrary e, if the cyclic e-error-correcting code is defined by e roots, then a necessary but not sufficient condition for this scheme to work is that every subset of $k \leqslant e$ roots generates a cyclic k-error-correcting code. We state this property in the following definition.

Definition: Let n be an integer and $\alpha_1, \ldots, \alpha_e$ be e different elements in the field $GF(2^n)$. Let the code $\mathcal{C}(\alpha_1, \ldots, \alpha_e)$ be a cyclic error-correcting code of length $2^n - 1$ with roots $\alpha_1, \ldots, \alpha_e$. The code $\mathcal{C}(\alpha_1, \ldots, \alpha_e)$ is called a **strong e-error-correcting code** if for every $1 \leqslant k \leqslant e$ and every set of k distinct elements $\alpha_{i_1}, \ldots, \alpha_{i_k} \in \{\alpha_1, \ldots, \alpha_e\}$, the code $\mathcal{C}(\alpha_{i_1}, \ldots, \alpha_{i_k})$ is a k-error-correcting code.

We note that finding strong e-error-correcting codes is a fascinating problem by itself but is beyond the scope of this paper. Next, we show how to choose the roots α_1, α_2, α_3 such that $\mathcal{C}(\alpha_1, \alpha_2, \alpha_3)$ is a strong triple-error-correcting code. For the following discussion, α is assumed to be a primitive element in $GF(2^n)$. The following result was proved by Kasami in [12].

Theorem 5: [12]. Let n be an odd integer and $\gcd(n, k) = 1$. Then, $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is a cyclic triple-error-correcting code.

In [1], the authors show an alternative proof to the last theorem and state the following lemma.

Lemma 6: Let n be an integer and $\gcd(n, k) = 1$. Then, $\mathcal{C}(\alpha, \alpha^{2^k+1})$ is a cyclic double-error-correcting code.

These two results imply the following lemma.

Lemma 7: Let n be an integer such that $\gcd(n, 6) = 1$, and let $k = \frac{n-1}{2}$. Then, the following properties hold.

1) The codes $\mathcal{C}(\alpha)$, $\mathcal{C}(\alpha^{2^k+1})$, $\mathcal{C}(\alpha^{2^{3k}+1})$ are cyclic single-error-correcting codes.
2) The codes $\mathcal{C}(\alpha, \alpha^{2^k+1})$, $\mathcal{C}(\alpha, \alpha^{2^{3k}+1})$ are cyclic double-error-correcting codes.
3) The code $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is a cyclic triple-error-correcting code.

Proof:

1) Since $k = \frac{n-1}{2}$, we know that $2^k + 1$ is a divisor of $2^{n-1} - 1$. Since $\gcd(2^n - 1, 2^{n-1} - 1) = 1$, we conclude that $\gcd(2^n - 1, 2^k + 1) = 1$. Therefore α^{2^k+1} is a primitive element in $GF(2^n)$ and the code $\mathcal{C}(\alpha^{2^k+1})$ is a cyclic single-error-correcting code. Since $\gcd(n, 6) = 1$, it follows also that $\gcd(2^n - 1, 2^{3k} + 1) = 1$, and therefore the code $\mathcal{C}(\alpha^{2^{3k}+1})$ is a cyclic single-error-correcting code as well.
2) Since $\gcd(n, k) = 1$, the condition of Lemma 6 holds and the code $\mathcal{C}(\alpha, \alpha^{2^k+1})$ is a double-error-correcting code. Similarly, since $\gcd(n, 6) = 1$ and $\gcd(n, k) = 1$, it follows that $\gcd(n, 3k) = 1$, and again by Lemma 6, the code $\mathcal{C}(\alpha, \alpha^{2^{3k}+1})$ is a double-error-correcting code.
3) Since $\gcd(n, 6) = 1$, n is necessarily an odd integer and since $\gcd(n, k) = 1$ the conditions of Theorem 5 hold. Therefore, the code $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is a triple-error-correcting code.

■

We note that at this point the code $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is "almost" a strong triple-error-correcting code. All that remains to be shown is that the code $\mathcal{C}(\alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is a double-error-correcting code. Before doing so, we state the definition of an almost perfect nonlinear mapping.

Definition: A mapping $f : GF(p^n) \to GF(p^n)$ is called an **almost perfect nonlinear (APN) mapping** if each equation

$$f(x + a) - f(x) = b$$

for $a, b \in GF(p^n)$ and $a \neq 0$ has at most two solutions in $GF(p^n)$. If f is an APN mapping and is of the form $f(x) = x^d$ then f is called an **almost perfect nonlinear power mapping**.

The next lemma was proved in [12].

Lemma 8: If n is an odd integer, $2 \leqslant k \leqslant \frac{n-1}{2}$, and $\gcd(n, k) = 1$ then the mapping $f(x) = x^{2^{2k}-2^k+1}$ over $GF(2^n)$ is an APN mapping.

The proof of the next lemma follows an outline similar to that of the proof of Theorem 1 in [1].

Lemma 9: If n, k are integers, $\gcd(n, 6) = 1$ and $k = \frac{n-1}{2}$, then $\mathcal{C}(\alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is a double-error-correcting code.

Proof: Note first that α^{2^k+1} is a primitive element in $GF(2^n)$ since $\gcd(2^n - 1, 2^k + 1) = 1$. Also, $\gcd(n, k) = 1$ and n is an odd integer, so, according to Lemma 8, $f(x) = x^d$ is an APN power mapping, where $d = 2^{2k} - 2^k + 1$. We denote $\gamma = \alpha^{2^k+1}$, and hence need to prove that $\mathcal{C}(\gamma, \gamma^d)$ is a double-error-correcting code.

