Not Just for Errors: Codes for Fast and Secure Flash Storage

Yuval Cassuto
School of Computer & Communication Sciences - ALGO Laboratory
EPFL, Lausanne, Switzerland
yuval.cassuto@epfl.ch

***Abstract*— Error-correcting codes are normally employed in storage devices to guarantee the integrity of data in the presence of errors. This paper presents two schemes where error-correcting codes are used for entirely different purposes. In the first part of the paper, a new coding paradigm is proposed to improve the write performance of multi-level flash devices. By slightly relaxing the accuracy of cell programming, significant speed-up can be achieved. The resulting write inaccuracies are then corrected by codes that are tailored for the appropriately restricted error model. In the second part, new low-complexity codes are proposed to protect the security of sensitive data in the presence of imperfect physical erasure processes. Codes that have optimal encoding and decoding complexities are constructed to allow fast storing and retrieval of secret data, and guarantee unconditional security of data against an adversary with access to parts of the secret that failed to erase.**

I. Introduction

The impact of error-correcting codes on communication systems has transcended into the data storage domain, where coding methods are shaping the way storage devices are designed and operated. At the individual-device level, error-correcting codes allow dense storage of information on physical media that are impaired by a multitude of phenomena like noise, jitter, inter-bit interference and defects. At the storage-system level, codes are used to protect the survivability and availability of data against failures of individual devices. Per their title, error-correcting codes are primarily applied to combat errors that are incident on the data read from the storage device. In these applications, the impairments from the read and write processes are combined and modeled through a *storage channel*, for which codes with appropriate parameters are chosen. Code design for storage channels arguably has substantial similarities to code design for communication channels.

Beyond the parallels between data storage and communication, storage devices exhibit unique properties and requirements that provide opportunities for rich applications of combinatorial structures in general, and error-correcting codes in particular. In addition to reliability and data integrity, other qualities of storage devices can be the targets of new coding frameworks. Two such important qualities are the subjects of this paper: write performance of flash devices, and data security of devices with imperfect physical erasure. In Part 1 of the paper, accurate programming of cells is identified as a bottleneck for high write performance in multi-level flash devices (section II). A scheme is proposed to relax the programming accuracy and correct the residual inaccuracies by codes that are specialized for the resulting error model (section III). Code design for this restricted error model allows to keep the required code redundancy low, and at the same time offer significant speed-up of the flash programming sequence. The potential time savings of this scheme are quantitatively evaluated using an analytical model for the programming sequence (section IV). In Part 2 of the paper (sections V and VI), new codes are proposed to improve the security of storage devices. Storage devices often contain sensitive customer data, such as financial, medical and personal items. The user usually chooses to erase these data upon disposal or transfer of the device, frequently following government regulations to do so. In these cases, the erasure of data must be secure, and shall not leave behind any trace of the data. Unfortunately, perfect physical erasure of data is either not possible, or very costly. The challenge of removing the data from the device is both due the imperfections of the physical erasure processes and, at a higher level, due to address-translation layers that may make it non-trivial to track all traces of sensitive data for erasure. The scheme proposed here addresses the imperfect erasure problem by using error-correcting codes to encode the secret data in a way that will guarantee its confidentiality, even if an adversary has access to a large number of physical bits. The new codes belong to a class of codes called *wire-tap II codes* [5], and they provide unconditional, information-theoretic (in contrast to schemes that assume an adversary that is computationally limited) security, so long as the number of bits accessible by the adversary is not greater than the code's security parameter. The key novelty of the presented code constructions is that they offer optimal encoding and decoding complexities given the security that they provide. Low encoding/decoding complexities allow fast writing/reading of dynamic sensitive data, and may consequently improve the write/read performance of the device. The codes are provided as infinite families of codes, allowing a system designer to choose the code parameters based on the specific architecture and security requirements of the target device.

