

Scaling efficient code-based cryptosystems for embedded platforms

Felipe P. Biasi, Paulo S. L. M. Barreto*, Rafael Misoczki, Wilson V. Ruggiero

Abstract—We describe a family of highly efficient codes for cryptographic purposes and dedicated algorithms for their manipulation. Our proposal is especially tailored for highly constrained platforms, and surpasses certain conventional and post-quantum proposals (like RSA and NTRU, respectively) according to most if not all efficiency metrics.

Index Terms—Algorithms, Cryptography, Decoding, Error correction

I. INTRODUCTION

QUANTUM computers, should they become a technological reality, will pose a threat to public-key cryptosystems based on certain intractability assumptions, like the integer factorization problem, or IFP (like RSA), and the discrete logarithm problem, or DLP (like Diffie-Hellman or DSA, in their elliptic curve version or otherwise). To face this scenario, several cryptosystems have been proposed that apparently resist attacks mounted with the help of quantum computers. The security of these so-called post-quantum cryptosystems [8] stems from quite distinct computational intractability assumptions. Such schemes are not necessarily new — for instance, cryptosystems based on coding theory (specifically, on the intractability of the syndrome decoding problem, or SDP) are known for nearly as long as the very concept of asymmetric cryptography itself, though they have only recently been attracting renewed interest.

However, being quantum-resistant is not the only interesting feature of many post-quantum proposals — some of them are equally remarkable because of their improved efficiency and simplicity for certain types of applications relatively to conventional schemes. Thus, schemes based on the SDP entirely avoid the multiprecision integer arithmetic typically needed by IFP or DLP cryptosystems, and their computational cost is usually a few orders of complexity smaller than those systems, reaching $\tilde{O}(n)$ instead of $\tilde{O}(n^2)$ or $\tilde{O}(n^3)$ which are commonplace in pre-quantum schemes. This indicates that post-quantum alternatives may have advantages even in situations where quantum attacks are not the main concern,

and justifies the investigation on how advantageous they can be.

Particularly interesting scenarios where such post-quantum schemes may have a positive impact are wireless sensor networks [2], [35] and the so-called “Internet of Things,” in which a wide range of devices are interconnected, from the most powerful clustered servers to embedded systems with extremely limited processing resources, storage, bandwidth occupation and power consumption, including micro-controllers [25], [43] and dedicated hardware [27].

One of the leading families of post-quantum cryptographic schemes is that of code-based cryptosystems [28], [34]. In contrast with the form in which these systems were originally proposed, where key sizes were typically large, modern approaches do offer far more space-efficient parameters, being fairly practical on general-purpose platforms. One can thus ask whether such schemes are suitable for highly constrained platforms as well.

Low-density parity-check (LDPC) codes and their quasi-cyclic variants (QC-LDPC) have been proposed for cryptographic applications [3], [4], [5], [6], [7], [24], [31], [41] although in a form still unsuitable for constrained platforms. Recently, quasi-cyclic moderate-density parity-check (QC-MDPC) codes have been designed to provide strong security assurances for McEliece-style cryptosystems [30]. Such codes are arguably ideal for modern general-purpose platforms, matching or surpassing the processing efficiency of conventional cryptosystems. However, no assessment of their suitability for constrained platform has been made, and indeed the traditional bit-flipping and belief-propagation decoding methods, even though they are quite processing-efficient, appear at first glance unsuitable for an Internet-of-Things scenario due to their considerable storage requirements.

A. Our Results

Our contributions in this paper are twofold:

- On the one hand, a family of linear error-correcting codes (so-called CS-MDPC codes) that are highly efficient for cryptographic applications in terms of reduced per-key and per-message bandwidth occupation;
- On the other hand, an efficient decoder for that family of codes that is especially tailored for (though not restricted to) constrained platforms.

Specifically, we show how to obtain code-based cryptosystems where the public keys and the space overhead incurred for each cryptogram are comparable in size to, or even smaller than, the corresponding values for the RSA cryptosystem at

* P. Barreto is the corresponding author: pbarreto@larc.usp.br

P. Barreto is supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under research productivity grant 303163/2009-7.

F. Biasi, P. Barreto and W. Ruggiero are supported by Cisco Research Award 2011-050 ‘Alternate Public Key Cryptosystems’

F. Biasi, W. Ruggiero and P. Barreto are with the Department of Computer and Digital Systems Engineering, Escola Politécnica, Universidade de São Paulo, Brazil. (e-mail: {fbiasi,pbarreto,wilson}@larc.usp.br)

R. Misoczki is with SECRET team, INRIA Paris-Rocquencourt, France. (e-mail: rafael.misoczki@inria.fr)

practical security levels. A careful selection of design features for the key generation, encoding, and decoding algorithms lead to very short processing times, and executable code size in software or area occupation in hardware (and thus potentially also energy consumption) tend to be considerably smaller than what can be attained with RSA or elliptic curve cryptosystems. Our proposed variant of the bit flipping decoding technique needs only $O(1)$ ancillary storage, in comparison with $O(n)$ (where n is the code length) as in previous variants of that technique.

B. Organization of the Paper

The remainder of this document is organized as follows. We provide theoretical preliminaries in Section II, including LDPC and MDPC codes, the hard decision decoding method, and code-based cryptosystems. We describe the new family of codes and assess its security properties in Section III. In Section IV we outline our proposed techniques to deploy code-based cryptosystems on embedded platforms, in particular an efficient bit-flipping decoder that takes only $O(1)$ ancillary storage instead of the usual $O(n)$ requirements. We illustrate some suggested parameters for typical security levels in Section V and assess the overall results of our proposal experimentally in Section VI. We conclude in Section VII.

II. PRELIMINARIES

A. General notation

Matrix and vector indices will be numbered from 0 through-out this paper, unless otherwise stated. Let p be a prime and let $q = p^m$ for some $m > 0$. The finite field of q elements is written \mathbb{F}_q . Given $h \in \mathbb{F}_2^r$, we denote by $\text{cir}(h)$ the circulant matrix

$$\text{cir}(h) = \begin{bmatrix} h_0 & h_1 & \dots & h_{r-1} \\ h_{r-1} & h_0 & \dots & h_{r-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \dots & h_0 \end{bmatrix}.$$

B. Error Correcting Codes

A (binary) linear $[n, k]$ error-correcting code \mathcal{C} is a subspace of \mathbb{F}_2^n of dimension k . Such a code is specified by either a *generator matrix* $G \in \mathbb{F}_2^{k \times n}$ such that $\mathcal{C} = \{uG \in \mathbb{F}_2^n \mid u \in \mathbb{F}_2^k\}$, or else by a *parity-check matrix* $H \in \mathbb{F}_2^{r \times n}$ such that $\mathcal{C} = \{v \in \mathbb{F}_2^n \mid vH^T = 0^r\}$ where $r = n - k$.

