Classic McEliece: conservative code-based cryptography: guide for security reviewers

23 October 2022

Contents

1 Overview

This document is aimed at security reviewers who are checking, qualitatively and quantitatively, the security provided by Classic McEliece. This document covers (1) known attacks and (2) what is known about the possibility of further attacks.

The primary focus of this document is IND-CCA2 security. Classic McEliece is designed to obtain high confidence in long-term IND-CCA2 security from the composition of

- high confidence in the OW-CPA security of the original 1978 McEliece cryptosystem, including a remarkably stable quantitative security level, and
- high confidence that switching from the original cryptosystem to the Classic McEliece KEM converts OW-CPA security $\geq 2^\lambda$ into IND-CCA2 security $\geq 2^{\lambda-\epsilon}$ for a small ϵ.

For simplicity, all of the selected KEM parameter sets use 256-bit secrets, even when the parameters could otherwise reach higher security levels; this limits the range of λ covered above. It would be easy to adjust the "ℓ" parameter choice to use larger secrets for users concerned about the possibility of an attacker guessing 256-bit secrets, but this is not a real-world threat.

Classic McEliece can be viewed as the result of a series of simple transformations applied to the original cryptosystem. The security analysis in this document is decomposed accordingly:

- Section 3 reviews the OW-CPA security of the original cryptosystem.
- Section 4.1 concludes that generalizing from $n = q$ to $n \leq q$ is safe, producing at most a marginal loss of OW-CPA security. (Taking $n < q$ could *gain* security, but the possibility of such gains is not logically relevant to the desired conclusion that Classic McEliece provides IND-CCA2 security *at least* $2^{\lambda-\epsilon}$.)
- Section 4.2 concludes that transmitting public keys in systematic form is safe.
- Section 4.3 concludes that using syndromes as ciphertexts is safe.
- Section 5.3 concludes that applying a generic CCA transform is safe, providing IND-CCA2 security if the previous system provides OW-CPA security.
- Section 5.4 concludes that particular improvements applied to the CCA transform are safe.
- Section 5.5 concludes that particular mechanisms of generating irreducible polynomials, field orderings, and fixed-weight vectors are safe.

The extent to which IND-CCA2 security has been *proven*, for the Classic McEliece KEM or for any other cryptosystem, should not be overstated. In particular, no proofs rule out the possibility of much faster attacks that break IND-CCA2 by breaking OW-CPA. Confidence in the OW-CPA security of the McEliece cryptosystem instead comes from how well the system has survived extensive cryptanalysis.

Section 6 covers various security properties beyond IND-CCA2.

2 Correctness

As a preliminary matter, the Classic McEliece KEM is correct. This means the following: for every key pair produced by KeyGen, and for every ciphertext and session key produced by Encap, the same session key will be produced by Decap as by Encap.

Correctness is often described as a functionality issue. An application encountering a decryption failure has to ask for the data to be retransmitted, so the application has to provide a two-way communication channel, and there can be problematic slowdowns if failures occur frequently. More subtly, correctness is a security issue, which is also why correctness is covered in this document. See, e.g., [42] for an attack exploiting occasional decryption failures in a lattice-based cryptosystem.

The following subsections give details to support security review for the functions described in the separate "cryptosystem specification" document using variable names defined there. Steps mentioned below refer to steps in algorithms stated in that document.

2.1 Goppa codes

Write $\Gamma = (g, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ for an input to MatGen. Recall that g is a monic irreducible polynomial in $\mathbb{F}_q[x]$ of degree t, and $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ are distinct elements of $\mathbb{F}_q$.

The corresponding "q-ary Goppa code" Γ_q is the set of vectors $c \in \mathbb{F}_q^n$ such that the polynomial $\sum_i c_i A/(x - \alpha_i)$ is a multiple of g in $\mathbb{F}_q[x]$, where $A = \prod_j (x - \alpha_j)$. The corresponding "binary Goppa code" Γ_2 is the set of vectors $c \in \mathbb{F}_2^n$ such that the polynomial $\sum_i c_i A/(x-\alpha_i)$ is a multiple of g in $\mathbb{F}_q[x]$; in other words, $\Gamma_2 = \Gamma_q \cap \mathbb{F}_2^n$.

Any vector in $\mathbb{F}_2^n$ can be written in *at most* one way as $c + e$ where $c \in \Gamma_2$, $e \in \mathbb{F}_2^n$, and e has Hamming weight at most t. In other words, Γ_2 has "minimum distance" at least $2t + 1$. Here are two ways for reviewers to check this fact:

- Reviewers who are also checking that a claimed KEM implementation correctly computes the specified KEM will, in particular, check the proof of a "decoding" algorithm that always outputs (c, e) given $c + e$; see, e.g., [7, Section 6]. Uniqueness of (c, e) follows immediately.
- Reviewers who are merely checking correctness of the specified KEM can rely on shorter proofs of uniqueness, such as [63, Theorem 4.4.5].

2.2 Parity-check matrices

The matrix $\tilde{H}$ constructed inside MatGen is a "parity-check matrix" for Γ_q. This means, for each $c \in \mathbb{F}_q^n$, that $\tilde{H}c = 0$ if and only if $c \in \Gamma_q$. See, e.g., [7, Theorem 7.2]; this theorem is due to Goppa. This implies that the matrix $\hat{H}$ constructed inside MatGen is a parity-check matrix for Γ_2.

Row reduction converts any parity-check matrix into another parity-check matrix. For $(\mu, \nu) = (0, 0)$, if MatGen(Γ) produces a non-failure output (T, Γ) then $H = (I_{mt} \mid T)$ is a matrix obtained by row reduction from $\hat{H}$, and thus a parity-check matrix for Γ_2.

For general (μ, ν), the following invariant holds inside MatGen: the matrix H is a parity-check matrix for Γ'_2, where $\Gamma' = (g, \alpha'_0, \ldots, \alpha'_{n-1})$. The point is that any column swaps that MatGen carries out on H are also carried out on $(\alpha'_0, \ldots, \alpha'_{n-1})$, preserving this invariant. Consequently, if MatGen(Γ) produces a non-failure output $(T, \ldots, \Gamma')$, then $H = (I_{mt} \mid T)$ is a parity-check matrix for Γ'_2.

2.3 Encoding and decoding

The next step is to check that Decode works as specified. This has two parts. First, any C of the form He, where $e \in \mathbb{F}_2^n$ has weight t, will have Decode output e. Second, any C that does not have the form He will have Decode output $\perp$. The logic is as follows.

For the first part, note that the "syndrome" Hv is C, because the first mt positions of v are multiplied by the identity matrix and the remaining positions are zero. Hence $Hv = He$. Define $c = v + e$; then $Hc = 0$, so $c \in \Gamma'_2$. This vector c has distance exactly t from v, and it is the unique element of Γ'_2 at distance $\leq t$ from v, since the minimum distance of Γ'_2 is at least $2t + 1$. Hence Step 2 finds c, Step 3 finds e, and Step 4 returns e.

For the second part, if Decode returns e in Step 4 then e has been verified to have weight t and to have $C = He$, so if C does not have this form then Decode must return $\perp$.

2.4 Encapsulation and decapsulation

The last step is to check that any ciphertext and session key output by Encap produce the same session key from Decap.

By assumption $C = He$ for some $e \in \mathbb{F}_2^n$ of weight t. The Decode algorithm will return e in Step 4, thus $b = 1$ and K matches the session key computed in encapsulation.

3 OW-CPA security of the original McEliece cryptosystem

The original McEliece cryptosystem encrypts a message a as $Ga + e$, where the matrix G is the public key and e is a weight-t vector. The OW-CPA problem for this cryptosystem is to recover a from $(G, Ga + e)$ when G is chosen as a random McEliece key and a and e are chosen uniformly at random. This section reviews the literature on attacks against this problem.

3.1 Types of attacks

For all parameters of interest, the fastest attacks known against the OW-CPA problem treat G as a general matrix with no evident structure. Sections 3.2, 3.3, 3.4, and 3.5 review the performance of algorithms to recover a from $(G, Ga + e)$ for general matrices G.

There are also much slower "structural attacks" known that try to exploit the secret structure of G, for example by recovering Goppa-code parameters for G. See Section 3.6.

A common way to formalize the distinction between these two types of attacks is as follows. Assume that it is difficult to recover a from $(G, Ga+e)$ when G is chosen as a uniform random matrix and m and e are chosen uniformly at random. An attack that breaks the OW-CPA problem is then a distinguisher between McEliece's public key and a uniform random matrix. In other words, OW-CPA security follows from the extra assumption that it is difficult to distinguish McEliece's public key from a uniform random matrix. This is not a two-way implication—perhaps there are distinguishers even if OW-CPA is secure—but it motivates studying distinguishers as a minimum bar for structural attacks. If there are no feasible distinguishers then the only possibility for better OW-CPA attacks is better attacks against uniform random matrices.

3.2 Prange's information-set-decoding algorithm

Assume that G is an $n \times k$ matrix over $\mathbb{F}_2$, that $a \in \mathbb{F}_2^k$, that $e \in \mathbb{F}_2^n$, and that e has weight t. Pick any set of k positions within the n positions in the corresponding ciphertext $C = Ga+e$. This set is error-free, meaning that these positions in e are all 0, with probability exactly $\binom{n-k}{t}/\binom{n}{t}$.

If this set is error-free then these positions in the known vector $C = Ga + e$ are the same as these positions in the vector Ga. One can then recover a from these positions in Ga by linear algebra. One can recognize whether this was successful by checking whether $C - Ga$ has weight t, and try another set otherwise.

A small difficulty here is that the $k \times k$ matrix obtained by restricting G to these positions is not necessarily invertible: i.e., these positions in Ga could leave multiple possibilities for a. The simplest solution is to require the set of positions to be an "information set", a set for which the $k \times k$ matrix is invertible, and simply try again if the matrix turns out not to be invertible. This attack algorithm is from 1962 Prange [56], the simplest form of "information-set decoding".

A uniform random $k \times k$ matrix has probability $(1 - 1/2)(1 - 1/4) \cdots (1 - 1/2^k) \approx 0.29$ of being invertible. Experiments for large k with McEliece's public keys, matrices secretly obtained from Goppa codes, obtain probabilities statistically indistinguishable from this. In other words, even if distinguishers exist between the public keys and uniform random matrices, the power of invertibility as a distinguisher is weak enough that one can treat these situations as equivalent for purposes of analyzing information-set decoding. Similarly, correlations of events inside Prange's algorithm (being invertible and being error-free across

multiple iterations) are shown by experiments to be negligible.

