Cryptanalysis of LEDAcrypt

No Author Given

No Institute Given

Abstract. We report on the concrete cryptanalysis of LEDAcrypt, a 2nd Round candidate in NIST's Post-Quantum Cryptography standardization process and one of 17 encryption schemes that remain as candidates for near-term standardization. LEDAcrypt consists of a public-key encryption scheme built from the McEliece paradigm and a key-encapsulation mechanism (KEM) built from the Niederreiter paradigm, both using a quasi-cyclic low-density parity-check (QC-LDPC) code.
In this work, we identify a large class of extremely weak keys and provide an algorithm to recover them. For example, we demonstrate how to recover 1 in $2^{47.79}$ of LEDAcrypt's keys using only $2^{18.72}$ guesses at the 256-bit security level. This is a major, practical break of LEDAcrypt. Further, we demonstrate a continuum of progressively less weak keys (from extremely weak keys up to all keys) that can be recovered in substantially less work than previously known. This demonstrates that the imperfection of LEDAcrypt is fundamental to the system's design.

Keywords: NIST PQC, LEDAcrypt, McEliece, QC-LDPC, Cryptanalysis

1 Introduction

Since Shor's discovery [28] of a polynomial-time quantum algorithm for factoring integers and solving discrete logarithms, there has been a substantial amount of research on quantum computers. If large-scale quantum computers are ever built, they will be able to break many of the public-key cryptosystems currently in use. This would gravely undermine the integrity and confidentiality of our current communications infrastructure on the Internet and elsewhere.

In response, the National Institute of Standards and Technologies (NIST) initiated a process [1] to solicit, evaluate, and standardize one or more quantum-resistant, public-key cryptographic algorithms. This process began in late 2017 with 69 submissions from around the world of post-quantum key-establishment mechanisms or KEMs (resp. public-key encryption schemes or PKEs), and digital signature algorithms. In early 2019, the list of candidates was cut from 69 to 24 (17 of which are PKEs), and the 2nd Round of the competition began [3]. The conclusion of Round 2 is now rapidly approaching.

LEDAcrypt [4] is one of the 17 remaining candidates for standardization as a post-quantum PKE scheme. It is based on the seminal works of McEliece [21] in 1978 and Niederreiter [24] in 1986, which are based on the NP-complete problem of decoding an arbitrary linear binary code [5]. More precisely, LEDAcrypt is composed of a PKE scheme based on McEliece but instantiated with a particular type of codes (called QC-LDPC) and a KEM in the variant style of Niederreiter. The specific origins of LEDAcrypt – the idea of using QC-LDPC codes with the McEliece paradigm – dates back a dozen years to [17].

At a very high level, the private key of LEDAcrypt is a pair of binary matrices H and Q, where H is a sparse, quasi-cyclic, parity-check matrix of dimension $p \times p \cdot n_0$ for a given QC-LDPC code and where Q is a random, sparse, quasi-cyclic matrix of dimension $p \cdot n_0 \times p \cdot n_0$. Here p is a moderately large prime and n_0 is a small constant. The intermediate matrix $L = [L_0|...|L_{n_0-1}] = H \cdot Q$ is formed by matrix multiplication. The public key M is then constructed from L by multiplying each of the L_i by $L_{n_0-1}^{-1}$. Given this key pair, information can be encoded into codeword vectors, then perturbed by random error-vectors of a low Hamming weight.[1] Security essentially states that it should be difficult to recover the originally-encoded information from the perturbed codeword unless a party possesses the factorization of the public key as H and Q.

The LEDAcrypt submission package in the 2nd Round of NIST's PQC process provides a careful description of the algorithm's history and specific design, a variety of concrete parameters sets tailored to NIST's security levels (claiming approximately 128-bit, 192-bit, and 256-bit security, under either IND-CPA or IND-CCA attacks), and a reference implementation in-code.

1.1 Our Results

In this work, we provide a novel, concrete cryptanalysis of LEDAcrypt. We begin by identifying a moderately-sized, very weak class of keys, which can be recovered with substantially less computational effort than expected. This is a major, practical break of the LEDAcrypt cryptosystem, which is encapsulated in the following theorem:

Theorem 1.1. *(Section 3) There is an algorithm that costs the same as* $2^{49.22}$ *AES-256 operations and recovers* 1 *in* $2^{47.79}$ *of LEDAcrypt's Category 5 (i.e. claimed 256-bit-secure) ephemeral / IND-CPA keys.*

Similarly, there is an algorithm that costs the same as $2^{57.49}$ *AES-256 operations and recovers* 1 *in* $2^{51.59}$ *of LEDAcrypt's Category 5 (i.e. claimed 256-bit-secure) long-term / IND-CCA keys.*

While most key-recovery algorithms can exchange computational time spent vs. fraction of the key space recovered, this trade-off will generally be 1-to-1 against a secure cryptosystem. However, we note in the above that both $49.22 + 47.79 = 97.01 \ll 256$ and $57.49 + 51.59 = 109.08 \ll 256$, making this attack quite significant. Additionally, we note that this class of very weak keys is present in every parameter set of LEDAcrypt.

While the existence of this class of imperfect keys is a serious concern, one might ask: Is it possible to identify such keys during KeyGen, reject them, and thereby save the scheme's design? We are able to answer this in the negative.

[1]We refer the reader to Section A.1 for further technical details of the construction.

Theorem 1.2 (Informal, Section 4). *There exists a continuum of sub-classes of progressively less weak keys, stretching from the weakest keys to all keys. Each sub-class of keys in this continuum is progressively larger and can be recovered with progressively more work per key (though, still, at a substantially lower computational cost than expected).*

Said another way, the existence of weaker-than-expected keys in LEDAcrypt is *fundamental* in the system's formulation and cannot be avoided without a major re-design of the cryptosystem. For example, one could try to instantiate the McEliece paradigm with a completely different flavor of code — but smaller changes in design will be insufficient.

Finally, we apply our new attack ideas to attempting key recovery in the average-case. Here we give only a partial, heuristic analysis, but which nonetheless demonstrates a mild improvement for the attacker.

Theorem 1.3 (Informal, Section 5). *LEDAcrypt's Category 5 ephemeral / IND-CPA keys can be recovered, on average, in no more than 2^{269} bit operations.*

We compare this to the computational cost of 2^{277} bit operations asserted in LEDAcrypt's specification sent to NIST. We also note that we consider just a single case of the average-case attack in our analysis, but a very large number of other, similar cases must exist as well. A more complete analysis of our average-case attack algorithm would necessarily further reduce the expected concrete security of LEDAcrypt in the average case. Regardless, this final analysis demonstrates that for real parameters of interest, our attack indeed impacts average-case security rather than just some class of weak keys that might be removed by more aggressive rejection sampling.

1.2 Technical Overview of Our New Attacks

Basic Approach: Exploiting the Product Structure. The typical approach to recovering keys for LEDAcrypt-like schemes is to use ordinary Information Set Decoding (ISD) algorithms, a class of techniques which can be used to search for low weight codewords in an arbitrary code. Generally speaking, these algorithms symbolically consider a row of an unknown binary matrix corresponding to the secret key of the scheme. From this row, they randomly choose a set of bit positions uniformly at random in the hope that these bits will (mostly) be zero. If the guess is correct and, additionally, the chosen set is an *information set* (i.e., a set in which all codewords differ at least in one position), then the key will be recovered with linear algebra computation. If (at least) one of the two requirements on the set is not met, then the procedure resets and guesses again.

For our attacks, intuitively, we will choose the information set in a non-uniform manner in order to increase the probability that the support of HQ, i.e. the non-zero coefficients of HQ, is (mostly) contained in the complement of the information set. At a high level, we will guess two sets of polynomials $H'_0, ..., H'_{n_0-1}$ and $Q'_{0,0}, ..., Q'_{n_0-1,n_0-1}$, then (interpreting the polynomials as $p \times$

p circulant matrices) group them into quasi-cyclic matrices H' and Q'. These matrices will be structured analogously to H and Q, but with non-negative coefficients defined over $\mathbb{Z}[x]/\langle x^p - 1\rangle$ rather than $\mathbb{F}_2[x]/\langle x^p + 1\rangle$. The hope is that the support of $H'Q'$ will (mostly) contain the support of HQ. It should be noted that a sufficient condition for this to be the case is that the support of H' contains the support of H and the support of Q' contains the support of Q. Assuming the Hamming weight of $H'Q'$ (interpreted as a coefficient vector) is chosen to be approximately W, then the information set can be chosen as the complement of the support of $H'Q'$ and properly passed to an ISD subroutine in place of a uniform guess.

Observe that the probability that the supports of H' and Q' contain the supports of H and Q, respectively, is maximized by making the Hamming weight of H' and Q' as large as possible while still limiting the Hamming weight of $H'Q'$ to W. An initial intuition is that this can be done by choosing the 1-coefficients of the polynomials $H'_0, ..., H'_{n_0-1}$ and $Q'_{0,0}, ..., Q'_{n_0-1,n_0-1}$ to be in a single, consecutive chunk. For example, by choosing the Hamming weight of the polynomials (before multiplication) as some value $B \ll W$, we can take $H'_0 = x^a + x^{a+1} + ... + x^{a+B-1}$ and $Q'_{0,0} = x^c + x^{c+1} + ... + x^{c+B-1}$.

Note that the polynomials H'_0 and $Q'_{0,0}$ (chosen with consecutive 1-coefficients as above) have Hamming weight B, while their product only has Hamming weight $2B - 1$. In the most general case, uniformly chosen polynomials with Hamming weight B would be expected to have a product with Hamming weight much closer to $\min(B^2, p)$. That is, for a fixed weight W required of $H'Q'$ by the ISD subroutine, we can guess around $W/2$ positions at once in H' and Q' respectively instead of something closer to $\sqrt{W}$ as would be given by a truly uniform choice of information set. As a result, each individual guess of H' and Q' that's "close" to this outline of our intuition will be more rewarding for searching the keyspace than the "typical" case of uniformly guessing information sets.

This constitutes the core intuition for our attacks against LEDAcrypt, but additional considerations are required in order to make the attacks practically effective (particularly when concrete parameters are considered). We enumerate a few of these observations next.

Different ring representations The idea of choosing the polynomials within H' and Q' with consecutive nonzero coefficients makes each iteration of an information set decoding algorithm using such an H' and Q' much more effective than an iteration with a random information set. However there is only a limited number of successful information sets with this form. We can vastly increase our range of options by observing that the ring $\mathbb{F}_2[x]/\langle x^p + 1\rangle$ has $p - 1$ isomorphic representations which can be mapped to one another by the isomoprhism $f(x) \to f(x^\alpha)$. This allows us many more equally efficient choices of the information set, since rather than restricting our choices to have polynomials H'_0 and $Q'_{0,0}$ with consecutive ones in the standard ring representation, we have the freedom to choose them with consecutive ones in any ring representation (provided the same representation is used for H'_0 and $Q'_{0,0}$.)

Equivalent keys. For each public key of LEDAcrypt, there exist many choices of private keys that produce the same public key. In particular, the same public key $M = (L_{n_0-1})^{-1}L$ produced by the private key

$$H = [H_0, H_1, \cdots, H_{n_0-1}],$$

$$Q = \begin{bmatrix} Q_{0,0} & Q_{0,1} & \cdots & Q_{0,n_0-1} \\ Q_{1,0} & Q_{1,1} & \cdots & Q_{1,n_0-1} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{n_0-1,0} & Q_{n_0-1,1} & \cdots & Q_{n_0-1,n_0-1} \end{bmatrix};$$

would also be produced by any private key of the form

$$H' = [x^{a_0}H_0, x^{a_1}H_1, \cdots, x^{a_{n_0-1}}H_{n_0-1}],$$

$$Q' = \begin{bmatrix} x^{b-a_0}Q_{0,0} & x^{b-a_0}Q_{0,1} & \cdots & x^{b-a_0}Q_{0,n_0-1} \\ x^{b-a_1}Q_{1,0} & x^{b-a_1}Q_{1,1} & \cdots & x^{b-a_1}Q_{1,n_0-1} \\ \vdots & \vdots & \ddots & \vdots \\ x^{b-a_{n_0}}Q_{n_0-1,0} & x^{b-a_{n_0}}Q_{n_0-1,1} & \cdots & x^{b-a_{n_0}}Q_{n_0-1,n_0-1} \end{bmatrix};$$

for any integers $0 < a_i, b < p, i \in \{0, \ldots, n_0-1\}$. These p^{n_0+1} equivalent keys improve the success probability of key recovery attacks as detailed in the following sections.

Different degree constraints for H' and Q'. While we have so far described H' and Q' as having the same Hamming weight B, this does not necessarily need to be the case. In fact, there are many, equivalent choices of H' and Q' which produce the same product $H'Q'$ based on this observation. For example, the product of

$$H'_0 = x^a + x^{a+1} + ... + x^{a+B-1}$$
$$Q'_{0,0} = x^c + x^{c+1} + ... + x^{c+B-1}$$

is identical to the product of

$$H'_0 = x^a + x^{a+1} + ... + x^{a+B-1-\delta}$$
$$Q'_{0,0} = x^c + x^{c+1} + ... + x^{c+B-1+\delta}$$

for any integer $-B < \delta < B$. More generally, this relationship (that if H' shrinks and Q' proportionally grows, or vice versa, then the product $H'Q'$ is the same) is independently true for any set of $\{H'_i, Q'_{i,0}, ..., Q'_{i,n_0-1}\}$ for $i \in \{0, ..., n_0 - 1\}$.

Attacks for $n_0 = 2$ imply similar-cost attacks for $n_0 > 2$. Our attacks are more easily described (and more effective) in the case $n_0 = 2$. In this case, we apply ISD to find low-weight codewords in the row space of the public key $[M_0 \mid M_1]$ to recover a viable secret key for the system. Naively extending this approach for the case $n_0 > 2$ to the entire public key $[M_0 \mid ... \mid M_{n_0}]$ requires constraints on the support of $n_0 + n_0^2$ polynomials (n_0 polynomials corresponding to H' and n_0^2 polynomials corresponding to Q'), so the overall work in the attack would increase quadratically as n_0 grows. However, even in the case that $n_0 > 2$, we observe that it is sufficient to find low weight codewords in the row space of only $[M_0 \mid M_1]$ in order to recover a working key, implying that the attack only needs to consider $3n_0$ polynomials $H_i, Q_{j,0}, Q_{k,1}$. So, increasing n_0 will make all of our attacks less effective, but not substantially so. More importantly, any attack against $n_0 = 2$ parameters immediately implies a similar-cost attack against parameters with $n_0 > 2$. Therefore, we focus on the case of $n_0 = 2$ in the remainder of this work.

