On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL

Martin R. Albrecht[⋆]

Information Security Group
Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
`martin.albrecht@royalholloway.ac.uk`

Abstract. We present novel variants of the dual-lattice attack against LWE in the presence of an unusually short secret. These variants are informed by recent progress in BKW-style algorithms for solving LWE. Applying them to parameter sets suggested by the homomorphic encryption libraries HElib and SEAL yields revised security estimates. Our techniques scale the exponent of the dual-lattice attack by a factor of $(2\,L)/(2\,L{+}1)$ when $\log q = \Theta(L \log n)$, when the secret has constant hamming weight h and where L is the maximum depth of supported circuits. They also allow to half the dimension of the lattice under consideration at a multiplicative cost of 2^h operations. Moreover, our techniques yield revised concrete security estimates. For example, both libraries promise 80 bits of security for LWE instances with $n = 1024$ and $\log_2 q \approx 47$, while the techniques described in this work lead to estimated costs of 68 bits (SEAL) and 62 bits (HElib).

1 Introduction

Learning with Errors (LWE), defined in Definition 1, has proven to be a rich source of cryptographic constructions, from public-key encryption and Diffie-Hellman-style key exchange (cf. [Reg09,Pei09,LPR10,DXL12,BCNS15,ADPS16,BCD+16]) to fully homomorphic encryption (cf. [BV11,BGV12,Bra12,FV12,GSW13,CS15]).

Definition 1 (LWE [Reg09]). *Let n, q be positive integers, χ be a probability distribution on $\mathbb{Z}$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s},\chi,q}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}$ according to χ and considering it in $\mathbb{Z}_q$, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

[⋆] This research was supported by EPSRC grants EP/L018543/1 "Multilinear Maps in Cryptography" and EP/P009417/1 "Bit Security of Learning with Errors for Post-Quantum Cryptography and Fully Homomorphic Encryption".

Decision-LWE *is the problem of deciding whether pairs* $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *are sampled according to* $L_{\mathbf{s},\chi,q}$ *or the uniform distribution on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$.
Search-LWE *is the problem of recovering* $\mathbf{s}$ *from* $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *sampled according to* $L_{\mathbf{s},\chi,q}$.

We may write LWE instances in matrix form $(\mathbf{A}, \mathbf{c})$, where rows correspond to samples $(\mathbf{a}_i, c_i)$. In many instantiations, χ is a discrete Gaussian distribution with standard deviation $\alpha\, q/\sqrt{2\pi}$. Though, in this work, like in many works on cryptanalysis of LWE, the details of the error distribution do not matter as long as we can bound the size of the error under additions.

The bit-security of concrete LWE instances is a prominent area of current cryptographic research, in particular in light of standardisation initiatives for LWE-based schemes and LWE-based (somewhat) homomorphic encryption being proposed for applications such as computation with medical data [KL15]. See [APS15] for a relatively recent survey of known (classical) attacks.

Applications such as [KL15] are enabled by progress in homomorphic encryption in recent years. The two most well-known homomorphic encryption libraries are HElib and SEAL. HElib [GHS12a,HS14] implements BGV [BGV12]. SEAL v2.0 [LP16] implements FV [Bra12,FV12]. Both schemes fundamentally rely on the security of LWE.

However, results on the expected cost of solving generic LWE instances do not directly translate to LWE instances as used in fully homomorphic encryption (FHE). Firstly, because these instances are typically related to the Ring-LWE assumption [LPR10,LPR13] instead of plain LWE. Secondly, because these instances are typically *small-secret* instances. In particular, they typically sample the secret $\mathbf{s}$ from some distribution $\mathcal{B}$ as defined below. We call such instances $\mathcal{B}$-secret LWE instances.

Definition 2. *Let* n, q *be positive integers. We call*

$\mathcal{B}$ *any distribution on* $\mathbb{Z}_q^n$ *where each component* ≤ 1 *in absolute value, i.e.* $\|\mathbf{s}_{(i)}\| \leq 1$ *for* $\mathbf{s} \leftarrow_{\$} \mathcal{B}$.
$\mathcal{B}^+$ *the distribution on* $\mathbb{Z}_q^n$ *where each component is independently sampled uniformly at random from* $\{0, 1\}$.
$\mathcal{B}^-$ *the distribution on* $\mathbb{Z}_q^n$ *where each component is independently sampled uniformly at random from* $\{-1, 0, 1\}$.
$\mathcal{B}_h^+$ *the distribution on* $\mathbb{Z}_q^n$ *where components are sampled independently uniformly at random from* $\{0, 1\}$ *with the additional guarantee that at most* h *components are non-zero.*
$\mathcal{B}_h^-$ *the distribution on* $\mathbb{Z}_q^n$ *where components are sampled independently uniformly at random from* $\{-1, 0, 1\}$ *with the additional guarantee that at most* h *components are non-zero.*

Remark 1. In [BLP+13], instances with $\mathbf{s} \leftarrow_\$ \mathcal{B}^+$ are referred to as binary-secret; $\mathcal{B}^+$ is used in [FV12]; $\mathcal{B}^-$ is used in Microsoft's SEAL v2.0 library[1] and [LN14]; $\mathcal{B}^-_{64}$ is the default choice in HElib, cf. [GHS12b, Appendix C.1.1] and [HS14].

It is an open question how much easier, if any, $\mathcal{B}$-secret LWE instances are compared to regular LWE instances. On the one hand, designers of FHE schemes typically ignore this issue [GHS12a,LN14,CS16]. This could be considered as somewhat justified by a reduction from [ACPS09] showing that an LWE instance with an arbitrary secret can be transformed into an instance with a secret following the noise distribution in polynomial time and at the loss of n samples. Hence, such instances are not easier than instances with a uniformly random secret, assuming sufficiently many samples are available. As a consequence, LWE with a secret following the noise distribution is considered to be in *normal form*. Given that the noise in homomorphic encryption libraries is also typically rather small — SEAL and HElib use standard deviation $\sigma \approx 3.2$ — the distribution $\mathcal{B}^-$ gives rise to LWE instances which could be considered relatively close to normal-form LWE instances. However, considering the actual distributions, not just the standard deviations, it is known that LWE with *error distribution* $\mathcal{B}$ is insecure once sufficiently many samples are available [AG11,ACFP14,KF15].

On the other hand, the best, known reduction from regular LWE to $\mathcal{B}^+$-secret LWE has an expansion factor of $\log q$ in the dimension. That is, [BLP+13] gives a reduction from regular LWE in dimension n to LWE with $\mathbf{s} \leftarrow_\$ \mathcal{B}^+$ in dimension $n \log q$.

In contrast, even for noise with width $\approx \sqrt{n}$ and $\mathbf{s} \leftarrow_\$ \mathcal{B}^-$ the best known lattice attacks suggest an expansion factor of at most $\log \log n$ [BG14], if at all. Overall, known algorithms do not perform significantly better for $\mathcal{B}$-secret LWE instances, perhaps reinforcing our confidence in the common approach of simply ignoring the special form of the secret.

One family of algorithms has recently seen considerable progress with regards to $\mathcal{B}$-secret instances: combinatorial algorithms. Already in [Reg09] it was observed that the BKW algorithm, originally proposed for LPN by Blum, Kalai and Wasserman [BKW00], leads to an algorithm in $2^{\Theta(n)}$ time and space for solving LWE. The algorithm proceeds by splitting the components of the vectors $\mathbf{a}_i$ into blocks of k components. Then, it searches for collisions in the first block in an "elimination table" holding entries for (possibly) all q^k different values for that block. This table is constructed by sampling fresh $(\mathbf{a}_i, c_i)$ pairs from the LWE oracle. By subtracting vectors with colliding components in the first block, a vector of dimension $n - k$ is recovered, applying the same subtraction to the corresponding c_i values, produces an error of size $\sqrt{2}\alpha\, q$. Repeating the process for consecutive blocks reduces the dimension further at the cost of an increase

[1] cf. `KeyGenerator::set_poly_coeffs_zero_one_negone()` at `https://sealcrypto.codeplex.com/SourceControl/latest#SEAL/keygenerator.h`

in the noise by a factor $\sqrt{2}$ at each level. This process either continues until all components of $\mathbf{a}_i$ are eliminated or when there are so few components left that exhaustive search can solve the remaining low-dimensional LWE instance.

A first detailed study of this algorithm when applied to LWE was provided in [ACF+15]. Subsequently, improved variants were proposed, for small secret LWE instances via "lazy modulus switching" [AFFP14], via the application of an FFT in the last step of the algorithm [DTV15], via varying the block size k [KF15] and via rephrasing the problem as the coding-theoretic problem of quantisation [GJS15]. In particular, the works [KF15,GJS15] improve the exploitation of a small secret to the point where these techniques improve the cost of solving instances where the secret is as big as the error, i.e. arbitrary LWE instances. Yet, combinatorial algorithms do not perform well on FHE-style LWE instances because of their large dimension n to accommodate the large modulus q.

1.1 Our contribution/outline

We first review parameter choices in HElib and SEAL as well as known algorithms for solving LWE and related problems in Section 2.

Then, we reconsider the dual-lattice attack (or "dual attack" in short) which finds short vectors $\mathbf{y}$ such that $\mathbf{y} \cdot \mathbf{A} \equiv 0 \bmod q$ using lattice reduction. In particular, we recast this attack as the lattice-reduction analogue of the BKW algorithm and adapt techniques and lessons learned from BKW-style algorithms. Applying these techniques to parameter sets suggested for HElib and SEAL, we arrive at revised concrete and asymptotic security estimates.