A smaller difficulty is that, for general matrices G, there might be colliding ciphertexts: another (a', e') might have $Ga' + e' = Ga + e$. This event limits the success probability of any OW-CPA attack, since the attack cannot tell whether the original target was a or a'. As an extreme case, if $G = 0$ then an OW-CPA attack cannot succeed with probability above $1/2^k$. However, colliding ciphertexts cannot occur for McEliece's public keys (see Section 2), and are extremely unlikely to occur for a uniform random matrix of the same size.

To summarize, under minor assumptions, each iteration of Prange's algorithm succeeds with probability approximately $0.29\binom{n-k}{t}/\binom{n}{t}$, and the average number of iterations before success is approximately $\binom{n}{t}/0.29\binom{n-k}{t}$.

3.3 Improvements to information-set decoding

McEliece's cryptosystem prompted many further papers studying algorithms to recover a from $(G, Ga + e)$ for general matrices G, including [24], [50], [51], [47], [61], [30], [25], [64], [31], [26], [21], [22], [65], [17], [18], [19], [9], [11], [35], [10], [52], [3], [39], [53], [62], [15], [16], [28], [34], and [20].

Reducing linear-algebra time has turned out to be an important source of quantitative improvements over Prange's algorithm. For example:

- Instead of guessing that an information set has 0 errors, 1988 Lee–Brickell [50] guess that it has (e.g.) 2 errors, reusing the costs of linear algebra on G for all $\binom{k}{2}$ choices of error positions.
- For each a obtained from a particular information set, instead of computing all positions of $C - Ga$ and checking whether $C - Ga$ has weight t, 1988 Leon [51] first computes a limited number of positions, requiring those positions to be 0.
- An idea from Omura (see [24, page 175]) is to have each iteration change just one position from the previous set, amortizing the costs of linear algebra across many iterations. It is easy to do this in a way that guarantees that each set will be an information set. One can use Markov chains to analyze the randomness of the number of errors in the set; see [19, Section III]. It turns out to be better to change somewhat more positions; see [9].

Further improvements have come from integrating Prange's algorithm with more sophisticated combinatorial searches for error positions in the information set. For example, 1989 Stern [61] uses a meet-in-the-middle search for error positions, and 2011 May–Meurer–Thomae [52] and 2012 Becker–Joux–May–Meurer [3] use a "representations" idea adapted from subset-sum algorithms.

All of these algorithms can be combined with "quantum walks" such as Grover's algorithm; see [4], [44], and [45]. This requires rebalancing various algorithm parameters, generally reducing the benefit of more sophisticated combinatorial searches, as pointed out in [4,

Section 3].

The literature has also explored various algorithmic ideas other than information-set decoding. One can, for example, try to find a low-weight vector in a code by taking a large list L of random vectors, computing a list L' of short differences from L, computing a list L'' of short(er) differences from L', etc. Compared to information-set decoding, the alternatives studied so far have turned out to be worse, at least for the parameter ranges of interest for McEliece's cryptosystem. A useful way to monitor progress is to quantify algorithm performance for a wider range of parameters; see, e.g., [20].

3.4 Asymptotic costs of information-set decoding

The ratio $\binom{n-k}{t}/\binom{n}{t}$, essentially the success probability of one iteration of Prange's algorithm, is $(1-k/n)(1-k/(n-1))\cdots(1-k/(n-t+1))$. The following easy asymptotic statement holds for any real number R with $0 < R < 1$: if $k \in (R+o(1))n$, and $t \in o(n)$, then $(1-k/n)(1-k/(n-1))\cdots(1-k/(n-t+1))$ is $(1-R+o(1))^t$ as $n \to \infty$. The total cost of Prange's algorithm is then $(1/(1-R)+o(1))^t$.

Remarkably, none of the improvements from Section 3.3 have done better than this. The improvements are merely changing the $o(1)$, without affecting the main constant $1/(1-R)$. In other words, the improvements are covering only fringe aspects of costs, without affecting the core combinatorial difficulty of finding e. For details of this analysis, see [11] and [10, Section 1] regarding Stern's algorithm and various other algorithms known up to that point, and [62] regarding the use of "representations".

In particular, if one takes $k \in (R+o(1))n$ in the McEliece system, then t is asymptotically $(1-R+o(1))n/\log_2 n$, so the assumption $t \in o(n)$ holds. The cost of Prange's algorithm is $(1/(1-R)^{1-R}+o(1))^{n/\log_2 n}$, and the cost of the most efficient attack known today is also $(1/(1-R)^{1-R}+o(1))^{n/\log_2 n}$. Meanwhile the ciphertext size is $(1-R+o(1))n$ bits, and the key size is $(R(1-R)+o(1))n^2$ bits. Security level 2^b thus uses key size $(c_0+o(1))b^2(\log_2 b)^2$ where $c_0 = R/(1-R)(\log_2(1-R))^2$. This c_0 reaches its minimum value, approximately 0.7418860694, when R is approximately 0.7968121300, as mentioned in the separate "design rationale" document.

Some papers indicate that asymptotic attack exponents have improved. This is because those papers are measuring their results for much larger $t \in \Theta(n)$, such as "half of the GV distance". This inflation of t increases cost from $2^{\Theta(n/\log_2 n)}$ to $2^{\Theta(n)}$, and also makes differences between algorithms more noticeable. For example, [53] reports $\mathcal{O}(2^{0.0473n})$ when t is half of the GV distance, compared to $\mathcal{O}(2^{0.0576n})$ from Prange's algorithm. Such large asymptotic sizes of t are of interest in coding theory but do not appear in the McEliece system.

Known quantum attacks have a simple effect on the asymptotics, replacing $1/(1-R)^{1-R}+o(1)$ with its square root.

3.5 Concrete costs of information-set decoding

It is important to realize that $o(1)$ does not mean 0: it means something that converges to 0 as $n \to \infty$. More detailed attack-cost evaluation is therefore required for any particular parameters.

For example, for $R = 0.8$, Prange's algorithm costs $(5 + o(1))^t$, and various improved algorithms also cost $(5 + o(1))^t$. It is incorrect to replace $o(1)$ with 0, obtaining the statement that Prange's algorithm costs 5^t and the statement that various improved algorithms also cost 5^t; the latter two statements contradict each other, given the experimentally verified fact that the improvements reduce costs.

Many of the papers cited above include precise operation counts and precise probability formulas for each attack iteration. A closer look shows that these operation counts tend to omit important components of attack costs. For example, the lowest operation counts in the literature ignore the costs of random access to a huge array, much larger than the public key being attacked. In reality, time and energy are required to move data across long distances, and to control which array elements are being accessed; meanwhile the same amount of attack hardware allows much more parallelism for low-memory attacks.

As numerical examples of the importance of memory access, Table 1 reports output of the Esser–Bellini security estimator [33] for each of the selected Classic McEliece parameter sets, under three different models supported by the estimator:

- The rows where "mem" is 0 use a model making a physically implausible assumption of free access to arbitrarily large amounts of memory. In this model, the 2011/2012 algorithms using "representations" provide noticeable speedups compared to Stern's 1989 algorithm.
- The rows where "mem" is 1/2 use a model assigning plausible square-root costs to memory access. The rows where "mem" is 1/3 use a model assigning cube-root costs to memory access. In these models, the algorithms using "representations" provide much smaller speedups compared to Stern's algorithm. In other words, most of the claimed speedup from these algorithms at cryptographic sizes relies on assigning unrealistically low costs to memory access.

Regarding quantum attacks, it is structurally clear that known quantum attacks produce somewhat less of a speedup for these algorithms than they do for, e.g., AES key search, since the application of Grover's method (or more general quantum walks) to information-set decoding suffers much more overhead in the inner loop. Using Grover's method also reduces the benefit of more sophisticated combinatorial searches, as noted in Section 3.3.

Comparison to AES security. A requirement in the NIST Post-Quantum Cryptography Standardization project is to compare concrete security levels to AES brute-force attacks and/or SHA-2 collision attacks. In particular, the minimum security level allowed is the security of brute-force AES-128 key search, which NIST estimates as 2^{143} bit operations.

param	mem	Prange	Stern	Dumer	BC	BJMM	pdw	MO	BM
348864	0	173.4	151.4	151.4	151.5	141.9	143.4	140.8	142.7
348864	1/3	176.7	159.3	159.7	159.7	159.1	157.8	156.4	157.5
348864	1/2	178.3	162.8	163.2	163.2	162.2	160.3	158.6	159.8
460896	0	216.8	193.3	193.2	193.3	180.8	182.6	179.8	182.1
460896	1/3	220.4	201.7	202.2	202.2	201.6	200.6	198.6	199.8
460896	1/2	222.1	205.3	205.8	205.8	204.8	203.4	201.0	202.2
6688128	0	295.7	268.0	267.9	268.0	247.3	248.8	246.2	249.6
6688128	1/3	299.4	279.2	279.5	279.5	278.9	277.6	274.9	276.4
6688128	1/2	301.2	283.0	283.3	283.3	282.3	280.4	278.2	279.4
6960119	0	296.8	268.4	268.3	268.4	246.6	248.4	245.7	249.2
6960119	1/3	300.4	280.1	280.5	280.5	279.4	278.6	275.8	277.3
6960119	1/2	302.2	283.9	284.3	284.3	283.3	281.4	279.2	280.3
8192128	0	334.1	303.2	307.4	303.4	277.7	279.1	275.6	281.1
8192128	1/3	337.7	316.8	317.1	317.1	316.3	315.5	311.9	313.3
8192128	1/2	339.5	320.7	321.0	321.0	320.0	318.6	315.6	317.0

Table 1: Output of the Esser–Bellini estimator for the selected Classic McEliece parameter sets. The "mem" column is 0 for estimates assuming free memory access, 1/3 for estimates using a cube-root assumption, and 1/2 for estimates using a square-root assumption. Some cost components are ignored in all estimates.

