A Practical Adaptive Key Recovery Attack on the LGM (GSW-like) Cryptosystem[⋆]

Prastudy Fauzi[1], Martha Norberg Hovd[1,2], and Håvard Raddum[1]

[1] Simula UiB, Norway
[2] University of Bergen, Norway

Abstract. We present an adaptive key recovery attack on the leveled homomorphic encryption scheme suggested by Li, Galbraith and Ma (Provsec 2016), which itself is a modification of the GSW cryptosystem designed to resist key recovery attacks by using a different linear combination of secret keys for each decryption. We were able to efficiently recover the secret key for a realistic choice of parameters using a statistical attack. In particular, this means that the Li, Galbraith and Ma strategy does not prevent adaptive key recovery attacks.

Keywords: Key recovery, somewhat homomorphic encryption, GSW, statistical attack

1 Introduction

Fully homomorphic encryption (FHE) is a powerful primitive which allows for meaningful computations to be performed on encrypted data, without the need for decryption. An FHE scheme allows for ciphertexts to be evaluated over any circuit without risking an erroneous decryption of the resulting ciphertext, i.e., $\mathrm{Dec}(C(\mathrm{Enc}(m))) \neq C(m)$ for some circuit C. There are other flavours of homomorphic encryption as well: leveled homomorphic encryption (LHE) and somewhat homomorphic encryption (SHE), which both allow for a *limited amount* of operations to be performed on a ciphertext before there is a risk of decryption failure. A key difference between LHE and SHE is that the key generation of LHE schemes takes an extra parameter as input, which specifies the depth of the deepest circuit the scheme is able to homomorphically evaluate.

Many F/L/SHE schemes rely on a quantum secure assumption, such as the hardness of learning with errors (LWE) and ring learning with errors (RLWE), to provide security against key recovery attacks and/or message recovery. In fact, these schemes are typically shown to achieve IND-CPA security, meaning an adversary with access only to the public key and parameters is provably unable to distinguish between the encryptions of any two messages. However, most existing F/L/SHE schemes are known to be susceptible to *adaptive* key recovery attacks [8,10]. In these attacks an adversary has temporary access to

[⋆] This paper appeared at PQCrypto 2021. The final authenticated publication is available online at `https://doi.org/10.1007/978-3-030-81293-5_25`.

a decryption oracle, and is able to recover the secret key based on information leaked from the decryption queries.

For example, schemes based on LWE or RLWE (such as GSW [11]) can leak one bit of the secret key from a small number of decryption queries. These schemes have public keys of the form $(\mathbf{A}, \mathbf{As}+\mathbf{e})$ with secret key $\mathbf{s}$, a matrix $\mathbf{A}$, and noise $\mathbf{e}$. Key recovery attacks either compute $\mathbf{s}$ directly, or first compute the noise $\mathbf{e}$ and use this to derive $\mathbf{s}$. Chenal and Tang used this approach to attack several (R)LWE schemes, one of which was GSW [8].

Li, Galbraith and Ma (LGM, [13]) proposed a technique to circumvent such key recovery attacks: instead of decrypting using a single secret key $\mathbf{s}$, they suggested changing secret keys for every decryption, so any information leaked from two different decryption queries would be unrelated, which should make it impossible for an adversary to recover any secret key. They constructed an LHE scheme based on GSW they claimed achieved IND-CCA1 security, though they were unable to provide a formal proof. [3]

Concretely, they start with the dual version of GSW, where the public key is of the form $(\mathbf{A}, \mathbf{As})$, but the secret key $\mathbf{s}$ must have small norm; security now depends on the hardness of the inhomogeneous short integer solution (ISIS) problem, which is also assumed to be quantum secure. Instead of having one secret key $\mathbf{s}$, they generate t distinct secret keys: $\mathbf{s}_1, \ldots, \mathbf{s}_t$. During decryption a random linear combination of the secret keys, $\mathbf{s}' = \sum_{i=1}^{t} \lambda_i \mathbf{s}_i$, is used, where the λ_i's are redrawn from a distribution for each decryption. The message space is $\mathbb{Z}_2$, so a decryption query leaks, at best, one bit of $\mathbf{s}'$: an unknown linear combination of secret keys unlikely to ever be reused, since the λ_is that generate it are redrawn for every decryption. The technique successfully thwarts the known adaptive key recovery attacks on GSW and similar schemes, and Li et al. therefore argue that their scheme achieves IND-CCA1 security, though they are unable to prove it.

In this paper we show that the LGM scheme is still susceptible to an adaptive key recovery attack, as we are able to recover a secret key using a statistical attack. We go even further to claim that the general approach of using a random linear combination of secret keys for each decryption query is susceptible to statistical adaptive key recovery attacks. To the best of our knowledge, the LGM scheme is currently the only concrete leveled homomorphic encryption scheme attempting to achieve IND-CCA1 security, which has proven a difficult security notion to achieve for SHE or LHE schemes. [4]

2 Preliminaries

Vectors are denoted by bold, lower case letters, and are assumed to be in column form. Logarithms are always base 2. For a real number x, let $\lfloor x \rceil = \lfloor x + \frac{1}{2} \rfloor$ be the

[3] The original paper was published in ProvSec 2016 [14] however, the ePrint version [13] of the paper contains major changes. In particular, the scheme we mount the adaptive key recovery attack on in this article is found in the ePrint version.

[4] There are suggestions for generic constructions achieving IND-CCA1 security (e.g., [7]), but there are no concrete instantiations of these constructions.

closest integer to x. For any integers x and q, let $x \mod q$ denote the modular reduction centered around zero. For a vector $\mathbf{v}$, let $\|\mathbf{v}\|$ be its Euclidean norm. Unless stated otherwise, we refer to somewhat, leveled, and fully homomorphic encryption schemes as simply *homomorphic encryption schemes.*

The gadget vector $\mathbf{g}$ is defined as the column vector $(1, 2, \ldots, 2^{l-1})^T$, and the gadget matrix is defined as $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} \in \mathbb{Z}_q^{n \times nl}$ (i.e., $\mathbf{G}$ is a matrix with $\mathbf{g}$ on the diagonal), for $l = \lfloor \log(q) \rceil + 1$.

Define $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times n'} \to \{0,1\}^{nl \times n'}$ to be the operation such that for any matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times n'}$, we have that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{M}) = \mathbf{M}$.

