DE GRUYTER
OPEN

TATRA
MOUNTAINS
Mathematical Publications

DOI: 10.2478/tmmp-2014-0025
Tatra Mt. Math. Publ. **60** (2014), 57–83

OVERVIEW OF THE MCELIECE CRYPTOSYSTEM AND ITS SECURITY

Marek Repka — Pavol Zajac

ABSTRACT. McEliece cryptosystem (MECS) is one of the oldest public key cryptosystems, and the oldest PKC that is conjectured to be post-quantum secure. In this paper we survey the current state of the implementation issues and security of MECS, and its variants. In the first part we focus on general decoding problem, structural attacks, and the selection of parameters in general. We summarize the details of MECS based on irreducible binary Goppa codes, and review some of the implementation challenges for this system. Furthermore, we survey various proposals that use alternative codes for MECS, and point out some attacks on modified systems. Finally, we review notable existing implementations on low-resource platforms, and conclude with the topic of side channels in the implementations of MECS.

1. Introduction

R. J. McEliece proposed in 1978 [37] a new public key cryptosystem based on the theory of algebraic codes, now called the McEliece Cryptosystem (MECS). Unlike RSA, it was not adopted by the implementers, mainly due to large public key sizes. The interest of researchers in MECS increased with the advent of quantum computing. Unlike systems based on integer factorisation problem and discrete logarithm problem, MECS security is based on the general decoding problem which is NP hard and should resist also attackers with access to the quantum computer.

In this article we provide an overview of the MECS, in its original form, and its alternatives. We start with the overview of the original system, basic attacks based on general decoding problem, and a brief overview of structural attacks. We conclude this section with a brief summary of MECS parameter selection.

2010 Mathematics Subject Classification: 94A60, 68P25.
Keywords: post-quantum cryptography, McEliece cryptosystem, side-channel attacks.
Support by NATOs Public Diplomacy Division in the framework of "Science for Peace", Project MD.SFPP 984520, is acknowledged.

In Section 3 we focus on CCA2-security conversions (i.e., a padding) that can prevent some of the attacks on basic MECS and lead to semantically secure versions of MECS. In Section 4 we provide more details for MECS based on irreducible binary Goppa codes, including an overview of the implementation options. Section 5 is an overview of MECS variants that use different codes. In Section 6 we provide a brief overview of selected MECS implementations on platforms with limited computational resources. Finally, in Section 7 we provide an overview of known side-channel attacks on MECS implementations.

2. McEliece cryptosystem and its security

Let G be a $k \times n$ generator matrix of a code $\mathcal{C}$, for which there is an efficient algorithm $Dec_{\mathcal{C}}$ that can decode any codeword with up to t errors. Let S be a random non-singular $k \times k$ matrix, and let P be a random $n \times n$ permutation matrix.

(Generalized) McEliece cryptosystem (MECS) is defined as follows:

Secret key: $(Dec_{\mathcal{C}}, S, P)$

Public key: $G' = S \cdot G \cdot P$

Encryption: Let m be a k-bit message, and let e be an random n-bit vector with $w_H(e) \leq t$. Then $c = m \cdot G' + e$ is a ciphertext.

Decryption: Decryption is given by the following algorithm:

1: $c' \leftarrow c \cdot P^{-1}$
2: $m' \leftarrow Dec_{\mathcal{C}}(c')$
3: $m \leftarrow m' \cdot S^{-1}$

Public key of MECS is a generator matrix G' of some linear code $\mathcal{C}'$, which is permutation equivalent to $\mathcal{C}$. MECS hides a message by transmitting a codeword of $\mathcal{C}'$ given by $mG' = m \cdot S \cdot G \cdot P$, further masked by an artificial error vector e. The weight of the error vector is not changed under codeword permutation. Thus the legitimate recipient can reverse the secret permutation P and decode the resulting codeword of $\mathcal{C}$. Finally, he can recover the original message by inverting the linear transform given by S.

2.1. Generic attacks based on information set decoding

There are two basic security assumptions for the classical McEliece cryptosystem:

(1) It is difficult to find a codeword of an arbitrary linear code with minimum distance to a given vector (general decoding problem). This problem is known to be NP-hard [4].

(2) There is no algorithm that can decode codewords in $\mathcal{C}'$ (without the knowledge of secret parameters) more efficiently than in an arbitrary linear code.

The second assumption does not seem to hold in general for an arbitrary choice of $\mathcal{C}$ (see Section 5). The original version of MECS [37] is based on irreducible binary Goppa codes [28] (see Section 4), for which the assumption 2 seems to hold. We remark that these two assumptions are not sufficient for a secure cryptosystem, see Section 3 for more details.

Although general decoding problem is NP hard, a careful choice of system parameters (n, k, t) is required to ensure that the (estimated) security level of MECS is high enough. The security level of MECS is estimated by the efficiency of the so-called (generalized) *information-set decoding* (ISD) attack. There are many variants of the ISD attack.

The basic form of the attack was already introduced by McEliece in [37]: Select k out of n coordinates of c in such a way, that no errors affect the selected coordinates. This can be done with probability roughly $(1 - t/n)^k$. The message can then be found by simple linear algebra (by solving k linear equations in k unknowns). Estimated running time of the attack is then $k^3(1 - t/n)^{-k}$. Lee and Brickell in [34] observed that k selected coordinates do not need to be error free. If the number of errors in the selected k coordinates is limited to some small bound j, a more efficient general decoding algorithm can be implemented. The work factor depends on j. For $j = 0$, a classical ISD attack is obtained. On the other hand, if we increase bound j to t, we get a brute force search through all error vectors with complexity $O(n^t)$. There are various improved versions of the algorithm using probabilistic selection of error positions [13], [14], [35], [68]. The selection of optimal j, as well as the estimate of the work factor depends on the actual version of the algorithm.

Stern's attack [58] uses a slightly different formulation of the problem. The ciphertext $c = mG' + e$ is at distance t from codewords of $\mathcal{C}'$. However, the minimal distance of $\mathcal{C}'$ is at least $2t + 1$, as it is a permutation equivalent to $\mathcal{C}$, which can correct up to t errors. Let us construct a new matrix

$$A = \begin{pmatrix} G' \\ c \end{pmatrix}.$$

The new code $\mathcal{A}$ generated by A has minimum distance t, and the only vector with weight t in $\mathcal{A}$ is the unknown error vector e (which is required to decode a hidden message m). The goal of the attack is to find the shortest vector in $\mathcal{A}$, with known weight t. The Stern's attack works with parity-check matrix H of the code. If the attacker can find a suitable set of t columns of H that sums to 0, he can immediately reconstruct e. The suitable set is found by normalizing H, precomputing sums of groups of j columns of H (corresponding to j in previous ISD attack in complexity analysis), and looking for collisions. The situation is illustrated in Figure 1. If no suitable collision is found, columns of H are permuted, and the attack is repeated. The method of Bernstein et al. [6]

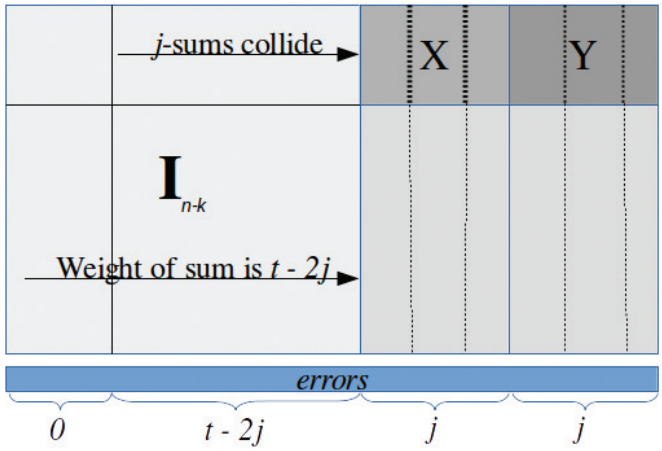


FIGURE 1. Illustration of Stern's attack.

uses more partitions of H, and more advanced methods to permute H and process collisions and vector sums.

The complexity of decoding general linear codes depends on all three code parameters $(n, k, d=2t+1)$. It is conjectured that the best generic decoding algorithm takes exponential time for any constant asymptotic code rate $R=k/n$ and constant asymptotic relative distance $D=d/n$. I.e., time complexity can be expressed as $2^{(\alpha(R,D)+o(1))n}$ for some positive real number $\alpha(R,D)$ as $n \to \inf$ [7]. With relative distance $D=0.04$ and rate $R=0.7577$ typical for McEliece cryptosystem, Stern's attack complexity can be upper bounded by $\alpha = 0.0809$ [3]. Recently, a series of new attacks: Ball decoding by Bernstein, Lange and Peters [7], algorithm based on subset sum problem by May, Meurer and Thomae [36], and its improvement by Becker, Joux et al. [3], have decreased this constant to $\alpha = 0.0672$. The non-asymptotic estimates of the ISD complexities for typical McEliece cryptosystem parameters are summarized in [30].

We remark that the complexity of ISD is lower, when the attacker only needs to decode one out of many cryptograms. An analysis by Sendrier [53] shows that ISD attacks can be speeded up by almost $\sqrt{N}$ using N instances of the problem. When the attacker has access to an unlimited number of instances, the complexity exponent is multiplied by value only slightly larger than 2/3. This type of attack scenario is critical for various compact versions of MECS, especially when considering quasi-cyclic variants (see Section 5) where any blockwise cyclic shift of the syndrome provides a proper new instance of the decoding problem.