If these requirements include "mem" being 0, counting only bit operations with free memory access, then Table 1 raises the following concern: the smallest number in the table, 140.8 for `mceliece348864`, is slightly below 143. However, the underlying estimator from [33] counts each vector operation as just 1 operation: in particular, finding C collisions between two lists, each list containing L vectors, is assigned cost $2L + C$, no matter what the vector length is. Consequently, the 140.8 actually counts more than 2^{143} bit operations. The exact numbers in Table 1 should be expected to be superseded by larger numbers from future estimators that count bit operations.

More importantly, the 140.8 is for an attack using memory $2^{86.6}$, according to the same estimator. Accounting for costs of memory access shows that all known attacks against `mceliece348864` are much more expensive than brute-force AES-128 key search.

Larger parameter sets can be above or below (e.g.) AES-256 depending on whether costs of memory are included. The Classic McEliece submission has always included the `6960119` parameter set, and has always explicitly distinguished two different methods of comparing attacks against this parameter set to attacks against AES-256:

- Imagining free memory access: The submission has always stated that this parameter set is breakable using fewer *bit operations* than brute-force AES-256 key search when the costs of memory access are ignored: "Subsequent ISD variants have reduced the number of bit operations considerably below 2^{256}." This is consistent with the $2^{246.6}$ number in Table 1.

- Counting realistic costs for memory: "We expect that switching from a bit-operation analysis to a cost analysis will show that this parameter set is more expensive to break than AES-256 pre-quantum and much more expensive to break than AES-256 post-quantum." This is consistent with the $2^{279.2}$ number in Table 1.

Given the fact that memory is not free in the real world, the submission has always assigned this parameter set to NIST's "Category 5" (AES-256).

Similarly, when the submission added further parameter sets, it assigned those parameter sets to "categories" on the basis of counting realistic costs for memory. One can object to the assignment of `460896` to "Category 3" (AES-192) since NIST estimates 2^{207} operations for brute-force AES-192 key search while Table 1 says 201.0, which is below 207; but, as noted above, the estimates in the table are underestimates of bit operations, for example counting each vector operation as just 1 operation.

Since the underlying facts have not changed, the submission continues to assign its selected parameter sets to "categories" 1, 3, 5, 5, 5 respectively. As before, these assignments are based on counting realistic costs for memory.

If NIST instead decides to make "category" assignments on the basis of bit operations with free memory access, then the correct assignments will instead be 1, 2, 4, 4, 5. This does not reflect any instability in the Classic McEliece security estimates: the submission has always been careful to distinguish between these two different types of accounting for the costs of attacks.

3.6 Structural attacks

McEliece's G is a random generator matrix for the binary Goppa code determined by the private key $(g, \alpha_0, \ldots, \alpha_{n-1})$, specifically with $n = q$; see Section 2.1. The description in McEliece's original paper is different but equivalent: there is a permutation matrix P that permutes a standard order of field elements into $(\alpha_0, \ldots, \alpha_{n-1})$, and there is an invertible matrix S that converts a standard generator matrix for the binary Goppa code into a random generator matrix for the same code.

Many followup cryptosystems have used alternatives to binary Goppa codes, sometimes switching to another metric (examples include rank-metric cryptography and lattice-based cryptography) and sometimes continuing to use the Hamming metric. This has motivated study of structural attacks against a wide variety of targets. Some of these alternative cryptosystems are broken while others are not. As noted in the separate "design rationale" document, authors of attacks on other codes often study the performance of their attacks against binary Goppa codes; see the examples cited there and in [54].

Potentially useful components of structural attacks sometimes appear as attacks using "partial information". For example, 2022 Kirshanova–May [46] presented a fast algorithm to recover the private key $(g, \alpha_0, \ldots, \alpha_{n-1})$ given the public key and partial information $(\alpha_0, \ldots, \alpha_{n-k})$. This improves on an earlier observation that one can recover g from

$(\alpha_0, \ldots, \alpha_{n-1})$. Consequently, given just the public key, one can recover the private key with a brute-force search through possibilities for $(\alpha_0, \ldots, \alpha_{n-k})$. There are $q(q-1)\cdots(q-n+k)$ possibilities; e.g., about $2^{4794.655}$ possibilities for McEliece's original parameters $(q, n, k, t) = (1024, 1024, 524, 50)$, and about $2^{9116.797}$ possibilities for $(q, n, k, t) = (4096, 3488, 2720, 64)$.

As another example (also mentioned in the separate "design rationale" document), 2000 Sendrier [58] introduced a "support-splitting algorithm" that quickly recovers $(\alpha_0, \ldots, \alpha_{n-1})$ from the public key, g, and $\{\alpha_0, \ldots, \alpha_{n-1}\}$, under minor assumptions. In the case $n = q$, the set $\{\alpha_0, \ldots, \alpha_{n-1}\}$ is known; consequently, given just the public key, one can recover the private key with a brute-force search through possibilities for g. There are about q^t/t possibilities; e.g., about $2^{494.356}$ possibilities for McEliece's original parameters, and about 2^{762} possibilities for $(q, t) = (4096, 64)$. Known symmetries provide only a small speedup.

4 OW-CPA security of modified cryptosystems

4.1 OW-CPA security of length below field size

As noted in Section 3.6, McEliece's original cryptosystem chose n specifically as q, a power of 2. More general algorithm statements allowing $n \leq q$, and proposals specifically to take $n < q$, appeared later, for example in [9, Section 7].

What happens if $n < q$ is breakable, while the original $n = q$ proposal is safe? Appealing to the long history of study of the original McEliece system does not answer this question: a different special case of a generalized system could have much less security than the original system.

Shortening as a reduction across parameters. A simple answer is that *any* attack A against parameters (q, n, t), for any $n \leq q$, can be used as an attack against parameters (q, q, t) as follows.

Consider a McEliece private key $\Gamma = (g, \alpha_0, \ldots, \alpha_{q-1})$ with parameters (q, q, t). The attacker is given the corresponding code Γ_2 (represented as a uniform random generator matrix) and a vector $w = c + e \in \mathbb{F}_2^q$, where $c \in \Gamma_2$, $e \in \mathbb{F}_2^q$, and e has weight t. The attacker's objective is to find c, or equivalently to find e.

The attack using A has five steps:

- Select, uniformly at random, an injection $\pi : \{0, \ldots, n-1\} \to \{0, \ldots, q-1\}$. Define Π as the corresponding linear map from $\mathbb{F}_2^n$ to $\mathbb{F}_2^q$, moving the coordinate at each position $j \in \{0, \ldots, n-1\}$ to position $\pi(j)$ and filling in 0 at other positions.
- Compute, by linear algebra, the subspace $\Gamma_2 \cap \Pi\mathbb{F}_2^n$. This is exactly $\Pi\Gamma_2'$ where $\Gamma' = (g, \alpha_{\pi(0)}, \ldots, \alpha_{\pi(n-1)})$.
- Compute, by linear algebra, a uniform random vector $w' \in \mathbb{F}_2^n$ such that $\Pi w' - w \in \Gamma_2$.

Abort if no such vector exists.

- Apply the parameter-(q, n, t) attack A to Γ_2' and w' to find $c' \in \Gamma_2'$ and $e' \in \mathbb{F}_2^n$ of weight t such that $w' = c' + e'$. Abort if A fails.
- Output $e = \Pi e'$ and $c = w - e$.

If this attack succeeds then $e = \Pi e'$ is supported on positions $\pi(0), \ldots, \pi(n-1)$, an event that occurs with probability $\binom{n}{t}/\binom{q}{t}$. Conversely, if e is supported on these positions, then e can be expressed uniquely as $\Pi e'$; there exist vectors $w' \in \mathbb{F}_2^n$ with $\Pi w' - w \in \Gamma_2$, such as $w' = e'$; and the inputs (Γ_2', w') to A have the correct distribution, namely a uniform random binary Goppa code with parameters (q, n, t) and a uniform random vector $c' + e'$ with $c' \in \Gamma_2'$, $e' \in \mathbb{F}_2^n$, and e' of weight t.

To summarize, the success probability of this attack against parameters (q, q, t) is exactly $\binom{n}{t}/\binom{q}{t}$ times the success probability of attack A against parameters (q, n, t). The cost of the attack is simply the cost of A plus some low-cost linear algebra.

The conventional perspective is that this reduction is not very tight: $\binom{n}{t}/\binom{q}{t}$ can be small if n is not close to q. However, this factor has to be compared to the target security level, which is not the same for parameters (q, q, t) and (q, n, t).

The starting assumption is that we understand the success probability of attacks (within the resources available to the attacker) against parameters (q, q, t), not much better than the success probability of Prange's algorithm for parameters (q, q, t), which is approximately $\binom{mt}{t}/\binom{q}{t}$ per iteration. The reduction then says that the success probability of attacks (within the same resources minus the cost of linear algebra in the reduction) against parameters (q, n, t) is at most $\binom{q}{t}/\binom{n}{t}$ times higher than this, and hence not much better than the success probability of Prange's algorithm for parameters (q, n, t), which is approximately $\binom{mt}{t}/\binom{n}{t}$ per iteration.

From this perspective, the tightness loss of the reduction is actually the ratio $S(q, q, t)/S(q, n, t)$, where S is the speedup factor of state-of-the-art algorithms compared to Prange's algorithm. Full quantification depends on the exact cost of state-of-the-art attacks, as in Section 3.5.

For example, in the setting of Table 1 with 1/2 for "mem", the tightness loss is just 0.8 bits for `460896`. To check this calculation, run the estimator for $(4608, 96)$ and $(8192, 96)$, and compare the security-level difference to $\binom{q}{t}/\binom{n}{t}$. In other words, instead of directly analyzing the cost of known attacks against $(4608, 96)$, one can, at the expense of 0.8 bits in the resulting security level, analyze the cost of known attacks against $(8192, 96)$—which is a parameter set for McEliece's original system, justifiably appealing to the long history of study of that system. As another example, the tightness loss is just 0.3 bits for `6960119`.

The cryptanalytic perspective. The literature on information-set decoding consistently handles arbitrary values of n, without regard to whether n is a power of 2. The obvious explanation for $S(q, q, t)$ being marginally larger than $S(q, n, t)$ is that q is larger than n, not that size n has been studied less than size q.

