WIDESEAS: A lattice-based PIR scheme implemented in EncryptedQuery

Dams, Dominic[*]

q2dominic@gmail.com

Lataille, Jeff[†]

jeff.lataille@envieta.com

Wade, John[‡]

john.wade@envieta.com

July 22, 2019

Abstract

We introduce the WIDESEAS protocol for lattice-based Private Information Retrieval (PIR), and we give performance numbers for its recent implementation in the EncryptedQuery open-source PIR software. This protocol uses the fully homomorphic Brakerski–Fan–Vercauteren (BFV) encryption scheme, as opposed to the Paillier scheme, which is used in all earlier versions of EncryptedQuery. We show that the homomorphic capabilities of BFV result in smaller query sizes (due to a query-shrinking technique based on batching and ciphertext multiplication), higher response generation rates (due to using relinearization to keep ciphertexts small; due to caching certain products of query elements in the NTT domain; due to using the distributive law to achieve a quadratic reduction in the total number of ciphertext multiplications; due to using lazy reduction to speed up modular multiplies; and, due to exploiting properties of inverse NTTs over periodic data, and forward NTTs over sparse data, for the purpose of accelerating plain multiplications), and comparable response sizes (due to using modulus switching to discard redundant

---

[*] Work completed while working as an intern at Envieta Systems, LLC.

[†] Senior Mathematician, Envieta Systems, LLC.

[‡] Senior Mathematician, Envieta Systems, LLC.

ciphertext information prior to transmitting the response). For instance, running a single thread (with Turbo Boost disabled) on a MacBook Pro equipped with a 2.8 GHz Intel core i7, and using a 20-bit hash and a 2^{15}-byte data chunk size (which allows us to search for a single targeted selector), our implementation can (i) generate a query of size 64 MiB in around 0.41 seconds, (ii) process a query against a 1 TiB database (comprised of 2^{20}-many 1 MiB records) at a rate of 23.67 MiB/s (which is at least two orders of magnitude faster than the Paillier-based version of EncryptedQuery), and (iii) generate a response of size 4 MiB in around 0.51 days. We expect a speed up on server class machines. Our implementation uses the Microsoft SEAL library along with a small amount of custom code.

Contents

1 Introduction 5

2 Detailed description of BFV 6
2.1 About SEAL . . . 7
2.2 Basic mathematical setup . . . 7
2.2.1 Algebra . . . 7
2.2.2 Probability . . . 10
2.3 Key generation . . . 12
2.3.1 On measurements . . . 14
2.4 Encryption . . . 15
2.5 Decryption . . . 16
2.5.1 Decryption of fresh ciphertexts . . . 21
2.6 Noise under homomorphic operations . . . 26
2.6.1 Ciphertext addition . . . 28
2.6.2 Plaintext multiplication . . . 33
2.6.3 Ciphertext multiplication . . . 36
2.6.4 Relinearization . . . 41
2.7 Modulus switching . . . 43
2.8 Batching . . . 45

3 WIDESEAS protocol 46
3.1 Naive approach . . . 46
3.2 Shrinking the query . . . 47
3.3 Concrete system parameters . . . 49
3.4 Noise growth . . . 51
3.5 Query size . . . 51
3.6 Response size . . . 52

4 Homomorphic optimizations 54
4.1 Caching subproducts . . . 55
4.2 Summing products . . . 56
4.3 NTT caching . . . 56
4.4 Lazy reduction . . . 57

4.5 Periodic NTT's 57

5 Performance numbers 59

5.1 A comparison with SEAL PIR 60

6 Conclusion 65

A Group algebras and Fourier analysis 65

B Fourier transform 66

C NTT's as twisted Fourier transforms 72

1 Introduction

A Private Information Retrieval (PIR) protocol allows a client to retrieve items from a database without revealing either the query term or the retrieved items. The database is assumed to be public, and there is no restriction that the client only learn the specific item. For example, the trivial PIR protocol is to download the entire database and then perform the search locally. A non-trivial PIR protocol aims to reduce the communication cost in terms of the data transmitted and received, as well as to reduce the computational costs for both the client and the database server.

A PIR protocol can be divided into three phases between two parties, a Querier and a Responder. In the first phase, called **query generation**, the Querier generates a private key and a query for a specific list of query terms, or (targeted) selectors. The Querier then transmits the query to the Responder. In the second phase, called **response generation**, the Responder reads items from the database and generates the response using the query. The Responder then transmits the response back to the Querier. In the third phase, called **response processing**, the Querier decrypts the response and extracts the underlying plaintext values. The computational and communication costs of a PIR protocol can be evaluated according to the following dimensions:

1. **Query Term Batching**: The number of query terms that can be processed in one query.
2. **Query Generation Time**: The per-core time to generate a query; this is closely related to the encryption speed of the underlying cryptosystem.
3. **Query Size**: The size of the query (in bytes, say).
4. **Response Processing Rate**: The per-core rate (in bytes per second, say) at which the database items are processed into the response.
5. **Response Size**: The size of the response (in bytes, say).
6. **Reponse Decryption Time**: The per-core time to decrypt the response; this is closely related to the decryption speed of the underlying cryptosystem.