2.2. Attacks on the secret key

Information set decoding represents the most efficient generic method to attack MECS via the general decoding problem. ISD methods try to decode a single encrypted message, and the attacker learns no information about the secret key.

Other types of attacks (so called structural attacks) target the code structure with the goal of reconstructing the secret key from a given public key. In MECS, the secret code with efficient decoding algorithm, and the public code are permutation equivalent, i.e., their codewords are just bit-wise permuted (by a secret permutation P). If the attacker knows the secret code, the permutation between two permutation equivalent linear codes can be reconstructed by the polynomial time support splitting algorithm [51]. The support splitting algorithm can be also used, if the secret code is taken from only a small set of admissible codes. E.g., in a traditional MECS based on Goppa codes (see section 4), the decoding algorithm is defined by the (secret) Goppa polynomial. If the set of possible Goppa polynomials is restricted to a set of irreducible polynomials with binary coefficients, the key space is too small to resist the attack based on enumerating potential binary Goppa polynomials and verifying the guess by the support splitting algorithm. However, if the coefficients of the Goppa polynomial are taken from a larger field, the keyspace becomes too large for an efficient support splitting attack.

Another type of key recovery is based on algebraic cryptanalysis [24], [67]. The problem of secret key recovery is rewritten as a system of polynomial equations parametrized by the public key. The solution of the system yields the unknown secret key. Algebraic methods can be used to break proposed McEliece variants with compact public keys [23]. Moreover, algebraic methods allow to distinguish generator matrices of high rate binary Goppa codes[1] from a generator matrix of a random linear code [21].

2.3. Recommended parameters

Original parameters $n = 1024$, $k = 524$, $t = 50$, suggested by McEliece in [37] are now considered insecure, as they only offer approximately 50-bit security [7]. We are not aware of any clear consensus on recommended parameters for MECS for various typical security levels. There have been various theoretical articles suggesting and analysing security of MECS parameters with respect to ISD complexity [6], [7], [30], [40]. Other sets of parameters can be found in implementation papers such as [12], [19]. The parameter selection for MECS is more complicated than for discrete log based systems, due to more degrees of freedom and various implementation constraints.

Although the complexity of ISD attacks depends only on public key, which can be considered a general linear code, parameter selection must take into

[1]These codes are used in code-based signature scheme CFS [16].

TABLE 1. Recommended (code-agnostic) parameter sizes for MECS.

Sec. Level	Ref.	(n, k, t)	PK size [kB]	
			Full PK	Systematic
50	[37]	(1024,524,50)	66	32
80	[6]	(2048,1751,27)	438	64
80	[40]	(1702,1219,45)	254	72
80	[12]	(2048,1696,32)	424	73
128	[7]	(3178,2384,68)	925	232
128	[12]	(4096,3604,41)	1802	217
256	[7]	(6944,5208,136)	4415	1104

account also the structure of the code used by the system. For MECS based on binary irreducible Goppa codes (see Section 4), the code parameters include underlying field size m. Typical implementation then selects t in such a way that the code rate based on $k = n - mt$ is the most secure choice for given $n = 2^m$. Alternative codes (see Section 5) might impose additional parameters for the system (i.e., the characteristic of the underlying field,...), or constraints (e.g., for cyclic codes the generator matrix can only have a specific size).

In Table 1 we summarize selected parameter recommendations for typical cryptosystem security levels (80, 128, and 256 bits). For each parameter selection we compute the size of the full public key as $k \cdot n$, and of the public key in systematic form as $k \cdot (n - k)$. Systematic public key can only be used with a proper CCA2 conversion of MECS, see Section 3. The parameters are typical for classical MECS based on irreducible binary Goppa codes. We do not include specific parameter selections and public key sizes for alternative codes, as the research in that area is very active with many new attacks invalidating prior security claims (see Section 5).

3. CCA2-secure versions of McEliece cryptosystem

The classical McEliece cryptosystem is not cryptographically secure (in a theoretical sense), even if we choose correct parameters. There exist various attacks that exploit the specifics of the encryption algorithm, rather than the underlying decoding problem. To prevent these attacks, MECS must be used with a proper CCA2 conversion (a more general form of padding). In ideal case, a proper CCA2-conversion transforms the original message (plaintext) into a random string of bits (cleartext), which is then encrypted with the classical MECS.

Once the recipient decodes the whole cleartext, he can decrypt the original message, as well as verify the integrity of the ciphertext. If a proper CCA2 conversion is used, the public key matrix can be stored in a systematic form.

3.1. Attacks on classical MECS without CCA2-conversion

Using the partially known plaintext attack [14], the attacker can reduce the code dimension, and thus the complexity of the ISD. Let us suppose that the attacker knows l bits of a message. Without the loss of generality, the message can be written as $m = (m_1, \ldots, m_l, m_{l+1}, \ldots, m_k)$, where the first part is known by the attacker. The encryption can be written as

$$c = (m_1, \ldots, m_l, m_{l+1}, \ldots, m_k) \cdot \begin{pmatrix} g_1 \\ \vdots \\ g_l \\ g_{l+1} \\ \vdots \\ g_k \end{pmatrix} + e$$

$$= (m_1, \ldots, m_l) \cdot \begin{pmatrix} g_1 \\ \vdots \\ g_l \end{pmatrix} + (m_{l+1}, \ldots, m_k) \cdot \begin{pmatrix} g_{l+1} \\ \vdots \\ g_k \end{pmatrix} + e = c_1 + c_2 + e.$$

The attacker can compute c_1, and obtain $c' = c_2 + e = c + c_1$. Then the attacker needs to decode c' (instead of c), but now c' can be decoded in a code with smaller dimension $k - l$.

MECS is also vulnerable to resend attack, and the attack with related messages [10]. If the attacker knows the difference between two plaintexts $\Delta m = m_1 + m_2$, he can easily compute

$$c_1 + c_2 = m_1 G' + e_1 + m_2 G' + e_2 = \Delta m G' + e_1 + e_2.$$

From this equation he can compute $\Delta e = e_1 + e_2$. The knowledge of Δe is usually enough to mount a very fast ISD attack on both messages.

Reaction attack [29] is a weaker version of the adaptive chosen ciphertext attack. Attacker sends an intercepted ciphertext with an added error e' (usually a single bit), and observes the reaction of the recipient. The resulting ciphertext can be written as $c' = mG' + e + e'$. If the recipient indicates an error in the decryption, then $w_H(e + e') > t$, and the attacker tries a different e'. Otherwise $w_H(e + e') \leq t$, and if the attacker knows that $w_H(e) = t$, he can immediately discern an error position indicated by e'. The attack continues until all error positions are known, or until a fast ISD attack is feasible.

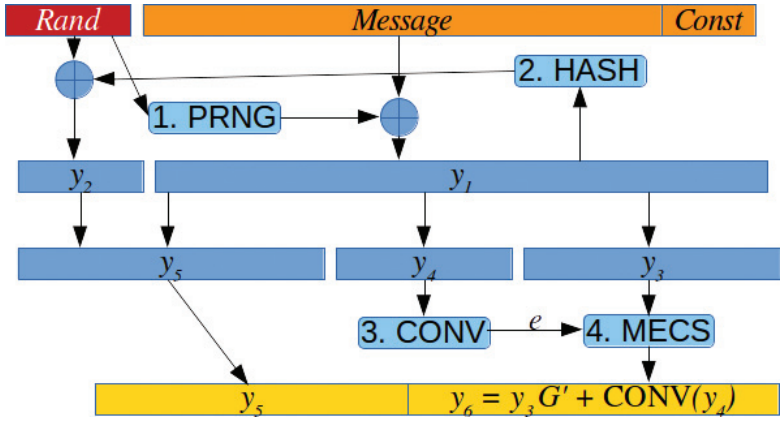


FIGURE 2. An illustration of the Kobara-Imai CCA-2 secure conversion γ of MECS.

Classical MECS does not provide non-malleability [66], as the adversary can modify a ciphertext c without knowing the original plaintext m. Suppose that the desired change of plaintext is Δm. The adversary sends $c' = \Delta m G' + c$, which is decrypted to $m + \Delta m$ by the recipient of the message.

3.2. CCA2-conversions for McEliece cryptosystem

In general, it is easy for the adversary to distinguish between encryptions of two messages m_1, m_2. The adversary computes $e_1 = m_1 G' + c$, and $e_2 = m_2 G' + c$, and outputs m_1 if $w_H(e_1) \leq t$, and m_2 if $w_H(e_2) \leq t$. Nojima et al. [42] showed that a randomized McEliece, where message is padded by a random string, is IND-CPA secure if the LPN (learning parity with noise) problem is hard. However, in practice, the random padding must have a sufficient length, as the attacker needs to decrypt only the random padding part (a partially known plaintext attack) to test whether c is the encryption of the message m.

To prevent more advanced attacks, various practical CCA2-secure versions of McEliece cryptosystem were proposed [33], [44]. There are also theoretical proposals for CCA2-secure code-based cryptosystem in a standard model [18], [47].

Kobara and Imai [33] point out two generic schemes that can be used with MECS to provide a CCA2-secure cryptosystem: Pointcheval's generic scheme [49], and Fujisaki-Okamoto scheme [25]. Unfortunately, these generic conversions introduce relatively high data redundancy (at least n bits). Kobara and Imai [33] thus propose 2 versions of MECS-specific conversions, that require only additional $(n - k + s)$ bits (where s is relative to the desired security level). Moreover, their third conversion utilizes also the entropy in the error vector, to further reduce the data overhead.