IND-CCA1. An encryption scheme $\mathcal{E} = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ achieves the security notion *indistinguishability under (non-adaptive) chosen ciphertext attack* (IND-CCA1) if any probabilistic polynomial time adversary $\mathbb{A}$ has at most a $1/2 + \mathrm{negl}$ chance of winning the following game against a challenger $\mathcal{C}$:

- $\mathcal{C}$ derives the parameters using $params \leftarrow \mathrm{Setup}(1^\kappa)$, draws a key pair $(pk, sk) \leftarrow \mathrm{KeyGen}(params)$, and sends pk and the parameters to $\mathbb{A}$.
- $\mathbb{A}$ sends ciphertexts c to her decryption oracle $\mathcal{O}_{\mathrm{Dec}}$, which returns $\mathrm{Dec}(c)$.
- $\mathbb{A}$ sends two messages of equal length (m_0, m_1) to $\mathcal{C}$.
- $\mathcal{C}$ returns $c \leftarrow \mathrm{Enc}(pk, m_b)$ to $\mathbb{A}$, for a randomly chosen bit $b \in \{0, 1\}$.
- $\mathbb{A}$ outputs the bit b^*, and wins if $b^* = b$.

The notion of IND-CPA security is defined in a similar way, but here $\mathbb{A}$ does not have access to a decryption oracle.

An adaptive key recovery attack is stronger than an IND-CCA1 attack, as recovering the secret key enables an adversary to decrypt *all* ciphertexts, not just distinguish between the encryptions of two chosen messages.

LWE. The *Learning With Errors* (LWE) distribution is defined as follows: for a fixed vector $\mathbf{s}$ drawn uniformly at random from $\mathbb{Z}_q^n$, sample a vector $\mathbf{a}$ uniformly at random from $\mathbb{Z}_q^n$ and an error e from some noise distribution χ, and output $(\mathbf{a}, b = \mathbf{a} \cdot \mathbf{s}^T + e \mod q)$. The search problem of LWE is to find $\mathbf{s}$ given m samples of the LWE distribution, where $\mathbf{s}$ is fixed for all the samples.

ISIS. Given a modulus q, a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, and a vector $\mathbf{u}$, the *Inhomogeneous Short Integer Solution* (ISIS) problem is to find a vector $\mathbf{e}$ drawn from a distribution χ with bound B such that $\mathbf{Be} = \mathbf{u} \mod q$, if such a vector exists. It is required that $m > n$ to prevent an adversary simply finding $\mathbf{e}$ using Gaussian elimination [13,5].

2.1 Distributions

For integers $a \leq b$, let $[a, b]$ denote the set of integers x such that $a \leq x \leq b$. A distribution over values $S = [a, b]$ for integers $a \leq b$ is *discrete uniform* if all $n = b - a + 1$ values $x \in S$ can occur with equal probability $1/n$. Such a distribution has mean $\frac{a+b}{2}$ and variance $\frac{n^2-1}{12}$.

A distribution over values $\mathbb{R}$ is *Gaussian* with mean μ and variance σ^2 if it follows the probability density function

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \quad .$$

A random variable following a Gaussian distribution is also said to be *normally distributed*. The value σ is also known as the *standard deviation*.

We provide, without proof, some known properties of Gaussian distributions.

Lemma 1. *Let $(X_i)_{i=1}^n$ be normally distributed independent random variables with mean μ_i and variance σ_i^2 for $i \in \{1, 2, \ldots, n\}$. Let $(a_i)_{i=1}^n$ be real numbers. Then $X = \sum_{i=1}^n a_i X_i$ is also normally distributed, with mean $\sum_{i=1}^n a_i \mu_i$ and variance $\sum_{i=1}^n a_i^2 \sigma_i^2$.*

Lemma 2. *Let X be a random variable following a Gaussian distribution with mean μ and variance σ^2. Then $Pr[|X - \mu| \geq t\sigma] = \mathbf{erf}(t/\sqrt{2})$, where $\mathbf{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ is the* error function. *In particular, $Pr[|X - \mu| \geq 5\sigma] \leq 2^{-20}$.*

Theorem 1 (Central limit theorem for sample). *Let $X_1, \ldots, X_n$ be independent random variables from a distribution with mean μ and variance σ^2. Let $X = 1/n \cdot \sum_{i=1}^n X_i$. Then if n approaches infinity, $X - \mu$ converges to a Gaussian distribution with mean 0 and variance σ^2/n.*

Informally, by taking a large enough sample size n, $X - \mu$ has mean ϵ_μ and variance $\sigma^2/n + \epsilon_\sigma$, where $\epsilon_\mu, \epsilon_\sigma$ may both be made arbitrarily small.

Discrete Gaussian Distribution. The *discrete Gaussian distribution* may be viewed as a Gaussian distribution where the values are restricted to a countable set, say $\mathbb{Z}$. To preserve the desirable properties of Gaussian distributions mentioned above, one should not simply sample a Gaussian and round to the closest integer. Instead, we adapt the definition of Gaussian distributions over $S \subseteq \mathbb{Z}$ presented by Micciancio and Walter [16]. For the more general definition over $S \subseteq \mathbb{Z}^n$, we refer to [1,13].

Definition 1. *Let S be a subset of $\mathbb{Z}$. For $c \in \mathbb{R}$ and a parameter $\sigma > 0 \in \mathbb{R}$, define $\rho_{\sigma,c}(x) = e^{-\pi \frac{(x-c)^2}{\sigma^2}}$ and $\rho_{\sigma,c}(S) = \sum_{x \in S} \rho_{\sigma,c}(x)$. The discrete Gaussian distribution over S with center c and standard deviation $\sigma$is defined as*

$$\forall x \in S : D_{S,\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(S)}.$$

We also state Theorem 7 (one dimensional leftover hash lemma) of [13], as it is central for the derivation of parameters for the LGM scheme. Li et al. present it as a special case of Theorem 2 of Agrawal et al. [2].

Theorem 2. *Let $\sigma, \epsilon \in \mathbb{R}$ be such that $\epsilon > 0$ and $\sigma > C$ for some absolute constant C (see [2]). Let $t, \sigma' \in \mathbb{R}$ be such that $t \geq 10\log(8t^{1.5}\sigma)$ and $\sigma' \geq 4t\log(1/\epsilon)$. Then the statistical difference between the following two distributions is bounded by 2ϵ.*

– *Choose a length t vector $\mathbf{x} \in \mathbb{Z}^t$ with entries chosen from the discrete Gaussian distribution on $\mathbb{Z}^t$ with parameter σ and a length t vector $\mathbf{z} \in \mathbb{Z}^t$ with entries chosen from a discrete Gaussian distribution on $\mathbb{Z}^t$ with parameter σ' and compute the output $\mathbf{x}^T\mathbf{z}$.*
– *Choose and output an element from the discrete Gaussian distribution on $\mathbb{Z}$ with parameter $\sigma\sigma'$.*

3 The LGM Scheme

The leveled homomorphic encryption scheme LGM [13] is also known as DMGSW since it uses a multi-key and dual version of GSW. We present it using mostly the original notation, but denote the secret keys as $\mathbf{s}_i = (\mathbf{r_i} \,\|\, -\mathbf{e}_i^T)^T$, as opposed to $\mathbf{e}_i = (\mathbf{I}_i \,\|\, -\mathbf{t}_i^T)^T$. Note also that we omit the details of homomorphic addition and multiplication, as they are not relevant for our attack.