First, in Section 3, we recall (the first stage of) BKW as a recursive dimension reduction algorithm for LWE instances. Each step transforms an LWE instance in dimension n to an instance in dimension $n - k$ at the cost of an increase in the noise by a factor of $\sqrt{2}$. This smaller instance is then reduced further by applying BKW again or solved using another algorithm for solving LWE; typically some form of exhaustive search once the dimension is small enough. To achieve this dimension reduction, BKW first produces elimination tables and then makes use of these tables to sample possibly many LWE samples in dimension $n-k$ relatively cheaply. We translate this approach to lattice reduction in the low advantage regime: we perform one expensive lattice reduction step followed by many relatively cheap lattice reductions on rerandomised bases. This essentially reduces the overall solving cost by a factor of m, where m is the number of samples required to distinguish a discrete Gaussian distribution with large standard deviation from uniform modulo q. We note that this approach applies to any LWE instance, i.e. does not rely on an unusually short secret and thus gives cause for a moderate revision of many LWE estimates based on the dual-attack in the low advantage regime. It does, however, rely on the heuristic that these cheap lattice reduction steps produce sufficiently short and random vectors. We give evidence that this heuristic holds.

Second, in Section 4, we observe that the normal form of the dual attack — finding short vectors $\mathbf{y}$ such that $\mathbf{y} \cdot \mathbf{A} \equiv \mathbf{x} \bmod q$ is short — is a natural analogue of "lazy modulus switching" [AFFP14]. Then, to exploit the unusually small secret, we apply lattice scaling as in [BG14]. The scaling factor is somewhat analogous to picking the target modulus in modulus switching resp. picking the (dimension of the) code for quantisation. This technique applies to any $\mathcal{B}$-secret LWE instance. For $\mathcal{B}_h^-$-secret instances, it reduces the cost of the dual attack by a factor of $2\,L/(2\,L+1)$ in the exponent when $\log q = \Theta\,(L \log n)$ for L the supported depth of FHE circuits and when h is a constant.

Third, in Section 5, we focus on $\mathbf{s} \leftarrow_{\$} \mathcal{B}_h^{\pm}$ and adapt the dual attack to find short vectors which produce zero when multiplied with a *subset* of the columns of $\mathbf{A}$. This, as in BKW, produces a smaller, easier LWE instance which is then solved using another algorithm. In BKW, these smaller instances typically have very small dimension (say, 10). Here, we consider instances with dimension of several hundreds. This is enabled by exploiting the sparsity of the secret and by relaxing the conditions on the second step: we recover a solution only with a small probability of success. The basic form of this attack does not rely on the size of the non-zero components (only on the sparsity) and reduces the cost of solving an instance in dimension n to the cost of solving an instance in dimension $n/2$ multiplied by 2^h where h is the hamming weight of the secret (other trade-offs between multiplicative cost increase and dimension reduction are possible and typically optimal). We also give an improved variant when the non-zero components are also small.

In Section 6, we put everything together to arrive at our final algorithm SILKE, which combines the techniques outlined above; inheriting their properties. We also give revised security estimates for parameter sets suggested for HElib and SEAL in Table 1. Table 1 highlights that the techniques described in this work can, despite being relatively simple, produce significantly revised concrete security estimates for both SEAL and HElib.

2 Preliminaries

Logarithms are base 2 if not stated otherwise. We write vectors in bold, e.g. $\mathbf{a}$, and matrices in upper-case bold, e.g. $\mathbf{A}$. By $\mathbf{a}_{(i)}$ we denote the i-th component of $\mathbf{a}$, i.e. a scalar. In contrast, $\mathbf{a}_i$ is the i-th element of a list of vectors. We write $\mathbf{I}_m$ for the $m \times m$ identity matrix over whichever base ring is implied from context. We write $\mathbf{0}_{m\times n}$ for the $m \times n$ zero matrix. A lattice is a discrete subgroup of $\mathbb{R}^n$. It can be represented by a basis $\mathbf{B}$. We write $\Lambda(\mathbf{B})$ for the lattice generated by the rows of the matrix $\mathbf{B}$, i.e. all integer-linear combinations of the rows of $\mathbf{B}$.

n	1024	2048	4096	8192	16384
			SEAL 80-bit		
$\log_2 q$	47.5	95.4	192.0	392.1	799.6
dual	83.1	78.2	73.7	71.1	70.6
$\text{SILKE}_{\text{small}}$	68.1	69.0	68.2	68.4	68.8
			HElib 80-bit		
$\log_2 q$	47.0	87.0	167.0	326.0	638.0
dual	85.2	85.2	85.3	84.6	85.5
$\text{SILKE}_{\text{sparse}}$	61.3	65.0	67.9	70.2	73.1
			HElib 128-bit		
$\log_2 q$	38.0	70.0	134.0	261.0	511.0
dual	110.7	110.1	109.3	108.8	108.9
$\text{SILKE}_{\text{sparse}}$	73.2	77.4	81.2	84.0	86.4

Table 1. Costs of dual attacks on HElib and SEAL. Rows "$\log_2 q$" give bit sizes for the maximal modulus for a given n, for SEAL it is taken from [LN14], for HElib it is chosen such that the expected cost is 2^{80} resp. 2^{128} seconds according to [GHS12a]. The rows "dual" give the log cost (in operations) of the dual attack according to our lattice-reduction estimates without taking any of our improvements into account; The row "$\text{SILKE}_{\text{small}}$" gives the log cost of Algorithm 3 with "sparse" set to false; The rows "$\text{SILKE}_{\text{sparse}}$" give the log cost of Algorithm 3 with "sparse" set to true. The "sparse" flag toggles whether the approach described in Section 5 is enabled or not in Algorithm 3.

Strategy	Dual	Decode	Embed
HElib	188.9	—	—
base line	124.2	116.6	114.5
Section 4	101.0	—	—
Section 5	97.1	111.0	110.9
Section 6	83.9	—	—

Table 2. Logarithms of algorithm costs in operations mod q when applied to example parameters $n = 2048$, $q \approx 2^{63.4}$, $\alpha \approx 2^{-60.4}$ and $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}_{64}$. The row "base line" gives the log cost of attacks according to our lattice-reduction estimates without taking any of our improvements into account.

We write $\Lambda_q(\mathbf{B})$ for the q-ary lattice generated by the rows of the matrix $\mathbf{B}$ over $\mathbb{Z}_q$, i.e. the lattice spanned by the rows $\mathbf{B}$ and multiples of q. We write $\mathbf{A}_{n:m}$ for the rows $n, \ldots, m-1$ of $\mathbf{A}$. If the starting or end point is omitted it is assumed to be 0 or the number of rows respectively, i.e. we follow Python's slice notation.

2.1 Rolling example

Throughout, we are going to use Example 1 below to illustrate the behaviour of the techniques described here. See Table 2 for an overview of complexity estimates for solving this set of parameters using the techniques described in this work.

Example 1. The LWE dimension is $n = 2048$, the modulus is $q \approx 2^{63.4}$, the noise parameter is $\alpha \approx 2^{-60.4}$, i.e. we have a standard deviation of $\sigma \approx 3.2$. We have $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}_{64}$, i.e. only $h = 64$ components of the secret are ± 1, all other components are zero. This set of parameters is inspired by parameter choices in HElib and produced by calling the function `fhe_params(n=2048,L=2)` of the LWE estimator from [APS15].

2.2 Parameter choices in HElib

HElib [GHS12a,HS14] uses the cost of the dual attack for solving LWE to establish parameters. The dual strategy reduces the problem of distinguishing LWE from uniform to the SIS problem [Ajt96]:

Definition 3 (SIS). *Given $q \in \mathbb{Z}$, a matrix $\mathbf{A}$, and $t < q$; find $\mathbf{y}$ with $0 < \|\mathbf{y}\| \leq t$ and*

$$\mathbf{y} \cdot \mathbf{A} \equiv \mathbf{0} \pmod{q}.$$

Now, given samples $\mathbf{A}, \mathbf{c}$ where either $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ or $\mathbf{c}$ uniform, we can distinguish the two cases by finding a short $\mathbf{y}$ which solves SIS on $\mathbf{A}$ and by computing $\langle \mathbf{y}, \mathbf{c} \rangle$. On the one hand, if $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, then $\langle \mathbf{y}, \mathbf{c} \rangle = \langle \mathbf{y} \cdot \mathbf{A}, \mathbf{s} \rangle + \langle \mathbf{y}, \mathbf{e} \rangle \equiv \langle \mathbf{y}, \mathbf{e} \rangle \pmod{q}$. If $\mathbf{y}$ is short then $\langle \mathbf{y}, \mathbf{e} \rangle$ is also short. On the other hand, if $\mathbf{c}$ is uniformly random, so is $\langle \mathbf{y}, \mathbf{c} \rangle$.