K o b a r a' s and I m a i' s third and the most efficient conversion is depicted in Figure 2. In the Figure:

- *Rand* is a random session key (with size given by the required security level for symmetric encryption).
- *PRNG* is a cryptographically secure pseudorandom generator (can be a symmetric encryption algorithm in practice).
- *Const* is a public constant (with size given by the required security level for authentication).
- *HASH* is a cryptographically secure hash function.
- *CONV* is a conversion function that (reversibly) computes an error vector e (size n string with exactly t ones) from a given bitstring.
- *MECS* is a standard MECS routine (with e as an additional input).

To decrypt the message, the legitimate recipient reconstructs y_3 and y_4 from y_6 using a classical MECS decryption routine (and the inverse function to *CONV*). Then he can reconstruct the session key and decrypt the message (using symmetric cryptography). It is possible to check the integrity of ciphertext by verifying the decryption of the public constant *Const*. Data overhead is increased by the size of *Rand*, and *Const*, respectively, but it is also decreased by the size of y_4, which is $\lfloor \log_2 \binom{n}{t} \rfloor$.

The cleartext of the CCA-2 secured version of MECS that is to be encrypted by the public key should be essentially a random string. In that case, a public key matrix can be stored in a systematic form[2] [6]. More precisely, O v e r b e c k and S e n d r i e r in [45] state that if the plaintext m is uniformly distributed, both versions (systematic and non-systematic generator matrix, respectively) are equally secure. Unfortunately, for some of the schemes that are proposed to be CCA2 secure, a public key stored in a systematic form can lead to weakened security for a specific choice of parameters [70]. E.g., in [56], a CCA2 conversion (a slightly different version of Pointcheval's scheme) is used in the following form:

(1) Plaintext for original MECS is $\tilde{m} = r_1 || hash(m||r_2)$, where r_1 has random $(k-l)$ bits (and the output of hash function can be considered undistinguishable from random l-bit string).

(2) Ciphertext is $(c_1, c_2, c_3) = (\tilde{m}G' + e, hash(r_1) + m, hash(e) + r)$.

If G' is in a systematic form, and m distinguishable from a random string, the attacker can extract r_1 part. This bit-string is "corrupted" with some of the errors from e (approximately $t' = t(k-l)/n$ errors), so the attacker knows some $r'_1 = r_1 + e_1$. Now the attacker can try to locate t' errors in $(k-l)$ positions, and verify their location by checking whether $c_2 + hash(r'_1 + e_1)$ is distinguishable

[2]If a systematic key matrix is used with a classical MECS, a low entropy plaintext can be reconstructed from the corresponding part of the ciphertext.

from random string. If the parameters (including l) are not chosen correctly, this attack can be more efficient than trying to decode whole $\tilde{m}$. A similar remark holds for the (proposed) CCA2-secure scheme presented in [44], and used in [59]. The technical details of the attack are presented in the Appendix A (and also in electronic archive [70]).

This type of attack does not apply to the original CCA2-secure schemes proposed by K o b a r a and I m a i [33], where the attacker needs to decrypt the whole MECS plaintext (and/or error vector) to learn anything, not just some part of it. If a MECS based scheme proposes to use a public key with a systematic matrix, it might be prudent to evaluate its security separately from a non-systematic case.

4. McEliece cryptosystem based on irreducible Goppa codes

The original M c E l i e c e proposal [37] is based on binary irreducible G o p p a codes [28], which can be efficiently decoded by the P a t t e r s o n algorithm [46] (among others). In this section we provide an overview of the basic building blocks of the MECS implementation (including Patterson algorithm), without delving into the deeper mathematical details.

Let $K = GF(2^m)$ be a finite field. Let $g \in K[x]$ be an irreducible polynomial of degree t. Let $L = \{\gamma_1, \gamma_2, \ldots, \gamma_n\} \subset K$.

The binary Goppa code $\mathcal{C}$ with Goppa polynomial g and support L is defined as a set of binary vectors $c \in GF(2)^n$ (indexed by elements of L), for which

$$\sum_{\gamma \in L} \frac{c_\gamma}{x - \gamma} \equiv 0 \mod g(x). \tag{1}$$

Code $\mathcal{C}$ has dimension $k \geq n - mt$, code distance at least $2t + 1$. An efficient algorithm to decode at most t errors was given by P a t t e r s o n in [46].

Let $c \in \mathcal{C}$ be a codeword, and let $r = c + e$ be a received word having at most t errors ($w_H(e) \leq t$). Let us define syndrome (polynomial) S as

$$S(x) \equiv \sum_{\gamma \in L} \frac{r_\gamma}{x - \gamma} \equiv \sum_{\gamma \in L} \frac{c_\gamma}{x - \gamma} + \sum_{\gamma \in L} \frac{e_\gamma}{x - \gamma} \equiv \sum_{\gamma \in L} \frac{e_\gamma}{x - \gamma} \mod g(x). \tag{2}$$

Furthermore, let us define an error locator polynomial σ as

$$\sigma(x) = \prod_{\gamma \in L, e_\gamma \neq 0} (x - \gamma).$$

For irreducible binary Goppa codes, we can compute $\sigma(x)$ from $S(x)$. The algorithm can be summarized in the following steps:

1: Determine the syndrome polynomial $S(x)$
2: **if** $S(x) = x$ **then return** $\sigma(x) \leftarrow x$
3: Compute $\tau(x) \leftarrow \sqrt{S(x) + x} \mod g(x)$
4: Find $a, b \in K[x]$, such that $a(x) = b(x)\tau(x) \mod g(x)$, and $\deg a \leq \lfloor t/2 \rfloor$
5: **return** $\sigma(x) \leftarrow a^2(x) + xb^2(x)$

The errors are found as roots of the error locator polynomial, i.e., $e_\gamma = 1$ if and only if $\sigma(\gamma) = 0$.

4.1. Implementation options

To implement MECS, it is necessary to at least provide a key generation, encryption and decryption algorithms, with optional CCA2 conversions (and paddings) and bit-string to constant-weight vector encoding/decoding routine.

The key generation routine provides a pair of public and secret key for a given set of parameters (e.g., code parameters (n, k, t)), along with additional precomputed values that can speed up the decryption routine.

When using classical binary irreducible Goppa codes, additional parameter m controls the size of the underlying finite field $K = GF(2^m)$, as well as the relation between (n, k, t). Typical implementation choice is that the support set L is the whole finite field K. With this choice $n = 2^m$, and $k = n - mt$, respectively. The support set L can be public [20], but Bernstein et al. in [8], [9] argue that the support set should be kept secret as an additional security measure against support splitting attacks [51].

The public key consists of the $k \times n$ public matrix $\hat{G}$. In some instances (if a correct CCA2-conversion is used), a public key can be converted to a systematic form, i.e., $\hat{G} = (I_k||R)$, where I_k is a $k \times k$ identity matrix. In this case, the public key size is reduced to $k \times (n - k)$ bits. If the size of the public key is too large to store in the encryption device, it can be retransmitted for each encryption [63]. However, this solution has a significant negative impact on the transmission rate and the speed of the encryption.

The private key must contain enough information for the decryption routine. In a classical general MECS, matrices S, P must be stored, along with the parameters of the decoding algorithm. In typical implementations based on Goppa codes, the parity check matrix and the Goppa polynomial are stored. It is essential to keep the Goppa polynomial secret, and chosen from a large set of polynomials. If the set of the possible Goppa polynomials is not large enough (e.g., if they are chosen only from $GF(2)[x]$), the support splitting attack [51] can be used to identify g, and to reconstruct the whole private key. Precomputed tables for fast arithmetic in $GF(2^m)$, and precomputed matrix for fast square root computation in $GF\big(2^m\big)/\big(g(x)\big)$ can also be stored as a part of the secret key.

The total information required for a private key storage is comparable to a public key size. However, in practice, it can be prohibitively expensive to provide enough secure memory for the private key and precomputed parameters. The effect of the secret matrix P is just permuting the support of the Goppa code. Thus, it is possible to store a secret key as a pair (g, L), where g is a secret Goppa polynomial, and L is a secret support (in a secret order), and compute the syndrome from equation (2) [63]. To further reduce the secret key, the secret parameters can be stored in a form of a seed for an appropriate fast pseudo-random generator and computed on the fly (similar to [19]). However, all these reductions in secret-key memory storage have an adverse effect on the speed of the decryption process. Thus, a suitable balance between the speed of the decryption and the cost of the decryption device must be found.

Encryption routine is a relatively straightforward matrix multiplication $m\hat{G}$. The public matrix $\hat{G}$ can be stored either in a row orientation, or in a column orientation. The row orientation can lead to a more efficient matrix multiplication, as each bit m_i just denote whether row $\hat{g}_i$ is added to an accumulator. Thus, the addition of rows corresponding to $m_i = 0$ can be skipped. This can, however, lead to some forms of side-channel attacks (if the algorithm is without any further protection).

The decryption process is more complicated, and provides a large variety of implementation options. Unfortunately, the complexity of the algorithm leads also to a variety of potential side-channel attacks (see the overview in Section 7). Typically, the extended Euclidean algorithm is used to compute inversions mod $g(x)$, as well as polynomials a, b. Square root computation is a linear operation in $K/\big(g(x)\big)$, and can be implemented as a matrix multiplication (with precomputed matrix). Typically, the most time consuming part of the decryption is to identify the errors using error locator polynomial σ. A standard procedure evaluates σ in each element of the support, but there are various alternative root finding procedures [11], [62].