A Continuum of Progressively Less Weak Keys. The attacker can recover keys with the highest probability per iteration of ISD by using a very structured pattern for L'. As we will see in section 3, in this pattern both L'_0 and L'_1 will have a single contiguous stretch of nonzero coefficients in some ring representation. The result is a practical attack, but one which is only capable of recovering weak keys representing something like 1 in 2^{40} or 1 in 2^{50} private keys.

However, if the attacker is willing to use a more complicated pattern for the information set, using different ring representations for different blocks of H' and Q', and possibly having multiple separate stretches of consecutive nonzero coefficients in each block, then the attacker will not recover keys with as high a probability per iteration, but the attack will extend to a broader class of slightly less weak keys. This may for example lead to a somewhat less practical attack that recovers 1 in 2^{30} keys, but still much faster than would be expected given the claimed security strength of the parameter set in question.

Improvements to Average-case Key Recovery. In section 5 we will take the continuum of progressively weaker keys to its logical extreme. We show that the attacks in this paper are asymptotically stronger than the standard attacks not just for weak keys, but for all keys. Moreover, we give an argument that the critical point at which the attacks in this paper overtake the standard attacks is within the range of parameters of cryptographic interest, in particular including some of the parameter sets proposed by the second round LEDAcrypt submission targeting higher security levels.

As we move away from the simpler information set patterns used on the weakest keys, the analysis becomes more difficult. To fully quantify the impact of our attack on average keys would require extensive case analysis of all scenarios that might lead to a successful key recovery given a particular distribution of information sets used by the attacker. We do not attempt such an exhaustive analysis, but rather analyze a single scenario for successful key recovery against a single parameter set, finding even this particular scenario is enough to recover typical keys faster than would be possible with the standard information set decoding algorithm used to analyze the security of the parameters in the submitted specification. The true complexity of the attack is likely lower than our estimate.

1.3 Related Work

The main attack strategies against cryptosystems based on QC-LDPC codes are known as information set decoding (ISD) algorithms. These algorithms are also applicable to a variety of other code-based cryptosystems including the NIST 2nd round candidates BIKE [23], HQC [8], Classic McEliece [9], and NTS-KEM [18]. Initiated by Prange [26] in 1962, these algorithms have since experienced substantial improvements during the years [2,7,14,15,19,20,29]. ISD algorithms can also be used to find low-weight codewords in a given, arbitrary code. ISD main approach is that of guessing a set of positions where such codewords contain a very low number of set symbols; when this set is actually an information set, then linear algebra computations yield the searched codeword (see A.3). ISD time complexity is estimated as the product between the expected number of required information set guesses and the cost of testing each set. Advanced ISD algorithms improve Prange's basic idea by reducing the average number of required guesses, at the cost of increasing the time complexity of the testing phase. Quantum ISD algorithms take into account Grover's algorithm [10] to quadratically accelerate the guessing phase. A quantum version of Prange's algorithm [6] was presented in 2010, while quantum versions of more advanced ISD algorithms were presented in 2017 [12].

In the case of QC-MDPC and QC-LDPC codes, ISD key recovery attacks can get a speed-up which is polynomial in the size of the circulant blocks [27]. This gain is due to the fact that there are more than one sparse vectors in the row space of the parity check matrix, and no modification to the standard ISD algorithms is required to obtain this speed-up. Another example of gains due to the QC structure is that of [16] which, however, works only in the case of the circulant size having a power of 2 among its factors (which is not the case we consider here).

ISD can be generally be described as a technique for finding low Hamming-weight codewords in a linear code. Most ISD algorithms are designed to assume that the low-weight codewords are random aside from their sparsity. However, in some cryptosystems that can be cryptanalyzed using ISD, these short codewords are not random in this respect, and modified versions of ISD have been used to break these schemes [22,25]. Our paper can be seen as a continuation of this line of work, since unlike the other second round NIST candidates where ISD is cryptanalytically relevant, the sparse codewords which lead to a key recovery of LEDAcrypt are not simply random sparse vectors, but have additional structure due to the product structure of LEDAcrypt's private key.

2 Notation

Throughout this work, we denote the finite field with 2 elements by $\mathbb{F}_2$. We denote the Hamming weight of a vector a (or a polynomial a, viewed in terms of its coefficient vector) as ω_a. For a polynomial a we use the representation $a = \sum_{i=0}^{p-1} a_i x^i$, and call a_i represents its i-th coefficients. We denote the support

– i.e. the non-zero coordinates – of a vector (or polynomial) a by $\mathrm{S}(a)$. In similar way, we define the *antisupport* of a, and denote it as $\bar{\mathrm{S}}(a)$, as the set of positions i such that $a_i = 0$.

We denote a polynomial a with support equal to a set of indices J by $a^{(J)}$. We denote the set of coefficients of a polynomial a that are indexed by a set of indices J by $a|_J$.

3 Attack on the Weakest Class of Keys

In this section, we present an attack against a class of weak keys in LEDAcrypt's design. We begin by identifying what appear to be the weakest class of keys (though large enough in number to constitute a serious, practical problem for LEDAcrypt). We proceed to provide a simple, single-iteration ISD algorithm to recover these keys, then analyze the fraction of all of LEDAcrypt's keys that would be recovered by this attack. Afterward, we show how to extend the ISD algorithm to more than one iteration, so as to enlarge the set of keys recovered by a similar enough of effort per key. We conclude by considering the effect of advanced ISD algorithms on the attack as well as the relationship between the rejection sampling step in LEDAcrypt's KeyGen and our restriction to attacking a subspace of the total key space.

Extending this attack from the class of weakest keys to a continuum of progressively less weak keys is considered in the subsequent section.

3.1 Attacking an example class of ultra-weak keys

The simplest and, where it works, most powerful version of the attack dramatically speeds up ISD for a class of ultra-weak keys chosen under parameter sets where $n_0 = 2$. One example class of ultra-weak keys are those keys where the polynomials L_0 and L_1 are of degree at most $\frac{p}{2}$. Such keys can be found by a single iteration of a very simple ISD algorithm:

The attacker chooses the information set to consist of the last $\frac{p-1}{2}$ columns of each block of M. If the key being attacked is one of these weak keys, the attacker can correctly guess the top row of L as being identically zero within the information set and linearly solve for the nonzero linear combination of the rows of M meeting this condition. The cost of the attack is one iteration of an ISD algorithm.

A sufficient condition for this class of weak key to occur is for the polynomials H_0, H_1, $Q_{0,0}$, $Q_{0,1}$, $Q_{1,0}$, and $Q_{1,1}$ to have degree no more than $\frac{p}{4}$. Since each of the $2m_0+2m_1+2d_v$ nonzero coefficients of these polynomials has a $\frac{1}{4}$ probability of being chosen with degree less than $\frac{p}{4}$, these weak keys represent at least 1 part in $4^{2m_0+2m_1+2d_v}$ of the key space.

3.2 Enumerating ultra-weak keys for a single information set

In fact, there are significantly more weak keys than this that can be recovered by the basic, one-iteration ISD algorithm using the information set described above. Intuitively, this is for two reasons:

1. **Equivalent keys:** There are p^2 private keys, not of this same, basic form, which nonetheless produce the same public key.
2. **Different degree contraints:**The support of the top row of L will also fall entirely outside the information set if the degree of H_0 is less than $\frac{p}{4} - \delta$ and the degrees of $Q_{0,0}$ and $Q_{0,1}$ are both less than $\frac{p}{4} + \delta$ for any $\delta \in \mathbb{Z}$ such that $-\frac{p}{4} < \delta < \frac{p}{4}$. Likewise for H_1 and $Q_{1,0}$ and $Q_{1,1}$, for a total of p keys.

Concretely, we derive the number of distinct private keys that are recovered by the one-iteration ISD algorithm in the following theorem.

Remark 1. There are p columns of each block of M. Instead of referring to pairs of $\frac{p-1}{2}$ and $\frac{p+1}{2}$ columns, we instead use $\frac{p}{2}$ for both cases.

Theorem 3.1. *The number of distinct private keys that can be found in a single iteration of the decoding algorithm described above (where the information set is chosen to consist of the last $\frac{p-1}{2}$ columns of each block of M) is*

$$\begin{aligned} & p^3 \cdot \sum_{A_0=d_v-1}^{\frac{p}{2}} \sum_{A_1=d_v-1}^{\frac{p}{2}} \left(\binom{A_0-1}{d_v-2} \binom{A_1-1}{d_v-2} \right. \\ & \cdot \left(\binom{\frac{p}{2}-A_0-2}{m_0-1} \binom{\frac{p}{2}-A_0-1}{m_1} \binom{\frac{p}{2}-A_1-1}{m_1} \binom{\frac{p}{2}-A_1-1}{m_0} \right. \\ & + \binom{\frac{p}{2}-A_0-1}{m_0} \binom{\frac{p}{2}-A_0-2}{m_1-1} \binom{\frac{p}{2}-A_1-1}{m_1} \binom{\frac{p}{2}-A_1-1}{m_0} \\ & + \binom{\frac{p}{2}-A_0-1}{m_0} \binom{\frac{p}{2}-A_0-1}{m_1} \binom{\frac{p}{2}-A_1-2}{m_1-1} \binom{\frac{p}{2}-A_1-1}{m_0} \\ & \left. \left. + \binom{\frac{p}{2}-A_0-1}{m_0} \binom{\frac{p}{2}-A_0-1}{m_1} \binom{\frac{p}{2}-A_1-1}{m_1} \binom{\frac{p}{2}-A_1-2}{m_0-1} \right) \right) \\ & \cdot \left(1 - O\left(\frac{m_0}{p} \right) \right) \end{aligned} \tag{1}$$

Proof. We are using the same information set for both $H_0Q_{0,0} + H_1Q_{1,0}$ and $H_0Q_{0,1} + H_1Q_{1,1}$. Therefore we count according the first nonzero bit of the sum $H_0Q_{0,0} + H_1Q_{1,0} + H_0Q_{0,1} + H_1Q_{1,1}$. Let l be the location of the first nonzero bit of M, let j_0, j_1 be the locations of the first nonzero bit of H_0, H_1, respectively. Suppose that the nonzero bits of H_0, H_1 are located within a block of length A_0, A_1, respectively.

We begin by considering H, Q such that HQ does not have full weight. Once j_0 is fixed, there are four blocks of Q which may influence the location l. We compute the probability that only one block of Q may influence l at a time.

If l is influenced by $Q_{0,0}$, there are $\binom{\frac{p}{2}-A_0-2}{m_0-1}$ ways the remaining bits of $Q_{0,0}$ can fall, $\binom{\frac{p}{2}-A_0-1}{m_1}$ arrangements of the bits of $Q_{0,1}$, $\binom{\frac{p}{2}-A_1-1}{m_1}$ arrangements of the bits of $Q_{1,0}$, and $\binom{\frac{p}{2}-A_1-1}{m_0}$ arrangements of the bits of $Q_{1,1}$. If l is influenced by $Q_{0,1}, Q_{1,0}$, or $Q_{1,1}$, similar estimates hold. We sum over the l locations considering each of the blocks of Q and their respective weights.

Failure to impose full weight requirements on HQ introduces double-counting, though the probability of which will not exceed $O(\frac{m_0}{p})$. □

We can now estimate the percentage of these recovered, ultra-weak keys out of all possible keys.

Theorem 3.2. *Let $m = m_0 + m_1, x = \frac{A_0}{p}, y = \frac{A_1}{p}$. Out of $\binom{p}{d_v}^2\binom{p}{m_0}^2\binom{p}{m_1}^2$ possible keys, we estimate the percentage of ultra-weak keys found in a single iteration of the decoding algorithm above as*

$$d_v^{\,2}(d_v-1)^2 m \int_{x=0}^{\frac{1}{2}}\int_{y=0}^{\frac{1}{2}} (xy)^{d_v-2}\left(\left(\frac{1}{2}-x\right)\left(\frac{1}{2}-y\right)\right)^m \left(\frac{1}{\frac{1}{2}-x}+\frac{1}{\frac{1}{2}-y}\right)\mathrm{d}x\mathrm{d}y.$$

Proof. Note that the lines $2-5$ of (1) are approximately

$$\binom{\frac{p}{2}-A_0}{m_0}\binom{\frac{p}{2}-A_0}{m_1}\binom{\frac{p}{2}-A_1}{m_1}\binom{\frac{p}{2}-A_1}{m_0}\left(\frac{m_0+m_1}{\frac{p}{2}-A_1}+\frac{m_0+m_1}{\frac{p}{2}-A_0}\right).$$

For $b \in \{0,1\}$, $\binom{\frac{p}{2}-A_b}{m_b} \approx \binom{p}{m_b}\left(\frac{1}{2}-\frac{A_b}{p}\right)^m$ and $\binom{A_b-1}{d_v-2} \approx \binom{p}{d_v-2}\left(\frac{A_b}{p}\right)^{d_v-2}$, since p is much larger than m_0, m_1, d_v. Then we rewrite (1) as

$$p^2\binom{p}{d_v-2}^2\binom{p}{m_0}^2\binom{p}{m_1}^2 m \sum_{A_0=d_v-1}^{\frac{p}{2}}\sum_{A_1=d_v-1}^{\frac{p}{2}}\left(\frac{A_0}{p}\frac{A_1}{p}\right)^{d_v-2}\left(\frac{1}{2}-\frac{A_0}{p}\right)^m$$
$$\left(\frac{1}{2}-\frac{A_1}{p}\right)^m\left(\frac{1}{\frac{1}{2}-\frac{A_0}{p}}+\frac{1}{\frac{1}{2}-\frac{A_1}{p}}\right)$$
$$\approx p^2\binom{p}{d_v}^2\binom{p}{m_0}^2\binom{p}{m_1}^2 m\frac{d_v^{\,2}(d_v-1)^2}{(p-d_v+2)^2(p-d_v+1)^2}\cdot$$
$$p^2\int_{x=0}^{\frac{1}{2}}\int_{y=0}^{\frac{1}{2}}(xy)^{d_v-2}\left(\frac{1}{2}-x\right)^m\left(\frac{1}{2}-y\right)^m\left(\frac{1}{\frac{1}{2}-x}+\frac{1}{\frac{1}{2}-y}\right)\mathrm{d}x\mathrm{d}y.$$

Dividing by $\binom{p}{d_v}^2\binom{p}{m_0}^2\binom{p}{m_1}^2$, the result follows. □

Evaluating this percentage with the claimed-256-bit ephemeral (CPA-secure) key parameters of LEDAcrypt — $d_v = 11, m = 13$ — we determine that 1 in $2^{72.8}$ ephemeral keys are broken by one iteration of ISD. Similarly for the long-term (CCA-secure) key setting, we evaluate with the claimed 256-bit parameters — $d_v = 13, m = 13$ — and conclude the number of long-term keys broken is 1 in $2^{80.6}$.