We will be particularly interested in *quasi-cyclic* codes, namely, codes that admit a parity-check matrix consisting of n_0 horizontally joined circulant square blocks of size $r \times r$. Thus:

$$H = [\text{cir}(h_0) \mid \text{cir}(h_1) \mid \dots \mid \text{cir}(h_{n_0-1})],$$

where $h_i \in \mathbb{F}_2^r$, $0 \leq i < n_0$. The representation advantages of such codes are obvious, since H can be compactly stored as n_0 sequences of r bits each.

The *syndrome decoding problem* (SDP) consists of computing an error pattern $e \in \mathbb{F}_2^n$ given a parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ for the underlying code, and a syndrome $s = eH^T \in \mathbb{F}_2^r$. In general the SDP is NP-hard, but sometimes the knowledge of certain structural code properties makes this problem solvable in polynomial time.

C. LDPC codes

LDPC codes were invented by Robert Gallager [20] and are linear codes obtained from sparse bipartite graphs. Suppose that \mathcal{G} is a graph with n left nodes (called message nodes) and r right nodes (called check nodes). The graph gives rise to a linear code of block length n and dimension at least $n - r$ in the following way: The n coordinates of the codewords are associated with the n message nodes. The codewords are those vectors (c_1, \dots, c_n) such that for all check nodes the sum of the neighboring positions among the message nodes is zero.

The graph representation is analogous to a matrix representation by looking at the adjacency matrix of the graph: let H be a binary $r \times n$ -matrix in which the entry (i, j) is 1 if and only if the i -th check node is connected to the j -th message node in the graph. Then the LDPC code defined by the graph is the set of vectors $c = (c_1, \dots, c_n)$ such that $H \cdot c^T = 0$. The matrix H is called a *parity check matrix* for the code. Conversely, any binary $r \times n$ matrix gives rise to a bipartite graph between n message and r check nodes, and the code defined as the null space of H is precisely the code associated to this graph. Therefore, any linear code has a representation as a code associated to a bipartite graph (note that this graph is not uniquely defined by the code). However, not every binary linear code has a representation by a *sparse* bipartite graph. If it does, then the code is called a low-density parity-check (LDPC) code.

An important subclass of LDPC codes that feature encoding advantages over other codes of the same class is that of quasi-cyclic low-density parity-check (QC-LDPC) codes [13], [40]. In general, an $[n, k]$ -QC-LDPC code satisfies $n = n_0b$ and $k = k_0b$ (thus also $r = r_0b$) for some b, n_0, k_0 (and r_0), and admits a parity-check matrix consisting of $n_0 \times r_0$ blocks of $b \times b$ sparse circulant submatrices. Of particular importance is the case where $b = r$ (and hence $r_0 = 1$ and $k_0 = n_0 - 1$), since a systematic parity-check matrix for this code is entirely defined by the first row of each $r \times r$ block. We say that the parity-check matrix is in *circulant form*.

D. QC-MDPC codes

A cryptographically interesting subclass of the QC-LDPC family is that of quasi-cyclic *moderate-density parity-check* (QC-MDPC) codes [30].

QC-MDPC codes in this sense are an entirely distinct class from a family of algebraic codes also known as ‘MDPC’ and designed by Ouzan and Be’ery [36], despite the name clash. The goal set forth in the latter approach is to obtain high-rate codes of short to moderate length whose duals contain known intermediate-weight words (and thus admit parity-check matrices of moderate density, hence the ‘MDPC’ name), but still have a good error correction capability in comparison with algebraic codes like BCH with similar length and rate. Examples from [36] indicate that typical densities are in the range 17% to 28% of the code length. Thus they are indeed intermediate between usual LDPC codes and general ones, but the density is too high for conventional LDPC decoding techniques, to the effect that those codes are not classical Gallager codes in the sense that the density of their dual

codes puts them beyond the capability of decoding techniques like plain belief propagation and bit flipping, and especially tailored decoders must thus be adopted.

By contrast, QC-MDPC codes in the sense of Misoczki *et al.* are oriented toward cryptographic purposes, with densities close enough to LDPC codes as to enable decoding by Gallager's simpler (and arguably more efficient) belief propagation and bit flipping methods, yet dense enough to prevent attacks based on the presence of too sparse words in the dual code like the Stern attack [39] and variants, without loosing too much of the error correcting capability so as to keep information-set decoding attacks [9], [10] infeasible as well. Furthermore, to prevent structural attacks as proposed by Faugère *et al.* [18] and by Leander and Gauthier [42], cryptographically-oriented codes must remain as unstructured as possible except for the hidden trapdoor that enables private decoding and, in the case of quasi-cyclic codes, external symmetries that allow for efficient implementation. Finally, the very circulant symmetry might introduce weaknesses as pointed out by Sendrier [38], but these induce only a polynomial (specifically, linear) gain in attack efficiency, and a small adjustment in parameters copes with it entirely. Typical densities in this case are in range 0.4% to 0.9% of the code length, one order of magnitude above LDPC codes but well below the published MDPC range above, and certainly within the realm of Gallager codes. Construction is also as random as possible, merely keeping the desired density and circulant geometry, and code lengths are much larger than typical MDPC values.

E. Gallager's Hard Decision (Bit Flipping) Decoding Method

We briefly recapitulate Gallager's hard decision decoding algorithm, closely following the very concise and clear description by Huffman and Pless [22]. This will provide the basis for the efficient variant we propose for embedded platforms.

Assume that the codeword is encoded with a binary LDPC code \mathcal{C} for transmission, and the vector c is received. In the computation of the syndrome $s = cH^T$, each received bit of c affects at most d_v components of that syndrome. If only the j -th bit of c contains an error, then the corresponding d_v components s_i of s will equal 1, indicating the parity check equations that are not satisfied. Even if there are some other bits in error among those that contribute to computation of s_i , one expects that several of the d_v components of s will equal 1. This is the basis of Gallager's *hard decision decoding*, or bit-flipping, algorithm.

- 1) Compute cH^T and determine the unsatisfied parity checks (namely, the parity checks where the components of cH^T equal 1).
- 2) For each of the n bits, compute the number of unsatisfied parity checks involving that bit.
- 3) Flip the bits of c that are involved in the largest number of unsatisfied parity checks.
- 4) Repeat steps 1, 2, and 3 until either $cH^T = 0$, in which case c has been successfully decoded, or until a certain bound in the number of iterations is reached, in which case decoding of the received vector has failed.