In this report, we introduce the WIDESEAS protocol for lattice-based Private Information Retrieval (PIR), and we give performance numbers for its recent implementation in the EncryptedQuery open-source PIR software[1]. This protocol is a lattice-based variant of the WIDESKIES protocol [1] that is used in all earlier versions of EncryptedQuery. The WIDESEAS protocol uses the fully homomorphic Brakerski–Fan–Vercauteren (BFV) encryption scheme, whereas the WIDESKIES protocol uses the Paillier scheme [2]. We show that the homomorphic capabilities of BFV result in smaller query sizes (due to a query-shrinking technique based on batching and ciphertext multiplication), higher response generation rates (due to using relinearization to keep ciphertexts small; due to caching certain products of query elements in the NTT domain; due to using the distributive law to achieve a quadratic reduction in the total number of ciphertext multiplications; due to using lazy reduction to speed up modular multiplies; and, due to exploiting properties of inverse NTTs over periodic data, and forward NTTs over sparse data, for the purpose of accelerating plain multiplications), and comparable response sizes (due to using modulus switching to discard redundant ciphertext information prior to transmitting the response). For instance, running a single thread (with Turbo Boost disabled) on a MacBook Pro equipped with a 2.8 GHz Intel core i7, and using a 20-bit hash and a 2^{15}-byte data chunk size (which allows us to search for a single targeted selector), our implementation can (i) generate a query of size 64 MiB in around 0.41 seconds, (ii) process a query against a 1 TiB database (comprised of 2^{20}-many 1 MiB records) at a rate of 23.67 MiB/s (which is at least two orders of magnitude faster than the Paillier-based version of EncryptedQuery), and (iii) generate a response of size 4 MiB in around 0.51 days. We expect a speed up on server class machines. Our implementation uses the Microsoft SEAL library[2] along with a small amount of custom code.

---

[1] `https://enquery.net/encryptedquery`

This report is organized as follows. In §2, we describe the BFV fully homomorphic cryptosystem as implemented in Microsoft's SEAL library. The operations required for the WIDESEAS protocol are emphasized. In §3, we provide a high-level summary of the WIDESEAS protocol for PIR. We begin by presenting a basic (naive) approach, and then we describe various enhancements. We also provide a concrete parameter set, and we analyze noise growth, query size and response size with respect to this set. In §4, we discuss certain optimizations we have made to the basic WIDESEAS computation described in §3. These optimizations, all of which are implemented in EncryptedQuery, do not change the underlying algorithm, but they greatly improve its performance. Finally, in §5, we present various performance measurements. These measurements were done using gcc-7 on an Oracle VM VirtualBox running Ubuntu 18.04.1 LTS (Bionic Beaver) on a MacBook Pro laptop running MacOS Sierra using a 2.8 GHz Intel core i7 and with Turbo Boost disabled.

2 Detailed description of BFV

This section describes the Brakerski–Fan–Vercauteren (BFV) fully homomorphic cryptosystem as implemented in Microsoft's SEAL library. The operations required for the WIDESEAS protocol are emphasized. See Fan–Vercauteren [3] or the SEAL manual [4] for more details. The scheme uses the following parameters, which will be explained below:

---

[2] `https://www.microsoft.com/en-us/research/project/microsoft-seal/`

$d = 2^k$	degree of ring extension, with $k \geq 0$
q	ciphertext modulus
t	plaintext modulus, with $2 \leq t < q$

Table 1: BFV Parameters

In the WIDESEAS protocol, t is prime, while q is square-free with four prime divisors. Moreover, both t and q are odd. Therefore, in the sequel, **we shall always assume that both t and q are odd**. This will somewhat simplify the analysis.

2.1 About SEAL

The Simple Encrypted Arithmetic Library (SEAL) is an easy-to-use open-source homomorphic encryption library developed by researchers in the Cryptography Research Group at Microsoft Research. SEAL is written in standard C++. It has no external dependencies, so it is easy to compile in many different environments. SEAL is licensed under the MIT license.

2.2 Basic mathematical setup

2.2.1 Algebra

Let $\Phi_{2d}(x) = x^d + 1$ be the $2d$-th cyclotomic polynomial. The central mathematical object in the BFV cryptosystem is the ring of integers

$$\mathcal{O} := \mathbb{Z}[x]/\Phi_{2d}(x)\mathbb{Z}[x]$$

in the $2d$-th cyclotomic field extension $\mathbb{Q} \otimes_{\mathbb{Z}} \mathcal{O}$ of $\mathbb{Q}$. We shall sometimes identify elements of $\mathcal{O}$ with their representatives in $\mathbb{Z}[x]$ of least degree.

Let n be a positive integer. By extension of scalars, the $\mathbb{Z}/n\mathbb{Z}$-algebra

$$\mathcal{O}_n := (\mathbb{Z}/n\mathbb{Z}) \otimes_{\mathbb{Z}} \mathcal{O}$$

is the reduction modulo n of $\mathcal{O}$. The natural projection

$$p_n : \mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$$

induces a natural projection

$$\pi_n : \mathcal{O} \to \mathcal{O}_n,$$

which, in turn, induces a natural projection

$$P_n : \mathcal{O}[X] \to \mathcal{O}_n[X],$$

where X is a formal variable distinct from x. In case n is a prime number, and, therefore, $\mathbb{Z}/n\mathbb{Z}$ is a (finite) field, we sometimes write $\mathbb{F}_n$ for $\mathbb{Z}/n\mathbb{Z}$, in which case $\mathcal{O}_n = \mathbb{F}_n \otimes_{\mathbb{Z}} \mathcal{O}$. The map p_n has a unique set-theoretic section

$$s_n : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}$$

with image $\big[-\lfloor n/2\rfloor, \lfloor (n-1)/2\rfloor\big]$. This map induces a set-theoretic section

$$\sigma_n : \mathcal{O}_n \to \mathcal{O}$$

for the projection π_n, which, in turn, induces a set-theoretic section

$$S_n : \mathcal{O}_n[X] \to \mathcal{O}[X]$$

for the projection P_n. The image of σ_n consists precisely of those elements of $\mathcal{O}$ represented in $\mathbb{Z}[x]$ by polynomials over Im s_n, that is to say,

$$\text{Im } \sigma_n = \left(s_n\left(\mathbb{Z}/n\mathbb{Z}\right)[x] + \Phi_{2d}(x)\mathbb{Z}[x]\right)/\Phi_{2d}(x)\mathbb{Z}[x].$$