Assume to the contrary that the code is not a double-error-correcting code. Clearly, there are no codewords of weight one or two and hence there exists a codeword of weight three or four. Assume there exists a codeword of weight four. Then, there exist four integers $0 \leqslant i_1 < i_2 < i_3 < i_4 \leqslant 2^n - 2$ such that

$$\gamma^{i_1} + \gamma^{i_2} + \gamma^{i_3} + \gamma^{i_4} = 0$$
$$(\gamma^{i_1})^d + (\gamma^{i_2})^d + (\gamma^{i_3})^d + (\gamma^{i_4})^d = 0.$$

The last two equations can be written as follows:

$$\gamma^{i_1} + \gamma^{i_2} = a = \gamma^{i_3} + \gamma^{i_4}$$
$$(\gamma^{i_1})^d + (\gamma^{i_2})^d = b = (\gamma^{i_3})^d + (\gamma^{i_4})^d,$$

for some $a, b \in GF(2^n)$, and $a \neq 0$. Hence, the equation

$$(x+a)^d + x^d = b$$

has four different solutions: $\gamma^{i_1}, \gamma^{i_2}, \gamma^{i_3}, \gamma^{i_4}$. This is a contradiction since x^d is an APN mapping. The case of a codeword of weight three is handled similarly. ■

From Lemma 7 and Lemma 9 we conclude the following theorem.

Theorem 10: If n, k are integers, $\gcd(n,6) = 1$, and $k = \frac{n-1}{2}$, then $\mathcal{C}(\alpha, \alpha^{2^k+1}, \alpha^{2^{3k}+1})$ is a strong triple-error-correcting code.

We are now ready to show the triple-error-correcting WOM-code construction. Again, we use the WOM-codes $\mathcal{C}$, $\mathcal{C}_{\mathcal{D}}$, and assume that $\gcd(\lceil \log_2(n+1) \rceil, 6) = 1$ and α is a primitive element in $GF(2^{\lceil \log_2(n+1) \rceil})$. The strong triple-error-correcting code is denoted by $\mathcal{C}_3^{\text{strong}}(\mathcal{E}_{\mathcal{C}_3^{\text{strong}}}, \mathcal{D}_{\mathcal{C}_3^{\text{strong}}})$. Its roots are $\alpha_1 = \alpha, \alpha_2 = \alpha^{2^k+1}, \alpha_3 = \alpha^{2^{3k}+1}$, where $k = \frac{\lceil \log_2(n+1) \rceil - 1}{2}$. There are $3r + t$ redundancy cells, divided into four groups:

1) The first t cells $\boldsymbol{q} = (q_0, \ldots, q_{t-1})$ are used with the n information cells to construct an $[n+t, k, t]$ single-error-detecting WOM-code $\mathcal{C}'(\mathcal{E}_{\mathcal{C}'}, \mathcal{D}_{\mathcal{C}'})$.
2) The other three groups $\boldsymbol{p}_1 = (p_0, \ldots, p_{r-1})$, $\boldsymbol{p}_2 = (p_r, \ldots, p_{2r-1})$, and $\boldsymbol{p}_3 = (p_{2r}, \ldots, p_{3r-1})$ contain r cells each. The ith group, $i = 1, 2, 3$, stores the $\lceil \log_2(n+1) \rceil$-bit syndrome $\boldsymbol{s}_i$ which corresponds to the root α_i.

To conclude, we describe an $[n+t+3r, k, t]$ triple-error-correcting WOM-code, $\mathcal{C}_{\text{TEC}}(\mathcal{E}_{\mathcal{C}_{\text{TEC}}}, \mathcal{D}_{\mathcal{C}_{\text{TEC}}})$.

Encoding Map $\mathcal{E}_{\mathcal{C}_{\text{TEC}}}$: The input is the memory-state vector $(\boldsymbol{c}, \boldsymbol{q}, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ and the new k-bit data vector $\boldsymbol{v}$. The output is either a new memory-state vector $(\boldsymbol{c}', \boldsymbol{q}', \boldsymbol{p}_1', \boldsymbol{p}_2', \boldsymbol{p}_3')$ or the erasure symbol E.

1. $(\boldsymbol{c}', \boldsymbol{q}') = \mathcal{E}_{\mathcal{C}'}((\boldsymbol{c}, \boldsymbol{q}), \boldsymbol{v})$;
2. if $((\boldsymbol{c}', \boldsymbol{q}') == \mathsf{E})$ return E;
3. $\boldsymbol{s}_1 = \sum_{i=0}^{n-1} c_i' \alpha_1^i$; $\boldsymbol{s}_2 = \sum_{i=0}^{n-1} c_i' \alpha_2^i$; $\boldsymbol{s}_3 = \sum_{i=0}^{n-1} c_i' \alpha_3^i$;
4. $\boldsymbol{p}_1' = \mathcal{E}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_1, \boldsymbol{s}_1)$; $\boldsymbol{p}_2' = \mathcal{E}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_2, \boldsymbol{s}_2)$;
   $\boldsymbol{p}_3' = \mathcal{E}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_3, \boldsymbol{s}_3)$;
5. if $((\boldsymbol{p}_1' == \mathsf{E})\mathtt{OR}(\boldsymbol{p}_2' == \mathsf{E})\mathtt{OR}(\boldsymbol{p}_3' == \mathsf{E}))$
   return E;
6. return$(\boldsymbol{c}', \boldsymbol{q}', \boldsymbol{p}_1', \boldsymbol{p}_2', \boldsymbol{p}_3')$;

The new k-bit data vector $\boldsymbol{v}$ is encoded in the information cells $\boldsymbol{c}$ and the first group of the redundancy cells $\boldsymbol{q}$ using the encoding map $\mathcal{E}_{\mathcal{C}'}$ (line 1). If this writing does not succeed the symbol E is returned (line 2). Otherwise, the three syndromes $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3$ are calculated from the n information cells (line 3) and are encoded in the last three groups of redundancy cells (line 4) while checking their writing success (line 5). If the last three writing operations succeed, the encoding map returns the new memory-state vector (line 6).