4.2. List decoding

B e r n s t e i n in [5] introduced a list-decoding algorithm for irreducible binary Goppa codes. This algorithm is an extension of Patterson's algorithm that allows the receiver to efficiently decode approximately $n\sqrt{n(n-2t-2)}$ errors (more than t in a typical setting). On the other hand, the list decoding algorithm can return more than one codeword within the specified distance (uniqueness is guaranteed only up to distance t). If MECS is implemented with a proper CCA2 conversion, the recipient can use integrity check to discard incorrect codewords. Thus, when using MECS with the list decoding algorithm, more than t errors can be introduced during the encryption to increase the complexity of the decoding attacks.

Recently, G l i g o r o s k i et al. proposed an alternative MECS based on list general decoding principles [27]. Instead of decoding any words within a fixed Hamming distance of the codeword, errors are restricted to specific error sets, and efficient list decoding algorithm is used for the underlying secret code. An interesting property of the proposed system is the straightforward algorithm for digital signatures.

5. Alternative codes for McEliece cryptosystem

MECS is a very efficient cryptosystem with a fast encryption and decryption routine, but its public key is prohibitively large for many applications. There have been many proposed solutions to decrease the key size by adopting alternative codes with specific symmetries, or other properties that allow shorter public keys. However, many of the proposals were quickly broken, and the research area is still very active with many open questions. Thus, in this section we provide just a short overview of the topic, but do not go into details. We focus mainly on code alternatives that remain unbroken (as far as we know).

Goppa codes are a subfamily of Alternant codes, which are a subfield subcodes of Generalised Reed-Solomon (GRS) codes. The use of GRS codes was proposed by N i e d e r r e i t e r [41] in his code-based cryptosystem. Niederreiter cryptosystem uses public parity-check matrix instead of generator matrix, and a message is encoded as an error vector, while a ciphertext is a syndrome. S i d e l n i k o v and S h e s t a k o v in [57] have shown that hidden structure of GRS codes can be reconstructed in polynomial time $O(n^4)$. This result applies to McEliece cryptosystem based on GRS codes as well. However, Niederreiter cryptosystem based on irreducible Goppa codes should have an equivalent security to McEliece cryptosystem with the same parameters.

There has been a large number of proposals that use compact representation via subclasses of Alternant or Goppa codes with non-trivial automorphisms. The automorphisms are used to build an efficient representation of the public key. However, such reductions are susceptible to algebraic key recovery attacks by using code folding techniques [22]. For a classical McEliece cryptosystem based on irreducible binary Goppa codes, the key recovery attacks are (believed to be) always more difficult than ISD attacks on ciphertext. However, for compact codes, the key recovery complexity is based on the parameters of the folded (compact) code, instead of the full code. Thus, the key recovery attack on some compact code variants of MECS can be much more efficient than ISD attack [23].

Non-binary variants of Goppa codes have been proposed by B e r n s t e i n et al. in [8], [9]. Instead of using irreducible Goppa polynomial over $\mathbb{F}_{2^m}$, Goppa code is defined over $\mathbb{F}_q$ by polynomials in the form fg^{q-1} ([9] version), where f, g

are coprime monic squarefree polynomials from some F_{q^r}. Goppa code defined in this way has dimension at least $nm\big(s + (q1)t\big)$ and can efficiently decode $\lfloor (s + qt)/2 \rfloor$ errors (using a specific decoder for Alternant codes). A larger base field allows to use a shorter public key for the same security level (with respect to generalized ISD attacks) [48]. We note that a care must be taken in choosing large enough extension degree r to avoid structural attacks [17].

Many attacks on the various proposals of MECS based on compact codes are possible due to some algebraic properties of the underlying code. M o n i c o et al. in [39] proposed to use Low Density Parity Check (LDPC) code [26] in McEliece cryptosystem. LDPC codes use parity check matrix with low row and column weight, without any specific algebraic structure. However, it is not sufficient to mask the hidden LDPC code by permutation matrix like in the original MECS, because the attacker can find low-weight codewords in the dual of the public code $\mathcal{C}'$, and reconstruct the low-weight parity check matrix that decodes ciphertexts.

B a l d i et al. in [2] proposed an improved variant of the LDPC-based MECS. The efficiency of the system is improved by using quasi-cyclic LDPC code. This code is given by a secret parity check matrix in the block form

$$H = (H_1 H_2 \dots H_{n_0}),$$

where H_i are $p \times p$ circulant matrices with row/column weight d_v (and H_{n_0} should be non-singular). Instead of a permutation matrix P, a sparse $n \times n$ non-singular matrix Q is used. If the LDPC code that is used in the system can correct t errors, and the weight of Q is at most m, the error vector added to the encrypted message can have weight at most $\lfloor t/m \rfloor$. On the other hand, scrambling matrix S must be dense to prevent attacks by O t m a n i et al. [43]. To preserve the quasi-cyclic structure, both S, Q have a quasicyclic structure, i.e., they consist of $p \times p$ circulant block matrices. The final public key is thus given as

$$G' = S^{-1} \cdot G \cdot Q^{-1},$$

where

$$G = \left(\begin{array}{c|c} I_{p(n_0-1)} & \begin{array}{c} (H_{n_0}^{-} 1 H_1)^T \\ (H_{n_0}^{-} 1 H_2)^T \\ \vdots \\ (H_{n_0}^{-} 1 H_{n_0-1})^T \end{array} \end{array} \right).$$

The final secret code has dimensions $n = pn_0$, $k = p(n_0 - 1)$, and at most $t' = \lfloor t/m \rfloor$ errors can be introduced during the encryption. For 80-bit security, B a l d i et al. [2] recommend parameters $n_0 = 3$, $d_v = 13$, and $p = 8192$. The proposed QC-LDPC code should be able to correct up to more than 470 errors per frame, which leads to parameters $t' = 40$ and $m = 11$. The public key

for the system can be stored in approximately $6kB$ (10-times less than a classical MECS based on irreducible Goppa codes with 80-bit security).

Even more efficient version of compact MECS (that is still considered secure) was proposed by M i s o c z k i et al. in [38]. This version is based on the so-called MDPC codes. MDPC codes are similar to LDPC codes, but their parity check matrix has "moderate" row weight w which scales with $O(\sqrt{n \log n})$. Such codes can correct significantly less errors then LDPC codes, but there is no need to use a special matrix Q to mask the low-weight codewords in the dual code. A quasi--cyclic variant (QC-MDPC) is again based on a secret parity check matrix in the block form

$$H = (H_1 H_2 \cdots H_{n_0}),$$

where each H_i is a $p \times p$ circulant matrix. The public key is just a systematic form of generator matrix

$$G' = \left(\begin{array}{c|c} I_{p(n_0-1)} & \begin{array}{c} (H_{n_0}^{-1} H_1)^T \\ (H_{n_0}^{-1} H_2)^T \\ \vdots \\ (H_{n_0}^{-1} H_{n_0-1})^T \end{array} \end{array} \right).$$

Code dimensions are thus $n = n_0 p$ and $k = (n_0 - 1)p$. A system with a systematic public matrix requires a proper CCA2 conversion, but it means that the public key is just k-bit long. Higher n_0 means longer public key, but better code rate as $k/n = 1 - 1/n_0$.

Various decoding algorithms for the LDPC/MDPC codes are available [26]. A fast bit-flipping algorithm was proposed to be used with MDPC codes in MECS implementations [38]. We remark that bit-flipping algorithm is stochastic and can lead to a decoding failure. A special treatment is required for these states, either by reducing the number of errors introduced during the encryption, by using more sophisticated (but slower) decoding algorithms, or using reencryption (CCA2 conversion prevents resend attacks). To keep the complexity of message recovery and key recovery attacks of the same order, w should be chosen to be of the form

$$w = \big(1 + o(1)\big) \sqrt{\frac{n \ln n \ln(1 - k/n)}{\ln(k/n)}}.$$

The bit-flipping algorithm then can correct a number of errors of the order

$$t = \big(1 + o(1)\big) \frac{n}{4w} \ln\big(w \cdot (1 - k/n)\big).$$

The recommended parameters for 80-bit security with code rate 1/2 are $n_0 = 2$, $p = 4800$, $w = 90$, $t = 84$ [38], with only $600B$ public key (only 4.7-times longer than RSA-1024 public key).

6. MECS on embedded and hardware platform

These is a large number of experimental implementations of MECS on embedded, hardware, and on software platforms. A good overview of the existing implementations can be found in [50]. In this section, we summarize selected contributions to demonstrate the progress in the implementation area, and challenges faced by the implementers.

The use of MECS on embedded devices is restricted due to the (relatively) large key structures. The first notable microprocessor based MECS implementation was published by E i s e n b a r t h et al. [19]. They implemented an instance of MECS with $n = 2048, t = 27$ (for estimated 80-bit security) on an 8-bit AVR microprocessor, and on a Xilinx Spartan-3AN FPGA, respectively. An external memory is required for public key structures, while private key structures (parity check matrix) were stored in 192 kB Flash memory. Scrambling matrix S is precomputed on the fly from the seed value.

FPGA implementations of MECS typically need to overcome the problem with large memory requirements. MECS processor architecture for a Virtex-5 FPGA was published by S h o u f a n et al. in [56]. The published architecture involves key generator, encryptor, as well as decryptor. Their implementation of MECS with $n = 2048, t = 50$ spent 84 % of slices and 50 % of BRAMs (2700 Kb).