This result merely determines the number of keys that can be recovered given that the information set of both blocks of M is chosen to be the last $\frac{p-1}{2}$ columns.[2] In the following, we turn to demonstrating a class of additional information sets that are as effective as this one.

3.3 Enumerating ultra-weak keys for all information sets

Now we will demonstrate a multi-iteration ISD attack that is effective against the class of all ultra-weak keys. To set up the discussion, we begin by highlighting two, further "degrees of freedom," which will allow us to find additional, relevant information sets to guess:

1. **Changing the ring representation:** Contiguity of indices depends on the choice of ring representation. The large family of ring isomorphisms on $\mathbb{Z}[x]/\langle x^p - 1\rangle$ given by $f(x) \to f(x^t)$ for $t \in [0, p]$ preserves Hamming weight. For example, we can use the family of polynomials

   $$H'_i = Q'_{i,j} = 1 + x^t + x^{2t} + ... + x^{\frac{p-1}{4}t}$$

   in this attack, since there exists one t such that H'_i has consecutive nonzero coefficients. Choices of $t \in \{1, \ldots, \frac{p-1}{2}\}$ yield independent information sets (noting that choices of t and $-t \mod p$ yield equivalent information sets).

2. **Changing the relative offset of the two consecutive blocks:** We can also change the beginning index of the consecutive blocks produced within L'_0 or L'_1 (by modifying the beginning indices of H'_i and $Q'_{i,j}$ to suit). Note that shifting both L'_0 and L'_1 by the same offset will recover equivalent keys. However, if we fix the beginning index of L'_0 and allow the beginning index of L'_1 to vary, we can find more, mostly independent information sets in order to recover more, distinct keys. The exact calculation of how far one should shift L'_1's indices for a practically effective attack is somewhat complex; we perform this analysis below in the remainder of this subsection.

Recall that in the prior 1-iteration attack, we considered *one* example class of ultra-weak keys – namely, those keys where the polynomials L_0 and L_1 are of degree at most $\frac{p}{2}$. Here, we will now take a broader view on the weakest-possible keys. We define the **class of ultra-weak keys** to be those where, in some ring representation, both $H_0Q_{0,0} + H_1Q_{1,0}$ and $H_0Q_{0,1} + H_1Q_{1,1}$ have nonzero coefficients that lie within a block of $\frac{p-1}{2}$-many consecutive (modulo p) degrees.

Our goal will be now to find a multi-iteration ISD algorithm — by estimating how far to shift the offset of L'_1 per iteration — that recovers as much of the class of ultra-weak keys as possible without "overly wasting" the attacker's computational budget. Toward this end, recall that we have a good estimate

[2]For the reader, we point out that if, hypothetically, we had a sufficiently large number of *totally independent* information sets that were equally "rewarding" in recovering keys, this would straightforwardly imply $\approx 2^{72.8}$-time and $\approx 2^{80.6}$-time "full" attacks against LEDAcrypt's claimed-256-bit parameters rather than weak-key attacks.

in Theorem 3.2 of the fraction of keys ($2^{-72.8}$, resp. $2^{-80.6}$) recovered by the best-case, single iteration of our ISD algorithm. In what follows, we will first calculate the fraction of ultra-weak keys as a part of the total key space.

Let 2^{-X} be the fraction of all keys recovered by the best-case, single iteration of our previous ISD algorithm. Let 2^{-Y} be the fraction of ultra-weak keys among all keys. On the assumption that every ring representation leads to independent information sets (chosen uniformly for each invocation of ISD) and on the assumption that independence of ISD key-recovery is maximized by shifting "as far as possible," we will compute an estimate of the number of index-shifts that should be performed by the optimal ultra-weak-key attacker as $2^Z = 2^{X-Y}$. Beyond 2^Z shifts per guess (but not until), the attacker should begin to experience diminishing returns in how many keys are recovered per shifted guess.

Therefore, given an index beginning at 1 out of p positions, the attacker will shift by $\frac{p(\frac{p-1}{2})}{2^Z}$ indices at each invocation (where the factor $\frac{p-1}{2}$ accounts for the effect of the different possible ring representations). By assumption, each such guess will be sufficiently independent to recover as many keys in expectation as the initial, best-guess case described by the 1-iteration algorithm. We note that additional, ultra-weak keys will certainly be obtained by performing more work — specifically by shifting less than $\frac{p(\frac{p-1}{2})}{2^Z}$ per guess — but necessarily at a reduced rate of reward per guess.

Toward this end, we now calculate the number of ultra-weak keys then the fraction of ultra-weak keys among all keys following the format of the previous calculation.

Theorem 3.3. *The total number of ultra-weak keys are*

$$\frac{p-1}{2}p^4 \sum_{A_0=d_v-1}^{\frac{p}{2}} \sum_{A_1=d_v-2}^{\frac{p}{2}} \binom{A_0-1}{d_v-2}\binom{A_1-1}{d_v-2} \tag{2}$$

$$\cdot \sum_{l_0}^{p-1} \left(\binom{\frac{p}{2}-A_0-1}{m_0-1}\binom{\frac{p}{2}-A_1-1}{m_1} + \binom{\frac{p}{2}-A_0-1}{m_0}\binom{\frac{p}{2}-A_1-1}{m_1-1} \right) \tag{3}$$

$$\cdot \sum_{l_1=0}^{p-1} \left(\binom{\frac{p}{2}-A_1-1}{m_0-1}\binom{\frac{p}{2}-A_1-1}{m_1} + \binom{\frac{p}{2}-A_0-1}{m_0-1}\binom{\frac{p}{p}-A_1-1}{m_0} \right). \tag{4}$$

Proof. The proof technique follows as in Theorem 3.1. Details are found in the appendix, B.1. □

Theorem 3.4. *Let $m = m_0 + m_1, x = \frac{A_0}{p}, y = \frac{A_1}{p}$. The fraction of ultra-weak keys out of all possible keys is*

$$\frac{p-1}{2} {d_v}^2 (d_v-1)^2 \int_{x=0}^{\frac{1}{2}} \int_{y=0}^{\frac{1}{2}} x^{d_v-2} y^{d_v-2} \left(\frac{1}{2}-x\right)^m \left(\frac{1}{2}-y\right)^m$$
$$\left(\frac{{m_0}^2+{m_1}^2}{(\frac{1}{2}-x)(\frac{1}{2}-y)} + \frac{m_0 m_1}{(\frac{1}{2}-x)^2} + \frac{m_0 m_1}{(\frac{1}{2}-y)^2} \right) \mathrm{d}x\mathrm{d}y.$$

Proof. Similar techniques apply. See appendix B.2 for details. $\square$

We evaluate the fraction of weak keys using the claimed CPA-secure parameters $p = 36877, m = 13, d_v = 11$ and determine that 1 in $2^{54.1}$ ephemeral keys are broken. Evaluating with one of the CCA-secure parameter sets $p = 152,267, m = 13, d_v = 13$, approximately 1 in $2^{59.7}$ long-term keys are broken.

Given the above, we can make an estimate as to the optimal shift-distance per ISD invocation as $\frac{36{,}877(\frac{36{,}876}{2})}{2^{72.8-54.1}} \approx 1597 \approx 2^{10.6}$ for the ephemeral key parameters and $\frac{152{,}267(\frac{152{,}266}{2})}{2^{80.6-59.7}} \approx 5925 \approx 2^{12.5}$ for the long-term key parameters.

The multi-iteration ISD algorithm against the class of ultra-weak keys, then, makes its first guess (except, one in each ring representation) as in the case of the 1-iteration ISD algorithm. It then shifts the relative offset of the two consecutive blocks by the values calculated above and repeats (again, in each ring representation).

This will not recover all ultra-weak keys, but it will recover a significant fraction of them. In particular, if the support of each block of L, rather than fitting in $\frac{p}{2}$ consecutive bits fits in blocks that are smaller by at least $\frac{1}{4}$ of the shift distance. We can therefore lower bound the fraction of recovered keys by replacing factors of $\frac{1}{2}$ with factors of $\frac{p}{2}$ minus half or a quarter of the offset, all divided by p, to find the sizes of sets of private keys of which we are guaranteed to recover all, or at least half of respectively.

The multi-iteration ISD algorithm attacking the ephemeral key parameters will make $2^{72.8-54.1} \approx 2^{18.7}$ independent guesses and recover at least 1 in $2^{56.0}$ of the total keys. The multi-iteration ISD algorithm attacking the long-term key parameters will make $2^{80.6-59.7} \approx 2^{20.9}$ independent guesses and recover at least 1 in $2^{61.6}$ of the total keys.

3.4 Estimating the effect of more advanced information-set decoding

Our attempts to enumerate all weak keys were based on the assumption that the adversary was using an ISD variant that required a row of L to be uniformly 0 on all columns of the information set. The state of the art in information set decoding still allows the adversary to decode provided that a row of L has weight no more than about 6 on the information set. For example, Stern's algorithm [29] with parameter 3 would attempt to find a low weight row of L as follows:

The information set is divided into two disjoint sets of $\frac{p}{2}$ columns. The first row of L to be recovered should have weight at most 3 within each of the two sets. Further, the same row of L should have have $\Omega(\log(p))$ many consecutive 0's in column-indices that are disjoint from those of the information set. If both of these conditions occur, then a matrix inversion is performed (even though 6 non-zero bits were contained in the information set).

Note that for reasonably large p, nearly a third of the sparse vectors having weight 6 in the information set will meet both conditions. The most expensive steps in the Stern's algorithm iteration are a matrix inversion of size p and a claw finding on functions with logarithmic cost in p and domain sizes of $\binom{\frac{p}{2}}{3}$. The claw finding step is similar in cost to the matrix inversion, both having computational cost $\approx p^3$. The matrix inversion step is present in all ISD algorithms. Therefore with Stern's algorithm we can recover in a single iteration with similar cost to a single iteration of a simpler ISD algorithm, $O(1)$ of the private keys where a row of L has weight no more than 6 on the information set columns.

Recall that we choose the information set to be of size $\approx \frac{p}{2}$ in L'. The distribution of the non-zero coordinates within a successful guess of information set will be more heavily weighted toward the middle of the set and approximately triangular shaped (since these coordinates are produced by convolutions of polynomials). In particular, we will *heuristically* model both of the tails of the distribution as small triangles containing 3 bits on the left side and three bits on the right that are missed by the choice of information set.

Let $W = 2d_v(m_0 + m_1)$ denote the number of non-zero bits in L'. Then the actual fraction ϵ that the information set (in the context of advanced information set decoding) should target within L, rather than $1/2$, can be estimated by geometric area as

$$\epsilon \cdot \left(1 - \sqrt{\frac{3}{W/2}}\right) = \frac{1}{2}$$

or, re-writing:

$$\epsilon = \frac{1}{2\left(1 - \sqrt{\frac{3}{W/2}}\right)}.$$

For the claimed-256-bit ephemeral key parameters, we have $t_{\mathsf{CPA}} = 286$. For the claimed-256-bit long-term key parameters, we have $t_{\mathsf{CCA}} = 338$. Therefore,

$$\epsilon_{\mathsf{CPA}} = \frac{1}{2\left(1 - \sqrt{\frac{3}{286/2}}\right)} \approx 0.585.$$

$$\epsilon_{\mathsf{CCA}} = \frac{1}{2\left(1 - \sqrt{\frac{3}{338/2}}\right)} \approx 0.577.$$

So – heuristically – we can model the effect of using advanced information set decoding algorithms by replacing the $\frac{1}{2}$'s in the calculations of the theorems earlier in this section by ϵ_{CPA} or ϵ_{CCA} respectively.

3.5 Rejection sampling considerations

We recall that LEDACrypt's KeyGen algorithm explicitly requires that the parity check matrix L be *full weight*. Intuitively full weight means that no cancellations occur in the additions or the multiplications that are used to generate L from H and Q. Formally, the full weight condition on L can be stated as:

$$\forall i \in \{0, \dots, n_0 - 1\}, \ \ \mathsf{weight}(L_i) = d_v \sum_{j=0}^{n_0} m_j.$$

When a weak key notion causes rejections to occur significantly more often for weak keys than non-weak keys, we will effectively reduce the probability of weak key generation compared to our previous analysis. As an extreme example, if, for a given weak key notion, rejection sampling rejects all weak keys, then no weak keys will ever be sampled. We therefore seek to measure the probability of key rejection for both weak keys and keys in general in order to determine whether the effectiveness of this attack is reduced via rejection sampling.

Let $\mathcal{K}$, $\mathcal{W} \subset \mathcal{K}$, and KeyGen be the public key space, the weak key space, and the key generation algorithm of LEDACrypt, respectively. Let $\mathcal{K}'$, $\mathcal{W}' \subset \mathcal{K}'$, and KeyGen' be the associated objects if rejection sampling were omitted from LEDACrypt. We observe that since KeyGen samples uniformly from $\mathcal{K}$,

$$\Pr\left[pk \in \mathcal{W} | (pk, sk) \leftarrow \mathsf{KeyGen}()\right] = \frac{|\mathcal{W}|}{|\mathcal{K}|}.$$

This equality additionally holds when rejection sampling does not occur. Since, until now, all of our analysis has ignored rejection sampling we have effectively been measuring $|\mathcal{W}'|/|\mathcal{K}'|$. We therefore seek to find a relation that allows us determine $|\mathcal{W}|/|\mathcal{K}|$ from $|\mathcal{K}'|$ and $\mathcal{W}'|$. We observe that

$$\frac{|\mathcal{W}|}{|\mathcal{K}|} = \frac{|\mathcal{W}|}{|\mathcal{K}|}\frac{|\mathcal{W}'|}{|\mathcal{W}'|}\frac{|\mathcal{K}'|}{|\mathcal{K}'|} = \frac{|\mathcal{W}'|}{|\mathcal{K}'|}\frac{|\mathcal{W}|}{|\mathcal{W}'|}\frac{|\mathcal{K}'|}{|\mathcal{K}|}.$$

Therefore it holds that the probability of generating a weak key when we consider rejection sampling for the first time in our analysis changes by exactly a factor of $(|\mathcal{W}|/|\mathcal{W}'|) \cdot (|\mathcal{K}'|/|\mathcal{K}|)$. This is precisely the probability that a weak key will not be rejected due to weight concerns divided by the probability that key will not be rejected due to weight concerns.