The bit-flipping algorithm is not the most powerful decoding method for LDPC codes; indeed, the belief propagation technique [20], [22] is known to exceed its error correction capability. However, belief-propagation decoders involve computing ever more refined *probabilities* that each bit of the received word c contains an error, thus incurring floating point arithmetic or suitable high-precision approximations and computationally expensive algorithms. In a scenario where the number of errors is fixed and known in advance, as is the case of cryptographic applications, parameters can be designed so that the more powerful but also more complex and expensive belief propagation methods are not necessary for decoding.

We therefore focus on the problem of designing an optimized variant of bit-flipping decoding for highly constrained platforms. Specifically, such methods still suffer from the drawback of requiring a large amount of ancillary memory for counters: if each column of H has Hamming weight d_v , step 2 requires $(\lfloor \lg d_v \rfloor + 1)$ bits to store the number of unsatisfied parity-checks for each of the n bits of c , hence $n(\lfloor \lg d_v \rfloor + 1)$ bits overall. Besides, steps 2 and 3 involve a loop of length n each, introducing processing inefficiency. We will show how to avoid these drawbacks in Section IV-C.

F. McEliece and Niederreiter encryption

The McEliece encryption scheme was proposed by R. McEliece [28] in 1978. In that scheme, the public key is a generator matrix for a certain code whose decoder is taken to be the private key. An equivalent scheme using a parity-check matrix as public key was proposed by H. Niederreiter in 1986 [34]. We briefly review these schemes, which consist of three algorithms (KeyGen, Encrypt, Decrypt) each.

1) McEliece:

- **KeyGen:** Select a binary t -error correcting $[n, k]$ -code \mathcal{C} with a decoding trapdoor \mathcal{D} and a $k \times n$ generator matrix G in systematic form. The public key is (G, t) , and the private key is the decoding trapdoor \mathcal{D} .
- **Encrypt:** To encrypt a plaintext $m \in \mathbb{F}_2^k$ into a cryptogram $c \in \mathbb{F}_2^n$, select a uniformly random error pattern $e \in \mathbb{F}_2^n$ and weight t , and compute $c \leftarrow m \cdot G + e$.
- **Decrypt:** To decrypt $c \in \mathbb{F}_2^n$, apply the decoding trapdoor \mathcal{D} to correct the t errors in c (thus finding the error pattern $e \in \mathbb{F}_2^n$ of weight t), then extract $m \in \mathbb{F}_2^k$ from the first k columns of $c - e$.

2) Niederreiter:

- **KeyGen:** Select a binary t -error correcting $[n, k]$ -code \mathcal{C} with a decoding trapdoor \mathcal{D} and an $r \times n$ parity-check matrix H in systematic form, where $r = n - k$. The public key is (H, t) , and the private key is the decoding trapdoor \mathcal{D} .
- **Encrypt:** To encrypt a plaintext $m \in \mathbb{Z}/\binom{n}{t}\mathbb{Z}$ into a cryptogram $c \in \mathbb{F}_2^n$, encode m into a vector $e \in \mathbb{F}_2^n$ of weight t via some conventional permutation unranking method, and compute $c \leftarrow e \cdot H^T$.
- **Decrypt:** To decrypt $c \in \mathbb{F}_2^n$, apply the decoding trapdoor \mathcal{D} to the syndrome c (thus finding the corresponding vector $e \in \mathbb{F}_2^n$ of weight t), then decode $m \in \mathbb{Z}/\binom{n}{t}\mathbb{Z}$ from e via permutation ranking.

Although the security of these two schemes is equivalent, Niederreiter is the more efficient [11], being therefore the method of choice for constrained platforms.

III. AN EFFICIENT FAMILY OF MDPC CODES

The QC-MDPC codes [30] are arguably among the most efficient settings for code-based cryptosystems. However, QC-MDPC parameters for practical security levels, specifically those corresponding to a cost between a legacy-level 2^{80} and a top-level 2^{256} steps to mount the best possible attacks, yield key and ciphertext space overheads well above the corresponding values achievable with the RSA cryptosystem, which is perhaps the most widely deployed asymmetric cryptography scheme today, and constitutes for that reason a practical upper bound for the corresponding parameters in other cryptographic schemes. Therefore one cannot claim that those codes are generically suitable for constrained platforms.

It turns out we can do better than that with a proper subset of QC-MDPC codes. To define it, we now introduce a class of matrices that admit a space-efficient representation:

Definition 1. *Given a ring \mathcal{R} and an integer p , the set of cyclosymmetric matrices of order p over \mathcal{R} is the set of $\Delta_p(\mathcal{R})$ of square $p \times p$ matrices over \mathcal{R} that are both circulant and symmetric.*

Cyclosymmetric matrices constitute a subring of the ring $\mathcal{R}^{p \times p}$ of $p \times p$ matrices over \mathcal{R} , which can be seen by the fact that the identity matrix is cyclosymmetric and that the product of symmetric matrices is itself symmetric iff the factors commute, and indeed circulant matrices are commutative: $(AB)^T = B^T A^T = BA = AB$ (all other properties are trivial). We call this the *cyclosymmetric ring* of order p over \mathcal{R} .

A cyclosymmetric ring can be itself defined over another cyclosymmetric ring and so on recursively, yielding a *multilayered* cyclosymmetric ring ultimately defined over a non-cyclosymmetric ring. This ring tower is written as $\Delta_{p_1}(\dots \Delta_{p_L}(\mathcal{R}_0))$ for successively embedded rings of orders p_1, \dots, p_L . Let $\text{lyr}(\mathcal{R})$ denote the number of layers of a multilayered cyclosymmetric ring \mathcal{R} . We define the number of layers of a non-cyclosymmetric ring \mathcal{R}_0 (e.g. a finite field) to be $\text{lyr}(\mathcal{R}_0) = 0$, and then recursively $\text{lyr}(\Delta_p(\mathcal{R}')) = \text{lyr}(\mathcal{R}') + 1$. Thus, $\text{lyr}(\Delta_{p_1}(\dots \Delta_{p_L}(\mathcal{R}_0))) = L$.