Part 1: Codes for Speed-up of Flash Programming

This paper surveys work that was partly done while the author was at the California Institute of Technology, and at Hitachi Global Storage Technologies. Research was supported in part by the ERC Advanced Researcher Grant of A. Shokrollahi entitled ECCSciEng.

II. The Challenge of Fast and Accurate Flash Programming

Flash storage devices are well known to exhibit a significant asymmetry between read speed (fast) and write speed (slower). The inferior write performance is attributed to the challenge of programming a flash cell to an accurate threshold level, as well as to the slow block erase operations. Programming flash cells is carried out by stochastic physical processes, hence accurate programming requires an iterative process of write-pulse applications and current measurements that comprise a program sequence approaching the desired target level. The sequence of write pulses applied to the cell during this iterative process needs to be carefully designed to avoid over-programming the cell beyond the target level. Such over-programming instance would result in a necessity for block erase, with the consequence of slowing down the write even further. Flash-programming optimization is studied in [4], and in many other published and unpublished works. A distinctive characteristic of a typical flash programming sequence is that most of the progress in changing the cell's threshold level is achieved by the few initial pulses, while many more pulses are needed to set the cell on an accurate final level. This property can be viewed graphically in Figure 1, showing an optimized floating-gate program sequence taken from [1]. Each of the curves in Figure 1 describes a program sequence with a different target level. Toward the end of each program curve, the vertical separation of adjacent sequence points becomes smaller, contributing to overall long program sequences that approach the target level asymptotically.

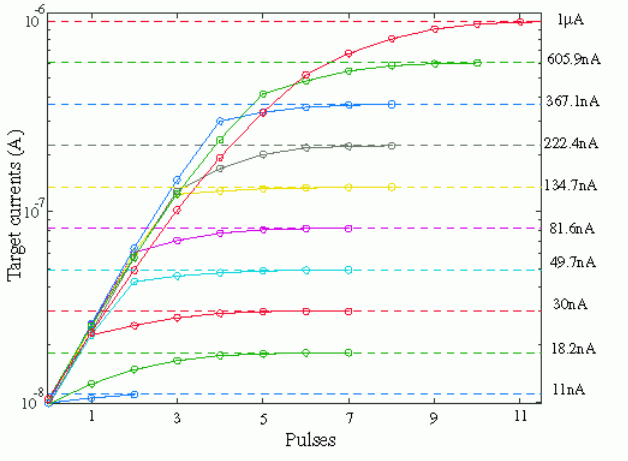

Figure 1. Performance of a Flash adaptive program sequence [1]. The circles on each curve describe the results of an iterative programming algorithm for a given target value.

If, as shown above, high-accuracy programming requires numerous program cycles, then a potential speed-up technique may be to alleviate the stringent accuracy requirement. By doing so, we would obviously introduce errors to the stored values, hence these errors need to be treated using an error-correcting code that is designed for the characteristics of the introduced errors. From the shape of the curves in Figure 1, it is clear that even if the program sequence is terminated very early, the resulting deficiency in the level of the programmed cell is relatively small. Therefore, an error-correcting code that corrects errors of *low magnitudes* may be sufficient for noticeable write speed-up. In addition, early termination of the program sequence results in values that are below the target value, thus the code may be designed for errors that are *asymmetric*. By the core wisdom of information theory, restricting the error model to asymmetric errors with limited magnitudes can provide codes that are more efficient, in both redundancy and complexity, than common codes for symmetric errors. The design and analysis of codes for this error model is the subject of the remainder of Part 1 of this paper.