We note that as long as the rejection probabilities for both keys and weak keys is not especially close to 0 or 1, then it is sufficient to sample many keys according to their distributions and observe the portion of these keys that would be rejected.

In order to practically measure the security gained by rejection sampling for the 1-iteration ISD attack against the ephemeral key parameters, we sample 10,000 keys according to KeyGen' and we sample 10,000 weak keys according to KeyGen' and we observe how many of them are rejected. We observe that approximately 39.2% of regular keys are rejected while approximately 67.4% of weak keys are rejected. We therefore conclude for this attack and this parameter set, $\frac{|\mathcal{W}|}{|\mathcal{K}|} = 0.582\frac{|\mathcal{W}'|}{|\mathcal{K}'|}$. Therefore, rejection sampling grants less than 1 additional bit of security back to LEDACrypt.

This attack analysis can be efficiently reproduced for additional parameter sets and alternative notions of weak key with the same result.

3.6 Putting it all together

Finally, we re-calculate the results of Section 3.2 using Theorems 3.2 and 3.4, but accounting for the attack improvement of using advanced information set decoding from Section 3.4 and accounting for the security improvement due to rejection sampling issues in Section 3.5. We re-write the formulas with the substitutions of ϵ_{CPA} (resp. ϵ_{CPA}) for the constant $\frac{1}{2}$ for the reader, and note that the definition of ultra-weak keys has been implicitly modified to have more liberal degree constraints to suit the advanced ISD subroutine being used now.

Let x, y, m be defined as in Theorem 3.4. For the case of claimed-256-bit security for ephemeral key parameters, the fraction of ultra-weak keys recovered by a single iteration of the advanced ISD algorithm is

$$d_v{}^2(d_v-1)^2 m \int_{x=0}^{\epsilon}\int_{y=0}^{\epsilon} (xy)^{d_v-2}\left((\epsilon-x)(\epsilon-y)\right)^m \left(\frac{1}{\epsilon-x}+\frac{1}{\epsilon-y}\right) \mathrm{d}x\mathrm{d}y,$$

and the fraction of these ultra-weak keys out of all possible keys is

$$(\epsilon p) d_v{}^2(d_v-1)^2 \int_{x=0}^{\epsilon}\int_{y=0}^{\epsilon} x^{d_v-2}y^{d_v-2}(\epsilon-x)^m(\epsilon-y)^m \left(\frac{m_0{}^2+m_1{}^2}{(\epsilon-x)(\epsilon-y)}+\frac{m_0m_1}{(\epsilon-x)^2}+\frac{m_0m_1}{(\epsilon-y)^2}\right)\mathrm{d}x\mathrm{d}y.$$

Evaluating with ephemeral key parameters $d_v = 11, m_0 = 7, m_1 = 6, p = 36,877$ and substituting $\epsilon = .577$ yields 1 key recovered in $2^{63.51}$ per single iteration, and 1 ultra-weak key in $2^{44.79}$ of all possible keys. This yields an algorithm making $2^{18.72}$ guesses and recovering 1 in $2^{47.79}$ of the ephemeral keys (accounting for the loss due to rejection sampling and the limited number of iterations).

Substituting $\epsilon = .585$ similarly and evaluating with long-term key parameters $d_v = 13, m_0 = 7, m_1 = 6, p = 152,267$ yields 1 key recovered in $2^{69.48}$ per single iteration and 1 ultra-weak key in $2^{48.59}$ of all possible keys. This yields an algorithm making $2^{20.89}$ guesses and recovering 1 in $2^{51.59}$ of the long-term keys (accounting for the loss due to rejection sampling).

To conclude, we would like to compare this result against the claimed security level of NIST Category 5. Formally, these schemes should be as hard to break as breaking 256-bit AES. Each guess in the ISD algorithms leads to a cost of approximately p^3 bit operations (due to linear algebra and claw finding operations). This is $2^{45.5}$ bit operations for the ephemeral key parameters and $2^{51.6}$ bit operations for the long-term key parameters. A single AES-256 operation costs approximately 2^{15} bit operations. This yields the main result of this section.

Theorem 3.5 (Main). *There is an advanced information set decoding algorithm that costs the same as* $2^{49.22}$ *AES-256 operations and recovers* 1 *in* $2^{47.79}$ *of LEDAcrypt's Category 5 ephemeral keys.*

Similarly, there is an advanced information set decoding algorithm that costs the same as $2^{57.49}$ *AES-256 operations and recovers* 1 *in* $2^{51.59}$ *of LEDAcrypt's Category 5 long-term keys.*

Remark 2. Note that $49.22+47.79 = 97.01 \ll 256$, $57.49+51.59 = 109.08 \ll 256$.

4 Continuum of Weak Keys

In the previous section we have described a family of LEDAcrypt secret keys, which we have defined *ultra-weak*, that can be easily recovered with advanced ISD algorithms. One may think that, to avoid such an issue, the key generation algorithm may be tweaked to reject such keys. However, ultra-weak keys represent only the tip of the iceberg, since there are many more typologies of keys that may be recovered with an ISD algorithm running with extremely low time complexity. Indeed, we can find many more pairs of matrices (H, Q) such that the distribution of ones in the corresponding $L = HQ$ is strongly biased, with respect to random a matrix with the same dimension and weights. With this criterion in mind, we can formally define a family of weak keys as follows.

Definition 4.1. *Let $\mathcal{K}$ be the public key space of LEDAcrypt with parameters n_0, p, d_v, m_0, m_1. Let $T \subseteq \{0, \cdots, p-1\}$ of cardinality $k = (n_0-1)p$ and $\mathcal{W} \subseteq \mathcal{K}$ be the set of all public keys corresponding to secret keys $sk = (H, Q)$ such that the corresponding $L = HQ$ contains (at least) a row whose support is disjoint with T. Finally, we define $\omega = n_0(m_0+m_1)d_v$ and $\mathcal{U}_\omega$ as the uniform distribution of (n_0p)-uples with weight ω. Then, we say that $\mathcal{W}$ is a set of* weak-keys *if*

$$\Pr\left[pk \in \mathcal{W} | (sk, pk) \leftarrow \mathsf{KeyGen}()\right] > \Pr\left[T \cap \mathrm{S}(a) = \varnothing | a \sim \mathcal{U}_\omega\right] = \frac{\binom{n_0p-\omega}{p}}{\binom{n_0p}{p}}.$$

In the remainder of this section, we describe a way to identify many families of keys matching the previous definition; we finally provide estimates on the time complexity required by an ISD searching for these keys, To simplify our analysis, without loss of generality, we significantly reduce the degrees of freedom that can be exploited to define such families of keys, yet, we are able to detect a continuum of families of weak keys. Our results show that considering all of these properties and rejecting such keys, in the key generation algorithm, is clearly infeasible.

4.1 Preliminary considerations on sparse polynomials multiplications

Let $N(c_i)$ denote the set of terms that contribute to the sum in Eq. (7), i.e.

$$N(c_i) = \left\{z \;\text{ s.t. }\; z \notin \bar{\mathrm{S}}(a) \;\text{ and }\; i - z \mod p \notin \bar{\mathrm{S}}(b)\right\}.$$

We now denote with $\tilde{a}$ and $\tilde{b}$ the polynomials obtained by lifting a and b over $\mathbb{Z}[x]/\langle x^p - 1\rangle$ i.e., by mapping the coefficients of a and b into $\{0, 1\} \subset \mathbb{Z}$. Let $\tilde{c} = \tilde{a}\tilde{b}$: we straightforwardly have that $c \equiv \tilde{c} \mod 2$, $|N(c_i)| = \tilde{c}_i$ and $\sum_{i=0}^{p-1} \tilde{c}_i = \mathrm{wt}(a) \cdot \mathrm{wt}(b)$. Let a' be a polynomial such that $\mathrm{S}(a') \supseteq \mathrm{S}(a)$, i.e., such that its support contains that of a (or, in another words, such that its antisupport is contained in that of a); an analogous definition holds for b'. We define $c' = a'b'$: it is immediately seen that $c'_i \geq \tilde{c}_i$ for all i; indeed, we can write $a' = \tilde{a} + s_a$ and $b' = \tilde{b} + s_b$, for two polynomials s_a and s_b taking values in $\{0, 1\} \subseteq \mathbb{Z}$. Then

$$a'b' = (\tilde{a} + s_a)(\tilde{b} + s_b) = \tilde{a}\tilde{b} + s_a\tilde{b} + s_b\tilde{a} + s_as_b = \tilde{c} + s_a\tilde{b} + s_b\tilde{a} + s_as_b.$$

Since $s_a\tilde{b}$, $s_b\tilde{a}$ and $s_a s_b$ have all non-negative coefficients, we have

$$c'_i \geq |N(c_i)| \geq 0, \forall i \in \{0, \cdots, p-1\}. \quad (5)$$

We now derive some properties that link the coefficients of c' to those of c; as we show, knowing portions of the antisupports of a and b is enough to gather information about the coefficients in their product.

Lemma 4.2. *Let $a, b \in \mathbb{F}_2[x]/\langle x^p + 1\rangle$, and $J_a, J_b \subseteq \{0, \cdots, p-1\}$ such that $J_a \supseteq \mathrm{S}(a)$ and $J_b \supseteq \mathrm{S}(b)$. Let $a', b' \in \mathbb{Z}[x]/\langle x^p - 1\rangle$ the polynomials whose coefficients are null, except for those indexed by J_a and J_b, respectively, which are set as 1. Let $c = ab \in \mathbb{F}_2[x]/\langle x^p + 1\rangle$ and $c' = a'b' \in \mathbb{Z}[x]/\langle x^p - 1\rangle$; then*

$$c'_i = 0 \implies c_i = 0.$$

Proof. The result immediately follows from (5) by considering that if $c'_i = 0$ then necessarily $|N(c_i)| = 0$ and, subsequently, $c_i = 0$. □

When the weight of $c = ab$ is maximum, i.e., equal to $\mathrm{wt}(a)\mathrm{wt}(b)$, the probability that a coefficient c_i is null can be related to that of c'_i as stated by the following Lemma.

Lemma 4.3. *Let $a, b \in \mathbb{F}_2[x]/\langle x^p + 1\rangle$, with respective weights ω_a and ω_b, with $\omega_a\omega_b \leq p$. Let $c = ab$ such that $\omega = \mathrm{wt}(c) = \omega_a\omega_b$. Let $J_a \supseteq \mathrm{S}(a)$ and $J_b \supseteq \mathrm{S}(b)$, and $a', b' \in \mathbb{Z}[x]/\langle x^p - 1\rangle$. Let c'_i be the i-th coefficient of $c' = a'b'$, and denote $M = |J_a| \cdot |J_b|$. Then*

$$\Pr\left[c_i = 0 | c'_i\right] = \gamma(M, \omega, c'_i) = \left(1 + \omega \cdot \frac{c'_i}{M + 1 - \omega - c'_i}\right)^{-1}.$$

Proof. The result follows from a combinatorial argument. See B.3 for details. □

Taking into account the previous Lemma, we can derive the probability that specific coefficients in the polynomial product are simultaneously null.

Lemma 4.4. *Let $a, b \in \mathcal{R}$, with respective weights ω_a and ω_b, with $\omega_a\omega_b \leq p$. Let $c = ab$ such that $\omega = \mathrm{wt}(c) = \omega_a\omega_b$. Let $J_a \supseteq \mathrm{S}(a)$ and $J_b \supseteq \mathrm{S}(b)$, and $a', b' \in \mathbb{Z}[x]/\langle x^p - 1\rangle$. Let c'_i be the i-th coefficient of $c' = a'b'$, and denote $M = |J_a| \cdot |J_b|$. Let $V = \{v_0, \cdots, v_{t-1}\} \subseteq \{0, \cdots, p-1\}$; then*

$$\Pr\left[\mathrm{wt}(c|_V) = 0 \mid c'\right] = \zeta(V, c', \omega) = \prod_{\ell=0}^{t-1} \gamma\big(M - \textstyle\sum_{j=0}^{\ell-1} c'_{v_j}, \omega, c'_{v_\ell}\big).$$

Proof. We model the computation of each coefficient c_i with the urn experiment described in the proof of Lemma 4.3. Then, the coefficients indexed by V are modeled through $t = |V|$ extractions without replacement: this is due to the fact that each product $a_\ell b_u$ contributes only to a single coefficient in c'. At the first extraction, the urn contains $M-\omega$ balls of the first color and ω of the second color. We have $c_{v_0}=0$ if and only if all extracted balls are of the first color. Then, at the second extraction, the urn contains $M-\omega-c'_{v_0}$ balls of the first color and ω of the second one. Iteration of this reasoning easily returns the probability expressed by the Lemma. □

4.2 Identifying families of weak keys

We are now ready to use the results presented in the previous section to describe how, in LEDAcrypt, families of weak keys as in Def. 4.1 can be defined. The idea we consider is particularly simple, and builds on the criteria described in Sec. 3. We consider "containers" for each polynomial in the secret key, i.e., polynomials over $\mathbb{Z}[x]/\langle x^p-1\rangle$ whose support contains that of the corresponding polynomials in $\mathbb{F}_2[x]/\langle x^p+1\rangle$. We then combine such containers, and use the results of Lemmas 4.3, 4.4 to find the positions that, with high probability, do not point at set coefficient in the polynomials in $L=HQ$. For the sake of simplicity, and without loss of generality, we describe our ideas for the practical case of $n_0=2$. A visual representation of our proposed constructive method to search for weak keys is found in Appendix C.