Similarly,

$$\text{Im } S_n = \sigma_n\left(\mathcal{O}_n\right)[X].$$

Put

$$t_n := s_n \circ p_n : \mathbb{Z} \to \text{Im } s_n,$$

$$\tau_n := \sigma_n \circ \pi_n : \mathcal{O} \to \text{Im } \sigma_n,$$

and

$$T_n := S_n \circ P_n : \mathcal{O}[X] \to \text{Im } S_n.$$

As $p_n \circ s_n = 1_{\mathbb{Z}/n\mathbb{Z}}$, it follows that each of t_n, τ_n and T_n is idempotent. Lest the reader suppose these maps are homomorphisms, we note the following results:

Lemma 1. *Let n be a positive integer. Regarding the maps t_n, τ_n, and T_n, the following are true:*

1. *They fail to commute with their respective addition operations.*
2. *They commute with their respective point-wise multiplication operations, if, and only if, $n \in \{1, 2, 3\}$.*

Proof. In order to prove the first claim above, we begin by observing that $t_n(n) = 0$ and $nt_n(1) = n$. As $0 \neq n$ in $\mathbb{Z}$, the result for t_n follows. The corresponding results for τ_n and T_n then follow by extension.

The second claim follows from the fact that Im t_n is a multiplicatively closed set, if, and only if, $n \in \{1, 2, 3\}$. □

The family of maps $(\tau_n)_{n\geq 1}$ will be used to provide connections between arbitrary members of the family $(\mathcal{O}_n)_{n\geq 1}$. Likewise, the family of maps $(T_n)_{n\geq 1}$ will be used to provide connections between arbitrary members of the family $(\mathcal{O}_n[X])_{n\geq 1}$. Indeed, for each positive integer n, the ring $\mathcal{O}_n$ is bijective with the set Im τ_n, via σ_n, and, similarly, the ring $\mathcal{O}_n[X]$ is bijective with the set Im T_n, via S_n. Furthermore, for all positive integers n_1 and n_2, with $n_1 \leq n_2$, the restriction to Im τ_{n_1} of τ_{n_2} agrees with idempotent τ_{n_1}, and, similarly, the restriction to Im T_{n_1} of T_{n_2} agrees with idempotent T_{n_1}. In particular, Im $\tau_{n_1} \subseteq$ Im τ_{n_2} and Im $T_{n_1} \subseteq$ Im T_{n_2}. The following pair of technical lemmas will be used frequently and implicitly in the sequel.

Lemma 2. *If n_1 and n_2 are positive integers, then*

$$t_{n_1 n_2}\left(n_1 t_{n_2}(z)\right) = t_{n_1 n_2}\left(n_1 z\right),$$

for all $z \in \mathbb{Z}$, and analogous results hold for $\tau_{n_1 n_2}$ and $T_{n_1 n_2}$.

Proof. The result for $t_{n_1 n_2}$ is an immediate consequence of the fact that the integers $t_{n_2}(z)$ and z are congruent modulo n_2, if, and only if, the integers $n_1 t_{n_2}(z)$ and $n_1 z$ are congruent modulo $n_1 n_2$. The results for $\tau_{n_1 n_2}$ and $T_{n_1 n_2}$ then follow easily by (coefficient-wise) extensions. □

Lemma 3. *Let n be a positive integer. If $f\left((X_i)_{i\in\mathcal{I}}\right) \in \mathbb{Z}\left[(X_i)_{i\in\mathcal{I}}\right]$ is a polynomial over $\mathbb{Z}$ in a family of formal variables $(X_i)_{i\in\mathcal{I}}$, then*

$$t_n\left(f\left((t_n(z_i))_{i\in\mathcal{I}}\right)\right) = t_n\left(f\left((z_i)_{i\in\mathcal{I}}\right)\right),$$

for any family of integers $(z_i)_{i\in\mathcal{I}}$, and analogous results hold for τ_n and T_n.

Proof. By definition, $t_n = s_n \circ p_n$, and the claim follows from the fact that p_n is a ring homomorphism and $p_n \circ t_n = p_n$. The lemma then follows by extension. □

Finally, the derivation-like map

$$d_n : \mathbb{Z} \to \mathbb{Z}$$

given by

$$d_n : z \mapsto (z - t_n(z))\, n^{-1}$$

induces a derivation-like map

$$\delta_n : \mathcal{O} \to \mathcal{O}$$

given by

$$\delta_n : r \mapsto (r - \tau_n(r))\, n^{-1},$$

which, in turn, induces a derivation-like map

$$D_n : \mathcal{O}[X] \to \mathcal{O}[X]$$

given by

$$D_n : f(X) \mapsto (f(X) - T_n(f(X)))\, n^{-1}.$$

For $z \in \mathbb{Z}$, $r \in \mathcal{O}$, and $f(X) \in \mathcal{O}[X]$, we have the developments

$$z = t_n(z) + n d_n(z),$$

$$r = \tau_n(r) + n\delta_n(r),$$

and

$$f(X) = T_n(f(X)) + n D_n(f(X))$$

to the base n.