To pick a target norm for $\mathbf{y}$, HElib picks $\|\mathbf{y}\| = q$ which allows distinguishing with good probability because q is not too far from q/σ since $\sigma \approx 3.2$ and q is typically rather large. More precisely, we may rely on the following lemma:

Lemma 1 ([LP11]). *Given an LWE instance characterised by n, α, q and a vector $\mathbf{y}$ of length $\|\mathbf{y}\|$ such that $\mathbf{y} \cdot \mathbf{A} \equiv 0 \pmod{q}$, the advantage of distinguishing $\langle \mathbf{y}, \mathbf{e} \rangle$ from random is close to*

$$\exp(-\pi(\|\mathbf{y}\| \cdot \alpha)^2).$$

To produce a short enough $\mathbf{y}$, we may call a lattice-reduction algorithm. In particular, we may call the BKZ algorithm with block size β. After performing BKZ-β reduction the first vector in the transformed lattice basis will have norm $\delta_0^m \cdot \det(\Lambda)^{1/m}$ where $\det(\Lambda)$ is the determinant of the lattice under consideration, m its dimension and the root-Hermite factor δ_0 is a constant based on the block size parameter β. Increasing the parameter β leads to a smaller δ_0 but also leads to an increase in run-time; the run-time grows at least exponential in β (see below).

In our case, the expression above simplifies to $\|\mathbf{y}\| \approx \delta_0^m \cdot q^{n/m}$ whp, where n is the LWE dimension and m is the number of samples we consider. The minimum of this expression is attained at $m = \sqrt{\frac{n \log q}{\log \delta_0}}$ [MR09].

Explicitly, we are given a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, construct a basis $\mathbf{Y}$ for its left kernel modulo q and then consider the q-ary lattice $\Lambda_q(\mathbf{Y})$ spanned by the rows of $\mathbf{Y}$. With high probability $\mathbf{Y}$ is an $(m-n) \times m$ matrix and $\Lambda_q(\mathbf{Y})$ has volume q^n. Let $\mathbf{L}$ be a basis for $\Lambda_q(\mathbf{Y})$, $m' = m - n$ and write $\mathbf{Y} = [\mathbf{I}_{m'} | \mathbf{Y}']$ then we have

$$\mathbf{L} = \begin{pmatrix} \mathbf{I}_{m'} & \mathbf{Y}' \\ 0 & q\,\mathbf{I}_n \end{pmatrix}.$$

In other words, we are attempting to find a short vector $\mathbf{y}$ in the integer row span of $\mathbf{L}$.

Given a target for the norm of $\mathbf{y}$ and hence for δ_0, HElib[2] estimates the cost of lattice reduction by relying on the following formula from [LP11]:

$$\log t_{BKZ}(\delta_0) = \frac{1.8}{\log \delta_0} - 110, \qquad (1)$$

where $t_{BKZ}(\delta_0)$ is the time in seconds it takes to BKZ reduce a basis to achieve root-Hermite factor δ_0. This estimate is based on experiments with BKZ in the NTL library [Sho01] and extrapolation.

2.3 LP model

The [LP11] model for estimating the cost of lattice-reduction is not correct.

Firstly, it expresses runtime in seconds instead of units of computation. As Moore's law progresses and more parallelism is introduced, the number of instructions that can be performed in a second increases. Hence, we first must translate

[2] `https://github.com/shaih/HElib/blob/a5921a08e8b418f154be54f4e39a849e74489319/src/FHEContext.cpp#L22`

Eq. (1) to units of computation. The experiments of Lindner and Peikert were performed on a 2.33 Ghz AMD Opteron machine, so we may assume that about $2.33 \cdot 10^9$ operations can be performed on such a machine in one second and we scale Eq. (1) accordingly.[3]

Secondly, the LP model does not fit the implementation of BKZ in NTL. The BKZ algorithm internally calls an oracle for solving the shortest vector problem in smaller dimension. The most practically relevant algorithms for realising this oracle are enumeration without preprocessing (Fincke-Pohst) which costs $2^{\Theta(\beta^2)}$ operations, enumeration with recursive preprocessing (Kannan) which costs $\beta^{\Theta(\beta)}$ and sieving which costs $2^{\Theta(\beta)}$. NTL implements enumeration without preprocessing. That is, while it was shown in [Wal15] that BKZ with recursive BKZ pre-processing achieves a run-time of $\mathrm{poly}(n) \cdot \beta^{\Theta(\beta)}$, NTL does not implement the necessary recursive preprocessing with BKZ in smaller dimensions. Hence, it runs in time $\mathrm{poly}(n) \cdot 2^{\Theta(\beta^2)}$ for block size β.

Thirdly, the LP model assumes a linear relation between $1/\log(\delta_0)$ and the log of the running time of BKZ, but from the "lattice rule-of-thumb" ($\delta_0 \approx \beta^{1/(2\beta)}$) and $2^{\Theta(\beta)}$ being the complexity of the best known algorithm for solving the shortest vector problem, we get:

Lemma 2 ([APS15]). *The log of the time complexity achieve a root-Hermite factor δ_0 with BKZ is*

$$\Theta\left(\frac{\log(1/\log\delta_0)}{\log\delta_0}\right)$$

if calling the SVP oracle costs $2^{\Theta(\beta)}$.

To illustrate the difference between Lemma 2 and Eq. (1), consider Regev's original parameters [Reg05] for LWE: $q \approx n^2$, $\alpha q \approx \sqrt{n}$. Then, solving LWE with the dual attack and advantage ϵ requires a log root-Hermite factor $\log\delta_0 = \log^2\left(\alpha\sqrt{\ln(1/\varepsilon)/\pi}^{-1}\right)/(4n\log q)$ [APS15]. Picking ε such that $\log\sqrt{\ln(1/\varepsilon)/\pi} \approx 1$, the log root-Hermite factor becomes $\log\delta_0 = \frac{9\log n}{32\,n}$. Plugging this result into Eq. 1, we would estimate that solving LWE for these parameters takes $\log t_{BKZ}(\delta_0) = \frac{32\,n}{5\log n} - 110$ seconds, which is subexponential in n.

2.4 Parameter choices in SEAL

SEAL v2.0 [LP16] largely leaves parameter choices to the user. However, it provides the `ChooserEvaluator::default_parameter_options()` function which returns

[3] The number of operations on integers of size $\log q$ depends on q and is not constant. However, constant scaling provides a reasonable approximation for the number of operations for the parameter ranges we are interested in here.

values from [LN14, Table 2]. This table gives a maximum $\log q$ for 80 bits of security for $n = 1024, 2048, 4096, 8192, 16384$. We reproduce these values for $\log q$ in Table 1. The default standard deviation is $\sigma = 3.19$.

The values of [LN14, Table 2] are based on enumeration costs and the simulator from [CN11,CN12]. Furthermore, to extrapolate from available enumeration costs from [CN12], [LN14] assumes calling the SVP oracle in BKZ grows only exponentially with β, i.e. as $2^{0.64\beta-28}$. Note that this is overly optimistic, as [CN12] calls enumeration with recursive preprocessing to realise the SVP oracle inside BKZ, which has a complexity of $\beta^{\Theta(\beta)}$.

Finally, we note that the SEAL v2.0 manual [LP16] cautions the user against relying on the security provided by the list of default parameters.

2.5 Lattice reduction

We will estimate the cost of lattice reduction using the following assumptions: BKZ-β produces vectors with $\delta_0 \approx \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}}$ [Che13]. The SVP oracle in BKZ is realised using sieving and sieving in blocksize β costs $t_\beta = 2^{0.292\,\beta+12.31}$ clock cycles. Here, $0.292\,\beta$ follows from [BDGL16], the additive constant $+12.31$ is based on experiments in [Laa15]. BKZ-β costs $c\,n \cdot t_\beta$ clock cycles in dimension n for some small constant c based on experiments in [Che13]; cf. [Che13, Figure 4.6]. This corresponds roughly to $2\,c$ tours of BKZ. We pick $c = 8$ based on our experiments with [FPL16].

This estimate is more optimistic than the estimate in [APS15], which does not yet take [BDGL16] into account and bases the number of SVP oracle calls on theoretical convergence results [HPS11] instead of experimental evidence. On the other hand, this estimate is more pessimistic than [BCD+16] which assumes *one* SVP call to be sufficient in order to protect against future algorithmic developments. While such developments, amortising costs across SVP calls during one BKZ reduction, are plausible, we avoid this assumption here in order not to "oversell" our results. However, we note that our improvements are somewhat oblivious to the underlying lattice-reduction model used. That is, while the concrete estimates for bit-security will vary depending on which estimate is employed, the techniques described here lead to improvements over the plain dual attack regardless of model. For completeness, we give estimated costs in different cost models in Appendix C.

According to the [LP11] estimate, solving Example 1 costs $2^{157.8}$ seconds or $2^{188.9}$ operations using the standard dual attack. The estimates outlined in this section predict a cost of $2^{124.2}$ operations for the same standard dual attack.

2.6 Related work

LWE. Besides the dual attack, via BKW or lattice-reduction, there is also the primal attack, which solves the bounded distance decoding (BDD) problem directly. That is, given $(\mathbf{A}, \mathbf{c})$ with $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ or $\mathbf{c} \leftarrow_{\$} \mathcal{U}\left(\mathbb{Z}_q^m\right)$ find $\mathbf{s}'$ such that $|\mathbf{w} - \mathbf{c}|$ with $\mathbf{w} = \mathbf{A} \cdot \mathbf{s}'$ is minimised. For this, we may employ Kannan's embedding [AFG14] or variants of Babai's nearest planes after lattice reduction [LP11,LN13]. For Example 1 the cost of the latter approach is $2^{116.6}$ operations, i.e. about a factor 190 faster than the dual attack.

Arora & Ge proposed an asymptotically efficient algorithm for solving LWE [AG11], which was later improved in [ACFP14]. However, these algorithms involve large constants in the exponent, ruling them out for parameters typically considered in cryptography. We, hence, do not consider them further in this work.