The interest in a cyclosymmetric ring resides in the fact that elements of $\Delta_p(\mathcal{R})$ can be represented as a sequence of $\lfloor p/2 \rfloor + 1$ elements from \mathcal{R} , asymptotically occupying only half the space required by a merely circulant matrix of order p over \mathcal{R} . To see this, just note that a circulant $p \times p$ matrix has the form $C_{ij} = c_{(j-i) \bmod p}$ where c is the first row of that matrix, while a symmetric matrix satisfies $C_{ij} = C_{ji}$, thus combining both conditions yields $c_{(j-i) \bmod p} = c_{(i-j) \bmod p}$, which for $i = 0$ (since the first row alone defines the entire matrix) simplifies to $c_j = c_{-j \bmod p}$, or $c_j = c_{p-j}$ for $j > 0$. Therefore, the sequence c_1, \dots, c_{p-1} is a palindrome (and c_0 is an arbitrary bit), and only $c_0, \dots, c_{\lfloor p/2 \rfloor + 1}$ are independent.

The space efficiency becomes more noticeable for rings with several layers: an element of $\Delta_{p_1}(\dots \Delta_{p_L}(\mathcal{R}_0))$ is represented as $\prod_{i=1}^L (\lfloor p_i/2 \rfloor + 1)$ elements of \mathcal{R}' , roughly a fraction $1/2^L$ of

the size of a generic circulant matrix of order $\prod_{i=1}^L p_i$ over \mathcal{R}_0 .

Extending the analogy, we define the family of *cyclosymmetric codes*:

Definition 2. *A cyclosymmetric (CS) code over \mathbb{F}_q is a code which admits a block parity-check matrix whose blocks correspond to elements of a (multiplayerd) cyclosymmetric ring.*

In other words, a cyclosymmetric $[n, n-r]$ -code admits an $r \times n$ parity-check matrix H with $r = r_0 p$, $n = n_0 p$, consisting of $r_0 \times n_0$ cyclosymmetric blocks of size $p \times p$ over some smaller ring. The natural advantage of these codes is the compact representation of such parity-check matrices. A particularly efficient case occurs when $r_0 = 1$, that is, H is a simple sequence of n_0 cyclosymmetric blocks of size $r \times r$: if H is in systematic form, $r = p_1 \cdots p_L$, and $\mathcal{R} = \Delta_{p_1}(\dots \Delta_{p_L}(\mathbb{F}_q) \dots)$, then H occupies only $(n_0 - 1) \prod_{i=1}^L (\lfloor p_i/2 \rfloor + 1) \lg q$ bits.

In cryptographic applications, the natural choice is to adopt binary codes, i.e. $q = 2$, and in particular MDPC codes, due to the simplicity of the decoding algorithm and the greatly reduced parameters that these codes allow for every desired security level. A cyclosymmetric MDPC code is, therefore, a CS-MDPC code. Moreover, in the same cryptographic context we not only propose the use CS-MDPC codes, but also to restrict *error patterns* to the same form as the concatenation of first rows of cyclosymmetric matrices, so that these patterns stand themselves for sequences of cyclosymmetric ring elements, as long as this does not affect the security level.

A disadvantage of a too large number of layers is that, on average, each ‘1’-bit among the $\prod_{i=1}^L (\lfloor p_i/2 \rfloor + 1)$ independent bits of each block of H represents about 2^L ‘1’-bits in the full $r \times r$ block, rapidly increasing the parity-check matrix density and therefore limiting the error correction capability of bit-flipping and related decoders. However, small values of L (one or two, in some cases possibly even three) yield potentially interesting parameters for cryptographic applications.

A. Security considerations

An immediate observation on the structure of cyclosymmetric codes is that one can optimize the Stern attack [9], [17], [39] and its variants by taking advantage of the form of each row of the parity-check matrix when performing linear algebra operations. Indeed, Stern tries to retrieve a row of low density from the dual code; since the first row consists of one element followed by a palindrome, and the remaining rows are rotated versions thereof, one can in principle reduce in half the overall effort incurred by row manipulations. However, this apparent improvement may turn out to be ineffective: linear algebra operations quickly destroy the palindrome structure within the rows, thwarting the optimization.

Leon’s attack [26] and related ones do not seem to benefit at all either, because they already ignore part of each row involved in linear algebra operations. Interestingly, a brute force attack would be faster than Leon’s against cyclosymmetric codes because it would need to test only $\binom{p/2}{w/2}$ rather than $\binom{p}{w}$ elements, yet for any practical choice of parameters

that number remains far above the cost of Stern's or similar attacks. For example, parameters for which the cost of the best known variants of Stern is about 2^{80} with block size $p = 4800$ and private code density $w = 45$, the cost of brute force would be $\binom{p/2}{w/2} \approx 2^{177}$.

Similar observation apply to information-set decoding attacks [9], [10]: at most, one would expect an improvement by a factor of 2^L in the attack cost for L -level CS-MDPC codes (recall that, in practice, $L \leq 2$).

There seems to be no essential restriction to the value of p , although a prudent choice would seem to be to take prime p to avoid the possibility of attacking smaller subcodes. No structural attacks along the lines of Faugère *et al.* [18] or Leander and Gauthier [42] seems to apply, since the CS-MDPC trapdoor is of a statistical rather than algebraic nature.

Apart from this, CS-MDPC codes appear to inherit most if not all of the security properties of the superclass of QC-MDPC codes, as indicated in Section II-D. One consequence of all these considerations is that, to the best of our knowledge, the best attacks against CS-MDPC codes are precisely the best attacks against *generic* QC-MDPC codes.

IV. SCALING THE IMPLEMENTATION TO EMBEDDED PLATFORMS

A. General operation

We use a representation of sparse matrices with a plain arithmetic: both matrix-matrix and matrix-vector products coalesce into (vector-vector) cyclic convolution, for which efficient algorithms like Karatsuba [23] and FFT are known. However, simple 'textbook' multiplication algorithms, slightly modified so as to operate on circulant matrices represented by their first row alone, are not only more compact, but at least as fast (and often faster) than more advanced counterparts because of the sparsity of the arguments. Indeed, the operations that actually occur in the Niederreiter cryptosystem always involve at least one sparse operand:

- inversion of a secret, sparse circulant matrix H yielding a public, dense matrix K : this is achieved with a carefully tuned extended Euclidean algorithm (see Section IV-B).
- Computation of the public syndrome of a sparse error vector. This syndrome is the product of the public, dense parity-check matrix K by the sparse vector e .
- Computation of the private, decodable syndrome $s^T = He^T$ corresponding to a given public syndrome $c^T = Ke^T$. This is the product $H_{n_0-1}c^T$ of the sparse secret matrix H_{n_0-1} by the given dense syndrome c^T , since $K = H_{n_0-1}^{-1}H$ and hence $H_{n_0-1}c^T = H_{n_0-1}Ke^T = He^T = s^T$.
- Additionally, our strategy recovers the decodable syndrome s from a modified but nonzero syndrome \hat{s} after a failed decoding attempt. Such an attempt yields an incorrect error vector \hat{e} of weight not exceeding $\text{HDDMARGIN}(t)$ satisfying the relation $s^T = \hat{s}^T + He^T$. Thus, recovering s involves the product He^T of a sparse matrix by a sparse vector.