In the BFV cryptosystem, the **plaintext space** is given by $\tau_t(\mathcal{O})$, while the **ciphertext space** is given by the subset $T_q(\mathcal{O}[X])$ of the polynomial ring $\mathcal{O}[X]$. A **short polynomial** is defined as an element of $\tau_3(\mathcal{O})$, which always shall be sampled in a uniformly random manner. Finally, we mention here that elements of $\mathcal{O}_n$ are measured (e.g., via the ℓ_∞-norm) indirectly, by measuring their corresponding lifts under σ_n in $\mathcal{O}$, or, more precisely, in $\mathbb{Z}[x]$, via the natural section (i.e., minimal degree pre-image) of the projection of $\mathbb{Z}[x]$ onto $\mathcal{O}$.

2.2.2 Probability

In the BFV cryptosystem, an **error polynomial** is an element of $\mathcal{O}$ sampled from the error distribution χ defined as the coefficient-wise extension of the truncated discrete Gaussian distribution having mean zero and standard deviation

$$\sigma = 8/\sqrt{2\pi} \approx 3.19.$$

Here, the SEAL library takes, by default, $B = \lceil 6 * \sigma \rceil = 20$, for the truncation bound, since sampling a value more than 6 standard deviations from the mean is highly unlikely for any Gaussian variate. With this choice of truncation bound, the truncated distribution is almost indistinguishable from the non-truncated distribution. Thus, letting

$$\nu = \sum_{x=-B}^{B} \exp\left(-2^{-1}\left(\frac{x-0}{\sigma}\right)^2\right)$$

be the normalization factor, and writing $f : \mathbb{Z} \to [0,1]$ for the probability mass function, we find, for each integer x,

$$f(x) = \begin{cases} \nu^{-1} \exp\left(-2^{-1}\left(\frac{x-0}{\sigma}\right)^2\right) & |x| \leq B \\ 0 & |x| > B. \end{cases}$$

Note that

$$\begin{aligned}
\nu &= \sum_{x=-B}^{B} \exp\left(-2^{-1}\left(\frac{x-0}{\sigma}\right)^2\right) \\
&\approx \int_{-B}^{B} \exp\left(-2^{-1}\left(\frac{x-0}{\sigma}\right)^2\right) \mathrm{d}x \\
&\approx \int_{-\infty}^{\infty} \exp\left(-2^{-1}\left(\frac{x-0}{\sigma}\right)^2\right) \mathrm{d}x \\
&= \sigma\sqrt{2\pi} \\
&= 8.
\end{aligned}$$

In hopes of aiding the reader's intuition, we provide the following approximation to the truncated discrete Gaussian distribution underlying χ on its support:

x	f(x)
-20	.0000000004
-19	.0000000025
-18	.0000000153
-17	.0000000852
-16	.0000004309
-15	.0000019763
-14	.0000082165
-13	.0000309626
-12	.0001057575
-11	.0003274214
-10	.0009188082
-9	.0023370329
-8	.0053879913
-7	.0112592915
-6	.0213263903
-5	.0366138798
-4	.0569765644
-3	.0803653565
-2	.1027456873
-1	.1190639915
0	.1250602760
1	.1190639915
2	.1027456873
3	.0803653565
4	.0569765644
5	.0366138798
6	.0213263903
7	.0112592915
8	.0053879913
9	.0023370329
10	.0009188082
11	.0003274214
12	.0001057575
13	.0000309626
14	.0000082165
15	.0000019763
16	.0000004309
17	.0000000852
18	.0000000153
19	.0000000025
20	.0000000004

Table 2: Truncated discrete Gaussian $\left(\mu = 0,\ \sigma = 8/\sqrt{2\pi},\ B = \lceil 6\sigma \rceil = 20\right)$

Intuitively, each uniform sample x between $\pm B$, inclusive, is accepted with probability $f(x)$. For example, approximately every eighth $x = 0$ sample is accepted, whereas only (approximately) every 2^{31}-st $x = 20$ sample is accepted.

2.3 Key generation

Key generation proceeds as follows:

```
Input: -
Output: the private/public key pair
1. $s \xleftarrow{\$} \tau_3(\mathcal{O})$
2. $a \xleftarrow{\$} \tau_q(\mathcal{O})$
3. $e \leftarrow \chi$
4. $b \leftarrow \tau_q(as + e)$
5. pubkey $= \big[ -b, a \big]$
6. privkey $= s$
7. return privkey, pubkey
```

Table 3: Key Generation

Here, and in the sequel, $x \xleftarrow{\$} X$ means that the value x is sampled in a uniformly random manner from the set X.

As a warm-up to working with the probability distributions from which the various polynomials in the BFV scheme are sampled, let us consider the extremal and expected sizes of the coefficients in $as + e$, prior to application of τ_q. This specific information will not be used in the sequel, but the arguments employed in deducing it will be used more generally throughout. It is the authors' hope that the reader will benefit from first hearing these arguments in an especially simple situation.