Setup$(1^\kappa, 1^L)$: Let $n = n(\kappa, L)$ and $m = m(\kappa, L)$ be parameters $n < m$ that depend on the security parameter κ and number of levels L. Choose a modulus q and bounded noise distribution $\chi = \chi(\kappa, L)$ on $\mathbb{Z}$ with bound B such that it achieves at least 2^κ security against known attacks. Choose the number of secret keys $t = O(\log n)$. Let $l = \lfloor \log q \rfloor + 1$ and $N = (t+m)l$. Output $params = (n, q, \chi, m, t, l, N)$.

KeyGen$(params)$: Uniformly sample $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$. For $i \in [1, t]$, sample $\mathbf{e}_i$ from χ^m, set $\mathbf{u}_i = \mathbf{B}\mathbf{e}_i$ and set $\mathbf{s}_i = (\mathbf{r}_i \,\|\, -\mathbf{e}_i^T)^T$, where $\mathbf{r}_i$ is the i-th row of the $t \times t$ identity matrix. Return the public key $\mathbf{A} = [\mathbf{u}_1 \| \ldots \| \mathbf{u}_t \| \mathbf{B}] \in \mathbb{Z}_q^{n \times (t+m)}$ and the secret key $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_t)$.

Enc$(\mathbf{A}, \mu \in \mathbb{Z}_2)$: Let $\mathbf{G}$ be the $(t+m) \times N$ gadget matrix. Sample $\mathbf{R} \leftarrow \mathbb{Z}_q^{n \times N}$ and $\mathbf{X} \leftarrow \chi^{(t+m) \times N}$. Output $\mathbf{C} = \mu \cdot \mathbf{G} + \mathbf{A}^T\mathbf{R} + \mathbf{X} \in \mathbb{Z}_q^{(t+m) \times N}$.

Dec$(\mathbf{s}, \mathbf{C})$: Sample $(\lambda_1, \ldots, \lambda_t) \in \mathbb{Z}_q^t \setminus \{0\}^t$ until the generated $\mathbf{s}' = \sum_{i=1}^t \lambda_i \mathbf{s}_i$ has small norm. Let $i \in [1, t], j, I = (i-1)l + j$ be integers such that $\lambda_i \neq 0$, $2^{j-1} \in (q/4, q/2]$ and $I \in [1, tl]$. Compute $u = \langle \mathbf{C}_I, \mathbf{s}' \rangle \mod q$, where $\mathbf{C}_I$ is the Ith column of the ciphertext matrix $\mathbf{C}$. Finally, output $|\lfloor u/2^{j-1} \rceil| \in \{0, 1\}$.

Correct decryption of honestly generated ciphertexts follows from the following observations: first, note that $\mathbf{A}\mathbf{s}_i = 0$ for all i by construction, which ensures that $\mathbf{A}\mathbf{s}' \equiv 0 \mod q$. Then, due to $\mathbf{s}'$ being small and the choice of I, $u = \mu\mathbf{G}^T\mathbf{s}' + \mathbf{X}^T\mathbf{s}' = \mu 2^{j-1} + E$ for some small E. It is clear that the rounded division with 2^{j-1} will result in the message μ.

Li et al. mainly focus on the case where the λ_is are drawn uniformly at random from $\{0, 1\}$, but they also discuss other possible distributions to sample from, such as a larger uniform distribution or a discrete Gaussian distribution. We consider the security of the scheme in all these cases.

Deducing a message from a ciphertext boils down to solving the LWE-like instance $\mathbf{B}^T\mathbf{R} + \mathbf{X}$, whilst security against (non-adaptive) key recovery attacks is based on the ISIS problem.

The intuition behind LGM's claimed IND-CCA1 security is that since a new secret key is being used to decrypt every time the oracle is called, an adversary will be unable to deduce anything meaningful about either the summands of the key or the key itself, as she gets at most one bit of information from each decryption query, since the message space is $\mathbb{Z}_2$. Li et al. argued that any information leaked from one decryption query cannot be combined with information from another query, since the secret keys are different every time.

3.1 Parameter derivation

The authors do not suggest a concrete parameter setting for the LGM scheme; we therefore derive a realistic choice for parameters based on the information and bounds provided in [13], which we also state here.

- The parameters m, n, q and the bound B must all be chosen so that the instantiated cases of LWE and ISIS problems are hard to solve.
- The inequality $tB + mB^2 < q/8$ must be satisfied in order to prevent an erroneous decryption of a fresh (i.e., unevaluated) ciphertext.
- Li et al. suggest setting $B = 6\sigma$.
- If the distribution of the values of $\langle \mathbf{C}_I, \mathbf{s}' \rangle \mod q = \langle \mathbf{C}_I, \sum_{i=1}^{t} \lambda_i \mathbf{s}_i \rangle \mod q$ resembles a uniform distribution over $\mathbb{Z}_q$, it must be indistinguishable from a uniform distribution over $\mathbb{Z}_q$. By the leftover hash lemma, we must have $t \geq \log(q) + 3\kappa$ where κ is the security parameter of the scheme for the two distributions to be indistinguishable.
- If the distribution of the values of $\langle \mathbf{C}_I, \mathbf{s}' \rangle \mod q = \langle \mathbf{C}_I, \sum_{i=1}^{t} \lambda_i \mathbf{s}_i \rangle \mod q$ resembles a discrete Gaussian distribution over $\mathbb{Z}_q$, t and σ must satisfy the bounds of Theorem 2, i.e., $t \geq 10 \log(8t^{1.5}\sigma)$ and $\sigma \geq C$. This ensures that statistical difference of the distribution of values of $\langle \mathbf{C}_I, \mathbf{s}' \rangle \mod q$ and a discrete Gaussian with parameter $\sigma\sigma'$ is bounded by 2ϵ.

We stress that the distribution mentioned in the two final points arise naturally during decryption, and that the properties of the distribution depends on $\mathbf{C}_I$, so both points must be taken into account. We start from the final point to derive $\sigma \leq C$, for $C \geq 18K\eta_\epsilon(\mathbb{Z})$, where $K > 1$ is some universal constant and $\eta_\epsilon(\mathbb{Z}) \leq \sqrt{\ln(\epsilon/44 + 2/\epsilon)/\pi}$ is the smoothing parameter of the integers [2,18]. Setting $\epsilon = 0.005$ to provide a statistical difference of 0.01 according to Theorem 2 and assuming $K \approx 1$, we derive $\sigma \geq 25$, so we choose $\sigma = 25$. Using the other bounds, we get $t = 400$, $m = 525$, $B = 150$ and $q = 94,980,001$, which ensures a 120-bit security against currently known attacks on LWE and ISIS [3,5]. [5]

[5] Seeing as we do not use n in our attack, we do not set it explicitly. We do note, though, that it affects the hardness of the LWE instance, and is implicitly set by the requirement $m > n$. We assume $m \approx n$.