In the decoding map, $\mathcal{D}_{\mathcal{C}_{\text{TEC}}}$, we use the decoding map of the double-error-correcting WOM-code $\mathcal{D}_{\mathcal{C}_{\text{DEC}}}$. Note that in the decoding map $\mathcal{D}_{\mathcal{C}_{\text{DEC}}}$, instead of using a double-error-correcting BCH code, we can use any other cyclic double-error-correcting code which is given by its two roots. Line 9 in the decoding map $\mathcal{D}_{\mathcal{C}_{\text{DEC}}}$ is modified by substituting the decoding map of the new cyclic double-error-correcting code. The input to the modified decoding map $\mathcal{D}'_{\mathcal{C}_{\text{DEC}}}$ is the two roots of the cyclic double-error-correcting code, the n information cells, and the $2r$ redundancy cells corresponding to the two syndromes of the two roots.

Decoding Map $\mathcal{D}_{\mathcal{C}_{\text{TEC}}}$: The input is the memory-state vector $(\boldsymbol{c}', \boldsymbol{q}', \boldsymbol{p}_1', \boldsymbol{p}_2', \boldsymbol{p}_3')$. The output is the decoded data vector $\boldsymbol{v}$.

1. $\boldsymbol{s}_1'' = \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_1')$; $\boldsymbol{s}_2'' = \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_2')$; $\boldsymbol{s}_3'' = \mathcal{D}_{\mathcal{C}_{\mathcal{D}}}(\boldsymbol{p}_3')$;
2. if $(\boldsymbol{s}_1'' == \mathsf{F})$
3. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\text{DEC}}}(\alpha_2, \alpha_3, \boldsymbol{c}', \boldsymbol{p}_2', \boldsymbol{p}_3')$; return $\boldsymbol{v}$;$\}$
4. if $(\boldsymbol{s}_2'' == \mathsf{F})$
5. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\text{DEC}}}(\alpha_1, \alpha_3, \boldsymbol{c}', \boldsymbol{p}_1', \boldsymbol{p}_3')$; return $\boldsymbol{v}$;$\}$
6. if $(\boldsymbol{s}_3'' == \mathsf{F})$
7. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\text{DEC}}}(\alpha_1, \alpha_2, \boldsymbol{c}', \boldsymbol{p}_1', \boldsymbol{p}_2')$; return $\boldsymbol{v}$;$\}$
8. $\boldsymbol{s}_1' = \sum_{i=0}^{n-1} c_i' \alpha_1^i$; $\boldsymbol{s}_2' = \sum_{i=0}^{n-1} c_i' \alpha_2^i$; $\boldsymbol{s}_3' = \sum_{i=0}^{n-1} c_i' \alpha_3^i$;
9. $\boldsymbol{e}_1 = \boldsymbol{s}_1' + \boldsymbol{s}_1''$; $\boldsymbol{e}_2 = \boldsymbol{s}_2' + \boldsymbol{s}_2''$; $\boldsymbol{e}_3 = \boldsymbol{s}_3' + \boldsymbol{s}_3''$;
10. if $((\sum_{i=0}^{n-1} c_i') == (\sum_{i=0}^{t-1} q_i'))$
11. $\{\boldsymbol{v} = \mathcal{D}'_{\mathcal{C}_{\text{DEC}}}(\alpha_1, \alpha_2, \boldsymbol{c}', \boldsymbol{p}_1', \boldsymbol{p}_2')$; return $\boldsymbol{v}$;$\}$
12. if $((\boldsymbol{e}_1^{2^k+1} == \boldsymbol{e}_2)\mathtt{OR}(\boldsymbol{e}_1^{2^{3k}+1} == \boldsymbol{e}_3)$
    $\mathtt{OR}(\boldsymbol{e}_2^{2^{2k}-2^k+1} == \boldsymbol{e}_3))$
13. $\{\boldsymbol{v} = \mathtt{maj}(\mathcal{D}'_{\mathcal{C}_{\text{SEC}}}(\alpha_1, \boldsymbol{c}', \boldsymbol{p}_1'), \mathcal{D}'_{\mathcal{C}_{\text{SEC}}}(\alpha_2, \boldsymbol{c}', \boldsymbol{p}_2'),$
    $\mathcal{D}'_{\mathcal{C}_{\text{SEC}}}(\alpha_3, \boldsymbol{c}', \boldsymbol{p}_3'))$; return $\boldsymbol{v}$;$\}$
14. $\boldsymbol{e}' = \mathcal{D}_{\mathcal{C}_3^{\text{strong}}}(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$;
15. $\boldsymbol{v} = \mathcal{D}_{\mathcal{C}}(\boldsymbol{c}' + \boldsymbol{e}')$;
16. return $\boldsymbol{v}$;

First, the three syndromes from the last three redundancy cell groups are decoded (line 1). If the decoded syndrome $\boldsymbol{s}_1''$ is the error flag F (line 2), then there is at least one error in the group $\boldsymbol{p}_1'$. In the information cells $\boldsymbol{c}'$ and redundancy cells $\boldsymbol{p}_2', \boldsymbol{p}_3'$ there are at most two errors. Therefore, we decode by applying the decoding map $\mathcal{D}'_{\mathcal{C}_{\text{DEC}}}$ to $\boldsymbol{c}'$ and $\boldsymbol{p}_2', \boldsymbol{p}_3'$ with the roots α_1, α_3 (line 3). The same procedure is applied if $\boldsymbol{s}_2''$ or $\boldsymbol{s}_3''$ is the error flag F (lines 4–7). Here, we use the property of $\mathcal{C}_3^{\text{strong}}$ that every two out of its three roots generate a cyclic double-error-correcting code.

After line 7, none of the syndromes $\boldsymbol{s}_1'', \boldsymbol{s}_2'', \boldsymbol{s}_3''$ is the error flag F. Therefore, if there are errors in these redundancy cells then the number of errors in each of the three redundancy cell groups is even and since there are at most three errors, at most one group has exactly two errors. The received syndromes $\boldsymbol{s}_1', \boldsymbol{s}_2', \boldsymbol{s}_3'$ from

the received cells and the differences $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$ are calculated (lines 8 and 9). If the condition in line 10 holds, then the cells $\boldsymbol{c}'$ and $\boldsymbol{q}'$ have zero or two errors. In both cases, the cells $\boldsymbol{c}'$, $\boldsymbol{p}'_1$, $\boldsymbol{p}'_2$ have at most two errors so it is possible to decode (line 11).