S t r e n z k e in [60] describes an implementation of MECS on a smart card with an Infineon SLE 76 chip. Strenzke tested two instances of MECS based on irreducible Goppa codes with systematic public key: $m = 10$, $n = 1024$, $t = 40$ errors (62-bit security, 93kB key structures), and $m = 11, n = 2048, t = 50$ (102-bit security, 257 kB key structures). Again, a non-volatile memory is used for storing key structures. Due to the complicated write access to non-volatile memory, key generation routines must be implemented outside the card.

A strong potential for low cost implementations lie in adopting some of the compact versions of MECS. H e y s e et al. in [32] report an 8-bit AVR implementation of QC-MDPC McEliece variant along with a Xilinx Virtex-6 XC6VLX240T FPGA implementation. Their compact implementation uses only 4,800 and 9,600 bits for the public and secret key at 80 bits of equivalent symmetric security. The results are on par, or even better than other currently used public key systems with similar security (ECC-160, RSA-1024, see Tables 2 and 3).

7. Side-channel attacks on MECS implementations

In practical applications, the security of cryptographic primitives can be compromised by a side-channel information leaked by a concrete implementation of cryptographic algorithms. Typical MECS implementation based on irreducible

TABLE 2. MECS on embedded platform.

PKC n, t, Security bits	**Device**	**Computation time Encrypt, Decrypt**	**Ref.**
Goppa MECS $n = 1024, t = 40$, 62-bit sec.	Infineon SLE76CF5120P controller, 16-bit CPU @ 33 MHz	970ms, 690ms	[60]
Goppa MECS $n = 2048, t = 50$, 102-bit sec.	Infineon SLE76CF5120P controller, 16-bit CPU @ 33 MHz	1390ms 1060ms	[60]
Goppa MECS $n = 2048, t = 27$, 80-bit sec.	AVR ATxMega192, 8-bit CPU @ 32MHz	450ms, 618ms	[19]
QC-MDPC MECS $n = 9600, t = 84$, 80-bit sec.	AVR ATxMega256A3, 8-bit CPU @ 32MHz	800ms, 2700ms	[32]
Cryptosystem	**Device**	**Computation time**	**Ref.**
ECC-P160 (SECG)	ATMega128@8MHz	203ms (at 32MHz)	[19]
RSA-1024	ATMega128@8MHz	20748ms (at 32MHz)	[19]

TABLE 3. MECS on embedded platform.

PKC n, t, Security bits	**Device**	**Computation time Encrypt, Decrypt**	**Ref.**
Goppa MECS $n = 2048, t = 27$, 80-bit sec.	Spartan-3AN 1400 FPGA, Enc@150MHz, Dec@85Mhz	1.07ms, 10.82ms	[19]
Goppa MECS $n = 2048, t = 50$, 102-bit sec.	Xilinx Virtex-5, 163MHz	0.5ms, 1.4ms	[56]
QC-MDPC MECS $n = 9600, t = 84$, 80-bit sec.	Xilinx Virtex-6 Enc@351.3MHz, Dec@190.6MHz	0.14ms, 0.86ms	[32]
Cryptosystem	**Device**	**Computation time**	**Ref.**
ECC-P160 (SECG)	Spartan-3 1000-4	5.1ms (at 32MHz)	[19]
RSA-1024	Spartan-3E 1500-5	51ms	[19]

binary Goppa codes uses Patterson's algebraic decoding algorithm [46] in the form presented in Figure 3 (see Section 4 for more details). This algorithm is relatively complex and there are many possibilities for various side channels in its implementation.

Require: $\mathbf{u} \in \mathbb{F}_2^n$ (a private code word with t errors), $\Gamma(\mathbf{\Lambda}, g)$.
Ensure: Error vector $\mathbf{e}$ such that $\mathbf{v} = \mathbf{u} + \mathbf{e}$, where $\mathbf{v} \in \Gamma(\mathbf{\Lambda}, g)$ is the code word.

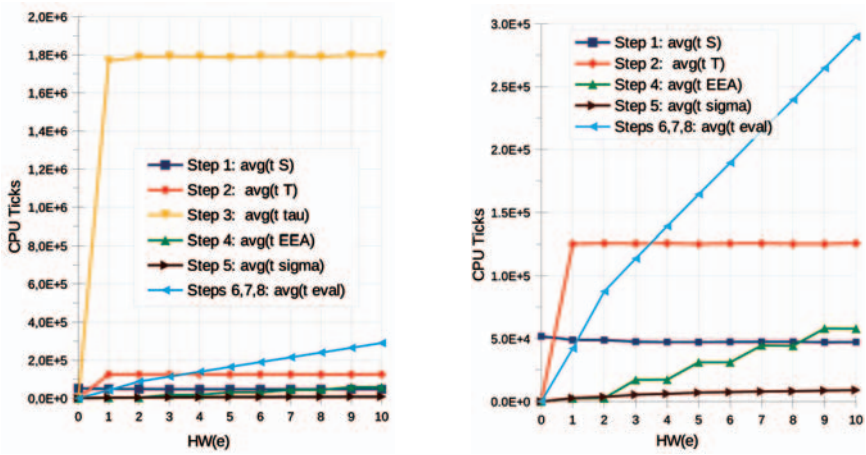
1: Compute syndrome $S(Z)$. ▷ Eq. (2).
2: $T(Z) \equiv S^{-1}(Z) + Z \bmod g(Z)$. ▷ Inversion.
3: $\tau(Z) \equiv \sqrt{T(Z)} \bmod g(Z)$. ▷ Square root.
4: Find α, β such that $\beta(Z)\tau(Z) \equiv \alpha(Z) \bmod g(Z)$. ▷ EEA.
5: $\sigma(Z) = \alpha^2(Z) + Z\beta^2(Z)$. ▷ Squaring.
6: Find roots of $\sigma(Z)$. ▷ Evaluation over $\mathbf{\Lambda}$.
7: Determine indexes of the roots in the support $\mathbf{\Lambda}$.
8: Set 1 in the determined indexes in error vector $\mathbf{e}$.
9: **return** $\mathbf{e}$.

FIGURE 3. Patterson's algebraic decoding algorithm.

The timing attack against the Patterson's algebraic decoding algorithm, first studied in S t r e n z k e et al. in 2008 [65], uses the dependence of the degree of error-locator polynomial σ on Hamming weight of the error vector $w_H(e)$. The total time of locating error position then depends on the degree of σ (in some implementations). Usually, a legitimate sender of the message uses error vectors with exactly t errors. The analysed attack is basically a reaction attack [29], when attacker introduces one or more additional errors, trying to remove some of the originally introduced errors. If the attacker guesses the error location correctly, the decryption time is slightly lower than in the case when error location was incorrect. In comparison with the original reaction attack, the timing side-channel can work even when CCA2 conversion is used. This attack was later extended in [54] to cover also timing information from different steps of Patterson's algorithm that depend on Hamming weight of error vector. A v a n z i et al. [1] extend these attacks further by using statistical profile of the decryption computation time regarding all the possible correctable error vectors of the desired Hamming weight. Thanks to this profiling phase of attack, the success of the attack has been improved significantly.

Figure 4 illustrates the dependence of computational time on Hamming weight of error vector for various steps of (schoolbook implementation of) Patterson's algorithm.

While the timing-enabled reaction attack can only recover a single message, other attacks are focused on key recovery. S t r e n z k e et al. in [65] discuss a possibility to reveal coefficients of the irreducible Goppa polynomial $g(Z)$ assuming that the code support $\mathbf{\Lambda}$ is known by analyzing the power consumption caused by the evaluation of the irreducible Goppa polynomial during the parity check matrix construction. We remark, that while originally this occurs only



(A) Top level view. (B) View without Step 3.

FIGURE 4. Computation time of Patterson's algebraic decoding algorithm with respect to $w_H(\mathbf{e})$. Measurements are averages based on 10 random instances of Goppa MECS with reduced parameters ($m = 8$, $n = 256$, $t = 10$).

in the key generation phase, some of the implementations can recompute parity check matrix for each decryption to save storage space.

Timing attacks enhanced with a specific post-processing can be used to reconstruct secret permutation [61] from the information provided by EEA step. Certain coefficients of the error-locator polynomial can be reconstructed from timing channel provided by the syndrome inversion, from which a further post-processing can reveal some information about the secret code support [64].

The first practical evaluation of a simple power analysis attack revealing the secret permutation P, and the scrambling matrix S, during the decryption process can be found in [31]. They conducted and evaluated attacks against MECS decryption algorithm implemented on the embedded device [19]. The SPA attacks are based on the ability to distinguish power traces for different instructions and Hamming weights of operands. The power trace obtained during the syndrome computation can leak information about permutation matrix P if the message is permuted before the syndrome computation, and the syndrome computation is different for ciphertext bits 0 and 1. Furthermore, if the attacker can accurately estimate the Hamming weight of the operands, he can reconstruct partial information about the parity check matrix from the syndrome computation, and about Goppa polynomial from the syndrome inversion (while the Goppa polynomial is loaded into the microprocessor). In some cases, this partial information can be used to reconstruct the whole private key after some post processing.

Classical MECS based on binary irreducible Goppa codes does not seem to be vulnerable to DPA attacks [31], as there is not enough information from syndromes to distinguish key-bit hypotheses. This however does not hold in general. Recently, a DPA attack on QC-MDPC implementation [69] was presented by C h e n et al. [15]. The attacker needs to reconstruct two vectors that define the secret sparse parity check matrix. These vectors are repeatedly rotated in a specific register during the syndrome computation. The attacker exploits the difference between power traces when a specific key bit is 0 or 1, leading to a possible full key recovery (this is implementation specific).