Small-secret LWE. As mentioned in [GHS12b], we can transform instances with an unusually short secret into instances where the secret follows the error distribution, but n samples have the old, short secret as noise [ACPS09].

Given a random $m \times n$ matrix $\mathbf{A}$ mod q and an m-vector $\mathbf{c} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$, let $\mathbf{A}_0$ denotes the first n rows of $\mathbf{A}$, $\mathbf{A}_1$ the next n rows, etc., $\mathbf{e}_0, \mathbf{e}_1, \ldots$ are the corresponding parts of the error vector and $\mathbf{c}_0, \mathbf{c}_1, \ldots$ the corresponding parts of $\mathbf{c}$. We have $\mathbf{c}_0 = \mathbf{A}_0 \cdot \mathbf{s} + \mathbf{e}_0$ or $\mathbf{A}_0^{-1} \cdot \mathbf{c}_0 = \mathbf{s} + \mathbf{A}_0^{-1}\mathbf{e}_0$. For $i > 0$ we have $\mathbf{c}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$, which together with the above gives $\mathbf{A}_i \mathbf{A}_0^{-1}\mathbf{c}_0 - \mathbf{c}_i = \mathbf{A}_i \mathbf{A}_0^{-1}\mathbf{e}_0 - \mathbf{e}_i$. The output of the transformation is $\mathbf{z} = \mathbf{B} \cdot \mathbf{e}_0 + \mathbf{f}$ with $\mathbf{B} = (\mathbf{A}_0^{-1} \mid \mathbf{A}_1 \cdot \mathbf{A}_0^{-1} \mid \ldots)$ and $\mathbf{z} = (\mathbf{A}_0^{-1}\mathbf{c}_0 \mid \mathbf{A}_1\mathbf{A}_0^{-1}\mathbf{c}_1 \mid \ldots)$ and $\mathbf{f} = (\mathbf{s}|\mathbf{e}_1 \mid \ldots)$. For Example 1, this reduces α from $2^{-60.4}$ to $\approx 2^{-60.8}$ and marginally improves the cost of solving.

An explicit variant of this approach is given in [BG14]. Consider the lattice

$$\Lambda = \{\mathbf{v} \in \mathbb{Z}^{n+m} \mid [\mathbf{A} \mid \mathbf{I}_m] \cdot \mathbf{v} \equiv 0 \bmod q\}.$$

It has an unusually short vector $(\mathbf{s}||\mathbf{e})$. When $\|\mathbf{s}\| \ll \|\mathbf{e}\|$, the vector $(\mathbf{s}||\mathbf{e})$ is uneven in length. To balance the two sides, rescale the first part to have the same norm as the second. When $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}$, this scales the volume of the lattice by σ^n. When $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{+}$, this scales the volume of the lattice by $(2\sigma)^n$ because we can scale by 2σ and then re-balance. When $\mathbf{s} \leftarrow_{\$} \mathcal{B}_h^{\pm}$, the volume is scaled depending on h. For our rolling example, this approach costs $2^{114.5}$ operations, i.e. is about a factor 830 faster than the dual attack.

Independently and concurrently to this work, a new key-exchange protocol based on sparse secret LWE was proposed in [CKH+16]. A subset of the techniques discussed here are also discussed in [CKH+16], in particular, ignoring components of the secret and using lattice scaling as in [BG14].

Combinatorial. This work combines combinatorial and lattice-reduction techniques. As such, it has some similarities with the hybrid attack on NTRU [HG07]. This attack was recently adapted to LWE in the $\mathcal{B}$-secret case in [BGPW16] and its complexity revisited in [Wun16].

Rings. Recently, [ABD16] proposed a subfield lattice-attack on the two fully homomorphic encryption schemes YASHE [BLLN13] and LTV [LTV12], showing that NTRU with "overstretched" moduli q is less secure than initially expected. Quickly after, [KF16] pointed out that the presence of subfields is not necessary for attacks to succeed. NTRU can be considered as the homogeneous version of Ring-LWE, but there is currently no indication that these attacks can be translated to the Ring-LWE setting. There is currently no known algorithm which solves Ring-LWE faster than LWE for the parameter choices (ring, error distribution, etc.) typically considered in FHE schemes.

3 Amortising costs

If the cost of distinguishing LWE from random with probability ε is c, the cost of solving is customary estimated as at least c/ε [LP11]. More precisely, applying Chernoff bounds, we require about $1/\varepsilon^2$ samples to amplify a decision experiment succeeding with advantage ε to a constant advantage. Hence, e.g. in [APS15], the dual attack is costed as the cost of running BKZ-β to achieve the target δ_0 multiplied by the number of samples required to distinguish with the target advantage, i.e. $\approx c/\varepsilon^2$.

In the case of the dual attack, this cost can be reduced by performing rerandomisation on the already reduced basis. If $\mathbf{L}$ is a basis for the lattice $\Lambda_q(\mathbf{Y})$, we first compute $\mathbf{L}'$ as the output of BKZ-β reduction where β is chosen to achieve the target δ_0 required for some given target advantage. Then, in order to produce sufficiently many relatively short vectors $\mathbf{y}_i \in \Lambda_q(\mathbf{Y})$ we repeatedly multiply $\mathbf{L}'$ by a fresh random sparse unimodular matrix with small entries to produce $\mathbf{L}'_i$. As a consequence, $\mathbf{L}'_i$ remains somewhat short. Finally, we run BKZ-β' with $\beta' \leq \beta$ on $\mathbf{L}'_i$ and return the smallest non-zero vector as $\mathbf{y}_i$. See Algorithm 1, where ε_d is chosen following Lemma 1 (see below for the expectation of $\|\mathbf{y}\|$) and m is chosen following [SL12].

That is, similar to BKW, which in a first step produces elimination tables which allow sampling smaller dimensional LWE samples in $\mathcal{O}\left(n^2\right)$ operations, we first produce a relatively good basis $\mathbf{L}'$ to allow sampling $\mathbf{y}_i$ relatively efficiently.

To produce the estimates in Table 1, we assume the same rerandomisation strategy as is employed in fplll's implementation [FPL16] of extreme pruning for BKZ 2.0.[4] This rerandomisation strategy first permutes rows and then adds three existing rows together using ± 1 coefficients, which would increase norms by a factor of $\sqrt{3} < 2$ when all vectors initially have roughly the same norm. For completeness, we reproduce the algorithm in Appendix A. We then run LLL, i.e. we set $\beta' = 2$, and assume that our $\mathbf{y}_i$ have their norms increased by a factor of two, i.e. $\mathrm{E}[\|\mathbf{y}_i\|] = 2 \cdot \delta_0^m q^{n/m}$.

```
Data: candidate LWE samples $\mathbf{A}, \mathbf{c} \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$
Data: BKZ block sizes $\beta, \beta' \geq 2$
Data: target success probability $\varepsilon$
$\varepsilon_d \leftarrow \exp(-\pi(\mathrm{E}[\|\mathbf{y}_i\|] \cdot \alpha)^2)$;
$m \leftarrow \lceil 2 \log(2 - 2\,\varepsilon)/\log(1 - 4\,\varepsilon_d^2) \rceil$;
$\mathbf{L} \leftarrow$ basis for $\{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} \cdot \mathbf{A} \equiv 0 \bmod q\}$;
$\mathbf{L}' \leftarrow$ BKZ-$\beta$ reduced basis for $\mathbf{L}$;
for $i \leftarrow 0$ to $m - 1$ do
    $\mathbf{U} \leftarrow_{\$}$ a sparse unimodular matrix with small entries;
    $\mathbf{L}_i \leftarrow \mathbf{U} \cdot \mathbf{L}'$;
    $\mathbf{L}'_i \leftarrow$ BKZ-$\beta'$ reduced basis for $\mathbf{L}_i$;
    $\mathbf{y}_i \leftarrow$ shortest row vector in $\mathbf{L}'_i$;
    $e'_i \leftarrow \langle \mathbf{y}_i, \mathbf{c} \rangle$;
end
if $e'_i$ follow discrete Gaussian distribution then
    return $\top$;
else
    return $\bot$;
end
```
Algorithm 1: SILKE₁: Amortising costs in BKW-style SIS strategy for solving LWE

Heuristic. We note that, in implementing this strategy, we are losing statistical independence. To maintain statistical independence, we would consider fresh LWE samples and distinguish $\langle \mathbf{y}_i, \mathbf{e}_i \rangle$ from uniform. However, neither HElib nor SEAL provides the attacker with sufficiently many samples to run the algorithm under these conditions. Instead, we are attempting to distinguish $\langle \mathbf{y}_i, \mathbf{e} \rangle$ from uniform. Furthermore, since we are performing only light rerandomisation our distribution could be skewed if our $\mathbf{y}_i$ in $\langle \mathbf{y}_i, \mathbf{e} \rangle$ are not sufficiently random. Just as in BKW-style algorithms [ACF+15] we assume the values $\langle \mathbf{y}_i, \mathbf{e} \rangle$ are distributed closely enough to the target distribution to allow us to ignore this issue.

Experimental verification. We tested the heuristic assumption of Algorithm 1 by rerandomising a BKZ-60 reduced basis using Algorithm 4 with $d = 3$ followed

[4] `https://github.com/fplll/fplll/blob/b75fe83/fplll/bkz.cpp#L43`

by LLL reduction several hundred times. In this experiment, we recovered fresh somewhat short vectors in each call, where somewhat short means with a norm at most twice that of the shortest vector of $\mathbf{L}'$. We give further experimental evidence in Section 6.