4 The Key Recovery Attack

First note that any non-zero linear combination of the secret vectors can be used to decrypt. Hence, to perform a successful key recovery attack we will only need to recover any single $\mathbf{s}_i = (\mathbf{r}_i \parallel -\mathbf{e}_i^T)^T$. We present our attack to recover the entire secret key $\{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$, by recovering the coefficients at a particular index across the t secrets $\mathbf{s}_i$, index by index. However, for the concrete experiments, we recover just one secret key.

We first assume the suggested variant of LGM where the values λ_i are chosen uniformly at random from $\mathbb{Z}_2$, and present the basic attack for this case. Afterwards we show that the attack generalises to the cases $\lambda_i \in [0, b-1]$ and $\lambda_i \in [-b, b]$, where b is some (very) small constant. This constraint on λ_i is necessary to ensure that $\|\mathbf{s}' = \sum \lambda_i \mathbf{s}_i\|$ is small, as is required by the scheme. We also discuss the security of the scheme for the case where the λ_is are sampled from a discrete Gaussian distribution.

$\boldsymbol{\lambda_i \in \{0,1\}}$. Recall that decryption works by first choosing $\lambda_1, \ldots, \lambda_t$ uniformly at random from $\{0,1\}$ and then generating a one-time decryption key $\mathbf{s}'$ as

$$\mathbf{s}' = \lambda_1 \mathbf{s}_1 + \ldots + \lambda_t \mathbf{s}_t = (\lambda_1, \ldots, \lambda_t, \sum_{i=1}^{t} \lambda_i e_{i,1}, \ldots, \sum_{i=1}^{t} \lambda_i e_{i,m}).$$

Next, a column $\mathbf{C}_I$ of the ciphertext matrix $\mathbf{C}$ is chosen, where I corresponds to some index k that satisfies $\lambda_k = 1$. Then $u = \langle \mathbf{C}_I, \mathbf{s}' \rangle \mod q$ is computed, and the decryption oracle returns $|\lfloor u/2^{j-1} \rceil|$, for the unique (and known) value j for which $q/4 < 2^{j-1} \leq q/2$.

In the following we focus on recovering the first component of each $\mathbf{e}_i$, namely the $e_{1,1}, e_{2,1}, \ldots, e_{t,1}$ that are linearly combined in position $t+1$ of $\mathbf{s}'$. The same attack can be carried out to recover all the other m positions with an easy adaptation. We construct our chosen ciphertexts from column vectors $c_{a,i}$ with some integer a in position i for $1 \leq i \leq t$, a 1 in position $t+1$ and 0 elsewhere:

$$c_{a,i} = (\underbrace{0, \ldots, a, \ldots, 0}_{\text{length } t,\, a \text{ in pos. } i}, \underbrace{1, 0, \ldots, 0}_{\text{length } m})^T.$$

Let D_α be the ciphertext matrix where for all i the column corresponding to λ_i is $c_{\alpha,i}$, and let $R_{a,i}$ be the ciphertext matrix where every column is $c_{a,i}$:

$$D_\alpha = \begin{bmatrix} \alpha & 0 & \cdots & 0 \\ 0 & \alpha & \cdots & 0 \\ 0 & 0 & \cdots & \alpha \\ 1 & 1 & \cdots & 1 \\ & & \mathbf{0}_{(m-1)\times t} & \end{bmatrix}, \quad R_{a,i} = \begin{bmatrix} & & \mathbf{0}_{(i-1)\times t} & \\ a & a & \cdots & a \\ & & \mathbf{0}_{(t-i)\times t} & \\ 1 & 1 & \cdots & 1 \\ & & \mathbf{0}_{(m-1)\times t} & \end{bmatrix}.$$

Asking for the decryption of D_α will result in the following expression for $u = u(D_\alpha)$, no matter which index k corresponds to the chosen column $\mathbf{C}_I$:

$$u(D_\alpha) = \langle c_{\alpha,i}, \mathbf{s}' \rangle = \alpha + \sum_{i=1}^{t} \lambda_i e_{i,1}.$$

This is because it is a requirement that $\lambda_k = 1$ for the chosen index I. The output of the decryption query will depend on the size of α, as well as the value of $\sum_{i=1}^{t} \lambda_i e_{i,1}$. In the attack we will only use values of α that are close to 2^{j-2}. In particular, we will always have $0 < \alpha + \sum_{i=1}^{t} \lambda_i e_{i,1} < q$, and thus will never have to consider any reductions modulo q. If $\alpha + \sum_{i=1}^{t} \lambda_i e_{i,1} < 2^{j-2}$ the decryption oracle will return 0, and if $\alpha + \sum_{i=1}^{t} \lambda_i e_{i,1} \geq 2^{j-2}$ it will return 1. Define the value $E = \sum_{i=1}^{t} \lambda_i e_{i,1}$. Asking for the decryption of D_α many times will make E a stochastic variable that takes its value according to a discrete Gaussian distribution over the interval $[E_{\min}, E_{\max}]$, where $E_{\min}$ and $E_{\max}$ are the minimum and maximum values E can take, respectively. Denoting the expected value of E by $\mathbb{E}(E)$, we get $\mathbb{E}(E) = 1/2 \sum_{i=1}^{t} e_{i,1}$. Approximately half of the time E will take a value that is smaller than $\mathbb{E}(E)$ and approximately half of the time the value of E will be greater than $\mathbb{E}(E)$.

Similarly, asking for the decryption of $R_{a,i}$ will give the following expression for $u = u(R_{a,i})$:

$$u(R_{a,i}) = \langle c_{a,i}, \mathbf{s}' \rangle = \lambda_i a + \lambda_i e_{i,1} + \sum_{k \neq i} \lambda_k e_{k,1}.$$

In this case we do not know if λ_i is 0 or 1. If $\lambda_i = 0$, the result is $u(R_{a,i}) = \sum_{k \neq i} \lambda_k e_{k,1} \ll 2^{j-2}$, and so the decryption will output 0. If $\lambda_i = 1$ the output of the decryption query will depend on the size of a and the value of the sum $\sum_{k \neq i} \lambda_k e_{k,1}$. Define E_i to be $E_i = \sum_{k \neq i} \lambda_k e_{k,1}$. In the same way as above, asking for decryptions of $R_{a,i}$ multiple times will make E_i be normally distributed over some interval, with an expected value $\mathbb{E}(E_i) = 1/2 \sum_{k \neq i} e_{k,1}$.

The main idea of the attack is to ask for many decryptions of D_α and $R_{a,i}$, and count how often the decryption oracle returns 1. Asking for sufficiently many decryptions makes the randomness of the unknown and varying λ_i's even out to their expected values. Counting how often the decryption oracle returns 1 for various values of α and a allows us to extract information about the size of $e_{i,1}$, and accurately estimate its value.