III. Codes for Asymmetric Limited-Magnitude Errors

A. Error model and code parameters

Suppose we have a flash device with cells that are programmed to one of q levels. For a particular technology and device architecture, we choose a parameter ℓ that defines the maximal difference (in discrete levels) between the target level and the programmed level. So if the target level is $x \in \{0,\ldots,q-1\}$, the programming outcome can be one of the levels in $\{x-\ell, x-\ell+1,\ldots,x-1,x\}$ (if $x<\ell$ we consider only the non-negative elements in the set). The case with the program outcome equals the target x is the no-error case. In addition, we define a code block of n cells, out of which at most t cells have asymmetric errors of magnitude limit ℓ. A general coding scheme to allow programming inaccuracies in a variety of scenarios thus consists of four integer parameters:

- q the code alphabet size
- n the code block size
- ℓ the maximum tolerated error magnitude
- t the maximum number of symbols in a code block with non-zero error values

A coding-theoretic framework to address asymmetric limited-magnitude errors with the parameters above has been proposed in [3]. Next we provide a brief summary of the construction, encoding and decoding of such codes.

B. Construction, encoding and decoding

The most favorable property of the codes from [3] is that they are obtained by reducing the problem of asymmetric limited-magnitude error correction, to the well-studied problem of coding for symmetric errors (with different parameters). In Construction 1 below, a code $\mathcal{C}$ that corrects t asymmetric limited-magnitude errors is obtained by employing a code Σ, defined over a smaller alphabet, that corrects t symmetric errors.

Construction 1. *Let Σ be a code over the alphabet Q' of size $\ell+1$. The code $\mathcal{C}$ over the alphabet Q of size q ($q>\ell+1$) is defined as*

$$\mathcal{C} = \{\boldsymbol{x} = (x_1,\ldots,x_n) \in Q^n : \boldsymbol{x} \bmod (\ell+1) \in \Sigma\}.$$

In other words, the codewords of $\mathcal{C}$ are the subset of the length n vectors over Q that are mapped to codewords of Σ when their symbols are reduced modulo $\ell+1$.

The way codes for asymmetric limited-magnitude errors are constructed, encoded and decoded is best explained with an example.

Example 1. *Let Σ_H be the binary Hamming code of length $n = 2^m - 1$, for some integer m. First we define the code $\mathcal{C}_H$ in the way of Construction* 1.

$$\mathcal{C}_H = \{\boldsymbol{x} = (x_1, \ldots, x_n) \in Q^n : \boldsymbol{x} \bmod 2 \in \Sigma_H\}.$$

By the properties of Σ_H, the code $\mathcal{C}_H$ corrects a single ($t = 1$) asymmetric $\ell = 1$ limited-magnitude error. As for encoding, when the code alphabet size is $q = 2^b$, for some integer b, Figure 2 *below describes a simple encoding function from $nb - m$ information bits to codewords of $\mathcal{C}_H$ over Q. In Figure* 2 *(a), $nb - m$ information bits are input to the encoder. The encoder then uses a binary Hamming encoder to encode $n - m$ of the information bits into a length n Hamming codeword (Figure* 2 *(b)). Finally, in Figure* 2 *(c), each q-ary symbol of the codeword $\boldsymbol{x} \in \mathcal{C}_H$ is constructed from b bits using the usual binary-to-integer conversion, the top row being the least-significant bits of $x_i \in Q$.*