We are left with the case where the parities of the cells $\boldsymbol{c}'$ and $\boldsymbol{q}'$ are not the same. That is, these cells have either one or three errors. We address this case in the next lemma.

Lemma 11: The condition in line 12 holds if and only if there is at most a single error in the information cells $\boldsymbol{c}'$.

Proof: If there is at most a single error in the information cells $\boldsymbol{c}'$ then at most one of the redundancy cell groups $\boldsymbol{p}'_1, \boldsymbol{p}'_2, \boldsymbol{p}'_3$ has two errors; that is, at least two of these groups do not have errors. If there is no error in the first and second groups and the ith information cell c'_i is in error, then $\boldsymbol{e}_1 = \alpha_1^i = \alpha^i$ and $\boldsymbol{e}_2 = \alpha_2^i = \alpha^{i(2^k+1)}$. Therefore, $\boldsymbol{e}_1^{2^k+1} = \boldsymbol{e}_2$. This condition clearly holds also if there are no errors in the information cells $\boldsymbol{c}'$. Similarly, if there is no error in $\boldsymbol{p}'_1$ and $\boldsymbol{p}'_3$ then $\boldsymbol{e}_1^{2^{3k}+1} = \boldsymbol{e}_3$, and if there is no error in $\boldsymbol{p}'_2$ and $\boldsymbol{p}'_3$ then $\boldsymbol{e}_2^{2^{2k}-2^k+1} = \boldsymbol{e}_3$. Therefore, if there is at most a single error in the information cells $\boldsymbol{c}'$ then the condition in line 12 holds.

Now assume that there is more than one error in the information cells $\boldsymbol{c}'$. That is, the information cells have two or three errors and in this case, there is no error in the redundancy cells $\boldsymbol{p}'_1, \boldsymbol{p}'_2, \boldsymbol{p}'_3$. Assume that the information cells have three errors in locations i, j, ℓ. Then

$$\begin{aligned} e_1 &= \alpha^i + \alpha^j + \alpha^\ell \\ e_2 &= \alpha^{i(2^k+1)} + \alpha^{j(2^k+1)} + \alpha^{\ell(2^k+1)} \\ e_3 &= \alpha^{i(2^{3k}+1)} + \alpha^{j(2^{3k}+1)} + \alpha^{\ell(2^{3k}+1)} \end{aligned}$$

for some $0 \leqslant i < j < \ell \leqslant n-1$. In this case, $\boldsymbol{e}_1^{2^k+1} \neq \boldsymbol{e}_2$. Otherwise, we get

$$\begin{aligned} &e_1 + \alpha^i + \alpha^j + \alpha^\ell = 0 \\ &e_1^{(2^k+1)} + \alpha^{i(2^k+1)} + \alpha^{j(2^k+1)} + \alpha^{\ell(2^k+1)} = 0 \end{aligned}$$

and $\mathcal{C}(\alpha, \alpha^{2^k+1})$ has a codeword of weight at most four, which is a contradiction. Similarly, $\boldsymbol{e}_1^{2^{3k}+1} \neq \boldsymbol{e}_3$ and $\boldsymbol{e}_2^{2^{2k}-2^k+1} \neq \boldsymbol{e}_3$. The case of two errors in the information cells is handled similarly. Hence, the condition in line 12 does not hold. ■

According to Lemma 11, if the condition in line 12 holds, then there is at most a single error in the information cells $\boldsymbol{c}'$. At most one of the redundancy cell groups $\boldsymbol{p}'_1, \boldsymbol{p}'_2, \boldsymbol{p}'_3$ has errors. Therefore, at least two out of the three decoding maps in line 13 succeed, and the function `maj`, which outputs the majority of the three decoded values, returns the correct value of $\boldsymbol{v}$. In line 14, there are at most three errors in the information cells and no errors in the redundancy cell groups $\boldsymbol{p}'_1, \boldsymbol{p}'_2, \boldsymbol{p}'_3$, so it is possible to find the error vector (line 14) and decode (line 15). We conclude with the following theorem.

Theorem 12: If $\mathcal{C}$ is an $[n, k, t]$ WOM-code, $\mathcal{C}_{\mathcal{D}}$ is an $[r, \lceil \log_2(n+1) \rceil, t]$ single-error-detecting WOM-code, and $\gcd(\lceil \log_2(n+1) \rceil, 6) = 1$, then $\mathcal{C}_{\mathrm{TEC}}$ is an $[n + 3r + t, k, t]$ triple-error-correcting WOM-code.

VII. Multiple Error-Correcting WOM-Codes

In this section, we study how to correct an arbitrary number of errors with a WOM-code. As described in the Introduction, a simple scheme to construct an e-error-correcting WOM-code is done by using an existing WOM-code and replicating each one of its cells $2e+1$ times. A first improvement upon this scheme can be achieved by replicating each cell only $e+1$ times. Then, instead of using a regular WOM-code, a single-error-detecting WOM-code is applied. In the decoding procedure, the value of each cell is the majority value among its replicas that are not detected to be in error. In the rest of the section we will show how to use similar ideas in order to construct better WOM-codes that correct any specified number of errors.

Let us first show another property of the triple-error-correcting WOM-code studied in Section VI.

Lemma 13: Let $\mathcal{C}_{\mathrm{TEC}}$ be an $[n+3r+t, k, t]$ triple-error-correcting WOM-code constructed in Theorem 12. Then the code $\mathcal{C}_{\mathrm{TEC}}$ can correct four erasures.