Interestingly, the McEliece cryptosystem does involve a product of a dense public generator matrix and a dense random vector in semantically secure constructions like Fujisaki-Okamoto [19]. This is further reason to adopt the Niederreiter cryptosystem on constrained platforms.

B. Space-efficient convolutional inverse

The extended Euclidean algorithm yields a time-efficient method to compute the inverse of a circulant matrix $H = \text{cir}(h)$. The technique consists of mapping the array h (with components h_j , $0 \leq j < r$) to a polynomial $h(x) = \sum_{0 \leq j < r} h_j x^j \in \mathbb{F}_2[x]$, computing the modular inverse $h(x)^{-1} \pmod{x^r - 1}$, and mapping $h(x)^{-1}$ back to an array denoted h^{-1} such that $H^{-1} = \text{cir}(h^{-1})$.

An apparently less widely known property of the extended Euclidean algorithm is that it admits a space-efficient implementation as well. In its most usual form, when computing $h^{-1} \pmod{m}$ the algorithm keeps track of four polynomials $f, g, b, c \in \mathbb{F}_2[x]$ (plus two additional polynomials $u, v \in \mathbb{F}[x]$ that are usually only implicit) related by the constraints $f = bh + um$ and $g = ch + vm$. This suggests a naive implementation requiring up to $4r$ bits of storage for those four polynomials. However, polynomials f and c can actually coexist on the same storage area, and similarly for polynomials g and b , as long as $r + 2$ bits are available for each of these pairs (totaling $2r + 4$ bits) because, at any step of the algorithm execution, it holds that $\deg(f) + \deg(c) \leq r$ and $\deg(g) + \deg(b) \leq r$. One can easily prove this by Floyd-Hoare logic.

C. A space-efficient decoder

The technique of bit-flipping decoding has received a substantial amount of attention in the literature since Gallager's discovery of LDPC codes [12], [16], [20], [21], [29], [32], [33], [44], [45], [46], [47], [48]. However, these are mostly concerned with improving the error correction capability rather than reducing computational resource requirements. Even techniques designed for VLSI like the soft bit-flipping (SBF) technique [14], [15], which might be potential candidates for implementation on the small processors typical of an Internet of Things scenario, turn out to take far more ancillary storage (namely, still $O(n)$ for a code of length n) than is typically available on those processors.

It turns out that one can entirely avoid the need for the large storage requirements of a bit-flipping decoder. For cryptographic applications, where the number of introduced errors is fixed and known beforehand, the error correction capability is not the central concern, as long as the desired security level can be attained while fitting the available resources. The variant we propose targets precisely this need. We now describe that variant, together with a rationale for each decision. The full method is summarized as Algorithm 1.

- *On-the-fly counter update*: The usual bit-flipping strategy requires two passes over the word variables at each step of the decoding process, namely, a first pass to determine the number of parity errors each variable is involved in (thus keeping an array of counters, one for each variable), and a second pass to tentatively correct the most suspicious variables, which are taken to be those whose parity error count is above a certain threshold. While the second pass could in principle be avoided by adopting a carefully designed data structure singling out the positions that do actually exceed the threshold, not only would maintaining such a structure be considerably

expensive, but the approach is not effective for the better part of the decoding process since a large fraction of the variables is expected (and experimentally observed) to be deemed suspicious, to the effect that this whole approach turns out to be easily outperformed by plain counters in both storage requirements and processing time.

We avoid the second pass, the complicated data structure, and even the need to keep an array of counters by counting on-the-fly the number of parity errors each variables is involved in, then deciding immediately whether it has to be tentatively corrected, and modifying the syndrome accordingly.

A consequence of this is that the relation between the actual parity-error counts evaluated on-the-fly and the bit flipping threshold value is likely to change at each such correction, and the decisions that will be taken for variables not yet reached may differ from what they would be if the counters were computed separately. In fact, the parity error threshold becomes known only approximately (unless one took the effort to update it by checking all variables again each time one of them is corrected), but this turns up not to be detrimental to a successful correction of all errors; on the contrary, this is empirically observed to *enhance* the chance of a successful decoding for practical parameters. This can be explained by considering that the number of false positives and false negatives in the error detection heuristic for bits not yet processed is reduced whenever one real error is corrected: in other words, there is a better signal-to-noise ratio in the bit reliability estimation that would be missed if all counters were computed before any actual correction is attempted.

- *Onset threshold estimation:* As we pointed out, bit flipping works not only with the exact value of the parity-error threshold, but also with a reasonable estimate thereof. This holds equally well at the onset of the process, so that not even the initial parity-error threshold θ_0 needs to be exact.

Analytically deriving a reasonable initial value, however, proves to be rather difficult, but it is easy to bypass this problem by adopting an experimental estimate. This is done by generating a number (say, of order 10^3) of codes uniformly at random, then performing for each one a number (say, of order 10^3 as well) of decodings of uniformly generated error patterns of suitable weight, and finally tallying the initial maximum count of parity errors influenced by each variable. The empirical estimate of the initial parity-error threshold θ_0 is then taken to be the average of those maximum counts. The standard deviation is observed to be fairly small, so this approximation, which lies around a fraction $0.7\text{--}0.8 d_v$, according to the values of r , t , and d_v itself, leads to a surprisingly stable decoding behavior.

- *Threshold fine tuning:* The actual parity-error threshold for bit-flipping does not need to be the very maximum current parity-error count among all variables. A faster variant is achieved by setting the threshold somewhere, say δ parity errors, below that maximum. Experimentally,

a fine-tuned δ can improve decoding speed by an order of magnitude, so this variant is worthwhile.

However, not only the decoding speed, but also the likeliness of decoding failure increases with growing δ , imposing a cutoff at a certain optimal point. As in the case of the initial threshold estimate, deriving an analytical value for the optimum δ is a difficult and elusive task. We therefore adopt an empirical estimate obtained from simulations here as well.

- *Decoding failure handling:* Because a large δ makes a decoding failure more likely, the decoder must be prepared to decrease the actual δ and restart the process. Fortunately, rewinding the process to recover the original is easy to do in-place, as the difference between the original syndrome and the current one is the syndrome of the partial error pattern constructed by the decoder up to the failure detection.