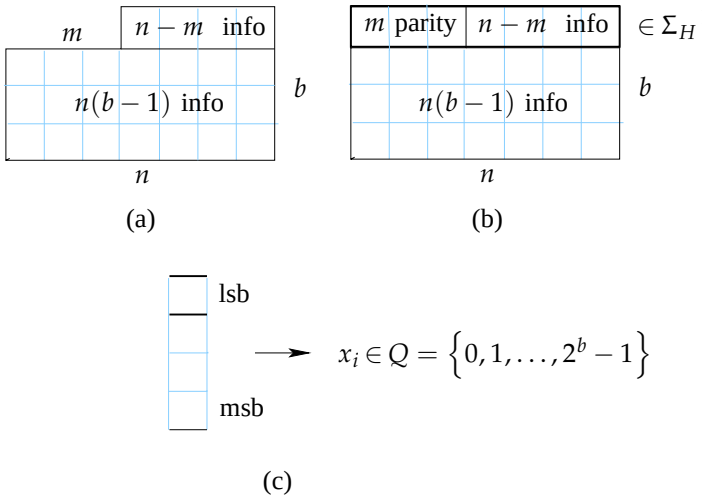

Figure 2. Encoding Procedure for $\mathcal{C}_H$

Decoding is carried out by using a Hamming decoder on the top row to find the limited-magnitude error location and magnitude (for binary Hamming codes the magnitude is always 1*). The top-row word is not corrected by the Hamming decoder, but rather the error magnitude is subtracted from the Q-ary word $\boldsymbol{y}$ to obtain a decoded codeword. To recover the information bits after decoding, the Q symbols are converted back to bits with the standard mapping, and the m parity bits are discarded.*

Despite its simplicity, Construction 1 was shown to provide the specified correction capability with small amounts of redundancy. In particular, the redundancy is smaller than any known code for related error models (including q-ary codes for symmetric errors), and the resulting code rate is asymptotically optimal for most useful parameter combinations [3]. Implementing the new codes in a multi-level flash device does not require significant hardware resources. In fact, given that asymmetric limited-magnitude codes use standard symmetric-error correcting codes as building blocks, their implementation will be very similar to the way error-correcting codes are already used in flash controllers. In Figure 3, a block sketch of implementation is included. The readings from the flash cells are passed modulo $\ell + 1$ to the decoder of an error-correcting code for symmetric errors over an alphabet of size $\ell + 1$. The error estimates from the decoder are then subtracted from the original symbols over the size-q alphabet.

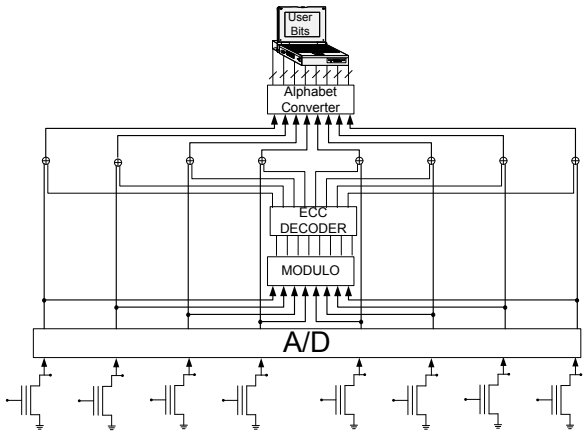

Figure 3. Proposed flash architecture with asymmetric limited-magnitude error-correcting codes.

IV. Flash Program Model and Speed-up Estimates

To actually utilize the proposed asymmetric limited-magnitude codes for write speed-up, a way to appropriately choose the code parameters is needed. Foremostly, given the number of levels q, how large should ℓ be for significant reduction in program time? An attempt to answer that question is made with the aid of an analytic model of flash program sequences. A program sequence is applied to move a cell from initial level I_0 to target level I_F. According to the model, starting from $i = 0$, each program iteration i starts from level I_i, and targets level I_F. The level change due to the pulse applied at iteration i is an exponentially distributed random variable with a λ parameter that depends on the difference $I_F - I_i$. The sequence terminates when it reaches a level that is higher than $I_F - \Delta$, where Δ is a prescribed accuracy parameter. The way the distribution parameter λ is chosen (for a given target increment $I_F - I_i$) is to fix the probability of over-programming (shooting over I_F) to some small constant ϵ. This criterion mimics a major consideration of real program-sequence designs to minimize the probability of the undesired over-shoot event. This model can now be used to compare the expected program time with and without codes for asymmetric limited-magnitude errors. When such codes are employed, the accuracy parameter Δ becomes larger, proportionally to the magnitude parameter ℓ. So by comparing the expected times for different ℓ values (including the case $\ell = 0$ when no code is used), one can obtain an estimate on the time required to program the cell to a level that will guarantee correct reading. The median[1] savings in program time, according to the model, is given by

$$\frac{\log(\ell + 1)}{\log(q/2)},$$

[1]The median is taken over all possible values of I_F

and are plotted in Figure 4 for cells with $q = 32$ levels for various values of ℓ.