REFERENCES

[1] AVANZI, R.—HOERDER, S.—PAGE, D.—TUNSTALL, M.: *Side-channel attacks on the McEliece and Niederreiter public-key cryptosystems*, J. Cryptogr. Eng. **1** (2011), 271–281.

[2] BALDI, M.—BODRATO, M.—CHIARALUCE, F.: *A new analysis of the McEliece cryptosystem based on QC-LDPC codes*, in: Proc. of the 6th Internat. Conf.—SCN '08, Amalfi, Italy, 2008 (R. Ostrovsky et al., eds.), Lecture Notes in Comput. Sci., Vol. 5229, Springer, Berlin, 2008, pp. 246–262.

[3] BECKER, A.—JOUX, A.—MAY, A.—MEURER, A.: *Decoding random binary linear codes in* $2^{n/20}$*: How* $1 + 1 = 0$ *improves information set decoding,* in: Advances in Cryptology—EUROCRYPT '12, 31st Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, Cambridge, UK, 2012 (D. Pointcheval et al., eds.), Lecture Notes in Comput. Sci., Vol. 7237, Springer, Berlin, 2012, pp. 520–536.

[4] BERLEKAMP, E.—MCELIECE, R.—VAN TILBORG, H.: *On the inherent intractability of certain coding problems,* IEEE Trans. Inform. Theory **24** (1978), 384–386.

[5] BERNSTEIN, D. J.: *List decoding for binary Goppa codes,* in: Coding and Cryptology, 3rd Internat. Workshop—IWCC '11, Qingdao, China, 2011 (Y. M. Chee et al., eds.), Lecture Notes in Comput. Sci., Vol. 6639, Springer, Berlin, 2011, 62–80.

[6] BERNSTEIN, D. J.—LANGE, T.—PETERS, C.: *Attacking and defending the McEliece cryptosystem,* in: Post-Quantum Cryptography, 2nd Internat. Workshop—PQCrypto '08 (J. Buchmann et al., eds.), Cincinnati, OH, USA, 2008, Lecture Notes in Comput. Sci., Vol. 5299, Springer, Berlin, 2008, pp. 31–46.

[7] BERNSTEIN, D. J.—LANGE, T.—PETERS, C.: *Smaller decoding exponents: Ball-collision decoding,* in: Advances in Cryptology—CRYPTO '11, 31st Annual Cryptology Conf., Santa Barbara, CA, USA, 2011, (P. Rogaway, ed.), Lecture Notes in Comput. Sci., Vol. 6841, Springer, Berlin, 2011, pp. 743–760.

[8] BERNSTEIN, D. J.—LANGE, T.—PETERS, C.: *Wild McEliece,* in: Selected Areas in Cryptography, 17th Internat. Workshop—SAC '10, Waterloo, Ontario, Canada, 2010 (A. Biryukov et al., eds.), Lecture Notes in Comput. Sci., Vol. 6544, Springer, Berlin, 2011, pp. 143–158.

[9] BERNSTEIN, D. J.—LANGE, T.—PETERS, C.: *Wild McEliece incognito,* in: Post--Quantum Cryptography, 4th Internat. Workshop—PQCrypto '11, Taipei, Taiwan, 2011 (B.-Y. Yang, ed.), Lecture Notes in Comput. Sci., Vol. 7071, Springer, Berlin, 2011, 244–254.

[10] BERSON, T. A.: *Failure of the McEliece public-key cryptosystem under message-resend and related-message attack,* in: Advances in Cryptology–CRYPTO '97, 17th Annual Internat. Cryptology Conf. Santa Barbara, CA, USA, 1997 (B. S. Kaliski, jr. ed.), Lecture Notes in Comput. Sci., Vol. 1294, Springer, Berlin, 1997, pp. 213–220.

[11] BISWAS, B.—HERBERT, V.: *Efficient root finding of polynomials over fields of characteristic 2,* in: Western European Workshop on Research in Cryptology—WEWORC '09, Graz, Austria, 2009 (C. Rechberger, ed.), Lecture Notes in Comput. Sci., Vol. 6429, Springer-Verlag, Berlin, 2009.

[12] BISWAS, B.—SENDRIER, N.: *McEliece cryptosystem implementation: theory and practice,* in: Post-Quantum Cryptography, 2nd Internat. Workshop—PQCrypto '08, Cincinnati, OH, USA, 2008 (J. Buchmann et al., eds.), Lecture Notes in Comput. Sci., Vol. 5299, Springer, Berlin, 2008, pp. 47–62.

[13] CANTEAUT, A.—CHABAUD, F.: *A new algorithm for finding minimum-weight words in a linear code: application to McElieces cryptosystem and to narrow-sense BCH codes of length 511.* IEEE Trans. Inform. Theory **44** (1998), 367–378.

[14] CANTEAUT, A.—SENDRIER, N.: *Cryptanalysis of the original McEliece cryptosystem,* in: Advances in Cryptology—ASIACRYPT '98, Internat. Conf. on the Theory and Application of Cryptology and Inform. Security, Beijing, China, 1998 (K. Ohta et al., eds.), Lecture Notes in Comput. Sci., Vol. 1514, Springer, Berlin, 1998, pp. 187–199.

[15] CHEN, C.—EISENBARTH, T.—VON MAURICH, I.—STEINWANDT, R.: *Differential power analysis of a McEliece cryptosystem.* Cryptology ePrint Archive, Report 2014/534, 2014, `http://eprint.iacr.org/`.

[16] COURTOIS, N. T.—FINIASZ, M.—SENDRIER, N.: *How to achieve a mceliece-based digital signature scheme,* in: Advances in Cryptology—ASIACRYPT '01, 7th Internat. Conf. on the Theory and Appl. of Cryptology and Inform. Security, Gold Coast, Australia, 2001 (C. Boyd, ed.), Lecture Notes in Comput. Sci., Vol. 2248, Springer, Berlin, 2001, pp. 157–174.

[17] COUVREUR, A.—OTMANI, A.—TILLICH, J.-P.: *Polynomial time attack on wild McEliece over quadratic extensions,* Cryptology ePrint Archive, Report 2014/112, 2014, `http://eprint.iacr.org/`.

[18] DÖTTLING, N.—DOWSLEY, R.—MÜLLER-QUADE, J.—NASCIMENTO, A. C. A.: *A CCA2 secure variant of the McEliece cryptosystem.* IEEE Trans. Inform. Theory **58** (2012), 6672–6680.

[19] EISENBARTH, T.—GÜNEYSU, T.—HEYSE, S.—PAAR, C.: *MicroEliece : McEliece for embedded devices,* in: Cryptographic Hardware and Embedded Systems—CHES '09, 11th Internat. Workshop Lausanne, Switzerland, 2009 (Ch. Clavier et al., eds.), Lecture Notes in Comput. Sci., Vol. 5747, Springer, Berlin, 2009, pp. 49–64.

[20] ENGELBERT, D.—OVERBECK, R.—SCHMIDT, A.: *A summary of McEliece-type cryptosystems and their security,* J. Math. Cryptol. **1** (2007), 151–199.

[21] FAUGÈRE, J.-C.—GAUTHIER-UMAÑA, V.—OTMANI, A.—PERRET, L.—TILLICH, J.-P.: *A distinguisher for high-rate McEliece cryptosystems,* IEEE Trans. Inform. Theory **59** (2013), 6830–6844.

[22] FAUGÈRE, J.-C. —OTMANI, A.—PERRET, L.—DE PORTZAMPARC, L.—TILLICH: *Folding alternant and Goppa codes with non-trivial automorphism groups,* arXiv preprint arXiv:1405.5101, 2014.

[23] FAUGÈRE, J.-C.—OTMANI, A.—PERRET, L.—DE PORTZAMPARC, F.—TILLICH, J.-P.: *Structural cryptanalysis of McEliece schemes with compact keys,* Cryptology ePrint Archive, Report 2014/210, 2014, `http://eprint.iacr.org/`.

[24] FAUGÈRE, J.-C.—OTMANI, A.—PERRET, L.—TILLICH, J.-P.: *Algebraic cryptanalysis of McEliece variants with compact keys,* in: Advances in Cryptology—EUROCRYPT '10, 29th Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, French Riviera, 2010 (H. Gilbert, ed.), Lecture Notes in Comput. Sci., Vol. 6110, Springer, Berlin, 2010, pp. 279–298.

[25] FUJISAKI, E.—OKAMOTO, T.: *Secure integration of asymmetric and symmetric encryption schemes,* in: Advances in Cryptology—CRYPTO '99, 19th Annual Internat. Cryptology Conf. Santa Barbara, CA, USA, 1999 (M. Wiener, ed.), Lecture Notes in Comput. Sci., Vol. 1666, Springer, Berlin, 1999, pp. 537–554.

[26] GALLAGER, R. G.: *Low-density parity-check codes,* IRE Trans. Inform. Theory **8** (1962), 21–28.

[27] GLIGOROSKI, D.—SAMARDJISKA, S.—JACOBSEN, H.—BEZZATEEV, S.: *McEliece in the world of Escher,* Cryptology ePrint Archive, Report 2014/360, 2014, `http://eprint.iacr.org/`.

[28] GOPPA, V. D.: *A new class of linear error correcting codes,* Probl. Pered. Inform. **6** (1970), 24–30.

[29] HALL, C.—GOLDBERG, I.—SCHNEIER, B.: *Reaction attacks against several public-key cryptosystem,* in: Information and Commun. Security, 2nd Internat. Conf.—ICICS '99, Sydney, Australia, 1999 (V. Varadharajan et al., eds.), Lecture Notes in Comput. Sci., Vol. 1726, Springer, Berlin, 1999, pp. 2–12.