Decoding failure is usually detected when a maximum number of decoding attempts is exceeded. Early detection is possible, however, by following the evolution of the weight of error pattern being reconstructed. Although that weight can temporarily surpass the final weight of t errors, in a successful decoding the provisional weight is very unlikely to be too large. A simple and sensible upper limit obtained from simulations is $3t/2$ errors (i.e. allow the decoding process to accumulate spurious errors up to 50% above the t limit before deciding for failure and decreasing δ), since no successful decoding has been observed to reach as high as this margin before the process begins to reduce it and converge to zero errors.

- *Simple supporting algorithms:* Sophisticated algorithms with a good asymptotic behavior turn out to be an unnecessary hindrance in the context of decoding at practical cryptographically-oriented parameters.

Thus, for instance, even though convolution-style algorithms may seem ideal to handle products of circulant matrices, in practice one of the factors is usually so sparse that the much simpler approach of just adding together a few rows or columns of the other factor as indicated by the other factor yields a faster outcome (and smaller executable code).

Likewise, representing the error pattern being reconstructed e as an unsorted list of error coordinates yields the most compact representation of e and is very efficient for cryptographic applications because of the relatively small target weight of e , even though this incurs sequential searches and updates.

Taking all this into account, we describe in Algorithm 1 an efficient variant of the hard-decision decoding method tailored for platforms with highly constrained data and code storage and processing power.

V. SUGGESTED PARAMETERS

For the sake of illustration, sample CS-MDPC parameters for typical security levels are listed on Tables I and II.

Although key sizes still fall short of reaching typical values for pre-quantum elliptic curve cryptosystems, CS-MDPC

Algorithm 1 Efficient hard-decision decoder for constrained platforms

INPUT: $H \in \mathbb{F}_2^{r \times n}$ (with $n = n_0 r$), a systematic quasi-cyclic low-density parity-check matrix with constant column weight d_v , represented as an array of n_0 lists of the d_v coordinates of the nonzero components in each cyclic block of H .

INPUT: $s \in \mathbb{F}_2^r$, a bit array representing the received syndrome.

INPUT: δ , a threshold margin.

INPUT: θ_0 , an estimate of the largest number of unsatisfied parity checks among the n variables (bits) of the codeword with errors.

INPUT: *iterBound*, a limit on the number of iterations for successful decoding (the heuristic default is *iterBound* = t),

OUTPUT: $e \in \mathbb{F}_2^n$, a sparse vector of weight $\text{wt}(e) \leq t$ represented as a list of coordinates of its nonzero components (but able to hold the coordinates of $\text{HDDMARGIN}(t) > t$ such coordinates), or \emptyset upon failure.

REMARK: compute mod remainders via iterated subtraction.

```

1: retry  $\leftarrow$  false
2: repeat
3:    $ew \leftarrow 0$   $\triangleright$  initialize  $e$  to no errors
4:    $iter \leftarrow 0$ 
5:    $\theta \leftarrow \theta_0$   $\triangleright$  initial estimate
6:   repeat
7:      $newmax \leftarrow 0$   $\triangleright$  new estimate for  $\theta$ 
8:     for  $j \leftarrow 0$  to  $n - 1$  do
9:        $L \leftarrow H[\lfloor j/r \rfloor]$ 
10:       $unsat \leftarrow 0$ 
11:      for  $z \leftarrow 0$  to  $d_v - 1$  do
12:        if  $s[(j + L[z]) \bmod r] = 1$  then
13:           $unsat \leftarrow unsat + 1$ 
14:        end if
15:      end for
16:       $newmax \leftarrow \max\{unsat, newmax\}$ 
17:      if  $unsat \geq \theta - \delta$  then  $\triangleright$  try to correct:
18:        if  $\exists q \in [0..ew - 1]$  such that  $e[q] = j$  then
19:           $ew \leftarrow ew - 1$ ,  $e[q] \leftarrow e[ew]$ 
20:        else if  $ew < \text{HDDMARGIN}(t)$  then
21:           $e[ew] \leftarrow j$ ,  $ew \leftarrow ew + 1$ 
22:        else  $\triangleright$  too many spurious errors introduced
23:          break  $\triangleright$  to line 31
24:        end if
25:        for  $z \leftarrow 0$  to  $d_v - 1$  do  $\triangleright$  update syndrome:
26:           $i \leftarrow (j + L[z]) \bmod r$ 
27:           $s[i] \leftarrow \neg s[i]$ 
28:        end for
29:      end if
30:    end for
31:     $\theta \leftarrow newmax$ 
32:     $\triangleright$  Iterate until the syndrome is zero (or until a bound on the
33:    number of iterations is reached)
34:     $iter \leftarrow iter + 1$ 
35:  until  $\text{wt}(s) = 0$  or  $iter = \text{iterBound}$ 
    
```

Algorithm 1 (Continued)

```

34: if ( $\text{wt}(s) \neq 0$  or  $ew > t$ ) and  $\delta > 0$  then
35:    $\delta \leftarrow \delta - 1$   $\triangleright$  threshold margin was too high
36:   for  $q \leftarrow 0$  to  $ew - 1$  do  $\triangleright$  revert syndrome to
37:   original form:
38:      $j \leftarrow e[q]$ 
39:      $L \leftarrow H[\lfloor j/r \rfloor]$ 
40:     for  $z \leftarrow 0$  to  $d_v - 1$  do
41:        $i \leftarrow (j + L[z]) \bmod r$ 
42:        $s[i] \leftarrow \neg s[i]$ 
43:     end for
44:   end for
45:    $retry \leftarrow \text{true}$ 
46: end if
47: until not  $retry$ 
48: if  $\text{wt}(s) = 0$  and  $ew \leq t$  then
49:   return  $e, ew$ 
50: else
51:   return  $\emptyset$ 
52: end if
    
```

TABLE I
CS-MDPC PARAMETERS FOR NIEDERREITER ENCRYPTION (1 LAYER; $n_0 = 2$)

r	$ pk $ (bits)	d_v	t	θ_0	δ	sec
4801	2401	45	84	37	9	2^{80}
7839	3919	65	117	48	4	2^{112}
9863	4931	71	134	55	5	2^{128}
20487	10243	105	198	75	8	2^{192}
32771	16386	137	264	105	10	2^{256}

Niederreiter encryption keys become competitive with pre-quantum RSA and post-quantum, size-optimal NTRU for non-legacy security levels, namely, 2^{112} onward. We also compare the key sizes with the previous smallest code-based parameters, namely, those attainable with QC-MDPC codes. This can be shown on Table III. Besides, as we will see in Section VI, the result is still competitive with elliptic curve implementations on constrained platforms according to other relevant metrics.