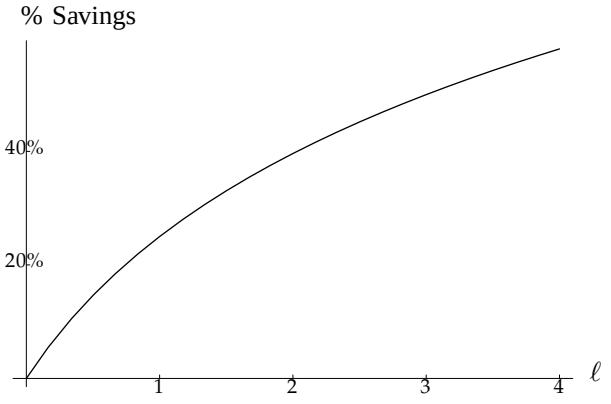

Figure 4. Percentage of program-time savings as a function of the code's magnitude limit parameter ℓ, with $q = 32$.

As seen in Figure 4, even small ℓ values (compared to q) suggest significant savings. On the other hand, increasing ℓ beyond some point exhibits diminishing returns and does not significantly contribute to increased savings in programming time. A similar conclusion can be reached experimentally from the shape of the curves in Figure 1. Thus both analytical and experimental evidence motivate the application of asymmetric limited-magnitude error-correcting codes (with small ℓ), as clearly codes for symmetric errors would consume much more redundancy, without significant return in write speed-up.

Part 2: Codes for Secure Erasure of Storage Devices

V. Codes for Data Security

When the stored data is very sensitive, leakage of even a few bits of information may be detrimental. In such cases, failure to physically erase secret bits must not reveal any information on the data to an adversary with access to the device after erasure. The task of storing data with resiliency to leakage of physical bits can be addressed with a coding-theoretic framework. In the "normal" use of error-correcting codes, redundancy is added to information such that restrictions on the stored code vectors can aid in detecting and correcting errors. When codes are used for security, their redundancy is used to *randomize* the data, such that an adversary with partial access to the code vector learns nothing about the secret data. The latter framework is called codes for wire-tap channel II [5], studied mainly in the context of secure communication in the presence of a wire-tapper. The general framework of [5] is now presented by an example.

Example 2. *Suppose we have a single bit of information, and we want to store it as n physical bits, such that an adversary with access to any $n-1$ or fewer bits can gain no information on the bit. Then the following coding system is shown to be the solution.*

Encoding

We draw at random a length n vector with even parity (or alternatively, a random codeword from the length n parity code). Then, the secret data bit is added modulo 2 to the first coordinate of the random vector.

Decoding

The legitimate reader of the data sums the elements of the stored vector modulo 2 (alternatively, computes the syndrome of the read vector with respect to the parity code) and recovers the secret.

Wire-Tapping

The wire-tapper observes $n-1$ bits of the stored vector. The number of ones out of these $n-1$ bits may be either even or odd. However, in both cases, the unavailable bit may be either a zero or a one, and each case reflects a different value of the secret bit. Hence with seeing $n-1$ bits the adversary cannot distinguish the two possibilities of the secret.

In the problem of secure storage erasure considered here, the wire-tapper is an adversary with access to bits that remain after physical erasure. Different parameters (number of information bits, block length, number of bits accessible by the adversary) are obtained by replacing the parity code in Example 2 with a different error-correcting code with different parameters. In such a case for encoding we take a random codeword from the chosen code, to which we add the k information bits at the k left positions. For decoding we simply compute the syndrome with respect to the chosen code.