Attack procedure. For a more detailed explanation of how the attack works, we start with the following definition.

Definition 2. *Let $h(\alpha)$ be the number of times the decryption oracle returns* 1 *when asked for a number of decryptions of D_α, and let $h_i(a)$ be the number of times the decryption oracle returns* 1 *when asked for a number of decryptions of $R_{a,i}$.*

The number of times we ask for the decryption of the same ciphertext is denoted by T, and its exact value will be determined later. By doing a binary search, find the integer α_0 such that $h(\alpha_0) < T/2 \leq h(\alpha_0 + 1)$. Next, make an interpolated value α_{est} that we estimate would give exactly $T/2$ decryptions returning 1 if we were allowed to ask for decryptions of D_α for $\alpha \in \mathbb{R}$:

$$\alpha_{est} = \frac{h(\alpha_0+1) - T/2}{h(\alpha_0+1) - h(\alpha_0)}\alpha_0 + \frac{T/2 - h(\alpha_0)}{h(\alpha_0+1) - h(\alpha_0)}(\alpha_0 + 1).$$

Note that the value α_{est} is a real value, and it is our best estimate for the equation $\alpha_{est} + 1/2 \sum_{i=1}^{t} e_{i,1} = 2^{j-2}$ to hold. To be precise, we get the equation

$$\alpha_{est} + 1/2 \sum_{i=1}^{t} e_{i,1} = 2^{j-2} + \epsilon, \tag{1}$$

where $|\epsilon|$ becomes small when the sample size T grows large.

Next, we repeat the process and ask for decryptions of $R_{a,i}$ for $i = 1, \dots, t$. Note that $\lambda_i = 0$ half of the time in these decryptions, which always causes the oracle to return the value 0. So the values $h_i(a)$ will be approximately half of $h(a)$. This is compensated for by finding the value a_0 such that $2h_i(a_0) < T/2 \leq 2h_i(a_0 + 1)$. Knowing that $\lambda_i = 1$ whenever we get a 1-decryption, we do the same interpolation as above and find an estimate $a_{est} \in \mathbb{R}$ such that

$$a_{est} + e_{i,1} + 1/2 \sum_{k \neq i} e_{k,1} = 2^{j-2} + \epsilon_i, \tag{2}$$

where $|\epsilon_i|$ is small. Subtracting (2) from (1) and rearranging we get

$$e_{i,1} = 2(\alpha_{est} - a_{est}) + 2(\epsilon_i - \epsilon). \tag{3}$$

Rounding the right-hand side value recovers the correct $e_{i,1}$, provided that T is large enough to make $|\epsilon_i| < 1/8$ and $|\epsilon| < 1/8$.

The attack can be repeated to recover all the $e_{i,x}$ for $x = 2, 3, \dots, m$ by setting the 1 in $c_{a,i}$ to be in position $t + x$. One can also focus on fully recovering only one of the vectors $\mathbf{e}_i$ by recovering $e_{i,x}$ for some fixed i and $x = 1, 2, \dots, m$. Note that the recovery of an entry of any $\mathbf{e}_i$-vector is independent of the recovery of any other entry. This is what enables us to recover a single $\mathbf{e}_i$-vector in its entirety, which can be used as a decryption key.

For the attack to work with high probability, we need $T \in O(t \cdot \sigma^2)$. In particular, we have the following:

Lemma 3. *If $T \geq 800 \cdot t \cdot \sigma^2$ then in Eq.* (1) *we have that $Pr[|\epsilon| \geq 1/8] \leq 2^{-20}$ and in Eq.* (2) *we have that $Pr[|\epsilon_i| \geq 1/8] \leq 2^{-20}$.*

Proof. Since λ_i is taken from the uniform distribution over $\{0, 1\}$ it has mean $1/2$ and variance $1/4$. Moreover, $e_{i,1}$ is taken from a Gaussian distribution with mean 0 and variance σ^2. Then $\lambda_i \cdot e_{i,1}$ is taken from a Gaussian distribution with mean $1/2 \cdot \sum e_{i,1}$ and variance $1/2 \cdot \sigma^2$. Hence $\epsilon = \sum_{i=1}^{t} \lambda_i \cdot e_{i,1} - \mathbb{E}(E)$ is a sum of Gaussians, which by Lemma 1 is also a Gaussian with mean 0 and variance $t/2 \cdot \sigma^2$. However, if we take an average of T samples of such a function, then by the central limit theorem for sample means we get a Gaussian X with mean 0 and variance $\frac{t}{2T} \cdot \sigma^2$. If $T \geq 800 \cdot t \cdot \sigma^2$ then X has standard deviation $\leq 1/40$, in which case $Pr[|\epsilon| \geq 1/8] \leq 2^{-20}$ by Lemma 2. Similarly, $Pr[|\epsilon_i| \geq 1/8] \leq 2^{-20}$. $\square$

Remark 1. We cannot get a smaller lower bound for T using Lemma 8.1 of [1] since there is no guarantee that the Gaussian X in the above proof is integral.

The running time will then be $O(Tm) = O(t\sigma^2 m)$. We have that t can only be polynomially large in the security parameter to have efficient encryption. Also, $\|\mathbf{s}'\|$ must be small in order for the underlying ISIS problem to be hard. Therefore, the attack runs in polynomial time.

4.1 Generalisation of the attack

The above attack assumes that the values λ_i were sampled uniformly at random from $\{0, 1\}$. We now investigate whether the attack can be prevented by choosing the λ_i from a larger set. The two generalisations we consider are $\lambda_i \in \{0, 1, \dots, b-1\}$, or $\lambda_i \in \{-b, \dots, b\}$. As before, we can take $T \in O(t \cdot \sigma^2)$. We show that the attack can be generalised to work in both cases.

$\boldsymbol{\lambda_i \in \{0, 1, \dots, b-1\}}$**.** The attack can be adapted to work when the λ_i are sampled uniformly at random from $\{0, 1, \dots, b-1\}$. We again focus on recovering the coefficients $e_{i,1}$, the other $e_{i,x}$'s are recovered by repeating the attack with the same adaptation as above. This also means that we may choose to recover a single $\mathbf{e}_i$-vector here as well.

When the decryption oracle is given the ciphertext matrix D_α, it will compute $u(D_\alpha) = \lambda_k \alpha + \sum_{i=1}^{t} \lambda_i e_{i,1}$, where $\lambda_k \neq 0$. When $\alpha \approx 2^{j-2}/(b-1)$, the decryption oracle will return 0 whenever $\lambda_k < b-1$, since $\lambda_k \alpha + \sum_{i=1}^{t} \lambda_i e_{i,1} < 2^{j-2}$ in this case. Only when $\lambda_k = b-1$ can we get decryptions that return 1. We know that $\lambda_k \neq 0$ when decrypting D_α, so the probability that $\lambda_k = b-1$ is $1/(b-1)$.