The question of whether $n = q$ has more effect on structural attacks. In particular, taking n considerably below q is an *extra* defense against support splitting, since the set $\{\alpha_0, \ldots, \alpha_{n-1}\}$ is kept secret: for `6960119`, there are $\binom{q}{n} \approx 2^{4997}$ possibilities for this set, although there could be ways to merge work across possibilities. Meanwhile there are no known attack strategies for which $n = q$ is an extra defense.

4.2 OW-CPA security of systematic-form public keys

The next modification made to McEliece's original system is sending public keys in systematic form. This modification has two effects. The first effect is to restrict attention to binary Goppa codes that have parity-check matrices of the form $(I_{mt} \mid T)$, restarting key generation whenever other codes occur. This form of parity-check matrix has been, for all parameters of interest, experimentally verified to occur for approximately 29% of binary Goppa codes.

Any attack with probability p against this restricted set of binary Goppa codes, combined with verifying that the input code is systematic, is an attack with probability approximately $0.29p$ against the full set of binary Goppa codes used in McEliece's cryptosystem. This is a tightness loss below 1.8 bits. There is no evidence that any such security loss occurs.

The second effect of the modification is to transmit T rather than transmitting McEliece's public key, a uniform random generator matrix for the same code. Any attack against T can be used to attack any generator matrix for the same code, and in particular McEliece's public key, since anyone given any generator matrix can quickly compute the systematic-form public key. This reduction preserves probability, and preserves cost aside from simple linear-algebra steps.

More generally, any attack with probability p against binary Goppa codes in semi-systematic form implies an attack with probability $p\sigma$ against McEliece's public key, where σ is the probability of a binary Goppa code being reducible to semi-systematic form; for example, the tightness loss is a small fraction of a bit for $(\mu, \nu) = (32, 64)$. The point is that anyone given McEliece's public key can reduce it to semi-systematic form (while permuting ciphertexts accordingly), aborting if the reduction fails, and then apply any semi-systematic-form attack.

4.3 OW-CPA security of syndromes as ciphertexts

Another standard modification is sending Niederreiter's ciphertexts He rather than McEliece's ciphertexts $Ga + e$.

For any distribution of parity-check matrices H publicly computable from the public code (e.g., the unique systematic-form parity-check matrix $(I_{mt} \mid T)$ for a systematic-form code), Niederreiter's OW-CPA problem is equivalent to McEliece's OW-CPA problem for the same code. In particular, any attack recovering e from Niederreiter's He and H can be used with negligible overhead to recover (a, e) from McEliece's $Ga+e$ and G. Specifically, given $Ga+e$ and G, compute a parity-check matrix H for the same code, multiply H by $Ga+e$ to obtain

$HGa + He = He$, apply the attack to recover e from He, subtract e from $Ga + e$ to obtain Ga, and recover a by linear algebra.

5 IND-CCA2 security

5.1 IND-CCA2 attacks against the original McEliece cryptosystem

McEliece's original PKE is trivially broken in the IND-CCA2 attack model. Here are three examples of attacks:

- The attacker chooses two different messages a, a'; is given a challenge ciphertext C, either $Ga + e$ or $Ga' + e'$; and checks whether $C - Ga$ has weight t.
- The attacker chooses $\delta \neq 0$; modifies a ciphertext $Ga + e$ into $Ga + G\delta + e$; is given the new plaintext $a + \delta$; and subtracts δ to obtain the target plaintext a.
- The attacker adds two errors to a ciphertext $Ga + e$. There is a noticeable chance that one error position is in e and the other is not, producing a valid new ciphertext $Ga + e'$, and then the attacker is given a.

Similarly trivial attacks work against Niederreiter ciphertexts. Here are three examples:

- The attacker chooses distinct e, e'; is given a challenge ciphertext C, either He or He'; and checks whether the ciphertext is He. (More generally, deterministic PKEs never provide IND-CCA2 security.)
- The attacker chooses a weight-2 vector δ; modifies a ciphertext He into $He + H\delta$; has a noticeable chance of receiving a new plaintext $e + \delta$; and subtracts δ to obtain the target plaintext e.
- Even in a weaker attack model that merely reveals *whether decryption succeeds* (see, e.g., [66]) rather than revealing plaintexts, the attacker tries many choices of weight-2 vectors δ, noting all positions used in vectors δ for which decryption of $He + H\delta$ succeeds. Each success notes one error position and one non-error position, and errors are at considerably under half of the positions, so the most popular positions will be the error positions.

5.2 Features of Classic McEliece supporting IND-CCA2 security

Classic McEliece adds extra defenses with the goal of upgrading OW-CPA security to IND-CCA2 security. Most importantly:

- Classic McEliece ensures PKE "rigidity". This means that the decryption function outputs a plaintext e only when the input ciphertext is He; for an input ciphertext

not of the form He, the decryption function outputs $\perp$.

- Classic McEliece wraps the PKE into a KEM, always choosing the plaintext e uniformly at random. This prevents the attacker from choosing plaintexts e, e'. The IND-CCA2 definition for KEMs does not allow the attacker to choose plaintexts.
- Classic McEliece does not output the plaintext e: it outputs only a hash $\mathsf{H}(e)$ (or, more precisely, $\mathsf{H}(1, e, C)$ where C is the ciphertext). An attacker seeing $\mathsf{H}(e + \delta)$ cannot determine $\mathsf{H}(e)$, unless there is a serious problem with the hash function H.
- Classic McEliece does not reveal decryption failures. Instead it uses "implicit rejection": if the PKE returns $\perp$ then the KEM outputs a pseudorandom function of the ciphertext (more precisely, $\mathsf{H}(0, s, C)$ where C is the ciphertext and s is a secret included in the private key).

1999 Fujisaki–Okamoto [37] introduced a generic transform where decryption checks a reencryption of the plaintext against the ciphertext; the "rigidity" abstraction is from 2018 Bernstein–Persichetti [12]. 2001 Shoup [60] introduced hashed KEMs, along with the KEM abstraction. Implicit rejection was introduced by 2012 Persichetti [55] in the context of code-based cryptography, and generalized by 2017 Hofheinz–Hövelmanns–Kiltz [41].

Review of the security of these defenses is simplified by two further features of the underlying PKE. First, the PKE is correct (see Section 2): the decryption function always correctly outputs e given He. Second, the PKE is deterministic; reencryption does not require the preliminary derandomization step from [37], a step that could lose security.

5.3 IND-CCA2 security with a generic transform

There are generic theorems that, for any correct deterministic PKE, *almost* say that IND-CCA2 security of a KEM using reencryption, hashing, and implicit rejection is tightly related to OW-CPA security of the PKE. More precisely, there is a simple theorem tightly guaranteeing ROM IND-CCA2 security, and a more complex theorem that almost as tightly guarantees QROM IND-CCA2 security.

Reviewers are cautioned that provable security, like cryptanalysis, can have gaps (e.g., the gap between QROM IND-CCA2 security and IND-CCA2 security) and errors (e.g., [12] gave counterexamples to some previously claimed IND-CCA2 theorems). This section surveys the structure of what is known.

5.3.1 ROM IND-CCA2 security

The state-of-the-art theorem is from 2018 Bernstein–Persichetti [12, Theorem 14.3]. The proof techniques are older but are factored in [12] into three separately verifiable theorems: [12, Theorem 6.4] handling reencryption, [12, Theorem 8.2] handling encryptions of oracle queries, and [12, Theorem 13.3] handling decapsulation queries. Part of this modularization comes from [41], which was factored into (1) one theorem for derandomization and

reencryption and (2) one theorem for the remaining proof steps.

The KEM construction considered in [12, Theorem 14.3], defined in [12, Definition 14.1], works as follows:

- Start from any correct deterministic PKE.
- The KEM public key is the PKE public key.
- The KEM private key includes the PKE private key, the PKE public key (for reencryption), and an implicit-rejection key s generated uniformly at random.
- The session key for plaintext p and PKE ciphertext C is $\mathsf{H}(1, p, C)$.
- The KEM ciphertext is the same as the PKE ciphertext.
- Decapsulation of C begins by decrypting C. If this produces a plaintext p that reencrypts to C then decapsulation outputs $\mathsf{H}(1, p, C)$. Otherwise decapsulation outputs $\mathsf{H}(0, s, C)$.

Note that this is slightly different from what Classic McEliece does: for example, the Classic McEliece private key does not include a copy of the public key. See Section 5.4.

The conclusion of [12, Theorem 14.3] is that the ROM IND-CCA2 success probability of an attack A against the KEM is at most the sum of the following terms:

- The OW-CPA success probability of an attack B against the KEM.
- The number of decapsulation queries divided by the size of the plaintext space.
- The number of hash queries, times 2, divided by the size of the implicit-rejection key space.

As noted in Section 1, the selected Classic McEliece parameter sets use 256-bit implicit-rejection keys. The third term is then negligible for any plausible number of hash queries. The second term is even much smaller; e.g., the number of plaintexts e is above 2^{456} for `348864`.

The cost overhead of B compared to A includes checking an encryption of each hash query. This can be a noticeable overhead: encryption is very fast in hardware, but hashing is also very fast in hardware. It is interesting to note that taking a slower hash function (for example, more rounds of hashing) could noticeably improve this aspect of proof tightness at negligible cost to the user, and would slow down any attack bottlenecked by hashing. However, the tightness loss is small in any case (at most a few bits), and the negligible second extra term above indicates that generic hashing is not a threat in the first place.

5.3.2 QROM IND-CCA2 security

For the broader class of QROM IND-CCA2 attacks, the state-of-the-art theorem is from 2019 Bindel–Hamburg–Hövelmanns–Hülsing–Persichetti [14, Theorem 2]. This considers the same KEM as above, with the following generalizations:

- The construction begins with any "ϵ-injective" deterministic PKE. Any correct deterministic PKE is ϵ-injective with $\epsilon = 0$.
- The implicit-rejection function $\mathsf{H}(0, s, C)$ is generalized to $F(s, C)$.

Classic McEliece does not use these generalizations.

Note that $\mathsf{H}(1, p, C)$ is denoted $\mathsf{H}(p, C)$ in [14]. This makes no difference for [14, Theorem 2], but if $F(s, C)$ were set to $\mathsf{H}(s, C)$ without input-space separation then the quantitative conclusions would need to be slightly adjusted to account for the probability that p collides with s. Also, the KEM in [14, Figure 3] does not keep a copy of the public key in the private key; decapsulation in [14, Figure 3] implicitly assumes that the public key can be recomputed from the private key.