Proof: Assume first that there are no erasures in the redundancy cell groups $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$, so we know the correct values of the syndromes $\boldsymbol{s}_1$, $\boldsymbol{s}_2$, and $\boldsymbol{s}_3$. Assume also that there are at most four erasures in the information cells $\boldsymbol{c}$. Since the code $\mathcal{C}_3^{\mathrm{strong}}$ corrects three errors, its minimum distance is at least seven and hence it can correct up to six erasures and, a fortiori, four erasures.

If each redundancy cell group $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ has at most one error, then it is still possible to successfully decode the three syndromes since each syndrome is stored using a single-error-detecting WOM-code, and we can then decode the erased information cells as in the first case.

If one of the three redundancy groups has at least two erasures, then the n information cells and two other redundancy groups contain at most two erasures. Therefore, it is again possible to successfully decode the erasure values. ■

The next theorem confirms the validity of the first construction for an e-error-correcting WOM-code.

Theorem 14: Let $\mathcal{C}_{\mathrm{TEC}}$ be an $[n, k, t]$ triple-error-correcting WOM-code. Then there exists an $[\lceil e/2 \rceil n, k, t]$ e-error-correcting WOM-code.

Proof: Let us denote the cells of the WOM-code $\mathcal{C}_{\mathrm{TEC}}$ by $c'_0, \ldots, c'_{n-1}$. The constructed e-error-correcting WOM-code is denoted by $\mathcal{C}_{\mathrm{eEC}}$ and its $\lceil e/2 \rceil n$ cells are denoted by $c_{0,0}, \ldots, c_{0,\lceil e/2 \rceil - 1}, \ldots, c_{n-1,0}, \ldots, c_{n-1,\lceil e/2 \rceil - 1}$. We use two transformations in the validation of the construction. The first transformation

$$f : \{0,1\}^{\lceil e/2 \rceil n} \to \{0,1,?\}^n$$

transforms a memory-state vector of $\lceil e/2 \rceil n$ cells

$$\boldsymbol{c} = (c_{0,0}, \ldots, c_{0,\lceil e/2 \rceil - 1}, \ldots, c_{n-1,0}, \ldots, c_{n-1,\lceil e/2 \rceil - 1})$$

into a memory-state vector of n cells

$$\boldsymbol{c}' = (c'_0, \ldots, c'_{n-1})$$

by taking the majority of every group of $\lceil e/2 \rceil$ cells. That is, for all $0 \leqslant i \leqslant n-1$

$$c'_i = \mathrm{maj}\{c_{i,0}, \ldots, c_{i,\lceil e/2 \rceil - 1}\}.$$

If the number of ones and number of zeros are equal, then $c'_i = ?$, the erasure symbol. The second transformation

$$g : \{0,1\}^n \to \{0,1\}^{\lceil e/2 \rceil n}$$

transforms a memory-state vector of n cells

$$\boldsymbol{c}' = (c'_0, \ldots, c'_{n-1})$$

to a memory-state vector of $\lceil e/2 \rceil n$ cells

$$\boldsymbol{c} = (c_{0,0}, \ldots, c_{0,\lceil e/2 \rceil - 1}, \ldots, c_{n-1,0}, \ldots, c_{n-1,\lceil e/2 \rceil - 1})$$

such that for all $0 \leqslant i \leqslant n-1$ and $0 \leqslant j \leqslant \lceil e/2 \rceil - 1$

$$c_{i,j} = c'_i.$$

That is, every cell is replicated $\lceil e/2 \rceil$ times.

In the encoding map $\mathcal{E}_{\mathcal{C}_{\mathrm{eEC}}}$, the new vector data $\boldsymbol{v}$ and memory-state vector $\boldsymbol{c}$ of $\lceil e/2 \rceil n$ cells are received. Then, the new memory-state vector is updated according to

$$g(\mathcal{E}_{\mathcal{C}_{\mathrm{TEC}}}(f(\boldsymbol{c}), \boldsymbol{v})).$$

First, a memory-state vector of n cells is generated by the transformation f on the memory-state vector of the $\lceil e/2 \rceil n$ cells, $\boldsymbol{c}$. Then, the encoding map $\mathcal{E}_{\mathcal{C}_{\mathrm{TEC}}}$ is invoked on the memory-state vector $f(\boldsymbol{c})$ and data vector $\boldsymbol{v}$. Finally, the new memory-state vector of n cells is transformed back to $\lceil e/2 \rceil n$ cells to generate the new memory-state vector.

In the decoding map $\mathcal{E}_{\mathcal{D}_{\mathrm{eEC}}}$, the memory-state vector $\boldsymbol{c}$ of $\lceil e/2 \rceil n$ cells is the input and is decoded according to

$$\mathcal{E}_{\mathcal{D}_{\mathrm{TEC}}}(f(\boldsymbol{c})).$$

As in the encoding map, first a memory-state vector of n cells is generated from the memory-state vector of $\lceil e/2 \rceil n$ cells and is the input to the decoding map of the WOM-code $\mathcal{E}_{\mathcal{D}_{\mathrm{TEC}}}$. The output data vector $\boldsymbol{v}$ from $\mathcal{E}_{\mathcal{D}_{\mathrm{TEC}}}$ is the output data vector of the decoding map.

If there are at most e errors in $\boldsymbol{c}$ then in the memory-state vector $f(\boldsymbol{c})$ there are at most three errors and erasures or exactly four erasures. Since $\mathcal{C}_{\mathrm{TEC}}$ is a triple-error-correcting WOM-code it can correct three errors and erasures and according to Lemma 13 it can correct four erasures as well. ■

The next example demonstrates how to use the previous construction in order to construct a four-error-correcting WOM-code.

Example 3: Let us illustrate how to construct a two-write four-error-correcting WOM-code. We start with the $[3, 2, 2]$ WOM-code by Rivest and Shamir [16]. Repeating this WOM-code ten times gives us a $[30, 20, 2]$ WOM-code. In order to apply Theorem 12, we use a $[10, 5, 2]$ single-error-detecting WOM-code obtained by repeating the $[3, 2, 2]$ WOM-code twice, appending two cells to store a single bit twice, and then appending two more cells for the error-detection. Thus, we get a $[62, 20, 2]$ triple-error-correcting WOM-code. Then, according to Theorem 14 there exists a $[124, 20, 2]$ four-error-correcting WOM-code. In order to use the simple construction we introduced at the beginning of this section, one needs to replicate five times a single-error-detecting WOM-code of the $[30, 20, 2]$ WOM-code. Thus the number of cells is at least $30 \cdot 5 = 150$.