We consider sets J_{H_i} such that $J_{H_i}\supseteq \mathrm{S}(H_i)$, for $i=0,1$; the cardinality of J_{H_i} is denoted as B_{H_i}. In analogous way, we define sets $J_{Q_{i,j}}$, for $i=0,1$ and $j=0,1$, with cardinalities $B_{Q_{i,j}}$. To each set J_{H_i} and $J_{Q_{i,j}}$ we associate the polynomials $H'_i, Q'_{i,j}\in\mathbb{Z}[x]/\langle x^p-1\rangle$, respectively. We think of these sets (and the respective polynomials) as "containers" for the polynomials in the secret key. We define

$$L'_{i,j}=H'_iQ'_{j,i}\in\mathbb{Z}[x]/\langle x^p-1\rangle,\ \ (i,j)\in\{0,1\}^2.$$

Each $L'_{i,j}$ is a container for for each product $H_iQ_{i,j}$, i.e., the support of $L'_{i,j}$ contains that of $H_iQ_{i,j}$. Furthermore, the coefficients of $L'_{i,j}$ give us the likelihood with which positions in $H_iQ_{j,i}$ are null. We can indeed apply Lemma 4.3: it is easily seen that, because of the maximum weight requirement in LEDAcrypt key generation, each polynomial $L'_{i,j}$ matches the hypothesis required by the Lemma. For instance, for $L'_{0,0}$, we have $\omega=m_0d_v$, and the coefficients with the lowest values are those that, with the highest probability, are actually null in $H_0Q_{0,0}$. Analogous reasoning applies to the other $L_{i,j}$.

We now consider that each polynomial in L is obtained as $L_i=H_0Q_{0,1}+H_1Q_{1,i}$. We now need to combine the coefficients of the polynomial containers $L'_{i,j}$, to identify positions that are very to be null in each L_i. We here consider a very simple criterion which, despite not being optimal, let us avoiding cumbersome notation and computations and yet leads to good results. We define

$$L'_i=L'_{i,0}+L'_{i,1}=H'_0Q'_{0,i}+H'_1Q'_{1,i}\in\mathbb{Z}[x]/\langle x^p-1\rangle.$$

Let $\pi_i\in S_n$, with $i=0,1$, be a permutation such that the coefficients of $\pi_i\big(L'_i\big)$ are in non decreasing order. We represent a permutation as an ordered set $\{\ell_0,\ell_1,\cdots,\ell_{n-1}\}$, such that the element in position ℓ_u goes in position u.

Then, as a criterion to determine our size-p set T, we choose the first $\frac{p}{2}$ entries of π_0, which we group in a set T_0, and the first $\frac{p}{2}$ ones of π_1, which we group in a set T_1. We then define T as $T = T_0 \cup \{p + \ell | \ell \in T_1\}$.

We now provide an estimate of the number of secret keys that meet the requirements we have imposed, i.e., keys leading to polynomials L_0 and L_1 that do not overlap with the chosen sets T_0 and T_1, respectively.

We analyze the simple case in which initial guesses for containers have constant size, i.e., $B_{H_i} = B_H$ and $B_{Q_{i,j}} = B_Q$, for all i and j; furthermore, we choose $J_{Q_{0,0}} = J_{Q_{0,1}}$ and $J_{Q_{1,0}} = J_{Q_{1,1}}$, for all i, j.

First of all, we compute the number of polynomials $H_iQ_{i,j}$ whose supports are contained in the sets J_{H_i} and $J_{Q_{i,j}}$. We denote with $\mathcal{J}$ the number of secret keys whose polynomials are contained by our initially chosen polynomials H'_i and $Q_{i,j}$; cardinality of this set gets estimated as

$$\mathcal{J} = \eta \left(\binom{B_H}{d_v} \binom{B_Q}{m_0} \binom{B_Q}{m_1} \right)^2,$$

where η is the acceptance ratio in key generation, i.e., the probability that a random choice of matrices H and Q leads to a matrix Q with full weight.

We now compute the number of keys in $\mathcal{J}$ that produce polynomials L_0 and L_1 corresponding to a correct choice for T_0 and T_1, i.e., such that their supports are disjoint with T_0 and T_1, respectively. Because of our choices in the selection of T, this quantity corresponds to the portion of keys in $\mathcal{J}$ that produce polynomials $H_0Q_{0,i}$ and $H_1Q_{1,i}$ with supports that do not overlap with T_i, for $i = 0, 1$. We can then use Lemma 4.4: the probability that a random key from $\mathcal{J}$ is such that the support of the associated polynomials $H_0Q_{0,0}$ does not overlap with T_0 is $\zeta\big(T_0, L'_{0,0}, m_0 d_v\big)$. Then, if we also take into account $H_1Q_{1,0}$, the probability that the support of L_0 does not overlap with T_0 is obtained as

$$\Pr\left[\mathtt{null}(T_0)\right] = \zeta\big(T_0, L'_{0,0}, m_0 d_v\big) \cdot \zeta\big(T_0, L'_{0,1}, m_1 d_v\big).$$

We now take into account T_1: given our restriction on the possible choices for the sets for $Q_{0,1}$ and $Q_{1,1}$, we have $\Pr\left[\mathtt{null}(T_0)\right] = \Pr\left[\mathtt{null}(T_1)\right]$. Then, the probability that a random key from $\mathcal{J}$ is associated to a valid L is

$$\begin{aligned}\Pr\left[\mathtt{null}(T)\right] &= \Pr\left[\mathtt{null}(T_0)\right] \cdot \Pr\left[\mathtt{null}(T_1)\right] \\ &= \left(\zeta\big(T_0, L'_{0,0}, m_0 d_v\big) \cdot \zeta\big(T_0, L'_{0,1}, m_1 d_v\big) \right)^2. \end{aligned} \tag{6}$$

Thus we conclude that, for each choice of sets J_{H_i} and $J_{Q_{i,j}}$, such that $J_{Q_{i,0}} = J_{Q_{i,1}}$ for $i = 0, 1$, we define a family of keys $\mathcal{W}$ with cardinality

$$|\mathcal{W}| \geq |\mathcal{J}| \cdot \Pr[\mathtt{null}(T)],$$

where inequality comes from the fact the above formula does not take into account existence of equivalent keys. Clearly, the coefficients of the covering polynomials $L'_{i,j}$ and, subsequently, the cardinality of $\mathcal{W}$, depend not only on B_H

and B_Q, but also on the actual elements in J_{H_i} and $J_{Q_{i,j}}$. In the following section we provide some concrete examples on how to choose such sets, and compute the portion of associated weak keys.

4.3 Results

In this section we provide practical examples on how to run Algorithm C to define actual families of weak keys. To this end, we need to define clear criteria on how the sets J_{H_i} and $J_{Q_{i,j}}$ can be selected; following the choice of the previous section, we choose $B_{H_i} = B_H$ for all i, $B_{Q_{i,j}} = B_Q$ for all i, j, and $J_{Q_{0,0}} = J_{Q_{0,1}}$, $J_{Q_{1,0}} = J_{Q_{1,1}}$. We then consider three different strategies to pick these sets.

I. For $i = 0, 1$, $a_i \in \{0, \cdots, p-1\}$, we choose

$$J_{H_i} = \{\ell \mod p \,|0 \leq \ell \leq B_H - 1\},$$
$$J_{Q_{i,j}} = \{a_i + \ell \mod p \,|0 \leq \ell \leq B_Q - 1\}.$$

The corresponding family of weak keys is denoted as $\mathcal{W}^{(I)}$.

II. For $i = 0, 1$, $a_i, \delta_i \in \{0, \cdots, p-1\}$, we choose

$$J_{H_i} = \{\ell\delta_i \mod p \,|0 \leq \ell \leq B_H - 1\},$$
$$J_{Q_{i,j}} = \{a_i + \ell\delta_i \mod p \,|0 \leq \ell \leq B_Q - 1\}.$$

The corresponding family of weak keys is denoted as $\mathcal{W}^{(II)}$.

III. For $i = 0, 1$, m_H and $\bar{B}_H$ such that $m_H B_H \leq p$, and $a_i^{(H)} \in \{0, \cdots, p-1\}$, we choose

$$J_{H_i} = \bigcup_{j=0}^{m_H - 1} \{a_i + \ell \mod p |\, 0 \leq \ell \leq B_H - 1\},$$

such that all sets $\{a_i + \ell \mod p|\, 0 \leq \ell \leq B_H - 1\}$ are disjoint. Analogous formalism and notation is used to choose sets $J_{Q_{i,j}}$. The corresponding family of weak keys is denoted as $\mathcal{W}^{(III)}$.

Tables 1, 2 display results testing various weak key families for two different LEDAcrypt parameters sets. The identified families of keys meet Definition 1, so can actually be considered weak. We did not perform optimization over all possible choices: our results, despite being referred to limited choices in the family selection, show that many families of weak keys exist.

5 Attack on All Keys

Assuming the LEDAcrypt approach is parameterized in a balanced way, that is H and Q are similarly sparse, and further assuming that n_0 is a constant, the complexity of the ordinary ISD attack (with a randomly chosen information set) has a complexity of $Exp(\tilde{O}(p^{\frac{1}{2}}))$, but a better asymptotic complexity of

Type	Family Parameters	$\Pr[pk \in \mathcal{W} \mid (sk, pk)]$
I	$B_H = B_Q = 7470$ $a_0 = a_1 = 0$	$\geq 2^{-99.88}$
I	$B_H = 8000, B_Q = 4000$ $a_0 = 2000, a_1 = 2000$	$\geq 2^{-85.25}$
II	$B_H = 6000, B_Q = 4000$ $a_0 = a_1 = 1000, \delta_0 = 200, \delta_1 = 100$	$\geq 2^{-116.32}$
II	$B_H = 8500, B_Q = 4000$ $a_0 = a_1 = 0, \delta_0 = \delta_1 = 127$	$\geq 2^{-90.23}$
III	$m_H = 2, b_H = 4500, m_Q = 2, b_Q = 2500$ $a_0^{(H)} = a_1^{(H)} = [0, 7000], a_0^{(H)} = a_1^{(H)} = [0, 8000]$	$\geq 2^{-101.53}$

Table 1. Fraction of weak keys, for LEDAcrypt instances designed for 128-bit security, with parameters $n_0 = 2$, $p = 14939$, $d_v = 11$, $m_0 = 4$, $m_1 = 3$, for which $\eta \approx 0.7090$. For this parameter set, probability of randomly guessing a null set of dimension p, in a vector of length $2p$ and weight $2(m_0 + m_1)d_v$, is $2^{-154.57}$.

Type	Family Parameters	$\Pr[pk \in \mathcal{W} \mid (sk, pk)]$
I	$B_H = 18000, B_Q = 9000$ $a_0 = a_1 = 9000$	$\geq 2^{-125.18}$
I	$B_H = 24000, B_Q = 12000$ $a_0 = a_1 = 0$	$\geq 2^{-184.21}$
II	$B_H = 18000, B_Q = 9000$ $a_0 = a_1 = 0, \delta_0 = \delta_1 = 5$	$\geq 2^{-125.18}$
II	$B_H = B_Q = 18439$ $a_0 = a_1 = 0, \delta_0 = 1, \delta_1 = 2$	$\geq 2^{-220.52}$
III	$m_H = 1, b_H = 21000, m_Q = 3, b_Q = 4000$ $a_0^{(H)} = a_1^{(H)} = [0]$ $a_0^{(H)} = a_1^{(H)} = [0, 10000, 20000]$	$\geq 2^{-270.30}$

Table 2. Fraction of weak keys, for LEDAcrypt instances designed for 128-bit security, with parameters $n_0 = 2$, $p = 36877$, $d_v = 11$, $m_0 = 7$, $m_1 = 6$, for which $\eta \approx 0.614$. For this parameter set, probability of randomly guessing a null set of dimension p, in a vector of length $2p$ and weight $2(m_0 + m_1)d_v$, is $2^{-286.80}$.

$Exp(\tilde{O}(p^{\frac{1}{4}}))$ may be obtained using structured information sets, even when not attacking weak keys.

To see this, imagine we are selecting the nonzero coefficients of H' and Q' completely at random, aside from a sparsity constraint. The sparsity constraint needs to be set in such a way that the row weight of the product $H'Q'$ (restricted to two cyclic blocks) has row weight no more than p. This further constrains the row weight of each cyclic block of H' and Q' to be approximately $\left(\frac{p\ln(2)}{n_0}\right)^{\frac{1}{2}} = O(p^{\frac{1}{2}})$. The probability of success per iteration is then at least $O\left(\left(\frac{\ln(2)}{pn_0}\right)^{\frac{1}{2}\cdot(\sum_{i=0}^{n_0} m_i + n_0 d_v)}\right)$. With balanced parameters, d_v and the m_i are $O(p^{\frac{1}{4}})$, thus the total complexity is indeed $Exp(\tilde{O}(p^{\frac{1}{4}}))$. Note that when H'

and Q' are random aside from the sparsity constraint, the probability that the supports of H' and Q' contain the supports of H and Q respectively does not depend on H and Q, so the structured ISD algorithm is asymptotically better than the unstructured ISD algorithm, even when we ignore weak keys.

These asymptotics however are not in and of themselves surprising. The very simple attack which simply tries to enumerate all the possible values of H and Q is also asymptotically $Exp(\tilde{O}(p^{\frac{1}{4}}))$. However, in this section, we show that for average keys, there are some proposed parameter sets in the Round 2 version of LEDAcrypt where we obtain a significantly improved concrete attack complexity by using a structured information set. To create the structured information set we will choose H' and Q' which is intermediate between the completely random and sparse distribution used to compute asymptotic complexity of the approach in the average case, and the denser, but decidedly nonrandom distribution used to attack extreme weak keys. As an exact calculation of the performance of the attack will be very difficult and perhaps infeasible, the analysis in this section will be more heuristic, and will aim to give a rough upper bound on the complexity of the structured ISD attack in the average case.

The parameter set we will focus on is the category 5 CPA parameter set from the Round 2 LEDAcrypt submission, namely $p = 36877; n_0 = 2; d_v = 11, m = (7, 6)$. We will estimate the complexity of both the standard ISD attack with a random information set and the structured information set attack by using a simplified model of advanced ISD algorithms, where the cost of an iteration is p^3 and the iteration succeeds whenever 6 or fewer bits of a row of L are inside the information set. Using this model to estimate the cost of the unstructured ISD attack for key recovery, we multiply the cost per iteration, p^3, by the probability of success per iteration. The probability of success is the probability that one of the p rows of L, which has weight $w = 2 * 11 * (7 + 6) = 286$ has at most 6 of its supports inside the information set. This probability is approximately $p \sum_{i=0}^{6} \binom{w}{i} \cdot 2^{-w}$. This gives a complexity for the standard attack of $p^3 \cdot \frac{2^w}{p \sum_{i=0}^{6} \binom{w}{i}} \approx 2^{277}$. This is quite close to the complexity of 2^{281} bit operations claimed by the LEDAcrypt submission for the cost of the BJMM ISD algorithm as applied to key recovery for this parameter set. The discrepancy is likely due to some small constant factors we are ignoring in our analysis. For a fair comparison between structured and ISD, we will use our simplified method to estimate complexity for both the standard ISD attack and the unstructured ISD attack.