[30] HAMDAOUI, Y.—SENDRIER, N.: *A non asymptotic analysis of information set decoding,* IACR Cryptology ePrint Archive, 2013:162, 2013.

[31] HEYSE, S.—MORADI, A.—PAAR, C.: *Practical power analysis attacks on software implementations of McEliece,* in: Post-Quantum Cryptography, 3rd Internat. Workshop––PQCrypto '10, Darmstadt, Germany, 2010 (N. Sendrier, ed.), Lecture Notes in Comput. Sci., Vol. 6061, Springer, Berlin, 2010, pp. 108–125.

[32] HEYSE, S.—VON MAURICH, I.—GÜNEYSU, T.: *Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices,* in: Cryptographic Hardware and Embedded Systems—CHES '13 15th Internat. Workshop, Santa Barbara, CA, USA, 2013, (G. Bertoni and J.-S. Coron, eds.), Lecture Notes in Comput. Sci., Vol. 8086, Springer, Berlin, 2013, pp. 273–292.

[33] KOBARA, K.—IMAI, H.: *Semantically secure McEliece public-key cryptosystems––conversions for McEliece PKC,* in: Public Key Cryptography, 4th Internat. Workshop on Practice and Theory in Public Key Cryptosystems—PKC '01, Cheju Island, Korea, 2001 (K. Kim, ed.), Lecture Notes in Comput. Sci., Vol. 1992, Springer, Berlin, 2001, pp. 19–35.

[34] LEE, P. J.—BRICKELL, E. F.: *An observation on the security of McElieces public-key cryptosystem,* in: Advances in cryptology—EUROCRYPT 88, Workshop on the Theory and Appl. of Cryptogr. Techniques, Davos, Switzerland, 1988 (D. Barstow et al., eds.), Lecture Notes in Comput. Sci., Vol. 330, Springer, Berlin, 1988, pp. 275–280.

[35] LEON, J.: *A probabilistic algorithm for computing minimum weights of large error-correcting codes,* IEEE Trans. Inform. Theory **34** (1988), 1354–1359.

[36] MAY, A.—MEURER, A.—THOMAE, E.: *Decoding random linear codes in* $\mathcal{O}(2^{0.054n})$, in: Advances in Cryptology—ASIACRYPT '11, 7th Internat. Conf. on the Theory and Appl. of Cryptology and Inform. Security, Seoul, South Korea, 2011 (D. H. Lee and X. Wang, eds.), Lecture Notes in Computer Science, Vol. 7073, Springer, Berlin, 2011, pp. 107–124.

[37] MCELIECE, R. J.: *A public-key cryptosystem based on algebraic coding theory,* DSN Progress Report **42** (1978), 114–116.

[38] MISOCZKI, R.—TILLICH, J.-P.—SENDRIER, N.—BARRETO, P. S. L. M.: *MDPC-McEliece: New McEliece variants from moderate density parity-check codes,* in: IEEE Internat. Symposium on Information Theory—ISIT '13, Istanbul, Turkey, 2013, IEEE, 2013, pp. 2069–2073.

[39] MONICO, C.—ROSENTHAL, J.—SHOKROLLAHI, A.: *Using low density parity check codes in the McEliece cryptosystem,* in: IEEE Internat. Symposium on Information Theory, Sorrento, Italy, 2000, IEEE, 2000, p. 215.

[40] NIEBUHR, R.—MEZIANI, M.—BULYGIN, S.—BUCHMANN, J.: *Selecting parameters for secure McEliece-based cryptosystems,* Int. J. Inf. Sec. **11** (2012), 137–147.

[41] NIEDERREITER, H.: *Knapsack-type cryptosystems and algebraic coding theory,* Probl. Control Inf. Theory **15** (1986), 159–166.

[42] NOJIMA, R.—IMAI, H.—KOBARA, K.—MOROZOV, K.: *Semantic security for the McEliece cryptosystem without random oracles,* Des. Codes Cryptography **49** (2008), 289–305.

[43] OTMANI, A.—TILLICH, J.-P.—DALLOT, L.: *Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes,* Math. Comput. Sci. **3** (2010), 129–140.

[44] OVERBECK, R.: *An analysis of side channels in the McEliece PKC (2008),* `www.cosic.esat.kuleuven.be/nato_arw/slides_participants/Overbeck\ slides_nato08.pdf`.

[45] OVERBECK, R.—SENDRIER, N.: *Code-based cryptography,* in: Post-Quantum Cryptography, 1st Internat. Workshop—PQCrypto '06, Leuven, The Netherland, 2006 (D. Bernstein et al., eds.), Springer, Berlin, 2009, pp. 95–145.

[46] PATTERSON, N. J.: *The algebraic decoding of Goppa codes,* IEEE Trans. Inform. Theory **21** (1975), 203–207.

[47] PERSICHETTI, E.: *On a CCA2-secure variant of McEliece in the standard model,* IACR Cryptology ePrint Archive, 2012:268, 2012.

[48] PETERS, C.: *Information-set decoding for linear codes over F_q,* in: Post-Quantum Cryptography, 3rd Internat. Workshop—PQCrypto '10, Darmstadt, Germany, 2010 (N. Sendrier, ed.), Lecture Notes in Comput. Sci., Vol. 6061, Springer, Berlin, 2010, pp. 81–94.

[49] POINTCHEVAL, D.: *Chosen-ciphertext security for any one-way cryptosystem,* in: Public Key Cryptography—PKC '00, 3rd Internat. Workshop on Practice and Theory in Public Key Cryptosystems, Melbourne, Victoria, Australia, 2000 (H. Imai et al., eds.), Lecture Notes in Comput. Sci., Vol. 1751, Springer, Berlin, 2000, pp. 129–146.

[50] REPKA, M.—CAYREL, P.-L.: *Cryptography based on error correcting codes: a survey,* in: Multidisciplinary Perspectives in Cryptology and Information Security, IGI Global, 2014, pp. 133–156.

[51] SENDRIER, N.: *Finding the permutation between equivalent linear codes: the support splitting algorithm,* Information Theory, IEEE Transactions **46** (2000), 1193–1203.

[52] SENDRIER, N. (ED.): Post-Quantum Cryptography, 3rd International Workshop––PQCrypto '10, Darmstadt, Germany, 2010 Lecture Notes in Comput. Sci., Vol. 6061, Springer, Berlin, 2010.

[53] SENDRIER, N.: *Decoding one out of many,* in: Post-Quantum Cryptography, 4th Internat. Workshop—PQCrypto '11, Taipei, Taiwan, 2011 (B.-Y. Yang, ed.), Lecture Notes in Comput. Sci., Vol. 7071, Springer, Berlin, 2011, pp. 51–67.

[54] SHOUFAN, A.—STRENZKE, F.—MOLTER, H. G.—STÖTTINGER, M.: *A timing attack against Patterson algorithm in the McEliece PKC,* in: Proc. of the 12th Internat. Conf. on Information Security and Cryptology—ICISC '09, Seoul, Korea, 2009 (D. Lee and S. Hong, eds.), Lecture Notes in Comput. Sci., Vol. 5984, Springer, Berlin, 2010, pp. 161–175.

[55] SHOUFAN, A.—WINK, T.—MOLTER, H. G.—HUSS, S. A.—KOHNERT, E.: *A novel cryptoprocessor architecture for the McEliece public-key cryptosystem,* IEEE Trans. Comput. **59** (2010), 1533–1546.

[56] SHOUFAN, A.—WINK, T.—MOLTER, H. G.—HUSS, S. A.—STRENZKE, F.: *A novel processor architecture for McEliece cryptosystem and FPGA platforms,* in: Application-Specific Systems, Architectures and Processors—ASAP '09, 20th IEEE Internat. Conf., Boston, MA, IEEE, 2009, pp. 98–105.

[57] SIDELNIKOV, V. M.—SHESTAKOV, S. O.: *On insecurity of cryptosystems based on generalized reed-solomon codes,* Discrete Math. Appl. **2** (1992), 439–444.

[58] STERN, J.: *A method for finding codewords of small weight,* in: Coding Theory and Applications, Proc. 3rd Int. Colloq., Toulon/France, 1988, Lecture Notes in Comput. Sci., Vol. 388, Springer, Berlin, 1989, pp. 106–113.

[59] STRENZKE, F.: *A side-channel secure and flexible platform-independent implementation of the McEliece PKC–flea version 0.1. 1–,* `http://www.cryptosource.de/flea_doc.pdf`.

[60] STRENZKE, F.: *A smart card implementation of the McEliece PKC,* in: Proc. of the 4th IFIP WG 11.2 Internat. Conf. on Information Security Theory and Practices: Security and Privacy of Pervasive Systems and Smart Devices—WISTP '10, Passau, Germany, 2010 (P. Samarati et al., eds.), Lecture Notes in Comput. Sci., Vol. 6033, Springer, Berlin, 2010, pp. 47–59.

[61] STRENZKE, F.: *A timing attack against the secret permutation in the McEliece PKC,* in: Post-Quantum Cryptography, 3rd Internat. Workshop—PQCrypto '10, Darmstadt, Germany, 2010 (N. Sendrier, ed.), Lecture Notes in Comput. Sci., Vol. 6061, Springer, Berlin, 2010, pp. 95–107.

[62] STRENZKE, F.: *Fast and secure root-finding for code-based cryptosystems*, IACR Cryptology ePrint Archive, 2011:672, 2011.