The conclusion of [14, Theorem 2] is that the QROM IND-CCA2 success probability of an attack A against the KEM is at most the sum of the following terms:

- 2 times the square root of the OW-CPA success probability of an attack B_1 against the PKE.
- The advantage of an attack B_2 at "finding failing ciphertexts" in the PKE. This advantage is 0 for a correct PKE such as the McEliece PKE.
- 2 times the PRF advantage of an attack B_3 against F. For the usual choice $F(s, C) = \mathsf{H}(0, s, C)$, [14, Corollary 1] says that the QROM PRF security of F is at most 2 times the number of hash queries divided by the square root of the size of the implicit-rejection key space.
- The quantity ϵ mentioned above in ϵ-injectivity. This is 0 for a correct PKE.

Each of the attacks uses "about the same time and resources" as A.

The square root of the size of the implicit-rejection key space corresponds to a Grover search. This is not a threat for 256-bit keys. The only tightness loss of interest thus comes from the first term: 2 times the square root of the OW-CPA success probability. For example, if the goal is to keep the IND-CCA2 success probability below 1/1000, then applying the theorem requires the OW-CPA success probability to be kept below 1/4000000.

In the context of Classic McEliece, the success probabilities of the best OW-CPA attacks known have the following shape: the best probability-δ non-quantum attacks use about $\delta 2^\lambda$ operations, and the best probability-δ^2 quantum attacks use about $\delta 2^{\lambda/2}$ operations. If these are optimal then, by [14], a QROM IND-CCA2 attack has success probability at most about 2δ using about $\delta 2^{\lambda/2}$ operations.

These proofs rely on the PKE being deterministic. QROM results are available in [48] for randomized PKEs, but those results are much looser: they need to assume a bound on PKE attack probabilities on the scale of δ/q, where q is the number of hash queries. For values of δ of interest, δ/q is much smaller than δ^2. These results also assume IND-CPA security, which is a stronger assumption than OW-CPA security. There are other results for randomized PKEs assuming only OW-CPA security, but these results are even less tight.

Classic McEliece uses a deterministic PKE, so it avoids all of these difficulties.

5.3.3 IND-CCA2 security

The theorems surveyed above do not rule out the possibility of an IND-CCA2 attack being faster than any QROM IND-CCA2 attack.

Consider, for example, an attack that predicts that the session key is 0. This is an IND-CCA2 attack with advantage almost exactly 1 if the chosen hash function H happens to always output 0. This does not contradict QROM IND-CCA2 security: the same attack has advantage almost exactly 0 against a uniform random function H.

It is easy to check experimentally that a standard hash function does not always output 0. However, it is not easy to check experimentally whether a standard hash function produces a particular output with probability, e.g., $1/2^{50}$, which would enable an attack with probability almost exactly $1/2^{50}$. This would also enable a feasible collision attack, so it is ruled out by collision resistance of the hash function, but this example shows that review of IND-CCA2 security relies on review of hash-function cryptanalysis.

One can construct artificial examples of hash functions for which it is easy to break IND-CCA2 even when it seems difficult to break the standard hash-function goals of collision resistance and preimage resistance. It would be surprising if this happened for "unstructured" hash functions, such as the standard SHAKE256 function selected in Classic McEliece; but further analysis is warranted regarding potential interactions between the security of hash functions and the security of public-key cryptography.

5.4 IND-CCA2 security with a more efficient transform

The KEM constructed in Section 5.3 is still not exactly the same as the Classic McEliece KEM, so, even if that KEM is IND-CCA2 secure, further steps are required to deduce IND-CCA2 security of Classic McEliece.

This section covers some of the differences, namely efficiency improvements in the CCA transform. These improvements make no changes to the input-output behavior visible to the attacker in the IND-CCA2 model, so they preserve IND-CCA2 security. The remaining differences are covered in Section 5.5.

5.4.1 Reducing costs of reencryption

After decapsulation has decrypted a ciphertext C to obtain a plaintext p, checking whether p reencrypts to C does not necessarily require a full computation of the encryption function.

In particular, in the Classic McEliece specification, the decryption function Decode is already guaranteed to output

- a weight-t vector e whose syndrome He is the decoding input C if such a vector exists, or
- $\perp$ otherwise.

Consequently, there is no need for the KEM to recompute He; this recomputation has no effect and is simply skipped. In other words, the PKE is already rigid.

Inside the DECODE specification, this rigidity is enforced by Step 4, which checks whether e has weight t and whether He matches the input. The matrix H is not provided as input to DECODE but, as noted in the separate "guide for implementors" document, can be recomputed via MATGEN, or skipped entirely in favor of a more efficient check of the same property.

The details of the more efficient check are relevant for reviewers who are checking that a claimed implementation correctly computes the specified decapsulation algorithm. In short, the problem is to check whether $c = v+e$ is in the Goppa code defined by $(g, \alpha_0, \ldots, \alpha_{n-1})$ (see Section 2.1), and this is simply a matter of checking whether the polynomial $\sum_i c_i A/(x-\alpha_i)$ is a multiple of g in the polynomial ring $\mathbb{F}_q[x]$. This computation typically reuses the fast syndrome-computation subroutine used at the beginning of decoding; see [7], [8], and [23].

Step 2 of DECODE is specified to either fail or find c with $Hc = 0$. The usual decoding algorithm is shown in [7, Section 7] to guarantee that $Hc = 0$ even without checking this; $Hc = 0$, in turn, implies that He matches the input. However, having a separate check that He matches the input makes the decapsulation procedure more robust; see [7, Section 8]. The specification is written with a separate check of He in Step 4 even though this is mathematically unnecessary. The specification also requires checking the weight of e.

5.4.2 Reducing space requirements for the private key

The private key in the KEM in Section 5.3 includes the PKE private key, the PKE public key, and the implicit-rejection key s. This space consumption can usually be improved: for example, if the public key can be computed efficiently from the private key, then it can be recomputed on demand, or optionally cached.

In the Classic McEliece specification, the situation is even simpler: decapsulation does not look at the PKE public key (see Section 5.4.1), so the PKE public key is simply eliminated from the KEM private key. This preserves IND-CCA2 security.

5.4.3 Simplifying generation of the implicit-rejection key

The round-1 Classic McEliece submission cited ROM/QROM IND-CCA2 proofs in which the implicit-rejection key was taken from the set of plaintexts. See, e.g., [41, Figure 12].

In Classic McEliece, the implicit-rejection key s is generated from a larger space than the plaintext space. It is simpler to generate a uniform random n-bit string than to generate a

uniform random weight-t n-bit string.

The documentation thus included an extra step to point out that this was safe: "The set of s enters into the security analysis solely for the indistinguishability of $\mathsf{H}_0(s, C)$ from uniform random." Formally, however, reviewing this statement required re-checking proof details rather than using theorems as black boxes.

Subsequent theorems allowed the implicit-rejection key space to be larger than the plaintext space (see, e.g., [57], [12], and [14]), so this extra step in the analysis is no longer required.

5.5 Generation of random objects

The result of all of the cryptosystem transformations described in previous sections is the "Model Classic McEliece KEM" defined in Section 5.5.3.

This is still not exactly Classic McEliece. For reasons explained in the separate "design rationale" document, Classic McEliece uses functions KeyGen and Encap that are different from the functions ModelKeyGen and ModelEncap in Model Classic McEliece:

- Where ModelKeyGen says "generate a uniform random monic irreducible polynomial g" and "generate a uniform random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ of n distinct elements of $\mathbb{F}_q$", KeyGen instead specifies a particular function generating $(g, \alpha_0, \ldots, \alpha_{n-1})$, using a source of uniform random *bits*.
- Where ModelEncap says "generate a uniform random vector $e \in \mathbb{F}_2^n$ of weight t", Encap instead specifies a particular function generating e, again using a source of uniform random bits.

Security analysis is factored through Model Classic McEliece. Previous sections of this document study IND-CCA2 security of Model Classic McEliece. Sections 5.5.4 and 5.5.5 study IND-CCA2 security of Classic McEliece, assuming IND-CCA2 security of Model Classic McEliece. This modularization shields most of the IND-CCA2 analysis from the details of how $g, \alpha_0, \ldots, \alpha_{n-1}, e$ are generated.

There is an overlap between this section and the separate "design rationale" document, but this section focuses on supporting IND-CCA2 security review of the specified functions, while the corresponding section of the other document is explaining *why* these functions were chosen.

5.5.1 Model key generation

The following randomized algorithm ModelKeyGen takes no input. It outputs a public key and private key. Here is the algorithm:

1. Generate a uniform random n-bit string s.
2. Generate a uniform random monic irreducible polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t.

3. Generate a uniform random sequence $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ of n distinct elements of $\mathbb{F}_q$.
4. Define $\Gamma = (g, \alpha_0, \alpha_1, \ldots, \alpha_{n-1})$.
5. Compute $(T, c_{mt-\mu}, \ldots, c_{mt-1}, \Gamma') = \textsc{MatGen}(\Gamma)$. If this fails, restart the algorithm.
6. Output T as the public key, and (Γ', s) as the private key.

The MatGen function used inside ModelKeyGen is defined in the separate "cryptosystem specification" document.

Compared to the output $(\delta, c, g, \alpha, s)$ of KeyGen where $\alpha = (\alpha'_0, \ldots, \alpha'_{n-1}, \alpha_n, \ldots, \alpha_{q-1})$, the output (Γ', s) of ModelKeyGen includes the information used by Decap, namely $\Gamma' = (g, \alpha'_0, \ldots, \alpha'_{n-1})$ and s; Decap does not use δ, c, or $\alpha_n, \ldots, \alpha_{q-1}$. See Section 5.5.5 for other differences between ModelKeyGen and KeyGen.

5.5.2 Model encapsulation

The following randomized algorithm ModelEncap takes as input a public key T. It outputs a ciphertext C and a session key K. Here is the algorithm:

1. Generate a uniform random vector $e \in \mathbb{F}_2^n$ of weight t.
2. Compute $C = \textsc{Encode}(e, T)$.
3. Compute $K = \mathsf{H}(1, e, C)$.
4. Output ciphertext C and session key K.