Our analysis of the structured information set attack will proceed as follows: First we will describe a distribution of information sets used in a version of the attack. Then, we will describe a possible scenario where the attack will succeed and compute the expected probability per iteration that the scenario obtains. In computing this probability we will take note of the factors in the computation of the probability of success that are dependent on the private key. In showing that these factors are large, i.e. $\approx \frac{1}{2}$, we will be demonstrating that the high per-iteration success probability relative to randomly chosen information sets does not depend upon the private key being chosen from a rare class of weak keys. Once we have demonstrated this and given a lower bound on the per iteration success probability we use this to compute an upper bound on the average case complexity for the attack.

We will choose the information set as in previous versions of the attack by choosing structured polynomials H' and Q' and letting the information set be the anti-support of $L' = H'Q'$. In this case, we will structure our information set as follows:

H'_0 will be chosen with $d'_v = 6$ chunks of consecutive nonzero bits in one ring representation. $Q'_{0,0}$ will be chosen with $m'_0 = 5$ chunks of consecutive nonzero bits and $Q'_{0,1}$ will be chosen with $m'_1 = 5$ chunks of consecutive nonzero bits, both in the same ring representation as used for H'_0.

Likewise H'_1 will be chosen with $d'_v = 6$ chunks of consecutive nonzero bits in an independently chosen ring representation from the one used for H'_0. $Q'_{1,1}$ will be chosen with $m'_0 = 5$ consecutive nonzero bits and $Q'_{1,0}$ will be chosen with $m'_1 = 5$ consecutive nonzero bits, both in the same ring representation as used for H'_1.

We will set the length of each consecutive chunk of nonzero bits to be equal to a quantity B that we compute so that the anti-support of a row of the matrix representation of $H'Q'$ is expected to have weight p. The product $H'Q'$ will contain two cyclic blocks, each of which will have rows containing $d'_v(m'_0 + m'_1) = 30$ chunks of size $2B - 1$ in one ring representation and 30 chunks of the same size in another ring representation. As these chunks may potentially overlap, the expected Hamming weight of the product is approximately $2 \cdot 36877 \left(1 - e^{-\frac{60(2B-1)}{36877}}\right)$. We can use this together with the requirement that the information set needs to be of size $p = 36877$ to bound B, resulting in the requirement $2B - 1 < \frac{36877 \ln(2)}{60}$. This gives a maximum chunk size of $B = 213$

The scenario we will consider for successful key recovery will be as follows: Assume the nonzero bits of H_0 are clustered in some ring representation, which we denote Rep_0, such that 4 pairs of nonzero bits are close enough that each pair will fit in a block of size B. Assume 1 pair of nonzero bits of $Q_{0,0}$ are close enough to fit in a single block of size B also in Rep_0. Likewise, assume the nonzero bits of H_1 are clustered in some other ring representation, Rep_1, such that 4 pairs of nonzero bits are close enough that each pair will fit in a block of size B. Assume 1 pair of nonzero bits of $Q_{1,1}$ are close enough to fit in a single block of size B in Rep_1. We call this the Clustering Assumption.

Our attack succeeds when, for H, Q or some equivalent private key,

1. $H'_0, Q'_{0,0},$ and $Q'_{0,1}$ are chosen using Rep_0
2. $H'_1, Q'_{1,0},$ and $Q'_{1,1}$ are chosen using Rep_1.
3. The close pairs of nonzero bits of H_0, H_1, $Q_{0,0}$, and $Q_{1,1}$ are each contained within a consecutive chunk of nonzero bits in H'_0, H'_1, $Q'_{0,0}$, and $Q'_{1,1}$ respectively.
4. All but one of the remaining nonzero bits in each of H_0, H_1, $Q_{0,0}$, $Q_{0,1}$, $Q_{1,0}$ and $Q_{1,1}$ are contained within a consecutive chunk of nonzero bits in H'_0, H'_1, $Q'_{0,0}$, $Q'_{0,1}$, $Q'_{1,0}$ and $Q'_{1,1}$.

5. At least 60 of the 66 nonzero bits of L not guaranteed to be contained within the support of L' by the above considerations, nevertheless happen to fall inside the support of L'.

We first examine the clustering assumption (which is dependent on the private key). There are several ways in which the support of H could be contained in the support of H', but we initially consider the case that 4 pairs of bits of H are clustered within regions of (cyclic) size ≤ 213. This occurs with probability 2^{-12}. See appendix B.4 for details. The probability that at least one pair of the nonzero bits of $Q_{0,0}$ cluster together is $(6+5+4+3+2+1)\frac{414}{36877} \approx \frac{1}{4}$. Thus we expect about $\frac{36876}{2} \cdot 2^{-12} \cdot \frac{1}{4} \approx 1$ of the $\frac{36876}{2}$ distinct ring representations to meet our criteria.

5.1 Conditions 1,2

Now we now examine, given the clustering assumption holds, the probability that either H_0' or H_1' is chosen using representation 0 while the other is chosen using representation 1. (Because in designing our attack we have set $m_0' = m_1'(= 5)$, we get the same distribution of information sets either way.) There are $\frac{p-1}{2}$ ring representations with a different notion of consecutivity. Thus the probability is approximately $\frac{8}{(p-1)^2}$.

5.2 Conditions 3,4

Given our criteria are met regarding the ring representation, we can estimate the probability that six of the seven regions containing nonzero bits of H_0, and all four of the regions containing a pair of bits, are contained within a nonzero chunk of H_0'. The nonzero regions may match with the nonzero chunks in any order. Therefore, the probability contains a factor of 6!. Additionally, any of the three unpaired bits in H_0 may be the one not covered by a chunk of nonzero bits in H_0'. Therefore, the probability contains an additional factor of 3. The probability that a given chunk covers a given region with one nonzero bit is $\frac{213}{36877}$, while the probability a chunk covers a region with two nonzero bits is (on average) half of this figure, i.e. $\frac{106.5}{36877}$, since we expect such regions to be uniformly distributed in size from 2 to 213 yielding probabilities that uniformly range from $\frac{212}{36877}$ to $\frac{1}{36877}$. Thus, our probability of covering H_0 with H_0' given a ring representation meeting our criteria is $6! \cdot 3 \cdot \frac{213}{36877}^6 \cdot 2^{-4}$. The same probability holds for matching H_1 with H_1' given an appropriate ring representation.

Likewise, given our criteria are met regarding the ring representation, we can estimate the probability that five of the six regions containing nonzero bits of $Q_{0,0}$, including the one containing a pair of bits, are contained within a nonzero chunk of $Q_{0,0}'$. The nonzero regions may match with the nonzero chunks in any order. Therefore, the probability contains a factor of 5!. Additionally, any of the five unpaired bits in $Q_{0,0}$ may be the one not covered by a chunk of nonzero bits in $Q_{0,0}'$. Therefore, the probability contains an additional factor of 5. The probabilities for a given region to be covered by a given chunk are the same as above. Thus, our probability of covering $Q_{0,0}$ with $Q'_{0,0}$ given a ring representation meeting our criteria is $5! \cdot 5 \cdot \frac{213}{36877}^5 \cdot 2^{-1}$. The same probability holds for matching $Q_{1,1}$ with $Q'_{1,1}$ given an appropriate ring representation.

We now estimate the probability, given our criteria are met regarding the ring representation, that five of the six nonzero bits of $Q_{0,1}$ are contained within a nonzero chunk of $Q'_{0,1}$. The nonzero regions may match with the nonzero chunks in any order. Therefore, the probability contains a factor of 5!. Additionally, any of the six unpaired bits in $Q_{0,1}$ may be the one not covered by a chunk of nonzero bits in $Q'_{0,1}$. Therefore, the probability contains an additional factor of 6. The probabilities for a given region to be covered by a given chunk are the same as above. Thus, our probability of covering $Q_{0,1}$ with $Q'_{0,1}$ given a ring representation meeting our criteria is $5! \cdot 6 \cdot \frac{213}{36877}^5$. The same probability holds for matching $Q_{1,0}$ with $Q'_{1,0}$ given an appropriate ring representation.

5.3 Condition 5

This leaves us to calculate the probability that at least 60 of the remaining 66 bits of L fall within the support of L'. As L', by construction has density $\frac{1}{2}$ we may approximate the probability that this occurs as $\sum i = 0^6 \binom{66}{i} \cdot 2^{-66}$. This is in fact an underestimate; due to the clustering assumption, and the corresponding assumptions regarding the ring representations used for the construction of L', the probabilities for some bits in question of L and L' overlapping are significantly correlated, making the above scenario more likely than it would be under the assumption of independence.

It should however be noted that this probability includes a factor dependent on the private key. We must check the probability for a given private key that it is possible, given the satisfaction of Conditions 1–4 in our scenario for the fifth to hold. We estimate this probability by trying to characterize the degrees of freedom in choosing information sets consistent with the first four success conditions. First, there is a degree of freedom due to the fact that we have 3 choices for the uncovered bit in each of H_0 and H_1, 5 choices for the uncovered bit in each of $Q_{0,0}$ and $Q_{0,1}$, and 6 choices for the uncovered bit in each of $Q_{0,1}$, and $Q_{1,0}$. This gives $3^2 * 5^2 * 6^2 = 8100$ essentially independent chances for a set of 66 bits to satisfy the fifth condition.

There are additional degrees of freedom, due to the fact that we may shift the locations of each chunk of H' or Q' that covers a single bit of H or Q to reach any of up to $2B-1$ different additional bit positions, and we may shift the locations of chunks of H' and Q' covering pairs of bits to reach any of up to $1.5B-1$ additional bit positions on average. This means that we expect $2p(1-e^{-\frac{217B-143}{p}}) \approx 52613$ of the $2p = 73754$ bits of L to be reachable by an L' consistent with the choices of bits of H and Q covered by H' and Q'. This gives a probability of 0.71 that each of the 66 bits in question will be reachable, and consequently a probability of $\sum_{i=0}^{6} \left(\binom{66}{i} \cdot 0.71^{66-i} 0.29^i\right) \approx 8.3 \times 10^{-5}$ that at least 60 of them will be reachable. This is a low probability, however, as stated before we have 8100 opportunities. Thus, accounting for all degrees of freedom, the probability that a given L can be covered by an L' meeting conditions 1 through 5, given that conditions 1 through 4 are met is $1 - e^{-8100 \cdot 8.3 \times 10^{-5}} \approx 0.47$. This is in fact likely to be an underestimate of the probability due to correlations between the probabilities various of the bits are reachable. Nonetheless, it is sufficiently high to illustrate that the attack works for typical private keys.

Finally, we must account for equivalent keys. All the probabilities we considered up to this point were based on the assumption that it was H and Q rather than some equivalent key $(x^{\alpha}H_0, x^{\beta}H_1, x^{\gamma-\alpha}Q_{0,0}, x^{\gamma-\alpha}Q_{0,1}, x^{\gamma-\beta}Q_{1,0}, x^{\gamma-\beta}Q_{1,1})$ that was being (mostly) covered by H' and Q'. The success probability is increased by a full factor of p due to the degree of freedom γ. This degree of freedom corresponds to a cyclic shift of each block of the information set. Due to the use of two different ring representations in constructing that information set, it is very likely that any shift will be large in at least one of the two ring representations and will completely change what bits of L are covered by L'. The other two degrees of freedom are not so easy to analyze, but we can see that shifting α only affects whether H or Q is covered if a nonzero bit of H_0, $Q_{0,0}$ or $Q_{0,1}$ enters or leaves the support of H' or Q'. $2(d_v' + m_0' + m_1') = 32$ bits enter or leave for every shift from one value of α to the next consecutive value in the appropriate ring representation. However, only $d_v + m_0 + m_1 = 24$ of the $3p$ bits of H_0, $Q_{0,0}$ and $Q_{0,1}$ are nonzero. Thus we only expect $\frac{32 \cdot 24}{3} = 256$ of the p possible cyclic shifts of α to result in independent outcomes. Likewise for β. So, considering equivalent keys increases our success probability per iteration by about $256^2 \cdot p$.

5.4 Putting it all together

We thus derive an approximate lower bound for the average-case per-iteration probability of key recovery with our structured information sets of:

$$\frac{8}{(p-1)^2}\left(3 \cdot 5 \cdot 6 \cdot 6! \cdot 5! \cdot 5! \cdot \left(\frac{213}{36877}\right)^1 6 \cdot \frac{1}{2^5}\right)^2 \sum_{i=0}^{6} \binom{66}{i} \frac{1}{2^{66}} \cdot 256^2 p \quad \approx \frac{1}{2^{224}}$$

Estimating a cost of 2^{45} bit operations for each iteration gives a cost of at most 2^{269} for attacking average keys, which is less than the estimated cost of 2^{277} for unstructured ISD. Note that we only considered one scenario by which our structured information sets might recover the private key, and there are several similarly probable scenarios which may also occur, so the real cost is probably less than our estimate. This demonstrates that for real parameters of interest, our attack affects average case security and not just a class of weak keys that might be removed by more aggressive rejection sampling. Due to the complexity of calculations involved, ruling out similar attacks via rejection sampling will be quite difficult.

6 Conclusion

In this work, we demonstrated a novel, real-world attack against LEDAcrypt – one of 17 remaining 2nd Round candidates for standardization in NIST's Post-Quantum Cryptography competition. The attack involved a customized form of Information Set decoding, which carefully guesses the information set in a non-uniform manner so as to exploit the unique product structure of the keys in LEDAcrypt's design. The attack was most effective against classes of weak keys in the proposed parameter sets asserted to have 256-bit security (demonstrating a trade-off between computational time and fraction of the key space recovered that was better than expected even of a 128-bit secure cryptosystem), but the attack also substantially reduced security of all parameter sets similarly.

Moreover, we demonstrated that these type of weak keys are present throughout the key space of LEDAcrypt, so that simple "patches" such as rejection sampling cannot repair the problem. This was done by demonstrating a continuum of progressively larger classes of less weak keys and by showing that the same style of attack reduces the average-case complexity of certain parameter sets.

Acknowledgements. We thank Corbin McNeill for his contributions to our analysis of the effect of rejection sampling on the weak key attack. This work is funded in part by NSF grant award number 1906360.