[63] STRENZKE, F.: *Solutions for the storage problem of McEliece public and private keys on memory-constrained platforms,* in: Inform. Security, 15th Internat. Conf.—ISC '12, Passau, Germany, 2012 (D. Gollmann and F. C. Freiling, eds.), Lecture Notes in Comput. Sci., Vol. 7483, Springer, Berlin, 2012, pp. 120–135.

[64] STRENZKE, F.: *Timing attacks against the syndrome inversion in code-based cryptosystems,* in: Post-Quantum Cryptography, 5th Internat. Workshop—PQCrypto '13, Limoges, France, 2013 (P. Gaborit, ed.), Lecture Notes in Comput. Sci., Vol. 7932, Springer, Berlin, 2013, pp. 217–230.

[65] STRENZKE, F.—TEWS, E.—MOLTER, H. G.—OVERBECK, R.—SHOUFAN, A.: *Side channels in the McEliece PKC,* in: Post-Quantum Cryptography, 2nd Internat. Workshop—PQCrypto '08, Cincinnati, OH, USA, 2008 (J. Buchmann et al., eds.), Lecture Notes in Comput. Sci., Vol. 5299, Springer, Berlin, 2008, pp. 216-229.

[66] SUN, H.-M.: *Further cryptanalysis of the McEliece public-key cryptosystem,* IEEE Commun. Letters **4** (2000), 18–19.

[67] UMANA, V. G. —LEANDER, G.: *Practical key recovery attacks on two McEliece variants,* Cryptology ePrint Archive, Report 2009/509, 2009, `http://eprint.iacr.org/`.

[68] VAN TILBURG, J.: *On the McEliece public-key cryptosystem,* in: Advances in Cryptology—CRYPTO 88, Santa Barbara, CA, USA, 1988 (S. Goldwasser, ed.), Lecture Notes in Comput. Sci., Vol. 403, Springer, Berlin, 1988, pp. 119–131.

[69] VON MAURICH, I.—GÜNEYSU, T.: *Lightweight code-based cryptography: QC-MDPC McEliece encryption on reconfigurable devices,* in: Proc. of the Conf. on Design, Automation and Test in Europe—DATE '14, Vol. 38, European Design and Automation Association, Leuven, Belgium, 2014, pp. 1–6.

[70] ZAJAC, P.: *A note on CCA2-protected McEliece cryptosystem with a systematic public key.* Cryptology ePrint Archive, Report 2014/651, 2014. `http://eprint.iacr.org/`.

Appendix A. Attack on CCA2-protected McEliece cryptosystem with a systematic public key

We summarize here the technical details about the attack on CCA2-protected McEliece cryptosystem with a systematic public key from [70] as a supplementary material for Section 3.1.

We use McEliece cryptosystem from Section 2, with code parameters (n, k, t). We will suppose that public key matrix G' is in a systematic form. I.e., $G' = (I_k||R)$, where I_k is a $k \times k$ identity matrix, and R is a public $k \times (n-k)$ matrix.

We will start with CCA2 conversion used in [55], [59]. Let us assume that message m is an l bit string. The full encryption algorithm with conversion works as follows:

1: $k_e \leftarrow$ a random $k - l$ bit string
2: $k_i \leftarrow$ a random l bit string
3: $e \leftarrow$ a random n-bit string with $w_H(e) = t$
4: $\tilde{m} \leftarrow k_e||hash(m||k_i)$
5: $c \leftarrow \tilde{m} \cdot G'$
6: $y \leftarrow c \oplus e||m \oplus hash(k_e)||k_i \oplus hash(e)$

Here, k_e is a session key used to mask the original message m, and k_i is a key used for protecting the integrity of the ciphertext. We note that the attacker only requires the key k_e, which can be used to decrypt the original message from the second part of the ciphertext.

If the McEliece public key is in a systematic form $G' = (I_k||R)$, we can rewrite the CCA2-protected ciphertext y as follows:

$$\begin{array}{ccccc} y_1|| & y_2|| & y_3|| & y_4|| & y_5 = \\ k_e \oplus e_1|| & hash(m||k_i) \oplus e_2|| & \tilde{m} \cdot R \oplus e_3|| & m \oplus hash(k_e)|| & k_i \oplus hash(e) \end{array}$$

The legitimate recipient will use his private key to

(1) decode the $y_1||y_2||y_3$ part, and extract $\tilde{m}$, and $e = e_1||e_2||e_3$, respectively,
(2) get k_e from $\tilde{m}$, and compute m from y_4 using k_e,
(3) decrypt integrity key k_i from y_5 using e,
(4) check the integrity using $hash(m||k_i)$.

The attacker can compromise the message, if he can decode the $y_1||y_2||y_3$ without the knowledge of the private key of the MECS. We model the security level by the complexity of the generalized information set decoding, which is supposed to be the most efficient attack the attacker has for a properly implemented system.

However, the attacker is really only interested in a $(k - l)$-bit string

$$y_1 = k_e \oplus e_1.$$

What the attacker can do is to guess e_1, and then compute $m = y_4 \oplus hash(y_1 \oplus e_1)$. Thus, instead of decoding whole $\tilde{m}$ with t errors in n bits, the attacker only needs to locate errors in the first $(k-l)$-bits to compute k_e. If the guess was incorrect, he gets a random string (property of the *hash* function). Otherwise he gets the original message, which we can assume is distinguishable from a random string. Even in the case when the message is just a random session key for some subsequent symmetric encryption, attacker can try to decrypt the symmetric message (which then can be distinguished from a random string).

We did not study the complexity of partial decoding problem, but even a simple brute-force attack can be more efficient than the prescribed security level, if the parameters are chosen incorrectly. Suppose that security level is s. Choose the encryption key of size $k - l = s$ (if it is shorter, the system is trivially broken just by guessing $hash(k_e)$). The attacker needs only to enumerate s-bit vectors within a certain Hamming distance (approximately $s \cdot \lceil t/n \rceil$) of y_1 to uncover k_e, instead of enumerating all 2^s vectors. Thus the attack is more efficient than the prescribed security level s.

If the attacker needs to recover $k - l$ bits of the cleartext, the complexity of the attack is approximately

$$\sum_{i=0}^{\lceil t/n \rceil \cdot (k-l)} \binom{k-l}{i} \leq 2^{H(t/n)(k-l)}.$$

The security of the system thus depends not only on the system parameters (n, k, t), but also on the choice of l. Usually, the selection of l is based on the typical length of the hash function output. However, larger values of l decrease a relative overhead of the analysed encryption scheme, thus it may seem more attractive for implementers to use longer messages and larger hash functions. Thus, we get a very counter-intuitive property of the system: Using a hash function with longer output decreases the overall security instead of increasing it.

Overbeck in [44] introduces a specific conversion for small messages (k–$3l \geq l$), where

$$\tilde{m} \leftarrow k_e || m \oplus hash(k_e) || k_i \oplus hash(e) || hash(m || k_i)$$

In this case, the ciphertext is just encoded $\tilde{m}$ with the added errors.

Similarly to the previous attack, if the public key is in a systematic form, attacker tries to correct errors in the k_e part of size $k - 3l$, and verify them using $m \oplus hash(k_e)$ part. As the size of k_e is much smaller than in the previous case, it is easier to recover k_e. However, the verification part is now also corrupted by approximately $l \cdot \lceil t/n \rceil$ errors. The success rate thus depends on the entropy of the message m. If m is a random session key used for further encryption, the attacker must correct errors in both parts. In this case the complexity of the attack is based on decoding errors in $k - 2l$ bits (out of n). However, the implementers of the system should always suppose that the worst case scenario, when

TABLE 4. Complexity of the attack for selected parameter choices of the MECS. Max. l denotes the maximum value of parameter l, for which the estimated attack cost is approximately equal to the expected security level. Choices of l larger than this maximum lead to insecure systems.

Sec. Level	(n, k, t)	$H(t/n)$	$l = 2s$	$l = 6s$	Max. l
50 [37]	(1024,524,50)	0.281	119	63	346
80 [6]	(2048,1751,27)	0.101	161	129	961
80 [40]	(1702,1219,45)	0.176	187	130	765
128 [7]	(3178,2384,68)	0.149	318	241	1526
256 [7]	(6944,5208,136)	0.139	653	511	3368

the message m can be easily distinguished even in the presence of errors, and the attacker only needs to correctly recover $k - 3l$ bits of k_e.

We summarize the complexity of the attack for some of the recommended parameter choices from [6], [7], [37], [40] in Table 4, along with the limits to parameter l based on security level. Maximum size l should be divided by 3 if one uses a conversion from [44] for small messages. If l is chosen to be double the security level (preventing collision attacks in hash function), the system can resist the attack in all analysed cases. The complexity estimate is based just on the estimated complexity of a simple brute force attack, and may be improved in the future by more advanced attacks.

The conversions presented by K o b a r a and I m a i [33] require the attacker to recover the whole cleartext $\tilde{m}$ to retrieve original message m. The complexity of the brute-force attack in this case is approximately $2^{H(t/n)k}$. However, if the attacker can repair all errors in k bits of encoded code-word, he has already succeeded in the information-set decoding attack, even if the public key matrix is not in a systematic form. Thus the attack is not relevant in this case, as it should be already thwarted by a proper choice of the parameters of the MECS.

Received October 18, 2013

Institute of Computer Science
and Mathematics
Faculty of Electrical Engineering
and Information Technology
Slovak University of Technology
in Bratislava
Ilkovičova 3
SK–812-19 Bratislava
SLOVAKIA
E-mail: pavol.zajac@stuba.sk
marek.repka@stuba.sk