As before, we scale the numbers $h(\alpha)$ with $b-1$ to do a binary search and find the value α_0 such that $(b-1)h(\alpha_0) < T/2 \leq (b-1)h(\alpha_0+1)$. We then use this to estimate the α_{est} for which we would expect $(b-1)h(\alpha_{est}) = T/2$ if we were allowed to ask for decryptions of D_α where $\alpha \in \mathbb{R}$.

When $\lambda_i \in \{0, 1, \dots, b-1\}$, the expected value of $\sum_{i=1}^{t} \lambda_i e_{i,1}$ is $\frac{b-1}{2} \sum_{i=1}^{t} e_{i,1}$. The α_{est} we find therefore gives the equation

$$(b-1)\alpha_{est} + \frac{b-1}{2} \sum_{i=1}^{t} e_{i,1} = 2^{j-2} + \epsilon, \tag{4}$$

where $|\epsilon|$ is small for large T.

In the same fashion we can ask for decryptions of $R_{a,i}$ where $a \approx 2^{j-2}/(b-1)$ and count the number of 1-decryptions we get. When decrypting $R_{a,i}$ we may have $\lambda_i = 0$, so the probability that $\lambda_i = b-1$ (which is necessary for the decryption oracle to return 1) is $1/b$. We therefore scale the values of $h_i(a)$

by b, and find an interpolated value for a_{est} based on the value a_0 for which $bh_i(a_0) < T/2 \leq bh_i(a_0+1)$. This yields the equation

$$(b-1)a_{est} + (b-1)e_{i,1} + \frac{b-1}{2}\sum_{k\neq i}^{t} e_{k,1} = 2^{j-2} + \epsilon_i, \tag{5}$$

where $|\epsilon_i|$ is small. Subtracting (5) from (4) gives

$$e_{i,1} = 2(\alpha_{est} - a_{est} + \frac{\epsilon_i - \epsilon}{b-1}),$$

and rounding this value recovers the correct $e_{i,1}$, provided $|\epsilon_i| < (b-1)/8$ and $|\epsilon| < (b-1)/8$.

The proof of the following lemma is almost identical to Lemma 3 and is thus omitted. In fact, the upper bound for $Pr[|\epsilon_i| \geq (b-1)/8]$ and $Pr[|\epsilon| \geq (b-1)/8]$ will be even smaller than 2^{-20} for $b > 2$.

Lemma 4. *If $T \geq 800 \cdot t \cdot \sigma^2$ then in Eq.* (4) *we have that $Pr[|\epsilon| \geq 1/8] \leq 2^{-20}$ and in Eq.* (5) *we have that $Pr[|\epsilon_i| \geq 1/8] \leq 2^{-20}$.*

$\boldsymbol{\lambda \in \{-b, \dots, b\}}$. We start by asking for T decryptions of D_α, where $\alpha \approx 2^{j-2}/b$, and count how often the decryption oracle returns 1. Recall that the decryption outputs the absolute value of $\lfloor u/2^{j-1} \rceil$, so there are now two cases where the decryption oracle can return 1, namely when $\lambda_i = b$ or $\lambda_i = -b$. There are $2b+1$ numbers in $\{-b, \dots, b\}$, but 0 cannot be chosen for λ_k when decrypting D_α, so the probability of having λ_k equal to $-b$ or b is $2/2b = 1/b$. We scale the numbers $h(\alpha)$ by a factor b to compensate for this. We then interpolate like before to find the value α_{est} such that we would expect $h(\alpha_{est}) = T/2$ if we were allowed to ask for decryptions of $D_{\alpha_{est}}$ for $\alpha \in \mathbb{R}$. When the set of values that λ_i can take is symmetric around 0, the expected value of $\sum \lambda_i e_{i,1}$ is 0. The equation we get for α_{est} is then simplified to

$$b\alpha_{est} = 2^{j-2} + \epsilon, \tag{6}$$

where $|\epsilon|$ is small. Note that we do not need to distinguish between the cases $\lambda_k = -b$ and $\lambda_k = b$, as this is incorporated in the probability $2/2b$ for having the possibility of 1-decryption. So the α_{est} we find covers both the cases $-b\alpha < -2^{j-2}$ and $b\alpha > 2^{j-2}$, which both result in 1-decryptions.

When decrypting $R_{a,i}$ we can have $\lambda_i = 0$, so the probability of $\lambda_i = -b$ or $\lambda_i = b$ is then $2/(2b+1)$. We ask T times for decryptions of $R_{a,i}$, and as before find the value a_{est} that is the best estimate for $\frac{2b+1}{2}h_i(a_{est}) = T/2$. We then get the equation

$$ba_{est} + be_{i,1} = 2^{j-2} + \epsilon_i, \tag{7}$$

where $|\epsilon_i|$ is small. Subtracting (7) from (6) gives us

$$e_{i,1} = \alpha_{est} - a_{est} + \frac{\epsilon_i - \epsilon}{b}.$$

Rounding this value to the nearest integer recovers the correct $e_{i,1}$, provided T is large enough to make $|\epsilon_i| < b/4$ and $|\epsilon| < b/4$.

The proof of the following lemma is almost identical to Lemma 3 and is thus omitted. In fact, the upper bound for $Pr[|\epsilon_i| \geq b/4]$ and $Pr[|\epsilon| \geq b/4]$ will be even smaller than 2^{-20}.

Lemma 5. *If $T \geq 800 \cdot t \cdot \sigma^2$ then in Eq.* (6) *we have that $Pr[|\epsilon| \geq 1/8] \leq 2^{-20}$ and in Eq.* (7) *we have that $Pr[|\epsilon_i| \geq 1/8] \leq 2^{-20}$.*

4.2 Implementation of the attack

We have implemented the attack and verified that it works as explained. The code for the attack was written in C, and can be found at [17]. The secret $\mathbf{e}_i$-vectors were sampled using the DGS library [4].

For testing the attack we have used a Dell server with 75 CPU cores (AMD Epyc 7451). We ran the attack twice, using the parameter sets deduced in Section 3.1. For the first attack we used $t = 190, m = 525, b = 2$, and $\sigma = 25$, which gives the sample size $T = 95,000,000$ according to Lemma 3. For the second attack we used $t = 400, m = 525, b = 2$, and $\sigma = 25$, for which Lemma 3 gives the sample size $T = 200,000,000$. In both attacks we drew the λ_i's uniformly from $\{0, 1\}$ and aimed to recover all the 525 coefficients of $\mathbf{e}_2$[6].

In the attack on the $t = 190$ case, 519 of the 525 coefficients were recovered correctly. The six coefficients that were wrong all had a difference 1 with the correct value. In the attack on the $t = 400$ case, all 525 coefficients of $\mathbf{e}_2$ were recovered correctly.