First, consider the extremal sizes of the coefficients in $as+e$. By the Triangle Inequality,

$$\|as + e\|_\infty \leq \|as\|_\infty + \|e\|_\infty,$$

so we may focus separately on as and e. Now, each coefficient in a is at most $(q-1)/2$ in magnitude, whereas each coefficient in s has magnitude no larger than unity. It follows that each coefficient in as is at most $d\left((q-1)/2\right)$ in magnitude, that is to say,

$$\|as\|_\infty \leq d\left((q-1)/2\right).$$

Each coefficient in e is at most B in magnitude, hence,

$$\|e\|_\infty \leq B.$$

Putting these two results together, we find

$$\begin{aligned}\|as + e\|_\infty &\leq \|as\|_\infty + \|e\|_\infty \\ &\leq d\left((q-1)/2\right) + B,\end{aligned}$$

which gives the extremal values of the coefficients in $as + e$. In WIDESEAS, we take $d = 8192$, $q \approx 2^{218}$, and $B = 20$, in which case $d\left((q-1)/2\right) + B$ is a 230-bit number, hence, each coefficient in $as + e$ is no larger than around $2^{230} \approx 4096q$ in magnitude. Of course, the coefficients in $\tau_q(as + e)$ lie strictly between $\pm(q - 1)/2$, inclusive, by definition of τ_q.

Second, consider the expected sizes of the coefficients in $as + e$. Each coefficient in as is the sum of d independent and identically distributed random variables, and each of these d random variables is the product of two independent random variables, one over a uniform distribution on $\{-1, 0, 1\}$, and the other over a uniform distribution on $\big\{-(q-1)/2, \ldots, (q-1)/2\big\}$. Since these two discrete variates are independent, the mean of their product is the product of their means, that is to say, the mean of their product is $0 \times 0 = 0$. Since these two independent variates each have mean zero, it follows that the variance of their product is the product of their variances. As the individual variances are

$$\frac{\left((1-(-1))+1\right)^2 - 1}{12} = \frac{2}{3}$$

and

$$\frac{\left(((q-1)/2 - (-(q-1)/2)) + 1\right)^2 - 1}{12} = \frac{q^2-1}{12},$$

it follows that the variance of the product is

$$\left(\frac{2}{3}\right)\left(\frac{q^2-1}{12}\right) = \frac{q^2-1}{18}.$$

Now, as mentioned above, each coefficient in as is the sum of d of these random variables. The mean of this sum is just the sum of the d means, which is zero, since all d variables involved have mean zero. Furthermore, as the d random variables being summed are independent, it follows that the variance of their sum is the sum of their variances, which is

$$d\left(\frac{q^2-1}{18}\right).$$

But, by the Central Limit Theorem, each coefficient in as is approximately described by a Gaussian random variable, since each coefficient is described by a sum of d independent and identically distributed random variables, and, in practice, d is large (i.e., $d = 8192$). Thus, in the light of our deductions above, we find that each coefficient in as is described by a Gaussian random variable with mean zero and variance $d\left(\frac{q^2-1}{18}\right)$. Finally, summing this Gaussian with (a smooth approximation to) the truncated discrete Gaussian from which the coefficients in e were sampled, and, using the fact that these two Gaussians are independent, we deduce that each coefficient in $as+e$ is described by a Gaussian random variable having mean zero and variance

$$d\left(\frac{q^2-1}{18}\right) + \sigma^2.$$

In WIDESEAS, we take $d = 8192$, $q \approx 2^{218}$, and $\sigma = 8/\sqrt{2\pi}$, in which case each coefficient in $as + e$ is described by a Gaussian with mean zero and variance approximately 444-bits, which is around $(16q)^2$. It follows that coefficients in $as + e$ almost always lie between $\pm 6 \times (16q) = \pm 96q$. Thus, even though in the paragraph above we were able to use the ℓ_∞-norm to quickly deduce that the coefficients in $as + e$ may be no larger than around $4096q$ in magnitude, a more careful statistical analysis has revealed that the coefficients in $as + e$ very rarely have magnitude larger than $96q$.

2.3.1 On measurements

In the previous section, we first measured elements of $\mathcal{O}$ using the ℓ_∞-norm, and then we made more careful measurements using statistical techniques based on the Central Limit Theorem (CLT). In the R-LWE literature, measurements typically are made using the **canonical norm**. In this short section, we show that the canonical norm-based approach is essentially equivalent to our CLT-based approach. As we find the CLT-based approach to be the more intuitive of the two, it alone shall be employed in the sequel.

The *canonical embedding*

$$\varepsilon : \mathbb{Q} \otimes_{\mathbb{Z}} \mathcal{O} \to \mathbb{C}^d$$

maps each field element

$$a = f(x) + \Phi_{2d}(x)\mathbb{Q}[x]$$

to the vector of the evaluations of the polynomial $f(x)$ at the complex roots of $\Phi_{2d}(x)$, that is to say,

$$\varepsilon\left(a\right) = \left(f\left(\omega_{2d}^{j}\right)\right)_j .$$

Here, $\omega_{2d} \in \mathbb{C}$ is a primitive $(2d)$-th root of unity, and the index j ranges over a complete set of reduced residues modulo $2d$. Note that polynomials congruent modulo $\Phi_{2d}(x)$ necessarily have identical evaluations at the complex roots of $\Phi_{2d}(x)$, hence, the canonical embedding is well-defined. The *canonical norm* $\|a\|_{\text{can}}$ is defined as the Hermitian norm of $\varepsilon(a)$, hence,

$$\|a\|_{\text{can}}^2 := \|\varepsilon(a)\|_2^2 = \sum_j \left|f(\omega_{2d}^j)\right|^2.$$

Thus, if we let

$$\rho : \mathbb{Q} \otimes_{\mathbb{Z}} \mathcal{O} \to \mathbb{Q}[x]$$

be the section map taking each field element onto its representative of least degree, then the above result may be written as

$$\|a\|_{\text{can}}^2 = \sum_j \left| \left(\rho(a)\right)\left(\omega_{2d}^j\right) \right|^2. \tag{1}$$