VI. Wire-tap Codes and Data Storage

Wire-tap codes that protect against adversaries with partial access have the usual parameters n and k to denote the code block length and dimension, respectively. In addition, the level of security of the codes is expressed in the parameter μ: the number of symbols accessible by the adversary for which the code guarantees complete equivocation. Characterizing a code as an $[n,k,\mu]$ wire-tap codes thus provides security guarantees that are most useful for evaluating security when the properties of the physical erasure process are known.

A. Design objectives

To generalize the scheme described in Example 2 to get an $[n,k,\mu]$ wire-tap code with different choices of parameters, one needs to employ an $[n,n-k]$ error-correcting code whose dual code has minimum Hamming distance of $\mu+1$ [6] (In Example 2, for $k=1$ the $[n,n-1]$ parity code was used. The dual of the parity code is the repetition code with minimum Hamming distance n that provides $\mu = n-1$ wire-tap security.). The intuition behind the duality of error-correcting and wire-tap codes is as follows. Error-correcting codes require *parity-check* matrices having column independence to achieve high-weight codewords. Wire-tap codes, on the other hand, require column independence in their *generator* matrices, such that the code is unrestricted on the coordinates visible to the adversary. Hence in general the problem of finding good wire-tap codes can be reduced to finding good error-correcting codes. Nevertheless, to allow fast reads and writes of secret data, the wire-tap code needs to minimize the encoding and decoding complexities, while achieving high μ security guarantees. In a typical secure storage device the wire-tap code will only be used to protect a small (but dynamic) fraction of the storage that contains encryption keys, therefore the benefit of reducing the encoding and decoding complexities prevails over the natural coding-theoretic emphasis on minimizing the code redundancy. This sets the design objective to be infinite families of wire-tap codes that have optimal

(lowest possible) encoding and decoding complexities given their security parameter μ.

B. Low complexity wire-tap codes

In wire-tap coding, both the encoding and decoding complexities are directly related to the density of the code parity-check matrix. This is in contrast to error-correcting codes, where low-density parity-check matrices do not readily translate to low encoding and decoding complexities. The *row-density* of a systematic binary parity-check matrix H is defined to be the total number of ones divided by the number of rows in H. The row-density is interpreted as the average number of encoding/decoding operations per information bit. It is not hard to see [2] that $\mu+1$ is a lower bound on the row-density of H with wire-tap security of μ. Wire-tap code constructions with optimal row-densities of $\mu+1$ are next presented, based on the work of [2].

The first code family, specified in Construction 2 below, gives $[k(k+1)/2, k, k-1]$ wire-tap codes, for any k. The construction is best understood through the example that follows.

Construction 2. *Let $A_k = \{1,2,\ldots,k,k+1\}$ be the set of integers between 1 and $k+1$ (inclusive), and $S_k = \{\{i,j\} : i \in A_k, j \in A_k, i \neq j\}$ be the set of unordered pairs from A_k, with $|S_k| = k(k+1)/2$. Let $P^{(k)}$ be the $(k+1) \times k(k+1)/2$ matrix whose columns correspond to the elements of S_k: each column has exactly two ones, at locations i, j specified by the pair of the given column (the other $k-1$ elements are zeros). The parity check matrix $H^{(k)}$ of the wire-tap code is obtained from $P^{(k)}$ by erasing any single row.*

Example 3. *For $k = 4$ we have $A_k = \{1,2,3,4,5\}$ and*

$$S_k = \{\{1,2\},\{1,3\},\{1,4\},\{1,5\},\{2,3\},\{2,4\},\{2,5\},\{3,4\},\{3,5\},\{4,5\}\}$$

Then the matrix $P^{(4)}$ is given by

$$P^{(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

which after erasing the top row results in

$$H^{(4)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (1)$$

The matrix $H^{(4)}$ in (1) defines a $[10,4,3]$ wire-tap code. The row-density of the code is 4.

The matrix $H^{(k)}$ resulting from Construction 2 has k ones in each row. Therefore, the row-density of $H^{(k)}$ is k. Any row combination of $H^{(k)}$ has Hamming weight at least k (proof omitted), hence Construction 2 yields wire-tap codes with $\mu = k-1$ and optimal encoding and decoding complexities.