References

1. National institute of standards and technology: Post-quantum cryptography project, 2016. `https://csrc.nist.gov/projects/post-quantum-cryptography`.
2. Decoding random binary linear codes in 2n/20: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 520–536, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
3. Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status report on the first round of the nist post-quantum cryptography standardization process. 2019.
4. Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. LEDAcrypt. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.
5. Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
6. Daniel J. Bernstein. Grover vs. mceliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography*, pages 73–80, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
7. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 743–760, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

8. Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos. HQC. available at `https://csrc.nist.gove/projects/post-quantum-cryptography/round-2-submission`, 2019. Technical report, National Institute of Standards and Technology.
9. Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Wen Wang. Classic McEliece. available at `https://csrc.nist.gove/projects/post-quantum-cryptography/round-2-submission`, 2019. Technical report, National Institute of Standards and Technology.
10. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th Annual ACM Symposium on the Theory of Computing*, pages 212–219, Philadephia, PA, May 1996.
11. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.
12. Ghazal Kachigar and Jean-Pierre Tillich. Quantum information set decoding algorithms. pages 69–89, 03 2017.
13. Kazukuni Kobara and Hideki Imai. Semantically secure McEliece public-key cryptosystems-conversions for McEliece PKC. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 19–35, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany.
14. P. Lee and E. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT 88*, volume 330, pages 275–280. Springer Verlag, 1988.
15. Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Information Theory*, 34(5):1354–1359, September 1988.
16. Carl Löndahl, Thomas Johansson, Masoumeh Koochak Shooshtari, Mahmoud Ahmadian-Attari, and Mohammad Reza Aref. Squaring attacks on mceliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Des. Codes Cryptography*, 80(2):359–377, August 2016.
17. Baldi M., Bodrato M., and Chiaraluce F. A new analysis of the mceliece cryptosystem based on qc-ldpc codes. *Ostrovsky R., De Prisco R., Visconti I. (eds) Security and Cryptography for Networks. SCN 2008.*, volume 5229, 2008.
18. Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, Martin Tomlinson. NTS-KEM. available at `https://csrc.nist.gove/projects/post-quantum-cryptography/round-2-submission`, 2019. Technical report, National Institute of Standards and Technology.
19. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in O(2**0.054n). In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 107–124, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
20. 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*, pages 203–228, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

21. Robert McEliece. A public-key cryptosystem based on algebraic coding theory. *The Deep Space Network (DSN) Progress Report*, 44:114–116, 1978.
22. Dustin Moody and Ray Perlner. Vulnerabilities of "mceliece in the world of escher". In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography*, pages 104–117, Cham, 2016. Springer International Publishing.
23. Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zemor. BIKE. available at `https://csrc.nist.gove/projects/post-quantum-cryptography/round-2-submission`, 2019. Technical report, National Institute of Standards and Technology.
24. Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Control and Inf. Theory*, 15(2):159–166, 1986.
25. Ray Perlner. Optimizing information set decoding algorithms to attack cyclosymmetric mdpc codes. In Michele Mosca, editor, *Post-Quantum Cryptography*, pages 220–228, Cham, 2014. Springer International Publishing.
26. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
27. Nicolas Sendrier. Decoding one out of many. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer Verlag, 2011.
28. P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 124–134, 1994.
29. Jacques Stern. A method for finding codewords of small weight. In *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*, pages 106–113, 1988.

A Preliminaries

A.1 Overview of LEDAcrypt: QC-LDPC codes

The 2nd Round submission to NIST's PQC standardization process, LEDAcrypt, includes a key-encapsulation mechanism (KEM) built from the Niederreiter cryptosystem (LEDAcrypt KEM) and a public-key encryption (PKE) scheme built from the McEliece cryptosystem (LEDAcrypt PKC), both based on linear error-correcting codes. LEDAcrypt crucially is built on top of QC-LDPC codes, or Quasi-Cyclic Low-Density Parity-Check codes. We briefly survey the construction of such codes (with emphasis on the details of LEDAcrypt) for the reader.

A $p \times p$ circulant matrix A is a matrix of the form

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{p-1} \\ a_{p-1} & a_0 & a_1 & \cdots & a_{p-2} \\ a_{p-2} & a_{p-1} & a_0 & \cdots & a_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{bmatrix},$$

that is, all of the rows (resp. columns) are cyclic shifts of the first row (resp. column).

A Quasi-Cyclic (QC) matrix is a matrix of the form

$$B = \begin{bmatrix} B_{0,0} & B_{0,1} & \cdots & B_{0,w-1} \\ B_{1,0} & B_{1,1} & \cdots & B_{1,w-1} \\ \vdots & \vdots & \ddots & \vdots \\ B_{z-1,0} & B_{z-1,1} & \cdots & B_{z-1,w-1} \end{bmatrix}$$

for two positive integers w and z, where each $B_{i,j}$ is a circulant matrix.

The set of $p \times p$ binary circulant matrices forms a ring under matrix addition and multiplication modulo 2. The algebra of the polynomial ring $\mathbb{F}_2[x]/\langle x^p + 1\rangle$ is isomorphic to the ring of $p \times p$ circulant matrices over $\mathbb{F}_2$.

Binary error correcting codes use a redundant representation of information primarily to detect and correct bit errors that occur during transmissions or storage. Let $\mathbb{F}_2^k$ denote the k-dimensional vector space defined on $\mathbb{F}_2$. A binary code, denoted $\mathcal{C}(n,k)$, is defined as a bijective map $\mathcal{C}(n,k) : \mathbb{F}_2^k \to \mathbb{F}_2^n, n, k \in \mathbb{N}, 0 < k < n$ between any binary k-tuple (called an *information word*) and a binary n-tuple (called a *codeword*). We call n the length of the code and k the dimension of the code.

Encoding via $\mathcal{C}(n,k)$ converts an information word $u \in \mathbb{F}_2^k$ into a codeword $c \in \mathbb{F}_2^n$. Given a codeword $\hat{c} = c + e$ corrupted by an error vector $e \in \mathbb{F}_2^n$ with Hamming weight $t > 0$, decoding aims to recover the original information word u and the error vector e. A code is called *t-error-correcting* if, for any value of e of Hamming weight t, given $\hat{c}$ there is an efficient decoding procedure that retrieves (u, e). Further, the code $\mathcal{C}(n,k)$ is *linear* if and only if the set of its 2^k codewords is a k-dimensional subspace of the vector space $\mathbb{F}_2^n$.

For any linear code, there are two equivalent descriptions of the code that are critical to its use: the generator matrix G and the parity-check matrix H.

Given a linear code $\mathcal{C}(n, k)$ and the vector subspace $\Gamma \subset \mathbb{F}_2^n$ containing its 2^k codewords, it is possible to choose k linearly independent codewords $\{g_0, g_1, ..., g_{k-1}\} \in \mathbb{F}_2^n$ to form a basis of Γ. That is, any codeword $c = (c_0, c_1, ..., c_{n-1})$ can be written as a linear combination of the basis vectors; i.e.

$$c = u_0 g_0 + u_1 g_1 + ... + u_{k-1} g_{k-1}$$

where the $u_i \in \{0, 1\}$ are the representatives of $u = (u_0, u_1, ..., u_{k-1})$ that the code maps into c. This can be re-written as $c = uG$, where G is a full row rank $k \times n$ binary matrix; i.e.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}.$$

The set of all n-bit vectors in $\mathbb{F}_2^n$ that are orthogonal to any codeword of the code subspace Γ is its orthogonal complement $\Gamma^\perp$. Its dimension is $\dim(\Gamma^\perp) = n - \dim(\Gamma) = n - k = r$. A basis of $\Gamma^\perp$ is similarly obtained by sampling r linearly independent vectors in $\Gamma^\perp$ and writing

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{r-1} \end{bmatrix}.$$

The $r \times n$ matrix H is known as the parity-check matrix of the code $\mathcal{C}(n, k)$, and for any n-bit vector $x \in \mathbb{F}_2^n$, the $r \times 1$ vector $s = Hx^T$ is known as the syndrome of x through H. Note that every codeword $c \in \Gamma$ satisfies the equality $Hc^T = 0_{r \times 1}$; that is a codeword belonging to $\mathcal{C}(n, k)$ has a null syndrome through H.

A QC code is defined as a linear (block) code $\mathcal{C}(n, k)$ with information word size $k = p \cdot k_0$ and codeword size $n = p \cdot n_0$, where n_0 is the basic block length of the code and each cyclic shift of a codeword by n_0 symbols gives another valid codeword. LEDAcrypt relies on a QC code $\mathcal{C}(p \cdot n_0, p \cdot k_0)$ where the generator and party-check matrices are composed of $p \times p$ circulant sub-matrices (i.e. blocks).

A Low-Density Parity-Check (LDPC) code $\mathcal{C}(n, k)$ is a special type of linear block code characterized by a highly-sparse parity-check matrix H. In particular, the Hamming weight of a column of H, denoted d_v, is significantly smaller than its length r and in fact decreases sub-linearly with it. LEDAcrypt is based on a QC-LDPC code customized to the security levels targeted by NIST's PQC standardization process.

A.2 Overview of LEDAcrypt's QC-LDPC code-based Niederreiter and McEliece cryptosystems

We briefly review the relevant portions of LEDAcrypt's QC-LDPC-based, public-key encryption algorithms, McEliece $\Pi^{\mathsf{McE}} = (\mathsf{KeyGen}^{\mathsf{McE}}, \mathsf{Enc}^{\mathsf{McE}}, \mathsf{Dec}^{\mathsf{McE}})$ and

Niederreiter $\Pi^{\mathtt{Nie}} = (\mathsf{KeyGen}^{\mathtt{Nie}}, \mathsf{Enc}^{\mathtt{Nie}}, \mathsf{Dec}^{\mathtt{Nie}})$. In LEDAcrypt, $\Pi^{\mathtt{McE}}$ is used to construct an IND-CCA2 public-key encryption scheme via the KI-γ transformation of Kobara-Imai [13] in the ROM, while $\Pi^{\mathtt{Nie}}$ is transformed via HHK [11] to produce an IND-CCA key-exchange mechanism in the ROM (and QROM).

However, our attacks will aim to recover the secret key of these cryptosystems from their public keys alone. That is, the form of encryption, decryption, encapsulation, and decapsulation will be immaterial here. Therefore, we will focus only on the underlying KeyGen algorithms, which are substantially similar (admitting the same form of attack). In particular, both key generation algorithms consider an appropriate QC-LDPC code $\mathcal{C}(n,k)$, have the same secret key material, but differ in whether they output a public key in the form of a parity-check matrix or a generator matrix systematic form.

Let $\mathsf{perm}(\cdot)$ denote the permanent of a matrix. Let $w(\cdot)$ denote the number of nonzero coefficients of a polynomial (its *weight*). Note that if 2 is a primitive element of $\mathbb{Z}_p$, then its order $\mathrm{ord_p}(2)$ is equal to the order of the multiplicative group of the field; i.e. $\mathrm{ord_p}(2) = \mathrm{p} - 1$.

Both cryptosystems' key generation algorithms rely on the following fact.

Fact A.1. *[4, cf. Theorem 1.1.14] Let $p > 2$ be a prime such that $\mathrm{ord_p}(2) = \mathrm{p}-1$ and Q is an $n_0 \times n_0$ matrix of elements of $\mathbb{F}_2[x]/\langle x^p+1\rangle$. If $\mathsf{perm}(w(Q))$ is odd and $\mathsf{perm}(w(Q)) < p$, then Q is non-singular.*

Key Generation. Now we describe the KeyGen algorithm(s).

Fix a code $\mathcal{C}(n,k)$. Let the codeword length be $n = p \cdot n_0$ and the information word length $k = p(n_0 - 1)$, where $n_0 \in \{2,3,4\}$ and p is a prime number such that $\mathrm{ord_p}(2) = \mathrm{p} - 1$.

1. $(H, Q) \leftarrow \mathsf{GenHQ}(\mathsf{seed})$: Find two random binary matrices corresponding to the secret quasi-cyclic $p \times p \cdot n_0$ parity-check matrix H of a given QC-LDPC code and also a $p \cdot n_0 \times p \cdot n_0$ quasi-cyclic sparse binary matrix Q.
   - Note that $H = [H_0, ..., H_{n_0-1}]$, where each H_i is a $p \times p$ circulant block with $w(H_i) = d_v$ for $0 \leq i < n_0$.
   - Note that Q is an $n_0 \times n_0$ block matrix with $w([Q_{i,0}, ..., Q_{i,n_0-1}]) = m$ for $0 \leq i < n_0$.
2. Compute $L := H \cdot Q$.
   - Writing $L = [L_0, ..., L_{n_0-1}]$, note that each $L_j = \sum_i H_i Q_{i,j}$ is a $p \times p$ circulant block.
3. If there is an index j so that $w(L_j) \neq d_v \times m$, start over.
4. For $i = 0$ to $n_0 - 2$, compute $M_i := (L_{n_0-1})^{-1} L_i$.
   - M is thus the parity-check matrix in systematic form and from this, the systematic generator matrix can easily be derived. Note that each M_i is a $p \times p$ circulant block.
5. Output public key $\mathsf{pk}^{\mathtt{Nie}} = [M_0|...|M_{n_0-2}|I_p]$ and secret key $\mathsf{sk}^{\mathtt{Nie}} = (H, Q)$.

In the case of McEliece style key generation, the final output is $\mathsf{sk}^{\mathtt{McE}} = \mathsf{sk}^{\mathtt{Nie}}$ and $\mathsf{pk}^{\mathtt{McE}} = [I_{p(n_0-1)}|[M_0|...|M_{n_0-2}]^T]$, where I_t denotes the $t \times t$ identity matrix.

Finally, the parameter sets of LEDAcrypt that we explicitly consider in this work are shown in Table 4 (although similar forms of our results hold for all parameter sets).

NIST Category	Security Type	p	d_v	m_0	m_1	n_0
1 (128-bit)	IND-CPA	14,939	11	4	3	2
5 (256-bit)	IND-CPA	36,877	11	7	6	2
5 (256-bit)	IND-CCA	152,267	13	7	6	2

Table 3. LEDAcrypt parameter sets that we consider in this paper.

A.3 The standard attack: Information Set Decoding

Decoding an error-affected codeword for random codes is known to be an NP-complete problem. [5]. The best known decoding algorithms which do not exploit code structure are based on information-set decoding (ISD), an approach introduced by Eugene Prange in [26]. The technique requires finding a large set of error-free coordinates in a noisy codeword vector such that the corresponding columns of the generator matrix form an invertible submatrix. The indices of the error-free coordinates are known as the *information set I*.