Now, assume the coefficients of $\rho(a)$ are given by independent random variables over identical distributions having mean 0 and variance σ^2. Consider the expected value of $\|a\|_{\text{can}}^2$. On the one hand, by taking expectations of both sides of equation (1), we find

$$\begin{aligned} E\left(\|a\|_{\text{can}}^2\right) &= E\left(\sum_j |\left(\rho(a)\right)(\omega_{2d}^j)|^2\right) \\ &= \sum_j E\left(|\left(\rho(a)\right)(\omega_{2d}^j)|^2\right) \\ &\approx \sum_j E\left(|\left(\rho(a)\right)(\omega_{2d})|^2\right) \\ &= dE\left(|\left(\rho(a)\right)(\omega_{2d})|^2\right). \end{aligned} \tag{2}$$

On the other hand, by Parseval's Theorem,

$$\sum_j \left| \left(\rho(a)\right)\left(\omega_{2d}^j\right) \right|^2 = d\sum_{i=0}^{d-1} \left|\rho(a)_i\right|^2,$$

where $(\rho(a)_i)_{i=0}^{d-1} \in \mathbb{Q}$ are the coefficients of $\rho(a)$. Hence,

$$\|a\|_{\text{can}}^2 = d\sum_{i=0}^{d-1} \left|\rho(a)_i\right|^2. \tag{3}$$

Taking expectations of both sides of equation (3), we find

$$E\left(\|a\|_{\text{can}}^2\right) = d\left(d\sigma^2\right). \tag{4}$$

Comparing equations (2) and (4), we deduce

$$dE\left(|\left(\rho(a)\right)(\omega_{2d})|^2\right) \approx d\left(d\sigma^2\right),$$

or, equivalently,

$$E\left(|\left(\rho(a)\right)(\omega_{2d})|^2\right) \approx d\sigma^2. \tag{5}$$

The expression on the left-hand side of (5) is a measure of the dispersion in a under the canonical norm-based approach to measurement. The expression on the right-hand side (5) is a measure of the dispersion in a under the CLT-based approach to measurement. Thus, these two approaches are essentially equivalent, as claimed.

2.4 Encryption

Messages $m \in \tau_t(\mathcal{O})$ are encrypted as follows:

```
Input: public key $\big[-b,a\big]$; message $m \in \tau_t(\mathcal{O})$
Output: linear ciphertext $c(X) \in T_q\left(\mathcal{O}[X]\right)$ encrypting $m$
1. $u \xleftarrow{\$} \tau_3(\mathcal{O})$
2. $e_1, e_2 \leftarrow \chi$
3. $c_0 \leftarrow \tau_q\left(((-b)u + e_1) + \Delta m\right)$
4. $c_1 \leftarrow \tau_q(au + e_2)$
5. $c(X) \leftarrow c_0 + c_1 X$
6. return $c(X)$
```

Table 4: Encryption

Here, Δ is defined using the division algorithm for natural numbers, via

$$q = t\Delta + r_t(q),$$

where $0 \leq r_t(q) < t$. In particular, $\Delta = \lfloor q/t \rfloor$, which, as $t \geq 2$, is no larger than $q/2$. Note that the sums $(-b)u + e_1$ and $au + e_2$ occurring during encryption are, effectively, "public keys" generated from the same "private key" u. In fact, the former sum has the form of a public key generated from u and based on the public parameter $-b$, while the latter sum has the form of a public key generated from u and based on the public parameter a. Thus, encryption is effected, first, by choosing a "private key" u; second, by generating a pair of public keys from u based on the public parameters given by the components of the public key $[-b,a]$; and, third, by using the public key created from $-b$ to create a masked version of the scaled plaintext Δm. Of course, the "private key" u need not actually be kept secret, insofar as it may be made public without violating the security model of the cryptosystem.

2.5 Decryption

Decryption of a ciphertext $c(X) \in T_q\left(\mathcal{O}[X]\right)$ works as follows:

```
Input: privkey $s$; ciphertext $c(X) \in T_q\left(\mathcal{O}[X]\right)$
Output: decryption of $c(X)$ in $\tau_t(\mathcal{O})$
1. $m' \leftarrow \tau_t\left(\lfloor q^{-1} \otimes \left(t\tau_q\left(c(s)\right)\right)\rceil\right)$
2. return $m'$
```

Table 5: Decryption

Here, for a given group element $a \in (qt)^{-2}\mathbb{Z} \otimes_{\mathbb{Z}} \mathcal{O}$, the corresponding ring element $\lfloor a \rceil \in \mathcal{O}$ is obtained by rounding each coefficient of a, with respect to the basis

$$\left(q^{-1} \otimes \left(x^j + \Phi_{2d}(x)\mathbb{Z}[x]\right)\right)_{j=0}^{d-1},$$

to its nearest integer.