For better security with a slightly higher (but still optimal) density, we include the following Construction 3 for $[k(k+1)(k+2)/6, k, k(k+1)/2-1]$ wire-tap codes, for any $k > 4$.

Construction 3. *Let $B_k = \{1,2,\ldots,k,k+1,k+2\}$ be the set of integers between 1 and $k+2$ (inclusive), and $T_k = \{\{i,j,l\} : i \in B_k, j \in B_k, l \in B_k, i \neq j, j \neq l, l \neq i,\}$ be the set of unordered triples from B_k, with $|T_k| = k(k+1)(k+2)/6$. Let $Q^{(k)}$ be the $(k+2) \times k(k+1)(k+2)/6$ matrix whose columns correspond to the elements of T_k: each column has exactly three ones, at locations i, j, l specified by the triple of the given column (the other $k-1$ elements are zeros). The parity check matrix $\mathcal{H}^{(k)}$ of the wire-tap code is obtained from $Q^{(k)}$ by erasing any two rows.*

To show that using triples in Construction 3 gives similarly optimal security given the row-density, we need to prove that any row linear combination of $\mathcal{H}^{(k)}$ has weight at least $k(k+1)/2$, the weight of a row of $\mathcal{H}^{(k)}$. Obtaining parity-check matrices combinatorially by taking all possible pair or triple combinations endows the nice property that the weight of each row linear combination depends only on the number s of summed rows. Moreover, this weight can be derived in closed form, and then simple calculus allows to prove that all possible row combinations give weights at least as large as the weight of an individual row. This property is formally stated in Lemma 1 below, which is the main tool to prove the security properties of Construction 3 (refer to [2] for more details).

Lemma 1. *Let $\boldsymbol{x}$ be a linear combination of s rows from $\mathcal{H}^{(k)}$, where $1 \leqslant s \leqslant k$. Then the Hamming weight of $\boldsymbol{x}$ is given by*

$$w_H(\boldsymbol{x}) = f_k(s) \triangleq s\tbinom{k+2-s}{2} + \tbinom{s}{3} \quad (2)$$

The weight of a linear combination of rows of $\mathcal{H}^{(k)}$ depends only on s, the number of rows in the combination, and is given by the function $f_k(s)$ in (2).

The same construction technique can be applied beyond pairs and triples to higher order combinations, using Lemma 1 to find the values of k for which the resulting codes give optimal-complexity wire-tap codes.

VII. Conclusion

In the two parts of the paper, new coding frameworks were developed to promote important storage features beyond data integrity and reliability. The codes are tailored to fast storage devices, flash based and potentially others, by optimizing the code designs to meet specific properties and constraints of such storage devices. Presented here through relatively simplified models, additional work is needed to adapt the new frameworks to more realistic implementation scenarios.

References

[1] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Programming analog computational memory elements to 0.2% accuracy over 3.5 decades using a predictive method," in *proc. of the IEEE International Symposium on Circuits and Systems*, 2005, pp. 2148–2151.

[2] Y. Cassuto and Z. Bandic, "Low-complexity wire-tap codes with security and error-correction guarantees," in *Proc. of the IEEE Information Theory Workshop*, Dublin, Ireland, 2010.

[3] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multi-level flash memories," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1582–1595, 2010.

[4] M. Grossi, M. Lanzoni, and B. Ricco, "Program schemes for multilevel Flash memories," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 594–601, 2003.

[5] L. Ozarow and A. D. Wyner, "Wire-tap channel II," *At&T Bell Laboratories Technical Journal*, vol. 63, no. 10, pp. 2135–2157, Dec. 1984.

[6] V. Wei, "Generalized Hamming weights for linear codes," *IEEE Transactions on Information Theory*, vol. 37, no. 5, pp. 1412–1418, 1991.