Roth [17] suggested another, recursive construction of multiple-error-correcting WOM-codes, based upon the following approach. Assume that $\mathcal{C}$ is an $[n, k, t]$ WOM-code, and assume that there exists a linear e-error-correcting code of length n and redundancy r. Then, a syndrome $\boldsymbol{s}$ of r bits is recursively stored using another e-error-correcting WOM-code. This process can be recursively repeated multiple times until we use an e-error-correcting WOM-code which can be constructed according to Theorem 14. We validate the recursive step of this construction in the next theorem and then show an example of how to use the construction.

Theorem 15: Let $\mathcal{C}_1$ be an $[n, k, t]$ WOM-code, $\mathcal{C}_2$ be a linear e-error-correcting-code of length n and redundancy r, and $\mathcal{C}_3$ be an $[m, r, t]$ e-error-correcting WOM-code. Then there exists an $[n+m, k, t]$ e-error-correcting WOM-code.

Proof: The e-error-correcting WOM-code we construct has $n+m$ cells which are partitioned into two groups. The first group has n cells and is denoted by $\boldsymbol{c} = (c_1, \ldots, c_n)$. The second group consists of m cells and is denoted by $\boldsymbol{p} = (p_1, \ldots, p_m)$.

In the encoding map the memory-state vector of $n+m$ cells, $(\boldsymbol{c}, \boldsymbol{p})$ and new data vector $\boldsymbol{v}$ are received. The output is a new memory-state vector $(\boldsymbol{c}', \boldsymbol{p}')$. The data vector $\boldsymbol{v}$ is stored in the first n cells using the encoding map of the WOM-code $\mathcal{C}_1$

$$\boldsymbol{c}' = \mathcal{E}_{\mathcal{C}_1}(\boldsymbol{c}, \boldsymbol{v}).$$

Let H be the parity check matrix of the linear e-error-correcting code $\mathcal{C}_2$. In the next step a syndrome $\boldsymbol{s}$ of r bits is calculated using the new value of the n bits

$$\boldsymbol{s} = H \cdot \boldsymbol{c}'.$$

Then, the syndrome $\boldsymbol{s}$ is stored in the m cells using the encoding map of the WOM-code $\mathcal{C}_3$

$$\boldsymbol{p}' = \mathcal{E}_{\mathcal{C}_3}(\boldsymbol{p}, \boldsymbol{s}).$$

In the decoding map, the memory-state vector $(\hat{\boldsymbol{c}}, \hat{\boldsymbol{p}})$ is the input and the output is the data vector $\boldsymbol{v}$ of k bits. We assume that $(\boldsymbol{c}', \boldsymbol{p}')$, defined above, is the stored memory-state vector and $\boldsymbol{s} = H \cdot \boldsymbol{c}'$. First, the syndrome of r bits is decoded by applying the decoding map of the e-error-correcting WOM-code $\mathcal{C}_3$

$$\boldsymbol{s}'' = \mathcal{D}_{\mathcal{C}_3}(\hat{\boldsymbol{p}}).$$

The success of this decoding map is guaranteed since there are at most e errors in $\boldsymbol{p}'$ and the WOM-code $\mathcal{C}_3$ can correct e errors, that is, $\boldsymbol{s}'' = \boldsymbol{s}$. Another syndrome is calculated from the n cells and the parity check matrix H

$$\hat{\boldsymbol{s}} = H \cdot \hat{\boldsymbol{c}}.$$

Let $\boldsymbol{e}$ be the error-vector in the first group of the memory-state vector whose weight is at most e, i.e., $\hat{\boldsymbol{c}} = \boldsymbol{c}' + \boldsymbol{e}$. Then

$$H \cdot \boldsymbol{e} = H \cdot (\boldsymbol{c}' + \hat{\boldsymbol{c}}) = H \cdot \boldsymbol{c}' + H \cdot \hat{\boldsymbol{c}} = \boldsymbol{s} + \hat{\boldsymbol{s}}.$$

Therefore, the syndrome that corresponds to the error vector $\boldsymbol{e}$ is $\boldsymbol{s} + \hat{\boldsymbol{s}}$ and it is possible to find it by applying the decoding map of the code $\mathcal{C}_2$ to $\boldsymbol{s} + \hat{\boldsymbol{s}}$

$$\boldsymbol{e} = \mathcal{D}_{\mathcal{C}_2}(\boldsymbol{s} + \hat{\boldsymbol{s}}).$$

Finally, the data vector $\boldsymbol{v}$ is decoded by applying the decoding map of the WOM-code $\mathcal{C}_1$ to the memory-state vector $\hat{\boldsymbol{c}} + \boldsymbol{e}$

$$\boldsymbol{v} = \mathcal{D}_{\mathcal{C}_1}(\hat{\boldsymbol{c}} + \boldsymbol{e}).$$

■

A necessary condition to efficiently apply this scheme recursively is that r, the number of redundancy bits of the e-error-correcting code, not be greater than the number of information bits k. Otherwise, the number of cells in the next step of the recursion would be greater than the total number of cells n. The code constructed in Example 3 cannot be used for just this reason. If we start with the $[2^k - 1, k, 5 \cdot 2^{k-4} + 1]$ WOM-code for $k \geqslant 4$, $\gcd(k, 6) = 1$, and then use a four-error-correcting-code, the number of redundancy bits is roughly $4k$ and so the number of information bits for the next WOM-code in the recursion is greater than the number of the information bits that the WOM-code needs to store. The next example shows another case where this scheme can outperform the construction in Theorem 14.