In carrying out the decryption algorithm, we need not look at elements of $\mathcal{O}$ any more closely than modulo qt, that is to say, at any point during the algorithm, we may safely replace any given quantity by its image under τ_{qt}. In fact, as

$$\left\|\tau_q\left(c(s)\right)\right\|_\infty \leq (q-1)\,2^{-1},$$

it follows

$$\begin{aligned}\left\|t\tau_q\left(c(s)\right)\right\|_\infty &\leq t\,(q-1)\,2^{-1}\\ &< (qt-1)\,2^{-1},\end{aligned}$$

where the last inequality follows from the assumption that $t > 1$. We deduce

$$\tau_{qt}\left(t\tau_q\left(c(s)\right)\right) = t\tau_q\left(c(s)\right),$$

and the claim follows immediately from the fact that ring elements a and b in $\mathcal{O}$ are congruent modulo qt, if, and only if, their corresponding group elements $q^{-1} \otimes a$ and $q^{-1} \otimes b$ in $(qt)^{-2}\mathbb{Z} \otimes_{\mathbb{Z}} \mathcal{O}$ are congruent modulo t, that is to say, in the quotient group

$$\left((qt)^{-2}\mathbb{Z} \otimes_{\mathbb{Z}} \mathcal{O}\right) / \left(t\mathbb{Z} \otimes_{\mathbb{Z}} \mathcal{O}\right).$$

Also, as

$$\begin{aligned}\left\|q^{-1} \otimes t\tau_q\left(c(s)\right)\right\|_\infty &\leq \left(t\,(q-1)\,2^{-1}\right)q^{-1}\\ &= \left(\left((q-t)\,2^{-1}\right) + q\left((t-1)\,2^{-1}\right)\right)q^{-1}\\ &= (1-qt^{-1})2^{-1} + (t-1)2^{-1}\\ &< 2^{-1} + (t-1)2^{-1},\end{aligned}$$

we deduce

$$\left\|\left\lfloor q^{-1} \otimes t\tau_q\left(c(s)\right)\right\rceil\right\|_\infty \leq (t-1)2^{-1},$$

and, therefore, the reduction modulo t operation at the close of the decryption algorithm is actually trivial.

Although the above algorithm for decryption might appear somewhat abstruse at first blush, it actually has a highly intuitive interpretation. In fact, the algorithm amounts to writing $t\tau_q\left(c\left(s\right)\right)$ in the mixed-radix system $\left\{1,\, q,\, tq\right\}$ relative to the sections τ_q and τ_t (i.e., in terms of the digits given by their images), so that

$$t\tau_q\left(c(s)\right) = \tau_q(d_0) + q\tau_t(d_1), \tag{6}$$

where $d_0,\, d_1 \in \mathcal{O}$, and then extracting the coefficient of q, that is, the "most significant digit" $\tau_t(d_1)$. From this point of view, it is apparent that in order for the decryption algorithm to correctly recover the underlying plaintext m, it is both necessary and sufficient that $\tau_t(d_1) = m$. In detail, write

$$\begin{aligned} t\tau_q\left(c(s)\right) &= \tau_q\left(t\tau_q\left(c(s)\right)\right) + q\delta_q\left(t\tau_q\left(c(s)\right)\right) \\ &= \tau_q\left(tc(s)\right) + q\delta_q\left(t\tau_q\left(c(s)\right)\right), \end{aligned}$$

and then write

$$\delta_q\left(t\tau_q\left(c(s)\right)\right) = \tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right) + t\delta_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right).$$

Deduce

$$t\tau_q\left(c(s)\right) = \tau_q\left(tc(s)\right) + q\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right) + qt\delta_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right).$$

As noted above,

$$\delta_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right) = 0_{\mathcal{O}},$$

and, therefore,

$$\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right) = \delta_q\left(t\tau_q\left(c(s)\right)\right),$$

so that

$$t\tau_q\left(c(s)\right) = \tau_q\left(tc(s)\right) + q\delta_q\left(t\tau_q\left(c(s)\right)\right). \tag{7}$$

We claim that the decryption algorithm returns the "most significant digit" $\delta_q\left(t\tau_q\left(c(s)\right)\right)$. In fact, scale $t\tau_q\left(c(s)\right)$ by q^{-1}, and then round each coefficient to its nearest integer. Thus, compute

$$\begin{aligned} q^{-1} \otimes t\tau_q\left(c(s)\right) &= q^{-1} \otimes \tau_q\left(tc(s)\right) + q^{-1} \otimes q\delta_q\left(t\tau_q\left(c(s)\right)\right) \\ &= q^{-1} \otimes \tau_q\left(tc(s)\right) + 1 \otimes \delta_q\left(t\tau_q\left(c(s)\right)\right), \end{aligned}$$

and then deduce

$$\left\lfloor q^{-1} \otimes t\tau_q\left(c(s)\right)\right\rceil = \delta_q\left(t\tau_q\left(c(s)\right)\right),$$

since

$$\left\|\tau_q\left(tc(s)\right)\right\|_\infty \leq (q-1)\,2^{-1}$$

implies

$$\left\|q^{-1} \otimes \tau_q\left(tc(s)\right)\right\|_\infty \leq \left(1 - q^{-1}\right)2^{-1} < 2^{-1},$$

which, in turn, implies

$$\left\lfloor q^{-1} \otimes \tau_q\left(tc(s)\right)\right\rceil = 0_{\mathcal{O}}.$$

Finally, (trivially) reduce modulo t, and thereby obtain the decrypt

$$\tau_t\left(\left\lfloor q^{-1} \otimes t\tau_q\left(c(s)\right)\right\rceil\right) = \delta_q\left(t\tau_q\left(c(s)\right)\right)$$

as the result of the decryption algorithm. Observe that we have thus extracted the "most significant digit" of $t\tau_q\left(c(s)\right)$, as claimed. Furthermore, observe that the decryption algorithm correctly recovers the underlying plaintext m if, and only if,

$$\delta_q\left(t\tau_q\left(c(s)\right)\right) = m,$$

or, equivalently

$$t\tau_q\left(c(s)\right) = \tau_q\left(tc(s)\right) + qm,$$

as claimed.