VI. EXPERIMENTAL RESULTS

We assessed the effectiveness of the techniques described herein according to the metrics of ROM and RAM usage by implementing the Niederreiter cryptosystem with the proposed parameters and decoder on the PIC24FJ32GA002-I/SP (32MHz) platform in the C programming language. No assembly language optimization has been attempted.

Mapping from raw plaintext (bit sequences) and error patterns is most efficiently achieved (in processing speed, data storage and executable code size requirements) with the Sendrier technique [37]. It was natural to adopt the same technique choosing CS-MDPC codes uniformly at random.

The observed program size (i.e. the ROM requirements for the deployed system) with the compiler employed is about 5.8 KiB. Storage (RAM) requirements are about 2.2 KiB overall, including the space needed for indices, counters and runtime bookkeeping (return addresses, stack management).

TABLE II
CS-MDPC PARAMETERS FOR NIEDERREITER ENCRYPTION (2 LAYERS; $n_0 = 2$)

r	$ pk $ (bits)	d_r	t	θ_0	δ	sec
$61 \times 79 = 4819$	$31 \times 40 = 1240$	45	84	37	9	2^{80}
$47 \times 167 = 7849$	$24 \times 84 = 2016$	65	117	48	4	2^{112}
$71 \times 139 = 9869$	$36 \times 70 = 2520$	71	134	55	5	2^{128}
$103 \times 199 = 20497$	$52 \times 100 = 5200$	105	198	75	8	2^{192}
$73 \times 449 = 32777$	$37 \times 225 = 8325$	137	264	105	10	2^{256}

TABLE III
PUBLIC KEY AND CRYPTOGRAM SIZE COMPARISON (SIZES IN BITS)

CS-MDPC	RSA	NTRU	QC-MDPC	sec
2016	2048	4411	7836	2^{112}
2520	3072	4939	9856	2^{128}
5200	7680	7447	20480	2^{192}
8325	15360	11957	32768	2^{256}

By contrast, a plain implementation of the bit flipping technique would take at least 7.2 KiB for the counters alone, far above the 3.8 KiB RAM available on a PIC24FJ32GA002 microcontroller. For simplicity, we limited the experiments to 1-layer CS-MDPC codes at the 80-bit security level.

In comparison, elliptic curve ElGamal encryption at the same security level on the ATMega128L platform using the state-of-the-art RELIC library [1] demands over 31 KiB ROM and 2.1 KiB RAM.

VII. CONCLUSION

We described how to scale code-based cryptosystems to platforms with very constrained storage and processing resources. Central to our proposal is the adoption of quasi-cyclic LDPC codes coupled with a storage-efficient algorithm for key pair generation, a carefully tailored variant of hard-decision decoding, and fine-tuned parameters. The efficiency of the result is competitive with traditional cryptosystems like those based on elliptic curves.

REFERENCES

- [1] D. F. Aranha and C. P. L. Gouvêa, "Relic is an efficient library for cryptography," <http://code.google.com/p/relic-toolkit/>.
- [2] T. Aysal and K. Barner, "Sensor data cryptography in wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 273–289, 2008.
- [3] M. Baldi, M. Bianchi, and F. Chiaraluce, "Security and complexity of the McEliece cryptosystem based on QC-LDPC codes," 2012, arXiv:1109.5827.
- [4] M. Baldi, M. Bodrato, and F. Chiaraluce, "A new analysis of the McEliece cryptosystem based on QC-LDPC codes," in *International Conference on Security and Cryptography for Networks – SCN 2008*. Berlin, Heidelberg: Springer, 2008, pp. 246–262. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85855-3_17
- [5] M. Baldi and F. Chiaraluce, "Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes," in *IEEE International Symposium on Information Theory – ISIT 2007*, June 2007, pp. 2591–2595.
- [6] M. Baldi, F. Chiaraluce, and R. Garelo, "On the usage of quasi-cyclic low-density parity-check codes in the McEliece cryptosystem," in *Proceedings of the First International Conference on Communication and Electronics – ICEE 2006*, 2006, pp. 305–310.
- [7] M. Baldi, F. Chiaraluce, R. Garelo, and F. Mininni, "Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem," in *IEEE International Conference on Communications – ICC 2007*, 2007, pp. 951–956.
- [8] D. J. Bernstein, J. Buchmann, and E. Dahmen, *Post-Quantum Cryptography*. Springer, 2008.
- [9] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the McEliece cryptosystem," in *Post-Quantum Cryptography Workshop – PQCrypto 2008*, ser. Lecture Notes in Computer Science, vol. 5299. Springer, 2008, pp. 31–46, <http://www.springerlink.com/content/68v69185x478p53g>.
- [10] —, "Smaller decoding exponents: Ball-collision decoding," in *Advances in Cryptology – CRYPTO 2011*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed. Springer Berlin / Heidelberg, 2011, vol. 6841, pp. 743–760, 10.1007/978-3-642-22792-942. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22792-9_42
- [11] A. Canteaut and N. Sendrier, "Cryptanalysis of the original mcEliece cryptosystem," in *Advances in Cryptology – Asiacrypt 1998*, ser. Lecture Notes in Computer Science, vol. 1514. Gold Coast, Australia: Springer, 1998, pp. 187–199.
- [12] C.-Y. Chang, Y. T. Su, Y.-L. Chen, and Y.-C. Liu, "Check reliability based bit-flipping decoding algorithms for LDPC codes," 2010, arXiv:1001.2503.
- [13] L. Chen, J. Xu, and I. Djurdjevic, "Near-shannon-limit quasicyclic low-density parity-check codes," in *IEEE Transactions on Communications*, vol. 52, no. 7. IEEE, 2004, pp. 1038–1042.
- [14] J. Cho, J. Kim, H. Ji, and W. Sung, "VLSI implementation of a soft bit-flipping decoder for PG-LDPC codes," in *IEEE International Symposium on Circuits and Systems –ISCAS 2009*. Taipei, Taiwan: IEEE, 2009, pp. 908–911.
- [15] J. Cho, J. Kim, and W. Sung, "VLSI implementation of a high-throughput soft-bit-flipping decoder for geometric LDPC codes," *IEEE Transactions on Circuits and Systems*, vol. 57, no. 5, pp. 1083–1094, 2010.
- [16] J. Cho and W. Sung, "Adaptive threshold technique for bit-flipping decoding of low-density parity-check codes," *IEEE Communications Letters*, vol. 14, no. 9, pp. 857–859, 2010.
- [17] D. Engelbert, R. Overbeck, and A. Schmidt, "A summary of McEliece-type cryptosystems and their security," *Journal of Mathematical Cryptology*, vol. 1, pp. 151–199, 2007.
- [18] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich, "Algebraic cryptanalysis of McEliece variants with compact keys," in *Advances in Cryptology – Eurocrypt 2010*, ser. Lecture Notes in Computer Science, vol. 6110. Springer, 2010, pp. 279–298.
- [19] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Advances in Cryptology – Crypto 1999*, ser. Lecture Notes in Computer Science, vol. 1666. Springer, 1999, pp. 537–554.
- [20] R. G. Gallager, "Low density parity-check codes," Monograph, MIT Press, Cambridge, MA, USA, 1963.
- [21] F. Guo and L. Hanzo, "Reliability ratio based weighted bit-flipping decoding for LDPC codes," in *IEEE Vehicular Technology Conference – VTC 2005*. Stockholm, Sweden: IEEE, 2005, pp. 709–713.
- [22] W. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.
- [23] A. Karatsuba and Y. Ofman, "Multiplication of many-digit numbers by automatic computers," *Proceedings of the USSR Academy of Sciences*, vol. 145, pp. 293–294, 1962.
- [24] D. Kline, H. Jeongseok, S. W. McLaughlin, J. Barros, and K. Byung-Jae, "LDPC codes for the Gaussian wiretap channel," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 532–540, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5740591
- [25] M. Koschuch, J. Lechner, A. Weitzer, J. Großschädl, A. Szekely, S. Tillich, and J. Wolkerstorfer, "Hardware/software co-design of elliptic curve cryptography on an 8051 microcontroller," in *International Conference on Cryptographic Hardware and Embedded Systems – CHES 2006*, ser. Lecture Notes in Computer Science, vol. 4249. Berlin, Heidelberg: Springer, 2006, pp. 430–444.
- [26] J. S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," *IEEE Transactions on Information Theory*, vol. 34, no. 6, pp. 1354–1359, 1988.
- [27] R. Maes, D. Schellekens, and I. Verbauwhede, "A pay-per-use licensing scheme for hardware IP cores in recent SRAM-based FPGAs," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 98–108, 2012.
- [28] R. McEliece, "A public-key cryptosystem based on algebraic coding theory," The Deep Space Network Progress Report, DSN PR 42–44, pp. 114–116, 1978. [Online]. Available: <http://ipnpr.jpl.nasa.gov/progressreport2/42-44/44N.PDF>
- [29] N. Miladinovic and M. P. C. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1594–1606, 2005.