Example 4: In this example, we start with the $[23, 11, 3]$ WOM-code constructed by Cohen *et al.* [2]. In order to use this WOM-code in a larger block of cells, one can simply repeat the WOM-code in successive groups of 23 cells. For example, repeating the code 89 times provides us with a $[2047, 979, 3]$ WOM-code. In order to construct a four-error-correcting WOM-code according to the construction in Theorem 14, it is necessary to first build a triple-error-correcting WOM-code. In this case $n = 2047$, $\lceil \log(n+1) \rceil = 11$, and we will construct a single-error-detecting WOM-code that stores 11 bits three times. This can be done according to Section III and the $[23, 11, 3]$ WOM-code, so we receive a $[26, 11, 3]$ single-error-detecting WOM-code. The condition of Theorem 12 holds, i.e., $\gcd(11, 6) = 1$, and thus we can construct a $[2047 + 3 \cdot 26 + 3 = 2128, 979, 3]$ triple-error-correcting WOM-code. Finally, by applying Theorem 14, we can construct a $[4256, 979, 3]$ four-error-correcting WOM-code.

Next, we construct the code according to Theorem 15. Again, let us start with the $[2047, 979, 3]$ WOM-code and use a four-error-correcting code of length 2047. Specifically, we use a four-error-correcting BCH code of $4 \cdot 11 = 44$ redundancy bits, so we need to store 44 bits three times while correcting four errors. In order to apply Theorem 14, we need to first construct a triple-error-correcting WOM-code which stores 44 bits three times. Note that now $n = 92$ and $\lceil \log(n+1) \rceil = 7$, so a single-error-detecting code that stores seven bits three times is required. Cohen *et al.* [2] also constructed a $[7, 3, 3]$ WOM-code and, therefore, there exists a $[14, 6, 3]$ WOM-code. By simply adding three more cells to store one more bit three times we construct a $[17, 7, 3]$ WOM-code. The latter WOM-code provides us with a $[20, 7, 3]$ single-error-detecting WOM-code. Since $\gcd(7, 6) = 1$, Theorem 12 again applies, and we can construct a $[92 + 3 \cdot 20 + 3 = 155, 44, 3]$ triple-error-correcting WOM-code. Next, by applying Theorem 14, we can construct a $[2 \cdot 155 = 310, 44, 3]$ four-error-correcting WOM-code. Finally, we get a $[2047 + 310 = 2357, 979, 3]$ four-error-correcting WOM-code, thereby improving upon the first construction.

VIII. Summary and Conclusions

In this paper, we constructed error-correcting WOM-codes. All the proposed constructions had the same structure in the sense that we started with an existing $[n, k, t]$ WOM-code and then added more redundancy cells that enabled the WOM-code to detect or correct errors. We started with a construction of a single-error-detecting WOM-code. Then, we showed how to use this construction along with Hamming codes in order to construct single-error-correcting WOM-codes. Building upon this, we constructed double-error-correcting WOM-codes when $\lceil \log(n+1) \rceil$ is an odd integer and when $\lceil \log(n+2) \rceil$ is an even integer.

We proceeded to construct triple-error-correcting WOM-codes. Here, we introduced the notion of strong cyclic error-correcting codes, for which the e roots of the generator polynomial have the property that any subset of k distinct roots generate a k-error-correcting code. We showed how to find strong triple-error correcting codes and used them in the construction of triple-error-correcting WOM-codes. The triple-error-correcting WOM-codes were used in one construction, which then formed the basis of a recursive construction that sometimes yields better multiple-error-correcting WOM-codes.

Acknowledgment

The authors thank the anonymous reviewer for valuable comments and suggestions.

References

[1] C. Bracken and T. Helleseth, "Triple-error-correcting BCH-like codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Seoul, Korea, Jun. 2009, pp. 1723–1725.

[2] G. D. Cohen, P. Godlewski, and F. Merkx, "Linear binary code for write-once memories," *IEEE Trans. Inf. Theory*, vol. 32, no. 5, pp. 697–700, Sep. 1986.

[3] H. Dobbertin, "Almost perfect nonlinear power functions on $\mathrm{GF}(2^n)$: The Welch case," *IEEE Trans. Inf. Theory*, vol. 45, no. 4, pp. 1271–1275, May 1999.

[4] A. Fiat and A. Shamir, "Generalized write-once memories," *IEEE Trans. Inf. Theory*, vol. 30, pp. 470–480, Sep. 1984.

[5] F. Fu and A. J. H. Vinck, "On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 308–313, Sep. 1999.

[6] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Comput. Surveys*, vol. 37, pp. 138–163, Jun. 2005.

[7] P. Godlewski, "WOM-codes construits à partir des codes de Hamming," *Discrete Math.*, no. 65, pp. 237–243, 1987.

[8] C. Heegard, "On the capacity of permanent memory," *IEEE Trans. Inf. Theory*, vol. 31, no. 1, pp. 34–42, Jan. 1985.

[9] A. Jiang, "On the generalization of error-correcting WOM codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 1391–1395.