The run time for the first attack was approximately 12 hours, and for the second attack approximately 48 hours, both using 75 CPU cores in parallel. However, the code can be optimised in several ways to reduce the run time. In particular, it is possible to abort early and not do all T decryptions when it is clear that $h(\alpha)$ or $h_i(a)$ will be much greater or smaller than $T/2$. We did not implement this optimisation, and computed all T decryptions every time.

Our attack does not necessarily recover a secret key flawlessly, as demonstrated above with the $t = 190$ case, where 6 out of 525 estimated coefficients were either -1 or 1 off from their true value. In these cases, we need a second phase to recover the entire secret key. It is straightforward to check whether such a second phase is necessary, as we may simply check if $\mathbf{A}\tilde{\mathbf{s}_i} = 0$, for an estimated secret key $\tilde{\mathbf{s}_i} = (\mathbf{r}_i, -\tilde{\mathbf{e}_i})^T$, where $\tilde{\mathbf{e}_i}$ is the estimate of $\mathbf{e}_i$ we get by running the attack. If $\mathbf{A}\tilde{\mathbf{s}_i} \neq 0$, there is at least one wrong entry of $\tilde{\mathbf{e}_i}$, implying $\mathbf{e}_i = \tilde{\mathbf{e}_i} + \epsilon$, where ϵ is a non-zero vector sampled from a Gaussian distribution with mean 0 and a very small standard deviation. Recall that the public key is $\mathbf{A} = [\mathbf{u}_1 \parallel \ldots \parallel \mathbf{u}_t \parallel \mathbf{B}]$, where $\mathbf{u}_i = \mathbf{B}\mathbf{e}_i$, so we can calculate $\mathbf{B}\epsilon = \mathbf{B}\tilde{\mathbf{e}_i} - \mathbf{u}_i$. Now, $\mathbf{B}\epsilon$ may be described as n highly unbalanced instantiations of the knapsack problem: only approximately 1% of the coefficients of ϵ are either -1 or 1. These instantiations are much simpler to solve than the standard knapsack

[6] We chose $\mathbf{e}_2$ arbitrarily; the attack works to recover any $\mathbf{e}_i$, $i \in \{1, \ldots, t\}$.

problem: one estimation for the time complexity for the $\mathbf{B}\epsilon$ case is $\tilde{O}(2^{0.03n})$[6], another is $\tilde{O}(2^{0.0473n})$[12], though the first algorithm does not guarantee finding the solution. Even though the memory requirement is higher in either case, the (potential) second phase of the attack does not contribute in any substantial way to the cost of the key recovery attack, and ensures that the secret key is completely recovered.

4.3 λ_i drawn from a non-uniform distribution

Whilst the authors of LGM mainly focus on the $\lambda_i \in \{0, 1\}$ case, they also discuss other distributions it would be possible to sample from, e.g., other uniform distributions, or a discrete Gaussian distribution. As shown above, sampling λ_i from other uniform distributions does not prevent our attack, however, a line of argument in the LGM paper suggests that a particular choice of a discrete Gaussian distribution might.[7]

The argument is as follows: if the values of $\langle \mathbf{C}_I, \sum_{i=1}^{t} \mathbf{s}_i \rangle$ resemble samples from a discrete Gaussian distribution, and the standard deviation σ' of the λ-distribution satisfies the condition of Theorem 2, the theorem itself is applicable to the distribution of the values of $\langle \mathbf{C}_I, \sum_{i=1}^{t} \lambda_i \mathbf{s}_i \rangle$. Then, seeing as $\langle \mathbf{C}_I, \sum_{i=1}^{t} \lambda_i \mathbf{s}_i \rangle$ is statistically close to a 'regular' discrete Gaussian distribution with standard deviation $\sigma\sigma'$ by Theorem 2, the result from the decryption oracle cannot leak any information about the secret key.

A rough estimate for the parameters required to achieve 120-bit security in these cases is: $t = 400, \sigma = 25, B = 150, \sigma' = 12,231, m = 940$, and q is a 40-bit number. These parameters prevent any practical use of the system, especially given the fact that the scheme encrypts a single bit at a time.

But that aside, could the scheme with this λ-distribution be regarded as a theoretical construct to demonstrate that IND-CCA1 security is achievable for homomorphic encryption schemes? We argue that the answer is no. Even if the λ-distribution is chosen according to Theorem 2, the theorem only guarantees that the distribution over $\langle \mathbf{C}_I, \sum_{i=1}^{t} \lambda_i \mathbf{s}_i \rangle$ is statistically close to a discrete Gaussian distribution *if* the matrix column $\mathbf{C}_I$ is such that the values of $\langle \mathbf{C}_I, \sum_{i=1}^{t} \mathbf{s}_i \rangle$ appear to be drawn from a discrete Gaussian distribution themselves.

We stress that the choice of $\mathbf{C}_I$ is *entirely* up to the adversary in an IND-CCA1 game, as she can simply submit a ciphertext matrix where every column is $\mathbf{C}_I$. It is therefore feasible for her to submit a $\mathbf{C}_I$ such that the values of $\langle \mathbf{C}_I, \sum_{i=1}^{t} \mathbf{s}_i \rangle$ do not appear to be drawn from a discrete Gaussian distribution, meaning Theorem 2 does not apply. It is therefore not possible to positively conclude that no useful information is leaked by a decryption query, even if it merely results in an adversary obtaining a non-negligible advantage in the IND-CCA1 game, and not a complete recovery of the secret key. Furthermore, the adversary can adapt her choice of $\mathbf{C}_I$, whilst the choice of the distribution from which the λ_is are drawn is fixed when the system is generated. The λ-distribution therefore cannot be constructed to fit both the situation where $\mathbf{C}_I$ is designed

[7] See discussion in Section 7 of [13].

to provide values of $\langle \mathbf{C}_I, \sum_{i=1}^{t} \mathbf{s}_i \rangle$ seemingly drawn from a discrete Gaussian distribution and when it is designed to *not* provide such values.

4.4 Thwarting the attack

We discuss some possible ideas to prevent the key recovery attack described above, and argue that they will not work.

Decryption oracle uses the same $(\boldsymbol{\lambda_1, \ldots, \lambda_t})$ for the same ciphertext. One can ensure that an attacker that queries the same ciphertext C (say, $C = D_\alpha$ as defined previously) multiple times will have the same set of λ-values chosen in every decryption. It will then be impossible to do the attack we presented, as it relies on having random and independent λ-values chosen for each query. This can be done by setting $(\lambda_1, \ldots, \lambda_t) = \mathsf{PRF}(C)$ for some pseudo-random function PRF that returns a vector of small values.

To circumvent this measure, the attacker can add a few 1's to the large part of the top t rows of D_α or $R_{a,i}$ that are defined to be 0. The number of 0's in this part of D_α or $R_{a,i}$ is $t(t-1)$. So if the attacker intends to ask for T decryptions of the same matrix, she can make the matrices unique by adding up to ρ 1's in all possible ways in the top t rows. The computation in the decryption will still be approximately the same. The largest number ρ of 1's that must be added in this way is the smallest integer that satisfies

$$\sum_{i=0}^{\rho} \binom{t(t-1)}{i} \geq T.$$

For the parameters used in our attack ($t = 190, T = 200.000.000$) this is satisfied already for $\rho = 2$.