The Encode function used inside ModelEncap is defined in the separate "cryptosystem specification" document.

5.5.3 The Model Classic McEliece KEM

By definition, Model Classic McEliece uses the ModelKeyGen function from Section 5.5.1 for key generation, the ModelEncap function from Section 5.5.2 for encapsulation, and the Decap function from the separate "cryptosystem specification" document for decapsulation.

5.5.4 Random objects in encapsulation

The only difference between ModelEncap and Encap is in how the vector e is generated: ModelEncap generates e as a uniform random weight-t vector, whereas Encap generates e by calling the FixedWeight subroutine from the separate "cryptosystem specification" document. The point of the following analysis is that the output of FixedWeight is a uniform random weight-t vector, so there is no change in IND-CCA2 security.

FixedWeight begins by generating a uniform random $\sigma_1\tau$-bit string, where $\sigma_1 \geq m$ and $\tau \geq t$. It takes $m\tau$ of these bits (from separate positions) to form τ m-bit integers $d_0, d_1, \ldots, d_{\tau-1}$.

The sequence $d = (d_0, d_1, \ldots, d_{\tau-1})$ is a uniform random element of $\{0, 1, \ldots, 2^m - 1\}^\tau$.

Define $u = \#\{j : d_j < n\}$. Define $a = (a_0, a_1, \ldots, a_{u-1})$ as the corresponding subsequence of d.

For any particular u, there are exactly n^u possibilities for the sequence a. For each choice of sequence a, there are exactly $\binom{\tau}{u}(q-n)^{\tau-u}$ sequences $(d_0, d_1, \ldots, d_{\tau-1})$ producing a: the factor $\binom{\tau}{u}$ is the number of choices of the set $\{j : d_j < n\}$, and the factor $(q-n)^{\tau-u}$ is the number of choices for the complementary subsequence. (As a double-check, $\sum_u \binom{\tau}{u}(q-n)^{\tau-u}n^u$ is q^τ, which is the number of possible sequences d.) Hence, given u, the conditional distribution of a is uniform.

FixedWeight restarts if $u < t$. Otherwise it computes $(a_0, a_1, \ldots, a_{t-1})$. For any particular $u \geq t$, the conditional distribution of $(a_0, a_1, \ldots, a_{t-1})$ is the uniform distribution on $\{0, \ldots, n-1\}^t$; hence the total distribution of $(a_0, a_1, \ldots, a_{t-1})$ is also the uniform distribution on $\{0, \ldots, n-1\}^t$.

FixedWeight then restarts if $a_0, a_1, \ldots, a_{t-1}$ are not all distinct. If this restart does not occur then the conditional distribution of $(a_0, a_1, \ldots, a_{t-1})$ is the uniform distribution on sequences of t distinct elements of $\{0, 1, \ldots, n-1\}$.

FixedWeight then outputs the vector $(e_0, \ldots, e_{n-1}) \in \mathbb{F}_2^n$ having bits set at exactly positions $a_0, a_1, \ldots, a_{t-1}$. This is a uniform random weight-t vector.

Formally, what this shows is that *if* FixedWeight terminates *then* its output is a uniform random weight-t vector. The algorithm terminates with probability 1, since, for example, the sequence $d = (0, 1, \ldots, \tau - 1)$ occurs with positive probability, namely $1/2^{m\tau}$, on each restart and produces output, namely $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$.

5.5.5 Random objects in key generation

There are more differences between ModelKeyGen and KeyGen. The following analysis covers these differences from bottom up.

Irreducible-polynomial generation. The function Irreducible takes $\sigma_1 t$ input bits; the parameter σ_1 is the same as in Section 5.5.4, with $\sigma_1 \geq m$. Assume for now that this is a uniform random element of $\{0, 1\}^{\sigma_1 t}$.

Irreducible takes mt of these bits to form t elements $\beta_0, \ldots \beta_{t-1}$ of $\mathbb{F}_q$ and then an element $\beta = \beta_0 + \cdots + \beta_{t-1}y^{t-1}$ of $\mathbb{F}_q[y]/F(y)$. This is a uniform random element of $\mathbb{F}_q[y]/F(y)$.

Irreducible then computes the unique monic irreducible polynomial $g \in \mathbb{F}_q[x]$ such that $g(\beta) = 0$, and returns g if it has degree t, else $\perp$.

Each degree-t monic irreducible polynomial in $\mathbb{F}_q[x]$ has exactly t distinct roots in the field $\mathbb{F}_q[y]/F(y)$ and is thus returned for exactly t choices of β. Consequently, if Irreducible does not fail, then the conditional distribution of g is the uniform distribution on degree-t

monic irreducible polynomials in $\mathbb{F}_q[x]$.

Field-ordering generation. The function FieldOrdering takes $\sigma_2 q$ input bits, with $\sigma_2 \geq 2m$. Assume for now that this is a uniform random element of $\{0,1\}^{\sigma_2 q}$.

FieldOrdering converts these bits into a uniform random sequence of q elements $a_0, a_1, \ldots, a_{q-1}$ of $\{0, 1, \ldots, 2^{\sigma_2} - 1\}^q$. It returns $\perp$ if any of these elements collide.

If no collision occurs then FieldOrdering has a uniform random sequence of q *distinct* elements of $\{0, 1, \ldots, 2^{\sigma_2} - 1\}^q$. There is then a unique permutation that sorts these elements, a uniform random permutation of $\{0, 1, \ldots, q-1\}$. The output of FieldOrdering is the same permutation applied to a standard ordering of $\mathbb{F}_q$; this output is a uniform random ordering of all field elements.

Key generation. KeyGen generates a uniform random ℓ-bit seed δ and then calls SeededKeyGen, which calls G to expand δ to a string of $n + \sigma_2 q + \sigma_1 t + \ell$ bits.

The latter string is indistinguishable from uniform under a standard PRG assumption on G. The selected Classic McEliece parameter sets use $\ell = 256$ seed bits, and derive G from SHAKE256; the review of IND-CCA2 security relies on review of the security of SHAKE256 as a stream cipher with a 256-bit key.

The rest of the analysis here relies on this indistinguishability and replaces the string with a uniform random string. This string is then decomposed into an n-bit string s, a $\sigma_2 q$-bit string given to FieldOrdering, a $\sigma_1 t$-bit string given to Irreducible, and an ℓ-bit string δ'; these are independent uniform random strings.

If FieldOrdering succeeds then it generates a uniform random ordering $(\alpha_0, \ldots, \alpha_{q-1})$ of $\mathbb{F}_q$, as in ModelKeyGen. If Irreducible succeeds then it generates a uniform random monic irreducible polynomial g of degree t, as in ModelKeyGen. If MatGen succeeds then it generates the same results as in ModelKeyGen. In short, the KeyGen output, like the ModelKeyGen output, is a parity-check matrix for a uniform random binary Goppa code in systematic form for $(\mu, \nu) = (0, 0)$, or semi-systematic form for general (μ, ν).

The fact that some seeds fail means that the final seed has "only" about 254 bits of entropy. The easiest way to check this is to experimentally observe the distribution of the number of restarts of SeededKeyGen on random seeds. Like most private keys in most cryptographic algorithms, the final seed is efficiently distinguishable from uniform; here by having the property that SeededKeyGen succeeds without restarts. A search through seeds still costs more than AES-256 key search.

6 Security notions beyond IND-CCA2

6.1 Multi-target security

In a multi-message attack scenario, the cost of finding the private key is spread across many messages. There are also faster multi-message attacks that do not rely on finding the private key; see, e.g., [43] and [59].

The difficult way to handle multi-message security is to incorporate multiple messages into every step of the security analysis, trying to identify and protect any system components that can enable multi-target attacks. The easy way to handle multi-message security is to take a very high security level and rely on the general fact that attacking T targets cannot gain more than a factor T. The point is that there is a standard way to use a T-target attack as a 1-target attack with probability $1/T$: simply insert the real target somewhere into a list of $T-1$ simulated targets, and hope that the T-target attack happens to pick the real target.

For example, for the recommended `6688128` and `6960119` parameter sets, one ciphertext is expected to be secure against an attacker without the resources to find an AES-256 key, so 2^{64} ciphertexts are expected to all be secure against an attacker without the resources to find an AES-192 key.

For quantum attacks, the same logic, accounting for the square-root loss of tightness in [14] (and making the worst-case assumption that there is a corresponding security loss), says that 2^{64} ciphertexts are expected to all be secure against an attacker without the resources to carry out a 1-target OW-CPA attack with probability roughly $1/2^{128}$. This is, when attacks have Grover-type scaling, roughly the same as an attacker without resources $R/2^{64}$, where R means the resources to carry out a high-probability 1-target attack.

6.2 Indistinguishability of ciphertexts

For KEMs, IND-CCA2 asks whether an attacker can distinguish a session key from uniform. For PKEs, IND-CCA2 asks whether an attacker can distinguish a ciphertext for plaintext m_0 from a ciphertext for plaintext m_1.

More broadly, the literature often asks which cryptographic objects can be distinguished from uniform random strings (of the same length), and which cryptographic objects (of the same length) can be distinguished from each other. For example, the definition of KEM "anonymity" in [38], motivated by protecting metadata, asks whether an attacker given two legitimately generated public keys can distinguish the output of encapsulation (the ciphertext and session key) for the first key from the output of encapsulation for the second key. The definition of "weak anonymity" asks merely whether the ciphertexts are distinguishable.

For the McEliece cryptosystem, cryptanalysis has focused primarily on one-wayness, recovering a weight-t vector e from $Ga+e$ or equivalently He. Breaking one-wayness would

- distinguish the resulting session key from uniform,
- distinguish He from a uniform random string, and
- distinguish He from a ciphertext for another public key.

Conversely, the difficulty of breaking one-wayness is the foundation of the IND-CCA2 security analysis for Classic McEliece in Section 5, but does not directly imply a lack of more general ciphertext distinguishers. As a trivial example, the `6960119` parameter set uses zero-padding to encode 1547-bit ciphertexts as 194-byte (1552-bit) strings, so those strings are distinguishable from uniform random 194-byte strings. Whether ciphertexts for two public keys are distinguishable from each other is an open question.

6.3 Key specificity of decryptable ciphertexts

Consider the following property of a cryptosystem: given two (legitimately generated) public keys, the attacker has negligible probability of finding a ciphertext for which dec succeeds (i.e., does not return $\perp$) under both of the corresponding private keys.