[10] A. Jiang, V. Bohossian, and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 1166–1170.
[11] A. Jiang and J. Bruck, "Joint coding for flash memory storage," in *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, Canada, Jul. 2008, pp. 1741–1745.
[12] T. Kasami, "The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes," *Inf. Contr.*, vol. 18, pp. 369–394, Sep. 1971.
[13] S. Kayser, E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple-write WOM-codes," in *Proc. 48th Ann. Allerton Conf. Commun., Contr. Comput.*, Monticello, IL, Sep. 2010.
[14] H. Mahdavifar, P. H. Siegel, A. Vardy, J. K. Wolf, and E. Yaakobi, "A nearly optimal construction of flash codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Seoul, Korea, Jul. 2009, pp. 1239–1243.
[15] F. Merkx, "Womcodes constructed with projective geometries," *Traitement Du Signal*, vol. 1, no. 2–2, pp. 227–231, 1984.
[16] R. L. Rivest and A. Shamir, "How to reuse a write-once memory," *Inf. Contr.*, vol. 55, no. 1–3, pp. 1–19, Dec. 1982.
[17] R. M. Roth, Pers. Commun. 2010.
[18] J. K. Wolf, A. D. Wyner, J. Ziv, and J. Korner, "Coding for a write-once memory," *AT&T Bell Labs. Tech. J.*, vol. 63, no. 6, pp. 1089–1112, 1984.
[19] Y. Wu, "Low complexity codes for writing write-once memory twice," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, Jun. 2010, pp. 1928–1932.
[20] Y. Wu and A. Jiang, "Position modulation code for rewriting write-once memories," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3692–3697, Jun. 2011.
[21] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," in *Proc. IEEE Inf. Theory Workshop*, Dublin, Ireland, Aug. 2010.
[22] E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple error-correcting WOM-codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, Jun. 2010, pp. 1933–1937.
[23] G. Zémor and G. D. Cohen, "Error-correcting WOM-codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 730–734, May 1991.
[24] G. Zémor, "Problèmes Combinatoires Liés à L'écriture sur des Mémoires," Ph.D., ENST, Paris, France, 1989.

Eitan Yaakobi (S'07) received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011.

He is currently a Joint Postdoctorate Researcher in electrical engineering at the California Institute of Technology, Pasadena, and with the University of California, San Diego, where he is associated with the Center for Magnetic Recording Research. His research interests include coding theory, algebraic error-correction coding, and their applications for digital data storage and, in particular, for nonvolatile memories.

Paul H. Siegel (M'82–SM'90–F'97) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology (MIT), Cambridge, in 1975 and 1979, respectively.

He held a Chaim Weizmann Postdoctoral Fellowship at the Courant Institute, New York University, New York. He was with the IBM Research Division, San Jose, CA, from 1980 to 1995. He joined the faculty of the School of Engineering, University of California, San Diego, in July 1995, where he is currently a Professor of Electrical and Computer Engineering. He is affiliated with the California Institute of Telecommunications and Information Technology, the Center for Wireless Communications, and the Center for Magnetic Recording Research, where he holds an endowed chair and served as Director from 2000 to 2011. His primary research interests lie in the areas of information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission. He holds several patents in the area of coding and detection.

Prof. Siegel was a member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and was re-elected for a three-year term in 2009. He served as Co-Guest Editor of the May 1991 Special Issue on "Coding for Storage Devices" of the IEEE TRANSACTIONS ON INFORMATION THEORY. He served the same TRANSACTIONS as Associate Editor for Coding Techniques from 1992 to 1995, and as Editor-in-Chief from July 2001 to July 2004. He was also Co-Guest Editor of the May/September 2001 two-part issue on "The Turbo Principle: From Theory to Practice" of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was a corecipient, with R. Karabed, of the 1992 IEEE Information Theory Society Paper Award and shared the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award with B. Marcus and J. K. Wolf. With J. B. Soriaga and H. D. Pfister, he received the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He was named a Master Inventor at IBM Research in 1994. He is a member of Phi Beta Kappa and the National Academy of Engineering.

Alexander Vardy (S'88–M'91–SM'94–F'98) was born in Moscow, U.S.S.R., in 1963. He received the B.Sc.(*summa cum laude*) degree from the Technion, Israel, in 1985, and the Ph.D. degree from the Tel-Aviv University, Israel, in 1991.

During 1985–1990, he was with the Israeli Air Force, where he worked on electronic counter measures systems and algorithms. During 1992 and 1993, he was a Visiting Scientist with the IBM Almaden Research Center, San Jose, CA. From 1993 to 1998, he was with the University of Illinois at Urbana-Champaign, first as an Assistant Professor then as an Associate Professor. He is currently a Professor with the Department of Electrical Engineering, the Department of Computer Science, and the Department of Mathematics, all at the University of California San Diego (UCSD). While on sabbatical from UCSD, he has held long-term visiting appointments with CNRS, France, the EPFL, Switzerland, and the Technion, Israel. His research interests include error-correcting codes, algebraic and iterative decoding algorithms, lattices and sphere packings, coding for digital media, cryptography and computational complexity theory, and fun math problems.

Dr. Vardy received an IBM Invention Achievement Award in 1993 and NSF Research Initiation and CAREER awards in 1994 and 1995, respectively. In 1996, he was appointed Fellow in the Center for Advanced Study at the University of Illinois, and received the Xerox Award for faculty research. In the same year, he became a Fellow of the Packard Foundation. He received the IEEE Information Theory Society Paper Award (jointly with Ralf Koetter) for the year 2004. In 2005, he received the Fulbright Senior Scholar Fellowship, and the Best Paper Award at the IEEE Symposium on Foundations of Computer Science (FOCS). During 1995–1998, he was an Associate Editor for Coding Theory and during 1998–2001, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. He was also an Editor for the SIAM *Journal on Discrete Mathematics*. He has been a member of the Board of Governors of the IEEE Information Theory Society from 1998 to 2006.

Jack Keil Wolf (S'54–M'60–F'73–LF'97) received the B.S.E.E. degree from the University of Pennsylvania, Philadelphia, and the M.S.E., M.A., and Ph.D. degrees from Princeton University, Princeton, NJ.

He is currently the Stephen O. Rice Professor of Magnetics in the Center of Magnetic Recording Research, University of California, San Diego, and a Vice President, Technology at Qualcomm, Incorporated. He is a member of the National Academy of Sciences, the National Academy of Engineering, and the American Academy of Arts and Sciences.

Dr. Wolf has been the recipient (or co-recipient) of the following IEEE awards: The 1990 E. H. Armstrong Achievement Award, the 1975 IEEE Information Theory Group Prize Paper Award, the 1992 Leonard G. Abraham Prize Paper Award, the 1998 Koji Kobayashi Award, the 2001 Information Theory Society Shannon Award, the 2004 Hamming Medal, and the 2007 Aaron D. Wyner Award.