When adding two 1's to D_α or $R_{a,i}$, the estimates in Eq. (1) and Eq. (2) will be disturbed by an extra 1 in approximately $(2/t) \cdot (1/2) = 1/t$ of the queries (the chosen column contains an extra 1 with probability $2/t$, and the λ_j-value it meets in the inner product will also be 1 with probability $1/2$). This error can be compensated for in the estimation of α_{est} and a_{est} by adding an extra term to the equations in Eq. (1) and Eq. (2), but both values will be the same and anyway cancel out in Eq. (3). Hence the attacker can overcome such a countermeasure.

Repeat multiple decryptions and return a value only if they are consistent. Alternatively, one can define a new decryption function which runs the original decryption function ℓ times and return bit b only if all ℓ evaluations return b, and abort (i.e., return $\perp$) if they are not all equal. However, note that we now have 3 return values $(0, 1, \perp)$ instead of 2, and can compute the expected value of each return value for every α similar to before.

For instance, consider querying D_α to the new decryption oracle, where α is a value such that the original decryption oracle would return 0 with probability p, and 1 with probability $1 - p$. Then the new decryption oracle aborts with probability $1 - (p^\ell + (1-p)^\ell)$, which achieves maximum at $p = 1/2$. Hence to detect an optimal α_{est} as in Eq. (1), one asks for T decryptions of $D_{\alpha_{est}}$ and makes sure $\approx (1-2^{1-\ell}) \cdot T$ of them abort. Note that the attack strategy fails if ℓ is sufficiently large (e.g., $\ell \in \Theta(\kappa)$), but a large choice of ℓ also severely restricts the LGM scheme's level of homomorphism, since a noisy ciphertext obtained from a homomorphic evaluation would also fail (i.e., return $\perp$ or the wrong bit) with non-negligible probability in the new decryption function.

Ciphertext checks. A plausible strategy to thwart our attack would be to add a ciphertext check during decryption, to ensure that the ciphertext to be decrypted has been honestly generated. Using a ciphertext check, Loftus et al. [15] constructed an SHE scheme that provably achieved IND-CCA1 security, although the underlying hardness assumption was later shown to be insecure; see discussion in [13] and the references therein. If such a ciphertext check is added to the decryption procedure, maliciously generated ciphertexts may simply be rejected by the decryption oracle, which will make it impossible to mount our attack. As illustrated by the previous idea, it is far from clear how to successfully add an efficient ciphertext check to the LGM scheme.

We argue that in general any such ciphertext check that uses the same secret key value both to check ciphertexts are well-formed and to decrypt will naturally give some information about the secret key. One can instead have two secret key values as in the CCA1-secure group homomorphic encryption scheme CS-lite [9], where the first value is used only for checking ciphertexts are well-formed, while the second value ensures indistinguishability even if the first value is revealed. We leave as an open problem how such a method can work with LGM or other homomorphic encryption schemes.

5 Conclusion

We have shown that the LGM scheme is susceptible to an adaptive key recovery attack, disproving the authors' claim that the scheme achieves IND-CCA1 security. The attack is practical for λ_i's drawn uniformly from $\{0, 1\}$, and is still practical and efficient for λ_i's drawn uniformly from a larger set of integers. We have also argued that the scheme is not secure even if the λ_i's are drawn from a discrete Gaussian distribution. In short, none of the distributions suggested by Li et al. ensures the IND-CCA1 security of the LGM scheme.

A plausible strategy to thwart our attack would be to add a ciphertext check during decryption, but we do not know if the strategy can be applied to the LGM scheme, and we know of no other strategies that may be applicable to the scheme to achieve IND-CCA1 security. We therefore do not know how to tweak the LGM scheme to be resistant to our proposed statistical attack.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (May / Jun 2010)
2. Agrawal, S., Gentry, C., Halevi, S., Sahai, A.: Discrete Gaussian leftover hash lemma over infinite domains. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 97–116. Springer, Heidelberg (Dec 2013)
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015), `http://eprint.iacr.org/2015/046`
4. Albrecht, M.R., Walter, M.: dgs, Discrete Gaussians over the Integers (2018), available at `https://bitbucket.org/malb/dgs`
5. Bai, S., Galbraith, S.D., Li, L., Sheffield, D.: Improved combinatorial algorithms for the inhomogeneous short integer solution problem. Journal of Cryptology 32(1), 35–83 (Jan 2019)
6. Becker, A., Coron, J.S., Joux, A.: Improved generic algorithms for hard knapsacks. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 364–385. Springer, Heidelberg (May 2011)
7. Canetti, R., Raghuraman, S., Richelson, S., Vaikuntanathan, V.: Chosen-ciphertext secure fully homomorphic encryption. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 213–240. Springer, Heidelberg (Mar 2017)
8. Chenal, M., Tang, Q.: On key recovery attacks against existing somewhat homomorphic encryption schemes. In: Aranha, D.F., Menezes, A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 239–258. Springer, Heidelberg (Sep 2015)
9. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (Aug 1998)
10. Dahab, R., Galbraith, S., Morais, E.: Adaptive key recovery attacks on NTRU-based somewhat homomorphic encryption schemes. In: Lehmann, A., Wolf, S. (eds.) ICITS 15. LNCS, vol. 9063, pp. 283–296. Springer, Heidelberg (May 2015)
11. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (Aug 2013)
12. Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 235–256. Springer, Heidelberg (May / Jun 2010)
13. Li, Z., Galbraith, S.D., Ma, C.: Preventing adaptive key recovery attacks on the gentry-sahai-waters leveled homomorphic encryption scheme. Cryptology ePrint Archive, Report 2016/1146 (2016), `http://eprint.iacr.org/2016/1146`
14. Li, Z., Galbraith, S.D., Ma, C.: Preventing adaptive key recovery attacks on the GSW levelled homomorphic encryption scheme. In: Chen, L., Han, J. (eds.) ProvSec 2016. LNCS, vol. 10005, pp. 373–383. Springer, Heidelberg (Nov 2016)
15. Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On CCA-secure somewhat homomorphic encryption. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 55–72. Springer, Heidelberg (Aug 2012)
16. Micciancio, D., Walter, M.: Gaussian sampling over the integers: Efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 455–485. Springer, Heidelberg (Aug 2017)

17. Raddum, H., Fauzi, P.: LGM-attack (2021), available at `https://github.com/Simula-UiB/LGM-attack`
18. Zheng, Z., Xu, G., Zhao, C.: Discrete Gaussian measures and new bounds of the smoothing parameter for lattices. Cryptology ePrint Archive, Report 2018/786 (2018), `https://eprint.iacr.org/2018/786`