This property was denoted "strong robustness" in [1]. Formally, the definition in [1] considers a large space of user identities, with a key for each identity, and asks the attacker to find two distinct identities and a ciphertext for which dec succeeds under both identities; also, the definition was given only for PKEs (with dec meaning decryption), not for KEMs (with dec meaning decapsulation). However, one can also consider the same definition, with or without identities, for KEMs. See, e.g., [38, Figure 2].

This property of KEMs is incompatible with building KEMs using implicit rejection: for implicit-rejection KEMs, decapsulation succeeds for every ciphertext. Most proposals for post-quantum IND-CCA2 KEMs use implicit rejection, and therefore do not provide this property, as noted in [38]. In particular, Classic McEliece uses implicit rejection, and therefore does not provide this property.

A KEM that does not provide this property can nevertheless be used to construct a PKE providing this property:

- By [38, Theorem 2], it suffices for the KEM to be "strongly collision-free", meaning that the attacker cannot find a ciphertext C for which $\textsc{Decap}(C, k_1) = \textsc{Decap}(C, k_2) \neq \perp$. The theorem also needs a suitable DEM, a requirement that [38] describes as "easily met".
- This collision-freeness property, in turn, is achieved by the following generic transformation: replace each session key k with a hash of (I, k), where I is the user identity. (In formalizations that do not include identities, define the identity as the public key, or as a precomputed hash of the public key, or simply enough leading bits of the public key to avoid collisions among legitimate keys.) Finding a collision of these hashes between two different identities implies finding a collision in the hash function.

The same generic transformation can equivalently be applied to the DEM, or kept as an

intermediate layer between the KEM and the DEM.

This generic transformation is not necessarily required. For example, for Classic McEliece, perhaps the collisions of weight-t ciphertexts (if C has weight t then it decrypts to $(C, 0, \ldots, 0)$ for every key, as noted in [38, Section 5.1]) are the only collisions that are feasible to construct, which would mean that it suffices to reject these ciphertexts. These ciphertexts have negligible chance of occurring legitimately, for the same reason that a single iteration of Prange's algorithm has negligible success chance; see Section 3.2. However, such KEM-specific analyses and modifications are less convincing and more work than adding a low-cost hash.

As noted in the separate "design rationale" document, Shoup argued in [60, Section 3.3] that KEMs should avoid hashing "labels" such as identities since "it is easier to implement labels in the data encapsulation mechanism"; Classic McEliece follows the principle that any generic transformation aiming at a goal beyond IND-CCA2 is out of scope for a KEM specification. This is not saying that further hashing should be avoided; it is saying that cryptographic systems should be appropriately modularized.

6.4 Security with cycling RNGs

Many different cryptographic specifications are defined using uniform random bits, and are implemented using bits that come from various RNGs. The security analysis of RNGs asks whether the RNG output is indistinguishable from uniform.

If an RNG is stuck in a short cycle, the standard response is that the RNG is deficient and must be replaced. A much more complicated response is to ask what impact this cycle has on applications using the RNG, and to try to modify applications to protect against this cycle.

Consider, for example, an application in which Alice receives a KEM public key from Bob and sends a ciphertext to that key, receives a KEM public key from Charlie and sends a ciphertext to that key, etc. A cycle in Alice's RNG will (for enough users, depending on the cycle length) repeat the randomness used for the ciphertexts. This easily breaks many cryptosystems.

In particular, for Classic McEliece, one can imitate the 1997 Berson attack [13], which showed how to break related messages in the original McEliece cryptosystem; or, more trivially, observe that a message He sent by Alice to Eve's KEM public key will give e to the attacker Eve, so if e is repeated in a ciphertext for another user then Eve can decapsulate that ciphertext too.

The following generic transformation ensures that randomness is not repeated across target identities even when the RNG is cycling: instead of taking random bits for encapsulation directly from the RNG, take them from a standard PRNG, where the PRNG seed is a hash of the identity and randomness from the RNG. (As in Section 6.3, appropriate extracts from the public key can be used as identities in formalizations that do not include identities.)

Because this is a generic transformation aiming at a goal beyond IND-CCA2, it is out of scope for the Classic McEliece specification. Commonly used RNGs, including NIST's standard RNGs (see [2, Section 8.7.2]), already provide an interface for applications to provide "additional input" to be hashed into the RNG state; applications are free to provide the target identity as additional input before calling encapsulation in any KEM.

Applications considering this generic transformation should beware that the security provided by the transformation is much weaker than the security provided by fixing the RNG. A cycling RNG will also repeat randomness used to generate ephemeral keys, randomness used to generate nonces, randomness provided to programs that the user's browser is running from Eve's web page, etc.; the resulting security problems go far beyond KEM encapsulation. Furthermore, the entire idea that deterministic postprocessing can rescue security when the RNG is repeating outputs relies on the assumption that there is enough entropy in the RNG; but the famous real-world example in [27] of an RNG repeating outputs was an example where the RNG had negligible entropy to begin with.

6.5 Security against natural private-key faults

The following extension of IND-CCA2 security is considered in [6]: some decapsulation queries occur; then a single bit is flipped at a random position in a stored private key; then further decapsulation queries occur against the modified private key. The motivation for considering this specific type of fault is that, even when attackers are stopped from triggering faults, statistics collected by Google regarding naturally occurring DRAM faults indicate that a noticeable fraction of keys stored in non-ECC DRAM, roughly 0.01% of all 256-bit keys, will be corrupted every year.

The current version of NTRU-HRSS is shown in [6] to not provide security in this model. The point is that the IND-CCA2 security provided by implicit rejection disappears when a bit is flipped in the implicit-rejection key. Plaintext confirmation (from [29]) does not have this issue, but NTRU-HRSS removed plaintext confirmation in 2019. Similarly, the round-4 version of Classic McEliece does not have plaintext confirmation, and does not provide security in this model.

However, the following generic transformation converts IND-CCA2 security into this extension of IND-CCA2 security: store the private key encoded using an error-correcting code of distance at least 3, and apply a 1-error-correcting decoder when reading the private key. A conventional choice of 1-error-correcting code is a distance-4 extended Hamming code.

Because this is a generic transformation aiming at a goal beyond IND-CCA2, it is out of scope for the Classic McEliece specification. Error correction is nevertheless recommended as low-cost protection against this type of fault, even in applications not otherwise concerned with faults. For devices without ECC DRAM, see [5] for software efficiently encoding and decoding arbitrary byte arrays using an extended Hamming code.

6.6 Security against back doors

The area of kleptography studies how to backdoor cryptosystems into leaking information so that only the holder of some secret back-door key can obtain plaintext from the leaked information. Typically the software differs dramatically from legitimate software; these back doors rely on implementations being used as a black box. A straightforward defense is to check the implementations: specifically, check that open-source software uses a trustworthy source of randomness, computes exactly the functions specified in the separate "cryptosystem specification" document, and is compiled by a trustworthy compiler into the binaries that users are running. Checking the function outputs is not enough: one also has to check that implementations are not leaking information through timing channels, electromagnetic channels, sending email to the attacker, etc.

For the case that black-box implementations are used, Young and Yung [67] define two types of SETUP (Secretly Embedded Trapdoor with Universal Protection): strong and weak SETUP. A back door is a weak SETUP if it is impossible for anybody but the owner of the back-door key to determine that the system is backdoored, given just the publicly observable outputs (public keys, ciphertexts, signatures, timing information) while it is a strong SETUP if this holds even if the secret inputs and outputs (private keys, plaintexts) are available additionally.

The paper [40] showed a strong SETUP against a version of McEliece they called Vanilla McEliece and a weak SETUP against Classic McEliece with uncompressed public keys. The underlying technique was already mentioned in [36, page 12]: "a malicious permutation-update mechanism can easily leak secrets through entries of the matrix that it produces as output". This SETUP is weak because a backdoored private key $(\delta, c, g, \alpha', s)$ has the field ordering α' inconsistent with the seed δ and thus is easily distinguished from a properly generated key for that δ. Note that checking consistency of α with δ is cheaper than recomputing α (see also "Double-checks on private keys" in the separate "guide for implementors" document).

The paper [40] does not provide any SETUP if the private key is compressed to (δ) or (δ, c) or (δ, c, g). However, as in other cryptosystems, further SETUPs could exist, so in all cases it is recommended to check the implementations.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9–11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, 2010. https://eprint.iacr.org/2008/440.

[2] Elaine Barker and John Kelsey. SP 800-90A rev. 1: Recommendation for random number generation using deterministic random bit generators, 2015. https://csrc.

nist.gov/publications/detail/sp/800-90a/rev-1/final.

[3] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1+1=0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology—EUROCRYPT 2012—31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15–19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012. https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/isd-extended.pdf.

[4] Daniel J. Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25–28, 2010. Proceedings*, volume 6061 of *Lecture Notes in Computer Science*, pages 73–80. Springer, 2010. https://cr.yp.to/papers.html#grovercode.

[5] Daniel J. Bernstein. `libsecded`, 2022. https://pqsrc.cr.yp.to/downloads.html.

[6] Daniel J. Bernstein. A one-time single-bit fault leaks all previous NTRU-HRSS session keys to a chosen-ciphertext attack, 2022. https://cr.yp.to/papers.html#ntrw, to appear at INDOCRYPT 2022.

[7] Daniel J. Bernstein. Understanding binary-Goppa decoding, 2022. https://cr.yp.to/papers.html#goppadecoding.

[8] Daniel J. Bernstein, Tung Chou, and Peter Schwabe. McBits: Fast constant-time code-based cryptography. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems—CHES 2013—15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 250–272. Springer, 2013. https://tungchou.github.io/papers/mcbits.pdf.

[9] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes A. Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17–19, 2008, Proceedings*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2008. https://eprint.iacr.org/2008/318.

[10] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology—CRYPTO 2011—31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14–18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer, 2011. https://eprint.iacr.org/2010/585.

[11] Daniel J. Bernstein, Tanja Lange, Christiane Peters, and Henk C. A. van Tilborg. Explicit bounds for generic decoding algorithms for code-based cryptography. In *Pre-proceedings of WCC 2009*, pages 168–180, 2009.

[12] Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification, 2018. https://eprint.iacr.org/2018/526.