Finally, we note that this process shares some similarities with random sampling reduction (RSR) [Sch03], where random linear combinations are LLL reduced to produce short vectors. While, here, we are only performing sparse sums and accept *larger* norms, the techniques used to analyse RSR might permit reducing our heuristic to a more standard heuristic assumption.

4 Scaled normal-form

The line of research improving the BKW algorithm for small secrets starting with [AFFP14] proceeds from the observation that we do not need to find $\mathbf{y} \cdot \mathbf{A} \equiv 0 \bmod q$, but if the secret is sufficiently small then any $\mathbf{y}$ such that $\mathbf{y} \cdot \mathbf{A}$ is short suffices, i.e. we seek short vectors $(\mathbf{w}, \mathbf{v})$ in the lattice

$$\Lambda = \{(\mathbf{y}, \mathbf{x}) \in \mathbb{Z}^m \times \mathbb{Z}^n : \mathbf{y} \cdot \mathbf{A} \equiv \mathbf{x} \bmod q\}.$$

Note that this lattice is the lattice considered in dual attacks on normal form LWE instances (cf. [ADPS15]).[5] Given a short vector in $(\mathbf{w}, \mathbf{v}) \in \Lambda$, we have

$$\mathbf{w} \cdot \mathbf{c} = \mathbf{w} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) = \langle \mathbf{v}, \mathbf{s} \rangle + \langle \mathbf{w}, \mathbf{e} \rangle .$$

Here, $\mathbf{v}$ corresponds to the noise from "modulus switching" or quantisation in BKW-style algorithms and $\mathbf{w}$ to the multiplicative factor by which the LWE noise increases due to repeated subtractions.

Now, in small secret LWE instances we have $\|\mathbf{s}\| < \|\mathbf{e}\|$. As a consequence, we may permit $\|\mathbf{v}\| > \|\mathbf{w}\|$ such that

$$\| \langle \mathbf{w}, \mathbf{s} \rangle \| \approx \| \langle \mathbf{v}, \mathbf{e} \rangle \|.$$

Hence, we consider the lattice

$$\Lambda_c = \{(\mathbf{y}, \mathbf{x}/c) \in \mathbb{Z}^m \times (1/c \cdot \mathbb{Z})^n : \mathbf{y} \cdot \mathbf{A} \equiv \mathbf{x} \bmod q\}$$

for some constant c, similar to [BG14]. The lattice Λ_c has dimension $m' = m + n$ and whp volume $(q/c)^n$. To construct a basis for Λ_c, assume $\mathbf{A}_{m-n:m}$ has full rank (this holds with high probability for large q). Then $\Lambda_c = \Lambda(\mathbf{L}')$ with

$$\mathbf{L}' = \begin{pmatrix} \frac{1}{c}\mathbf{I}_n & \mathbf{0}_{n\times(m-n)} & \mathbf{A}_{m-n:m}^{-1} \\ & \mathbf{I}_{m-n} & \mathbf{B}' \\ & & q\mathbf{I}_n \end{pmatrix}$$

where $[\mathbf{I}_{m-n}|\mathbf{B}']$ is a basis for the left kernel of $\mathbf{A} \bmod q$.

[5] The strategy seems folklore, we were unable to find a canonical reference for it.

Remark 2. In our estimates for HElib and SEAL, we typically have $m = n$ and $[\mathbf{I}_{m-n}|\mathbf{B}'] \in \mathbb{Z}^{0\times n}$.

It remains to establish c. Lattice reduction produces a vector $(\mathbf{w}, \mathbf{v})$ with

$$\|(\mathbf{w}, \mathbf{v})\| \approx \delta_0^{m'} \cdot (q/c)^{n/m'}, \tag{2}$$

which translates to a noise value

$$e = \mathbf{w} \cdot \mathbf{A} \cdot \mathbf{s} + \langle \mathbf{w}, \mathbf{e}\rangle = \langle c \cdot \mathbf{v}, \mathbf{s}\rangle + \langle \mathbf{w}, \mathbf{e}\rangle$$

and we set

$$c = \frac{\alpha\, q}{\sqrt{2\,\pi\, h}} \cdot \sqrt{m' - n}$$

to equalise the noise contributions of both parts of the above sum.

As a consequence, we arrive at the following lemma, which is attained by combining Equation (2) with Lemma 1.

Lemma 3. *Let $m' = 2\,n$ and $c = \frac{\alpha\, q}{\sqrt{2\,\pi\, h}} \cdot \sqrt{m' - n}$. A lattice reduction algorithm achieving δ_0 such that*

$$\log \delta_0 = \frac{\log\left(\frac{2\,n \log^2 \varepsilon}{\pi \alpha^2 h}\right)}{8\,n}$$

leads to an algorithm solving decisional LWE with $\mathbf{s} \leftarrow_{\$} \mathcal{B}_h^-$ instance with advantage ε and the same cost.

Remark 3. We focus on $m' = 2\,n$ in Lemma 3 for ease of exposure. For the instances considered in this work, $m' = 2\,n$ is a good approximation for m' (see Section 6).

For Example 1 we predict at a cost of $2^{107.4}$ operations mod q for solving Decision-LWE when applying this strategy. Amortising costs as suggested in Section 3 reduces it further to $2^{101.0}$ operations mod q.

Asymptotic behaviour. The general dual strategy, without exploiting small secrets, requires

$$\log \delta_0 = \frac{\log\left(-\frac{2\,\log \varepsilon}{\alpha^2 q}\right)}{4\,n}$$

according to [APS15]. For HElib's choice of $8 = \alpha\, q$ and $h = 64$ and setting ε constant, this expression simplifies to

$$\log \delta_0 = \frac{\log q + C_d}{4\,n},$$

for some constant C_d. On the other hand, Lemma 3 simplifies to

$$\log \delta_0 = \frac{\log q + \frac{1}{2}\log n + C_m}{4\,n}, \tag{3}$$

for some constant $C_m < C_d$.

For a circuit of depth L, BGV requires $\log q = L \log n + \mathcal{O}(L)$ [GHS12b, Appendix C.2]. Applying Lemma 2, we get that

$$\lim_{\kappa\to\infty} \frac{\mathrm{cost}_m}{\mathrm{cost}_d} = \lim_{n\to\infty} \frac{\mathrm{cost}_m}{\mathrm{cost}_d} = \frac{2\,L}{2\,L+1},$$

where cost_d is the log cost of the standard dual attack, cost_m is the log cost under Lemma 3 and κ the security parameter. The same analysis applies to any constant h. Finally, when $h = 2/3\,n$, i.e. $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}$, then the term $1/2 \cdot \log n$ vanishes from (3), but $C_m > C_d$.

5 Sparse secrets

Recall that BKW-style algorithms consist of two stages or, indeed, sub-algorithms. First, in the reduction stage, combinatorial methods are employed to transform an LWE instance in dimension n into an instance of dimension $0 \leq n' \leq n$, typically with increased noise level α. This smaller LWE instance is then, in the solving stage, is solved using some form of exhaustive search over the secret.

Taking the same perspective on the dual attack, write $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ with $\mathbf{A}_0 \in \mathbb{Z}_q^{m\times(n-k)}$ and $\mathbf{A}_1 \in \mathbb{Z}_q^{m\times k}$ and find a short vector in the lattice

$$\Lambda = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y}\cdot\mathbf{A}_0 \equiv 0 \bmod q\}.$$

Each short vector $\mathbf{y} \in \Lambda$ produces a sample for an LWE instance in dimension k and noise rate $\alpha' = \mathrm{E}[\|\mathbf{y}\|]\cdot\alpha$. Setting $k = 0$ recovers the original dual attack. For $k > 0$, we may now apply our favourite algorithm for solving small dimensional, easy LWE instances. Applying exhaustive search implies $\log_2 k < \kappa$ for $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{+}$ resp. $\log_3 k < \kappa$ for $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}$ when κ is the target level of security.

The case $\mathbf{s} \leftarrow_{\$} \mathcal{B}_h^{\pm}$ permits much larger k by relaxing the conditions we place on solving the k-dimensional instance. Instead of solving with probability one, we solve with some probability p_k and rerun the algorithm in case of failure.

For this, write $\mathbf{A}\cdot\mathbf{P} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ and $\mathbf{s}\cdot\mathbf{P} = [\mathbf{s}_0 \mid \mathbf{s}_1]$ where $\mathbf{P}$ is a random permutation matrix. Now, over the choice of $\mathbf{P}$ there is a good chance that $\mathbf{s}_1 = 0$ and hence that $\mathbf{A}_1 \cdot \mathbf{s}_1 \equiv 0 \bmod q$. That is, the right choice of $\mathbf{P}$ places all non-zero components of $\mathbf{s}$ in the $\mathbf{s}_0$ part.

In particular, with probability $1 - h/n$ a coordinate $\mathbf{s}_{(i)}$ is zero. More generally, picking k components of $\mathbf{s}$ at random will pick only components such that $\mathbf{s}_{(i)} = 0$ with probability

$$p_k = \prod_{i=0}^{k-1} \left(1 - \frac{h}{n-i}\right) = \frac{\binom{n-h}{k}}{\binom{n}{k}} \approx \left(1 - \frac{h}{n}\right)^k.$$

Hence, simply treating $k > 0$ in the solving stage the same as $k = 0$ succeeds with probability p_k. The success probability can be amplified to close to one by repeating the elimination and solving stages $\approx 1/p_k$ times assuming we distinguish with probability close to 1.