- [30] R. Misoczki, N. Sendrier, J.-P. Tillich, and P. S. L. M. Barreto, “MDPC-McEliece: New McEliece variants from moderate density parity-check codes,” *Cryptology ePrint Archive*, Report 2012/409, 2012, <http://eprint.iacr.org/2012/409>.
- [31] C. Monico, J. Rosenthal, and A. Shokrollahi, “Using low density parity check codes in the McEliece cryptosystem,” in *IEEE International Symposium on Information Theory – ISIT 2000*. Sorrento, Italy: IEEE, 2000, p. 215.
- [32] T. M. N. Ngatched, M. Bossert, A. Fahrner, and F. Takawira, “Two bit-flipping decoding algorithms for low-density parity-check codes,” *IEEE Transactions on Communications*, vol. 57, no. 3, pp. 591–596, 2009.
- [33] T. M. N. Ngatched, F. Takawira, and M. Bossert, “A modified bit-flipping decoding algorithm for low-density parity-check codes,” in *IEEE International Conference on Communications – ICC 2007*. Glasgow, Scotland: IEEE, 2007, pp. 653–658.
- [34] H. Niederreiter, “Knapsack-type cryptosystems and algebraic coding theory,” *Problems of Control and Information Theory*, vol. 15, no. 2, pp. 159–166, 1986.
- [35] P. Oliveira and J. Barros, “A network coding approach to secret key distribution,” *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 414–423, 2008.
- [36] S. Ouzan and Y. Be’ery, “Moderate-density parity-check codes,” *CoRR*, vol. abs/0911.3262, 2009.
- [37] N. Sendrier, “Encoding information into constant weight words,” in *IEEE International Symposium on Information Theory – ISIT 2005*. IEEE, 2005, pp. 435–438.
- [38] —, “Decoding one out of many,” in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science, B.-Y. Yang, Ed. Springer Berlin / Heidelberg, 2011, vol. 7071, pp. 51–67, 10.1007/978-3-642-25405-5_4. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-25405-5_4
- [39] J. Stern, “A method for finding codewords of small weight,” *Coding Theory and Applications*, vol. 388, pp. 106–133, 1989.
- [40] R. M. Tanner, “Spectral graphs for quasi-cyclic LDPC codes,” in *IEEE International Symposium on Information Theory – ISIT 2001*. Washington, DC, USA: IEEE, 2001, p. 226.
- [41] A. Thangaraj, S. Dihidar, A. R. Calderbank, S. W. McLaughlin, and J.-M. Merolla, “Applications of LDPC codes to the wiretap channel,” *IEEE Transactions on Information Theory*, vol. 53, no. 8, pp. 2933–2945, 2007.
- [42] V. G. Umaña and G. Leander, “Practical key recovery attacks on two McEliece variants,” in *International Conference on Symbolic Computation and Cryptography – SCC 2010*. Egham, UK: Springer, 2010.
- [43] M. Vogt, A. Poschmann, and C. Paar, “Cryptography is feasible on 4-bit microcontrollers – a proof of concept,” in *IEEE International Conference on RFID*, 2009, pp. 241–248.
- [44] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, “Gradient descent bit flipping algorithms for decoding LDPC codes,” 2008, arXiv:0711.0261.
- [45] X. Wu, C. Ling, M. Jiang, E. Xu, C. Zhao, and X. You, “New insights into weighted bit-flipping decoding,” *IEEE Transactions on Communications*, vol. 57, no. 8, pp. 2177–2180, 2009.
- [46] P. Zarrinkhat and A. H. Banihashemi, “Hybrid hard-decision iterative decoding of regular low-density parity-check codes,” *IEEE Communications Letters*, vol. 8, no. 4, pp. 250–252, 2004.
- [47] —, “Threshold values and convergence properties of majority-based algorithms for decoding regular low-density parity-check codes,” *IEEE Transactions on Communications*, vol. 52, no. 12, pp. 2087–2097, 2004.
- [48] X. S. Zhou, B. F. Cockburn, and S. Bates, “Improved iterative bit flipping decoding algorithms for LDPC convolutional codes,” in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing – PACRIM 2007*. Victoria, Canada: IEEE, 2007, pp. 541–544.