[13] Thomas A. Berson. Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1997. https://doi.org/10.1007/BFb0052237.

[14] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography—17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 61–90. Springer, 2019. https://eprint.iacr.org/2019/590.

[15] Leif Both and Alexander May. Optimizing BJMM with nearest neighbors: Full decoding in $2^{2n/21}$ and McEliece security, 2017. International Workshop on Coding and Cryptography (WCC 2017). https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/bjmm+.pdf.

[16] Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Lange and Steinwandt [49], pages 25–46. https://eprint.iacr.org/2017/1139.

[17] Anne Canteaut and Herve Chabanne. A further improvement of the work factor in an attempt at breaking McEliece's cryptosystem. In Pascale Charpin, editor, *Livre des résumés—EUROCODE 94, Abbaye de la Bussière sur Ouche, France, October 1994*, 1994. https://hal.inria.fr/inria-00074443.

[18] Anne Canteaut and Florent Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. Information Theory*, 44(1):367–378, 1998. https://www.rocq.inria.fr/secret/Anne.Canteaut/Publications/Canteaut_Chabaud98.pdf.

[19] Anne Canteaut and Nicolas Sendrier. Cryptanalysis of the original McEliece cryptosystem. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology—ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18–22, 1998, Proceedings*, volume 1514 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 1998. https://www.rocq.inria.fr/secret/Anne.Canteaut/Publications/Canteaut_Sendrier98.pdf.

[20] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. *CoRR*, abs/2208.02201, 2022. https://doi.org/10.48550/arXiv.2208.02201.

[21] Herve Chabanne and Bernard Courteau. Application de la méthode de décodage itérative d'Omura à la cryptanalyse du système de McEliece, 1993. Université de Sherbrooke, Rapport de Recherche, number 122.

[22] Florent Chabaud. Asymptotic analysis of probabilistic algorithms for finding short codewords. In Paul Camion, Pascale Charpin, and Sami Harari, editors, *Eurocode '92: proceedings of the international symposium on coding theory and applications held in Udine, October 23–30, 1992*, pages 175–183. Springer, 1993.

[23] Tung Chou. McBits revisited. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems—CHES 2017—19th International Conference, Taipei, Taiwan, September 25–28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 213–231. Springer, 2017. https://tungchou.github.io/papers/mcbits_revisited.pdf.

[24] George C. Clark, Jr. and J. Bibb Cain. *Error-correcting coding for digital communication*. Plenum, 1981.

[25] John T. Coffey and Rodney M. Goodman. The complexity of information set decoding. *IEEE Transactions on Information Theory*, 35:1031–1037, 1990.

[26] John T. Coffey, Rodney M. Goodman, and P. Farrell. New approaches to reduced complexity decoding. *Discrete and Applied Mathematics*, 33:43–60, 1991. https://core.ac.uk/reader/81155220.

[27] Debian. Debian security advisory DSA-1571-1 openssl – predictable random number generator, 2008. https://www.debian.org/security/2008/dsa-1571.

[28] Thomas Debris-Alazard, Léo Ducas, and Wessel P. J. van Woerden. An algorithmic reduction theory for binary codes: LLL and more. *IEEE Trans. Inf. Theory*, 68(5):3426–3444, 2022. https://eprint.iacr.org/2020/869.

[29] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2003. https://eprint.iacr.org/2002/174.

[30] Ilya I. Dumer. Two decoding algorithms for linear codes. *Problemy Peredachi Informatsii*, 25:24–32, 1989. http://www.mathnet.ru/eng/ppi635.

[31] Ilya I. Dumer. On minimum distance decoding of linear codes. In Grigori A. Kabatianskii, editor, *Fifth joint Soviet-Swedish international workshop on information theory, Moscow, 1991*, pages 50–52, 1991.

[32] Orr Dunkelman and Stefan Dziembowski, editors. *Advances in Cryptology—EUROCRYPT 2022—41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III*, volume 13277 of *Lecture Notes in Computer Science*. Springer, 2022. https://doi.org/10.1007/978-3-031-07082-2.

[33] Andre Esser and Emanuele Bellini. Syndrome decoding estimator. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography—PKC 2022—25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8–11, 2022, Proceedings, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 112–141. Springer, 2022. https://eprint.iacr.org/2021/1243.

[34] Andre Esser, Alexander May, and Floyd Zweydinger. McEliece needs a break—solving McEliece-1284 and Quasi-Cyclic-2918 with modern ISD. In Dunkelman and Dziembowski [32], pages 433–457. https://eprint.iacr.org/2021/1634.

[35] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In Mitsuru Matsui, editor, *Advances in Cryptology—ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6–10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009. https://eprint.iacr.org/2009/414.

[36] Classic McEliece Comparison Task Force. Classic McEliece vs. NTS-KEM. 2018. https://classic.mceliece.org/nist/vsntskem-20180629.pdf.

[37] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology—CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[38] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Dunkelman and Dziembowski [32], pages 402–432. https://eprint.iacr.org/2021/708.

[39] Yann Hamdaoui and Nicolas Sendrier. A non asymptotic analysis of information set decoding, 2013. https://eprint.iacr.org/2013/162.

[40] Tobias Hemmert, Alexander May, Johannes Mittmann, and Carl Richard Theodor Schneider. How to backdoor (Classic) McEliece and how to guard against backdoors. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography—13th International Workshop, PQCrypto 2022, Virtual Event, September 28–30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 24–44. Springer, 2022. https://eprint.iacr.org/2022/362.

[41] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography—15th International Conference, TCC 2017, Baltimore, MD, USA, November 12–15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017. https://eprint.iacr.org/2017/604.

[42] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In Dan Boneh, editor, *Advances in Cryptology—CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer, 2003. https://iacr.org/archive/crypto2003/27290225/27290225.pdf.

[43] Thomas Johansson and Fredrik Jönsson. On the complexity of some cryptographic problems based on the general decoding problem. *IEEE Trans. Information Theory*, 48(10):2669–2678, 2002. https://doi.org/10.1109/TIT.2002.802608.

[44] Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography—8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26–28, 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 69–89. Springer, 2017. https://eprint.iacr.org/2017/213.

[45] Elena Kirshanova. Improved quantum information set decoding. In Lange and Steinwandt [49], pages 507–527. https://arxiv.org/abs/1808.00714v1.

[46] Elena Kirshanova and Alexander May. Decoding McEliece with a hint—secret Goppa key parts reveal everything. In Clemente Galdi and Stanislaw Jarecki, editors, *Security and Cryptography for Networks—13th International Conference, SCN 2022, Amalfi, Italy, September 12–14, 2022, Proceedings*, volume 13409 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2022. https://eprint.iacr.org/2022/525.

[47] Evgueni A. Krouk. Decoding complexity bound for linear block codes. *Problemy Peredachi Informatsii*, 25:103–107, 1989. http://www.mathnet.ru/eng/ppi665.

[48] Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology—EUROCRYPT 2020—39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 703–728. Springer, 2020. https://doi.org/10.1007/978-3-030-45727-3_24.

[49] Tanja Lange and Rainer Steinwandt, editors. *Post-Quantum Cryptography—9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9–11, 2018, Proceedings*, volume 10786 of *Lecture Notes in Computer Science*. Springer, 2018. https://doi.org/10.1007/978-3-319-79063-3.

[50] Pil Joong Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In Christoph G. Günther, editor, *Advances in Cryptology—EUROCRYPT '88, Workshop on the Theory and Application of of Cryptographic Techniques, Davos, Switzerland, May 25-27, 1988, Proceedings*, volume 330 of *Lecture Notes*

in Computer Science, pages 275–280. Springer, 1988. https://doi.org/10.1007/3-540-45961-8_25.

[51] Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Information Theory*, 34(5):1354–1359, 1988. https://doi.org/10.1109/18.21270.

[52] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology—ASIACRYPT 2011—17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4–8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011. https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/decoding.pdf.

[53] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology—EUROCRYPT 2015—34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26–30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228. Springer, 2015. https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/paper/codes.pdf.

[54] Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 95–145. Springer Berlin Heidelberg, 2009.

[55] Edoardo Persichetti. *Improving the efficiency of code-based cryptography*. PhD thesis, University of Auckland, 2012. https://www.math.auckland.ac.nz/~sgal018/EdoardoPhD.pdf.

[56] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, IT-8:S5–S9, 1962.

[57] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology—EUROCRYPT 2018—37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29–May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 520–551. Springer, 2018. https://eprint.iacr.org/2017/1005.pdf.

[58] Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Trans. Information Theory*, 46(4):1193–1203, 2000. https://doi.org/10.1109/18.850662.

[59] Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography—4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November*

29–December 2, 2011. Proceedings, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2011. https://eprint.iacr.org/2011/367.

[60] Victor Shoup. A proposal for an ISO standard for public key encryption, 2001. https://eprint.iacr.org/2001/112.

[61] Jacques Stern. A method for finding codewords of small weight. In Gérard D. Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2–4, 1988, Proceedings*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988. https://doi.org/10.1007/BFb0019850.

[62] Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography—7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24–26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2016. https://hal.inria.fr/hal-01244886v1/document.

[63] Henk C. A. van Tilborg. *Coding theory, a first course*. 1993. https://www.win.tue.nl/~henkvt/images/CODING.pdf.

[64] Johan van Tilburg. On the McEliece public-key cryptosystem. In Shafi Goldwasser, editor, *Advances in Cryptology—CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21–25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 119–131. Springer, 1988. https://doi.org/10.1007/0-387-34799-2_10.

[65] Johan van Tilburg. *Security-analysis of a class of cryptosystems based on linear error-correcting codes*. PhD thesis, Technische Universiteit Eindhoven, 1994.

[66] Eric R. Verheul, Jeroen M. Doumen, and Henk C. A. van Tilborg. Sloppy Alice attacks! Adaptive chosen ciphertext attacks on the McEliece public-key cryptosystem. In Mario Blaum, Patrick G. Farrell, and Henk C. A. van Tilborg, editors, *Information, coding and mathematics*, volume 687 of *Kluwer International Series in Engineering and Computer Science*, pages 99–119. Kluwer, 2002.

[67] Adam L. Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology—EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11–15, 1997, Proceedings*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1997. https://cryptome.org/2013/09/klepto-crypto.pdf.