It is clear that the same strategy translates to the primal attack by simply dropping random columns before running the algorithm. However, for the dual attack, the following improvement can be applied. Instead of considering only $\mathbf{s}_1 = 0$, perform exhaustive search over those solutions that occur with sufficiently high probability. In particular, over the choice of $\mathbf{P}$, the probability that $\mathbf{s}_1$ contains $k - j$ components with $\mathbf{s}_{1,(i)} = 0$ and exactly j components with $\mathbf{s}_{1,(i)} \neq 0$ is

$$p_{k,j} = \frac{\binom{n-h}{k-j}\binom{h}{j}}{\binom{n}{k}},$$

i.e. follows the hypergeometric distribution.

Now, assuming $\mathbf{s} \leftarrow_{\$} \mathcal{B}_h^-$, to check if any of those candidates for $\mathbf{s}_1$ is correct, we need to compare $\binom{k}{j} \cdot 2^j$ distributions against the uniform distribution mod q.

Thus, after picking a parameter ℓ we arrive at Algorithm 2 with cost:

1. m calls to BKZ-β in dimension $n - k$.
2. $m \cdot \sum_{i=0}^{\ell} \binom{k}{i} \cdot 2^i \cdot i$ additions mod q to evaluate m samples on all possible solutions up to weight ℓ.

Assuming m is chosen such that distinguishing LWE from uniform succeeds with probability close to one, then Algorithm 2 succeeds with probability $\sum_{j=0}^{\ell} p_{k,j}$.

Asymptotic behaviour. We arrive at the following simple lemma:

Lemma 4. *Let $0 \leq h < n$ and $d > 1$ be constants, $p_{h,d}$ be some constant depending on h and d, $c_{n,\alpha,q}$ be the cost of solving LWE with parameters n, α, q with probability $\geq 1 - 2^{-p_{h,d}^2}$ Then, solving LWE in dimension n with $\mathbf{s} \leftarrow_{\$} \mathcal{B}_h^{\pm}$ costs $\mathcal{O}\left(c_{n-n/d,\alpha,q}\right)$ operations.*

```
Data: $m \times n$ matrix $\mathbf{A}$ over $\mathbb{Z}_q$
Data: $m$ vector $\mathbf{c}$ over $\mathbb{Z}_q$
Data: density parameter $0 \leq \ell \leq 64$
Data: dimension parameter $0 \leq k \leq n$
$\mathbf{P} \leftarrow_{\$}$ $n \times n$ permutation matrices;
$[\mathbf{A}_0 \mid \mathbf{A}_1] \leftarrow \mathbf{A} \cdot \mathbf{P}$ with $\mathbf{A}_0 \in \mathbb{Z}_q^{m \times (n-k)}$;
$\mathbf{L} \leftarrow$ basis for scaled-dual lattice of $\mathbf{A}_0$;
for $i \leftarrow 0$ to $m - 1$ do
    $\mathbf{y}_i \leftarrow$ a short vector in the row span of $\mathbf{L}$;
    $e'_i \leftarrow \langle \mathbf{y}_i, \mathbf{c} \rangle$;
end
if $e'_i$ follow discrete Gaussian distribution then
    return $\top$;
end
foreach $\mathbf{s}'$ in the set of $\sum_{i=0}^{\ell} \binom{k}{i} \cdot 2^i$ candidate solutions do
    for $i \leftarrow 0$ to $m - 1$ do
        $e''_i = e'_i + \left\langle \mathbf{A}_{1,(i)}, \mathbf{s}' \right\rangle$;
    end
    if $e''_i$ follow discrete Gaussian distribution then
        return $\top$;
    end
end
return $\perp$;
```

Algorithm 2: SILKE$_2$: Sparse secrets in BKW-style SIS strategy for solving LWE.

Proof. Observe that $p_{h,d} = \lim_{n\to\infty} \binom{n-h}{n/d} / \binom{n}{n/d}$ is a constant for any constant $0 \leq h < n$ and $d > 1$. Hence, solving $\mathcal{O}(1/p_{h,d}) = \mathcal{O}(1)$ instances in dimension $n - n/d$ solves the instance in dimension n. □

Remark 4. Picking $d = 2$ we get $\lim_{n\to\infty} \binom{n-h}{n/2} / \binom{n}{n/2} = 2^{-h}$ and an overall costs of $\mathcal{O}\left(2^h \cdot c_{n/2,\alpha,q}\right)$. This improves on exhaustive search, which costs $\mathcal{O}\left(2^h \cdot \binom{n}{h}\right)$, when $c_{n/2,\alpha,q} \in o\left(\binom{n}{h}\right)$.

6 Combined

Combining the strategies described in this work, we arrive at Algorithm 3 (SILKE). It takes a flag *sparse* which enables the sparse strategy of Algorithm 2. In this case, we enforce that distinguishing LWE from uniform succeeds with probability $1 - 2^{-\kappa}$ when we guessed $\mathbf{s}'$ correctly. Clearly, this parameter can be improved, i.e. this probability reduced, but amplifying the success probability is relatively cheap, so we forego this improvement.

```
Data: candidate LWE samples $\mathbf{A}, \mathbf{c} \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$
Data: BKZ block sizes $\beta, \beta' \geq 2$
Data: target success probability $\varepsilon$
Data: $sparse$ flag toggling sparse strategy
Data: scale factor $c \geq 1$
Data: dimension parameter $0 \leq k \leq n$, 0 when $sparse$ is set
Data: density parameter $0 \leq \ell \leq k$, 0 when $sparse$ is set
// distinguishing advantage per sample from $\beta, \beta'$
$\varepsilon_d \leftarrow \exp(-\pi(\mathrm{E}[\|\mathbf{y}_i\|] \cdot \alpha)^2)$;
if $sparse$ then
    $\varepsilon_t \leftarrow 1 - 1/2^\kappa$; // for security parameter $\kappa$
    $r \leftarrow \max\left(\lceil \log(1-\varepsilon)/\log(1 - \sum_{j=0}^{\ell} p_{k,j}) \rceil, 1\right)$;
else
    $\varepsilon_t, r \leftarrow \varepsilon, 1$;
end
// required number of samples for majority vote
$m \leftarrow \lceil 2\,\log(2 - 2\,\varepsilon_t)/\log(1 - 4\,\varepsilon_d^2) \rceil$;
repeat $r$ times
    $\mathbf{P} \leftarrow_\$ n \times n$ permutation matrices;
    $[\mathbf{A}_0 \mid \mathbf{A}_1] \leftarrow \mathbf{A} \cdot \mathbf{P}$ with $\mathbf{A}_0 \in \mathbb{Z}_q^{m \times (n-k)}$;
    $\mathbf{L} \leftarrow$ basis for $\{(\mathbf{y}, \mathbf{x}/c) \in \mathbb{Z}^m \times (1/c \cdot \mathbb{Z})^n : \mathbf{y} \cdot \mathbf{A}_0 \equiv \mathbf{x} \bmod q\}$;
    $\mathbf{L}' \leftarrow$ BKZ-$\beta$ reduced basis for $\mathbf{L}$;
    for $i \leftarrow 0$ to $m-1$ do
        $\mathbf{U} \leftarrow_\$$ a sparse unimodular matrix with small entries;
        $\mathbf{L}_i \leftarrow \mathbf{U} \cdot \mathbf{L}'$;
        $\mathbf{L}'_i \leftarrow$ BKZ-$\beta'$ reduced basis for $\mathbf{L}_i$;
        $(\mathbf{w}_i, \mathbf{v}_i) \leftarrow$ shortest row vector in $\mathbf{L}'_i$;
        $e'_i \leftarrow \langle \mathbf{w}_i, \mathbf{c} \rangle$;
    end
    if $e'_i$ follow discrete Gaussian distribution then
        return $\top$;
    end
    foreach $\mathbf{s}'$ in the set of $\sum_{i=1}^{\ell} \binom{k}{i} \cdot 2^i$ candidate solutions do
        for $i \leftarrow 0$ to $m-1$ do
            $e''_i = e'_i + \langle \mathbf{A}_{1,(i)}, \mathbf{s}' \rangle$;
        end
        if $e''_i$ follow discrete Gaussian distribution then
            return $\top$;
        end
    end
return $\perp$;
```

Algorithm 3: SILKE : (Sparse) BKW-style SIS Strategy for solving LWE

We give an implementation of Algorithm 3 for $sparse$ = false in Appendix B. For brevity, we skip the $sparse$ = true case. We also tested our implementation on several parameter sets:

1. Considering an LWE instance with $n = 100$ and $q \approx 2^{23}$, $\alpha = 8/q$ and $h = 20$, we first BKZ-50 reduced the basis $\mathbf{L}$ for $c = 16$. This produced a short vector $\mathbf{w}$ such that $|\langle \mathbf{w}, \mathbf{c}\rangle| \approx 2^{15.3}$. Then, running LLL 256 times, we produced short vectors such that $\mathrm{E}[|\langle \mathbf{w}_i, \mathbf{c}\rangle|] = 2^{15.7}$ and standard deviation $2^{16.6}$.
2. Considering an LWE instance with $n = 140$ and $q \approx 2^{40}$, $\alpha = 8/q$ and $h = 32$, we first BKZ-70 reduced the basis $\mathbf{L}$ for $c = 1$. This took 64 hours and produced a short vector $\mathbf{w}$ such that $|\langle \mathbf{w}, \mathbf{c}\rangle| \approx 2^{23.7}$, with $\mathrm{E}[|\langle \mathbf{w}, \mathbf{c}\rangle|] \approx 2^{25.5}$ conditioned on $|\mathbf{w}|$. Then, running LLL 140 times (each run taking about 50 seconds on average), we produced short vectors such that $\mathrm{E}[|\langle \mathbf{w}_i, \mathbf{c}\rangle|] = 2^{26.0}$ and standard deviation $2^{26.4}$ for $\langle \mathbf{w}_i, \mathbf{c}\rangle$.
3. Considering the same LWE instance with $n = 140$ and $q \approx 2^{40}$, $\alpha = 8/q$ and $h = 32$, we first BKZ-70 reduced the basis $\mathbf{L}$ for $c = 16$. This took 65 hours and produced a short vector $\mathbf{w}$ such that $|\langle \mathbf{w}, \mathbf{c}\rangle| \approx 2^{24.7}$ after scaling by c, cf. $\mathrm{E}[|\langle \mathbf{w}, \mathbf{c}\rangle|] \approx 2^{24.8}$. Then, running LLL 140 times (each run taking about 50 seconds on average), we produced short vectors such that $\mathrm{E}[|\langle \mathbf{w}_i, \mathbf{c}\rangle|] = 2^{25.5}$ and standard deviation $2^{25.9}$ for $\langle \mathbf{w}_i, \mathbf{c}\rangle$.
4. Considering again the same LWE instance with $n = 140$ and $q \approx 2^{40}$, $\alpha = 8/q$ and $h = 32$, we first BKZ-70 reduced the basis $\mathbf{L}$ for $c = 1$. This took 30 hours and produced a short vector $\mathbf{w}$ such that $|\langle \mathbf{w}, \mathbf{c}\rangle| \approx 2^{25.2}$, cf. $\mathrm{E}[|\langle \mathbf{w}, \mathbf{c}\rangle|] \approx 2^{25.6}$. Then, running LLL 1024 times (each run taking about 50 seconds on average), we produced 1016 short vectors such that $\mathrm{E}[|\langle \mathbf{w}_i, \mathbf{c}\rangle|] = 2^{25.8}$ and standard deviation $2^{26.1}$ for $\langle \mathbf{w}_i, \mathbf{c}\rangle$.
5. Considering an LWE instance with $n = 180$ and $q \approx 2^{40}$, $\alpha = 8/q$ and $h = 48$, we first BKZ-70 reduced the basis $\mathbf{L}$ for $c = 8$. This took 198 hours[6] and produced a short vector $\mathbf{w}$ such that $|\langle \mathbf{w}, \mathbf{c}\rangle| \approx 2^{26.7}$, cf. $\mathrm{E}[|\langle \mathbf{w}, \mathbf{c}\rangle|] \approx 2^{25.9}$. Then, running LLL 180 times (each run taking about 500 seconds on average), we produced short vectors such that $\mathrm{E}[|\langle \mathbf{w}_i, \mathbf{c}\rangle|] = 2^{26.6}$ and standard deviation $2^{26.9}$ for $\langle \mathbf{w}_i, \mathbf{c}\rangle$.

All our experiments match our prediction bounding the growth of the norms of our vectors by a factor of two. Note, however, that in the fourth experiment 1 in 128 vectors found with LLL was a duplicate of previously discovered vector, indicating that re-randomisation is not perfect. While the effect of this loss on the running time of the overall algorithm is small, it highlights that further research is required on the interplay of re-randomisation and lattice reduction.

Applying Algorithm 3 to parameter choices from HElib and SEAL, we arrive at the estimates in Table 1. These estimates were produced using the Sage [S$^+$15]

[6] We ran 49 BKZ tours until fplll's auto abort triggered. After 16 tours the norm of the then shortest vector was by a factor 1.266 larger than the norm of the shortest vector found after 49 tours.

code available at `http://bitbucket.org/malb/lwe-estimator` which optimises the parameters c, ℓ, k, β to minimise the overall cost.

For the HElib parameters in Table 1 we chose the sparse strategy. Here, amortising costs as in Section 3 did not lead to a significant improvement, which is why we did not use it in these cases. All considered lattices have dimension $< 2\,n$. Hence, one Ring-LWE sample is sufficient to mount these attacks. Note that this is less than the dual attack as described in [GHS12a] would require (two samples).

For the SEAL parameter choices in Table 1, dimension $n = 1024$ requires two Ring-LWE samples, larger dimensions only require one sample. Here, amortising costs as in Algorithm 1 does lead to a modest improvement and is hence enabled.

Finally, we note that reducing q to $\approx 2^{34}$ resp. $\approx 2^{560}$ leads to an estimated cost of 80 bits for $n = 1024$ resp. $n = 16384$ for $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}_{64}$. For $\mathbf{s} \leftarrow_{\$} \mathcal{B}^{-}$, $q \approx 2^{40}$ resp. $q \approx 2^{660}$ leads to an estimated cost of 80 bits under the techniques described here. In both cases, we assume $\sigma \approx 3.2$.

Acknowledgements. We thank Kenny Paterson and Adeline Roux-Langlois for helpful comments on an earlier draft of this work.

References

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Heidelberg, August 2016.

[ACF$^+$15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74:325–354, 2015.

[ACFP14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. http://eprint.iacr.org/2014/1018.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[ADPS15] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. Cryptology ePrint Archive, Report 2015/1092, 2015. http://eprint.iacr.org/2015/1092.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016.

[AFFP14] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 429–445. Springer, Heidelberg, March 2014.

[AFG14] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[BCD$^+$16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1006–1018. ACM Press, October 2016.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24. ACM-SIAM, January 2016.

[BG14] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014.

[BGPW16] Johannes A. Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 24–43. Springer, Heidelberg, April 2016.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd ACM STOC*, pages 435–440. ACM Press, May 2000.

[BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, Heidelberg, December 2013.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Safavi-Naini and Canetti [SNC12], pages 868–886.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Paris 7, 2013.

[CKH$^+$16] Jung Hee Cheon, Jinsu Kim, Kyoo Hyung Han, Yongha Son, and Changmin Lee. Practical post-quantum public key cryptosystem based on LWE. In Seokhie Hong and Jong Hwan Park, editors, *19th Annual International Conference on Information Security and Cryptology (ICISC)*, Lecture Notes in Computer Science. Springer, 2016.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011.

[CN12] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates (full version). `http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf`, 2012.

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In Oswald and Fischlin [OF15], pages 513–536.

[CS16] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340. Springer, Heidelberg, February / March 2016.

[DTV15] Alexandre Duc, Florian Tramèr, and Serge Vaudenay. Better algorithms for LWE and LWR. In Oswald and Fischlin [OF15], pages 173–202.

[DXL12] Jintai Ding, Xiang Xie, and Xiaodong Lin. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. http://eprint.iacr.org/2012/688.

[FPL16] The FPLLL development team. FPLLL 5.0, a lattice reduction library, 2016. Available at https://github.com/fplll/fplll.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [SNC12], pages 850–867.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. Cryptology ePrint Archive, Report 2012/099, 2012. http://eprint.iacr.org/2012/099.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Gennaro and Robshaw [GR15], pages 23–42.

[GR15] Rosario Gennaro and Matthew J. B. Robshaw, editors. *CRYPTO 2015, Part I*, volume 9215 of *LNCS*. Springer, Heidelberg, August 2015.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, Heidelberg, August 2007.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464. Springer, Heidelberg, August 2011.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Gennaro and Robshaw [GR15], pages 43–62.

[KF16] Paul Kirchner and Pierre-Alain Fouque. Comparison between subfield and straightforward attacks on NTRU. *IACR Cryptology ePrint Archive*, 2016:717, 2016.

[KL15] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(5):1–12, 2015.

[Laa15] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Gennaro and Robshaw [GR15], pages 3–22.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 293–309. Springer, Heidelberg, February / March 2013.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 318–335. Springer, Heidelberg, May 2014.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011.

[LP16] Kim Laine and Rachel Player. Simple encrypted arithmetic library - seal (v2.0). Technical report, Microsoft Research, September 2016.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. Cryptology ePrint Archive, Report 2013/293, 2013. http://eprint.iacr.org/2013/293.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, Berlin, Heidelberg, New York, 2009.

[OF15] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*. Springer, Heidelberg, April 2015.

[Pei09] Chris Peikert. Some recent progress in lattice-based cryptography. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, page 72. Springer, Heidelberg, March 2009.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[S^+15] William Stein et al. *Sage Mathematics Software Version 7.1.* The Sage Development Team, 2015. Available at http://www.sagemath.org.

[Sch03] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2003.

[Sho01] Victor Shoup. NTL: A library for doing number theory. http://www.shoup.net/ntl/, 2001.

[SL12] Jayalal Sarma and Princy Lunawat. IITM-CS6840: Advanced Complexity Theory — Lecture 11: Amplification Lemma. http://www.cse.iitm.ac.in/~jayalal/teaching/CS6840/2012/lecture11.pdf, 2012.

[SNC12] Reihaneh Safavi-Naini and Ran Canetti, editors. *CRYPTO 2012*, volume 7417 of *LNCS*. Springer, Heidelberg, August 2012.

[Wal15] Michael Walter. Lattice point enumeration on block reduced bases. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15*, volume 9063 of *LNCS*, pages 269–282. Springer, Heidelberg, May 2015.

[Wun16] Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. Cryptology ePrint Archive, Report 2016/733, 2016. http://eprint.iacr.org/2016/733.