Let G be a generator matrix of a code C. Let I be an information set such that G restricted to I, denoted G_I, forms an invertible submatrix. Let m be an information word and $c = mG + e$, a noisy codeword. Then $c_I = (mG + e)_I = (mG)_I + e_I = (mG)_I$, and m can be recovered by computing $c_I \cdot G_I^{-1} = (mG)_I \cdot G_I^{-1}$. This essentially outlines a *message recovery attack*.

The McEliece and Niederreiter cryptosystems use error-correcting codes such that the public key is a random-looking representation of a code, while the secret key is a representation of the same code that allows efficient decoding. In the case of LEDAcrypt, the dual of the public code contains low-weight codewords that coincide with the rows of the sparse parity-check matrix of the public code. Recovering a sparse parity-check matrix of the code allows for efficient decoding, and can thus be considered a *key recovery attack*.

A modified version of ISD can be used to to recover low-weight codewords from the dual of the public code. In the case where $n_0 = 2$, one must select a subset of half the columns of L so that the support of a row of L is outside of those columns. If successful, then the row of L can be recovered by simple linear algebra and recognized as the private key by having low hamming weight. Many improvements of Prange's ISD algorithm have been published, and in particular the series of improved ISD algorithms from [29] to [20] can be used to recover low-weight code words from the dual of the public code with little additional cost, when the subset of half the columns of L is instead chosen so that no more than 4–6 of the support columns of a row of L are within the chosen subset.

A.4 Sparse polynomials

We now recall some basic fact about polynomial multiplication in the rings $\mathbb{F}_2[x]/\langle x^p+1\rangle$ and $\mathbb{Z}[x]/\langle x^p-1\rangle$, which will be useful for our treatment.

Let $a, b \in \mathcal{R}$ and $c = ab$; we then have

$$c_i = \bigoplus_{z=0}^{p-1} a_z b_{z'}, \quad z' = i - z \mod p,$$

where the operator $\bigoplus$ highlights the fact that the sum is performed over $\mathbb{F}_2$. Taking into account antisupports, we can rewrite the previous equation as

$$c_i = \bigoplus_{\substack{z \notin \bar{\mathrm{S}}(a) \\ z'=i-z \mod p,\ z' \notin \bar{\mathrm{S}}(b)}}^{p-1} a_z b_{z'}. \tag{7}$$

B Proofs

B.1 Proof of Theorem 3.3

Proof. Let l_0 be the location of the first nonzero bit of $H_0Q_{0,0} + H_1Q_{1,0}$, l_1 the location of the first nonzero bit of $H_0Q_{0,1} + H_1Q_{1,1}$. Let j_0, j_1 be the first non zero bit of H_0, H_1, respectively. Suppose that the nonzero bits of H_0, H_1 are located within a block of length A_0, A_1, respectively.

Once j_0 is fixed, there are two blocks of Q which may influence the location l_0. If l_0 is influenced by $Q_{0,0}$, there are $\binom{\frac{p}{2}-A_0-1}{m_0-1}$ arrangements of the bits of $Q_{0,0}$ and $\binom{\frac{p}{2}-A_1-1}{m_1}$ possible arrangements of the bits of $Q_{1,0}$. Otherwise, if l_0 is influenced by $Q_{1,0}$, there are $\binom{\frac{p}{2}-A_0-1}{m_0}\binom{\frac{p}{2}-A_1-1}{m_1-1}$ arrangements of $Q_{0,0}$, $Q_{1,0}$, respectively. A similar calculation holds for the locations of l_1 once j_1 is fixed.

Summing over all $j_0, A_0, j_1, A_1, l_0, l_1$ we obtain

$$\begin{aligned}
&\frac{p-1}{2} \sum_{j_0=0}^{p-1} \sum_{A_0=d_v}^{\frac{p}{2}} \binom{A_0-1}{d_v-2} \sum_{j_1=0}^{p-1} \sum_{A_1=d_v}^{\frac{p}{2}} \binom{A_1-1}{d_v-2} \\
&\cdot \sum_{l_0}^{p-1} \left(\binom{\frac{p}{2}-A_0-1}{m_0-1} \binom{\frac{p}{2}-A_1-1}{m_1} + \binom{\frac{p}{2}-A_0-1}{m_0} \binom{\frac{p}{2}-A_1-1}{m_1-1} \right) \\
&\cdot \sum_{l_1=0}^{p-1} \left(\binom{\frac{p}{2}-A_1-1}{m_0-1} \binom{\frac{p}{2}-A_1-1}{m_1} + \binom{\frac{p}{2}-A_0-1}{m_0-1} \binom{\frac{p}{2}-A_1-1}{m_0} \right).
\end{aligned}$$

Simplifying the expression, the result holds. □

B.2 Proof of Theorem 3.4

Proof. The expressions in (3,4) are approximately equal to

$$\binom{\frac{p}{2}-A_0-1}{m_0}\binom{\frac{p}{2}-A_1-1}{m_1}\binom{\frac{p}{2}-A_0-1}{m_1}\binom{\frac{p}{2}-A_1-1}{m_0}\cdot$$
$$\left(\frac{{m_0}^2+{m_1}^2}{(\frac{p}{2}-A_0)(\frac{p}{2}-A_1)}+\frac{m_0m_1}{(\frac{p}{2}-A_0)^2}+\frac{m_0m_1}{(\frac{p}{2}-A_1)^2}\right).$$

Applying the same approximations as in the proof of Theorem 3.2, we rewrite expressions (2,3,4) as

$$\begin{aligned}
&\frac{p-1}{2}p^2\binom{p}{m_0}^2\binom{p}{m_1}^2\binom{p}{d_v-2}^2\\
&\sum_{A_0}^{\frac{p}{2}}\sum_{A_1=0}^{\frac{p}{2}}\left(\frac{A_0}{p}\frac{A_1}{p}\right)^{d_v-2}\left(\frac{1}{2}-\frac{A_0}{p}\right)^m\left(\frac{1}{2}-\frac{A_1}{p}\right)^m\\
&\left(\frac{{m_0}^2+{m_1}^2}{(\frac{p}{2}-A_0)(\frac{p}{2}-A_1)}+\frac{m_0m_1}{(\frac{p}{2}-A_0)^2}+\frac{m_0m_1}{(\frac{p}{2}-A_1)^2}\right)\\
\approx&\frac{p-1}{2}p^2\binom{p}{m_0}^2\binom{p}{m_1}^2\binom{p}{d_v}^2\frac{{d_v}^2(d_v-1)^2}{(p-d_v+2)^2(p-d_v+1)^2}\\
&p^2\int_{x=0}^{\frac{1}{2}}\int_{y=0}^{\frac{1}{2}}(xy)^{d_v-2}\left(\frac{1}{2}-x\right)^m\left(\frac{1}{2}-y\right)^m\\
&\left(\frac{{m_0}^2+{m_1}^2}{(\frac{1}{2}-x)(\frac{1}{2}-y)}+\frac{m_0m_1}{(\frac{1}{2}-x)^2}+\frac{m_0m_1}{(\frac{1}{2}-y)^2}\right)\mathrm{d}x\mathrm{d}y.
\end{aligned}$$

Dividing by $\binom{p}{d_v}^2\binom{p}{m_0}^2\binom{p}{m_1}^2$, the result follows. □

B.3 Proof of Lemma 4.3

Proof. Let $\tilde{a}$ and $\tilde{b}$ be the polynomials obtained by lifting a and b, respectively, from $\mathbb{F}_2[x]/\langle x^p+1\rangle$ to $\mathbb{Z}[x]/\langle x^p-1\rangle$. Since $c\equiv\tilde{c} \mod 2$, we have

$$\mathrm{wt}(c)=\mathrm{wt}(\tilde{c})-|\{j\in\{0,\cdots,p-1\}\ \text{ s.t. }\ \tilde{c}_j\geq 2\}|.$$

Since $\mathrm{wt}(c)=\omega_a\omega_b$, by hypothesis, and $\mathrm{wt}(\tilde{c})\leq\omega_a\omega_b$, this means that

$$|\{j\in\{0,\cdots,p-1\}\ \text{ s.t. }\ \tilde{c}_j\geq 2\}|=0.$$

Then, the coefficients of $\tilde{c}$ take values in $\{0,1\}\subset\mathbb{Z}$, and exactly $\omega_a\omega_b$ out of them are equal to 1.

We now define $N(c'_i)$ analogously to $N(c_i)$, i.e., $N(c'_i)$ contains all indices ℓ such that $\ell\in\mathrm{S}(a')$ and $\ell-i \mod p\in\mathrm{S}(b')$; clearly $N(c_i)\subseteq N(c'_i)$. Then, each c'_i is obtained as the sum of exactly c'_i products in the form $a_\ell b_u$, for proper indices ℓ and u. We can then consider the i-th coefficient of c' as the sum of c'_i

values which, because of the condition on the maximum weight, are either all null, or all null expect one.

With a combinatorial perspective, we can describe the computation of c_i' through an urn experiment, containing M balls of two colors. Balls of the first colors correspond to coefficients $a_\ell b_u = 0$, while balls of the second color correspond to coefficients $a_\ell b_u = 1$. For the i-th coefficient in c', we randomly extract c_i' balls, with the restriction that either zero on one ball of the second color can be selected. The number of balls in the urn, which we denote with M, equals the sum of coefficients in c', i.e., the cardinality of $\bigcup_{i=0}^{p-1} N(c_i')$; since all such sets are disjoint, we have $M = \sum_{i=0}^{p-1} c_i' = |J_a| \cdot |J_b|$. We have $M - \omega$ balls of the first color and $\omega = \mathrm{wt}(c) = \omega_a \omega_b$ of the second color. For the i-th coefficient in c', we extract c_i' balls: if all of them are of the first color, then $c_i' = 0$, otherwise $c_i' = 1$. The number of favourable events for having $c_i' = 0$ is then obtained as $\binom{M-\omega}{c_i'}$, while the number of possible extractions, taking into account that a maximum of one ball of the second color can be selected, is equal to $\binom{M-\omega}{c_i'} + \omega\binom{M-\omega}{c_i'-1}$. Dividing these these two quantities and simple computations yield the formula in the statement of the Lemma. □

B.4 Counting *close* pairs of bits

The clustering assumption holds when there are at least four pairs of bits of H that are located within a stretch of 213 consecutive indices. To capture the probability of this event, we consider the 11 nonzero bits being sampled one at a time. Let $N_{x,y}$ capture the probability that after x bits have been sampled, y *close* pairs of bits are formed.

Suppose the first bit is sampled. The probability that this forms zero pairs of bits is 1. Then $N_{1,0} = 1$.

The second bit is sampled. The probability that this forms one pair of bits within a span of 213 is $\frac{414}{p}$ and the probability that this second bit does not form a pair is $1 - \frac{414}{p}$. Then $N_{2,1} = \frac{414}{p}$, $N_{2,0} = 1 - \frac{414}{p}$, respectively.

The third bit is sampled. The probability that this forms two pairs of bits is 0. The probability that this forms one pair of bits is $2\frac{414}{p}$ and the probability that this third bit forms zero pairs of bits is $1 - 2\frac{414}{p}$. Then $N_{3,1} = 0.0334275, N_{3,0} = 0.9665725$, respectively.

Continuing in this way, we find that $N_{11,4} = 0.000232026 \approx 2^{-12}$ and $N_{11,5} = 0.000001655 \approx 2^{-19}$. A complete table of values is given below.

C Constructing information sets

For the sake of completeness, in this Appendix we provide a procedural description of the methodology that, in Section 4, is used to define information sets for a chosen families of weak keys. Such a procedure is depicted, in Algorithm 1.

x	$y = 0$	$y = 1$	$y = 2$	$y = 3$	$y = 4$	$y = 5$
1	1	-	-	-	-	-
2	0.98877349	0.01122651	-	-	-	-
3	0.96657254	0.03342746	-	-	-	-
4	0.93401883	0.06560589	0.00037527	-	-	-
5	0.89207575	0.10607593	0.00184832	-	-	-
6	0.84200126	0.15257783	0.00540016	0.00002075	-	-
7	0.78528485	0.20244257	0.01213058	0.00014200	-	-
8	0.72357279	0.25279101	0.02308564	0.00054896	0.00000159	-
9	0.65858722	0.30074882	0.03907672	0.00157332	0.00001392	-
10	0.59204449	0.34365703	0.06051776	0.00371380	0.00006675	0.00000016
11	0.52557856	0.37925841	0.08730590	0.00762345	0.00023203	0.00000166

Table 4. Probabilities of x bits forming y *close* pairs

Algorithm 1 Constructing information sets for LEDAcrypt weak keys with $n_0 = 2$

Input: sets $J_{H_0}, J_{H_1}, J_{Q_{0,0}}, J_{Q_{0,1}}, J_{Q_{1,0}}, J_{Q_{1,1}} \subseteq \{0, 1, \cdots, p-1\}$, such that J_{H_i} has cardinality B_{H_i} and $J_{Q_{i,j}}$ have respective cardinalities B_{H_i} and $B_{Q_{i,j}}$, for $i = 0, 1$ and $j = 0, 1$.

Output: sets $T_0, T_1 \subseteq \{0, \cdots, p-1\}$, with cardinality $\frac{p}{2}$.

```
1: for $i \leftarrow 0$ to 1 do
2:     $H'_i \leftarrow$ polynomial with support $J_{H_i}$ and coefficients over $\{0,1\} \subseteq \mathbb{Z}$
3:     for $j \leftarrow 0$ to 1 do
4:         $Q'_{i,j} \leftarrow$ polynomial with support $J_{Q_{i,j}}$ and coefficients over $\{0,1\} \subseteq \mathbb{Z}$
5:     end for
6: end for
7: for $i \leftarrow 0$ to 1 do
8:     for $j \leftarrow 0$ to 1 do
9:         $L'_{i,j} \leftarrow H'_i Q'_{i,j} \in \mathbb{Z}[x]/\langle x^p - 1\rangle$
10:    end for
11: end for
12: for $i \leftarrow 0$ to 1 do
13:     $L'_i = L'_{0,i} + L'_{1,i}$
14:     $\pi_i \leftarrow$ permutation such that $\pi_i(L'_i)$ has non decreasing coefficients
15:     $T_i \leftarrow$ first $\frac{p}{2}$ coefficients of $T_i$
16: end for
17: return $\{T_0, T_1\}$
```