A Rerandomisation

```
Data: $n \times m$ matrix $\mathbf{L}$
Data: density parameter $d$, default $d = 3$
Result: $\mathbf{U} \cdot \mathbf{L}$ where $\mathbf{U}$ is a sparse, unimodular matrix.
for $i \leftarrow 0$ to $4 \cdot n - 1$ do
    $a \leftarrow_{\$} \{0, n-1\}$;
    $b \leftarrow_{\$} \{0, n-1\} \setminus \{a\}$;
    $\mathbf{L}_{(b)}, \mathbf{L}_{(a)} \leftarrow \mathbf{L}_{(a)}, \mathbf{L}_{(b)}$ ;
end
for $a \leftarrow 0$ to $n - 2$ do
    for $i \leftarrow 0$ to $d - 1$ do
        $b \leftarrow_{\$} \{a+1, n-1\}$;
        $s \leftarrow_{\$} \{0, 1\}$;
        $\mathbf{L}_{(a)} \leftarrow \mathbf{L}_{(a)} + (-1)^s \cdot \mathbf{L}_{(b)}$;
    end
end
return $\mathbf{L}$;
```

Algorithm 4: Rerandomisation strategy in the fplll library [FPL16].

B Implementation

```python
# -*- coding: utf-8 -*-

from sage.all import shuffle, randint, ceil, next_prime, log, cputime, mean, variance, set_random_seed, sqrt
from copy import copy
from sage.all import GF, ZZ
from sage.all import random_matrix, random_vector, vector, matrix, identity_matrix
from sage.stats.distributions.discrete_gaussian_integer import DiscreteGaussianDistributionIntegerSampler \
    as DiscreteGaussian
from estimator.estimator import preprocess_params, stddevf


def gen_fhe_instance(n, q, alpha=None, h=None, m=None, seed=None):
    """
    Generate FHE-style LWE instance

    :param n:     dimension
    :param q:     modulus
    :param alpha: noise rate (default: 8/q)
    :param h:     hamming weight of the secret (default: 2/3n)
    :param m:     number of samples (default: n)

    """
    if seed is not None:
        set_random_seed(seed)

    q = next_prime(ceil(q)-1, proof=False)
    if alpha is None:
        alpha = ZZ(8)/q

    n, alpha, q = preprocess_params(n, alpha, q)

    stddev = stddevf(alpha*q)

    if m is None:
        m = n
    K = GF(q, proof=False)
    A = random_matrix(K, m, n)

    if h is None:
        s = random_vector(ZZ, n, x=-1, y=1)
    else:
        S = [-1, 1]
        s = [S[randint(0, 1)] for i in range(h)]
        s += [0 for _ in range(n-h)]
        shuffle(s)
        s = vector(ZZ, s)
    c = A*s

    D = DiscreteGaussian(stddev)

    for i in range(m):
        c[i] += D()

    return A, c


def dual_instance0(A):
    """
    Generate dual attack basis.

    :param A: LWE matrix A

    """
    q = A.base_ring().order()
    B0 = A.left_kernel().basis_matrix().change_ring(ZZ)
    m = B0.ncols()
    n = B0.nrows()
    r = m-n
    B1 = matrix(ZZ, r, n).augment(q*identity_matrix(ZZ, r))
    B = B0.stack(B1)
    return B


def dual_instance1(A, scale=1):
    """
    Generate dual attack basis for LWE normal form.

    :param A: LWE matrix A

    """
    q = A.base_ring().order()
    n = A.ncols()
    B = A.matrix_from_rows(range(0, n)).inverse().change_ring(ZZ)
    L = identity_matrix(ZZ, n).augment(B)
    L = L.stack(matrix(ZZ, n, n).augment(q*identity_matrix(ZZ, n)))

    for i in range(0, 2*n):
        for j in range(n, 2*n):
            L[i, j] = scale*L[i, j]

    return L


def balanced_lift(e):
    """
    Lift e mod q to integer such that result is between -q/2 and q/2

    :param e: a value or vector mod q

    """
    from sage.rings.finite_rings.integer_mod import is_IntegerMod

    q = e.base_ring().order()
    if is_IntegerMod(e):
        e = ZZ(e)
        if e > q//2:
            e -= q
        return e
    else:
        return vector(balanced_lift(ee) for ee in e)


def apply_short1(y, A, c, scale=1):
    """
    Compute `y*A`, `y*c` where y is a vector in the integer row span of
    ``dual_instance(A)``

    :param y: (short) vector in scaled dual lattice
    :param A: LWE matrix
    :param c: LWE vector
    """
    m = A.nrows()
    y = vector(ZZ, 1/ZZ(scale) * y[-m:])
    a = balanced_lift(y*A)
    e = balanced_lift(y*c)
    return a, e


def log_mean(X):
    return log(mean([abs(x) for x in X]), 2)
```

```
def log_var(X):
    return log(variance(X).sqrt(), 2)


def silke(A, c, beta, h, m=None, scale=1, float_type="double"):
    """

    :param A:    LWE matrix
    :param c:    LWE vector
    :param beta: BKW block size
    :param m:    number of samples to consider
    :param scale: scale rhs of lattice by this factor

    """
    from fpylll import BKZ, IntegerMatrix, LLL, GSO
    from fpylll.algorithms.bkz2 import BKZReduction as BKZ2

    if m is None:
        m = A.nrows()

    L = dual_instance1(A, scale=scale)
    L = IntegerMatrix.from_matrix(L)
    L = LLL.reduction(L, flags=LLL.VERBOSE)
    M = GSO.Mat(L, float_type=float_type)
    bkz = BKZ2(M)
    t = 0.0
    param = BKZ.Param(block_size=beta,
                      strategies=BKZ.DEFAULT_STRATEGY,
                      auto_abort=True,
                      max_loops=16,
                      flags=BKZ.VERBOSE|BKZ.AUTO_ABORT|BKZ.MAX_LOOPS)
    bkz(param)
    t += bkz.stats.total_time

    H = copy(L)

    import pickle
    pickle.dump(L, open("L-%d-%d.sobj"%(L.nrows, beta), "wb"))

    E = []
    Y = set()
    V = set()
    y_i = vector(ZZ, tuple(L[0]))
    Y.add(tuple(y_i))
    E.append(apply_short1(y_i, A, c, scale=scale)[1])

    v = L[0].norm()
    v_ = v/sqrt(L.ncols)
    v_r = 3.2*sqrt(L.ncols - A.ncols())*v_/scale
    v_l = sqrt(h)*v_

    fmt = u"{\"t\": %5.1fs, \"log(sigma)\": %5.1f, \"log(|y|)\": %5.1f, \"log(E[sigma]):\" %5.1f}"

    print
    print fmt%(t,
               log(abs(E[-1]), 2),
               log(L[0].norm(), 2),
               log(sqrt(v_r**2 + v_l**2), 2))
    print
    for i in range(m):
        t = cputime()
        M = GSO.Mat(L, float_type=float_type)
        bkz = BKZ2(M)
        t = cputime()
        bkz.randomize_block(0, L.nrows, stats=None, density=3)
        LLL.reduction(L)
        y_i = vector(ZZ, tuple(L[0]))
        l_n = L[0].norm()
        if L[0].norm() > H[0].norm():
            L = copy(H)
        t = cputime(t)

        Y.add(tuple(y_i))
        V.add(y_i.norm())
        E.append(apply_short1(y_i, A, c, scale=scale)[1])
        if len(V) >= 2:
            fmt =  u"{\"i\": %4d, \"t\": %5.1fs, \"log(|e_i|)\": %5.1f, \"log(|y_i|)\": %5.1f,"
            fmt += u"\"log(sigma)\": (%5.1f,%5.1f), \"log(|y|)\": (%5.1f,%5.1f), |Y|: %5d}"
            print fmt%(i+2, t, log(abs(E[-1]), 2), log(l_n, 2), log_mean(E), log_var(E), log_mean(V), log_var(V), len(Y))

    return E
```

C Alternative Cost Models

[LP11]					
n	1024	2048	4096	8192	16384
SEAL 80-bit					
q	47.5	95.4	192.0	392.1	799.6
dual	107.9	97.4	88.0	82.0	78.8
small	80.5	81.1	78.2	76.4	75.9
HElib 80-bit					
q	47.0	87.0	167.0	326.0	638.0
dual	111.5	112.4	111.5	111.2	111.2
sparse	58.1	62.6	65.4	69.2	71.5
HElib 128-bit					
q	38.0	70.0	134.0	261.0	511.0
dual	162.0	162.1	160.1	159.1	159.5
sparse	76.3	81.9	85.8	86.2	90.3
[LN14,APS15], $8-16$ BKZ tours					
n	1024	2048	4096	8192	16384
SEAL 80-bit					
q	47.5	95.4	192.0	392.1	799.6
dual	101.2	91.7	83.1	78.3	76.1
small	74.5	76.0	74.1	73.5	73.2
HElib 80-bit					
q	47.0	87.0	167.0	326.0	638.0
dual	105.1	107.1	106.8	107.7	108.8
sparse	54.1	59.1	62.8	65.8	68.9
HElib 128-bit					
q	38.0	70.0	134.0	261.0	511.0
dual	158.4	159.8	158.6	158.3	160.0
sparse	72.0	77.4	81.4	84.3	87.1

Table 3. Costs of dual attacks on HElib and SEAL in the [LP11] cost model resp. assuming SVP in dimension β costs $2^{0.64\beta-28}$ operations as in [LN14] plugged into the estimator from [APS15]; cf. Table 1.