In order to study more closely the possible obstructions to correct decryption, write

$$t\tau_q\left(c(s)\right) = \left(t\tau_q\left(c(s)\right) - qm\right) + qm,$$

and put

$$\nu\left(c\left(s\right)\right) := \tau_{qt}\left(t\tau_q\left(c(s)\right) - qm\right) = \tau_{qt}\left(tc(s) - qm\right),$$

so that

$$t\tau_q\left(c(s)\right) = \tau_{qt}\left(t\tau_q\left(c(s)\right)\right) = \tau_{qt}\left(\nu\left(c\left(s\right)\right) + qm\right).$$

The polynomial $\nu\left(c\left(s\right)\right)$ is called the **noise** in the ciphertext $c(X)$. Note that the development of $t\tau_q\left(c(s)\right) - qm$ in the mixed-radix system $\left\{1,\, q,\, qt\right\}$ is given by

$$t\tau_q\left(c(s)\right) - qm = \tau_q\left(tc\left(s\right)\right) + q\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right) + qt\delta_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right).$$

Here, the "most significant digit" $\delta_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right)$ has ℓ_∞-norm no larger than unity. In fact, since

$$\delta_q\left(t\tau_q\left(c(s)\right)\right) = \tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right)\right)$$

and

$$m = \tau_t(m),$$

we deduce

$$\begin{aligned}\left\|\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right\|_\infty &\leq 2\left(\left(t-1\right)2^{-1}\right)\\ &= t-1\\ &= (-1) + t\cdot 1,\end{aligned}$$

from which the claim follows. In any case, we deduce

$$\nu\left(c\left(s\right)\right) = \tau_q\left(tc\left(s\right)\right) + q\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right).$$

It follows from the discussion above that $c(X)$ decrypts to m, if, and only if,

$$\nu\left(c\left(s\right)\right) = \tau_q\left(tc(s)\right),$$

if, and only if,

$$\|\nu\left(c\left(s\right)\right)\|_\infty \leq \left(q-1\right)2^{-1},$$

if, and only if,

$$\|\nu\left(c\left(s\right)\right)\|_\infty < q2^{-1},$$

if, and only if,

$$-\log_2\left(\left(2\|\nu\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right) > 0.$$

The quantity

$$\beta\left(c(s)\right) := -\log_2\left(\left(2\|\nu\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right)$$

is called the **noise budget** in the ciphertext $c(X)$. It is measured in bits. In actual practice (e.g., in the SEAL library), the noise budget is usually only estimated. An exact calculation of the noise budget would require a knowledge of not only the private key, but also the underlying plaintext. Since noise budget calculations are usually done only while choosing parameters for implementing a given circuit, and, since random plaintexts are typically used in this situation, estimating the noise budget in a way that depends on the private key but not on the plaintext relieves the programmer of having to store random plaintexts. The method used for estimating the noise budget hinges on the equality

$$\tau_q\left(tc\left(s\right)\right) = \tau_q\left(\nu\left(c\left(s\right)\right)\right),$$

since this equality shows that the reduction modulo q of the noise can be computed solely from a knowledge of $c\left(s\right)$. Thus, if we estimate the noise by its reduction modulo q, then we can estimate the noise budget without a knowledge of the underlying plaintext, via

$$\begin{aligned}\beta\left(c(s)\right) &\approx -\log_2\left(\left(2\|\tau_q\left(\nu\left(c\left(s\right)\right)\right)\|_\infty\right)q^{-1}\right)\\ &= -\log_2\left(\left(2\|\tau_q\left(tc\left(s\right)\right)\|_\infty\right)q^{-1}\right).\end{aligned}$$

Write

$$\widetilde{\nu}\left(c(s)\right) := \tau_q\left(\nu\left(c(s)\right)\right)$$

and

$$\widetilde{\beta}\left(c(s)\right) := -\log_2\left(\left(2\|\widetilde{\nu}\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right)$$

for the estimated noise and the estimated noise budget, respectively. In this way, we are discarding all but the "least significant digit" information in the noise, that is to say, we are discarding the "most significant digit" in our estimate. As it is precisely the latter digit that determines whether or not the noise has carried over into the plaintext, it follows that we can never actually detect noise overflow using SEAL's estimation procedure. Nevertheless, we can monitor the estimated noise budget, and then set parameters so that it never falls too close to zero. Equivalently, we can set parameters so that the norm of the estimated noise never climbs too close to $(q-1)2^{-1}$.

In summary, decryption of $c(X)$ is performed by writing

$$t\tau_q\left(c(s)\right) = \tau_q\left(tc(s)\right) + q\delta_q\left(t\tau_q\left(c(s)\right)\right),$$

and then extracting the "digit" $\delta_q\left(t\tau_q\left(c(s)\right)\right)$. The decryption algorithm recovers the underlying plaintext if, and only if, the noise

$$\nu\left(c(s)\right) := \tau_{qt}\left(t\tau_q\left(c(s)\right) - qm\right)$$

has positive budget

$$\beta\left(c(s)\right) := -\log_2\left(\left(2\|\nu\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right).$$

In practice, we use the estimates

$$\widetilde{\nu}\left(c(s)\right) := \tau_q\left(\nu\left(c(s)\right)\right)$$

and

$$\widetilde{\beta}\left(c(s)\right) := -\log_2\left(\left(2\|\widetilde{\nu}\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right)$$

for the noise and budget, respectively, and we set parameters so that the estimated budget always is sufficiently positive.

Although the above decryption algorithm applies to ciphertexts

$$c(X) = \sum_i c_i X^i \in T_q\left(\mathcal{O}[X]\right)$$

of arbitrary degree, we will not need to decrypt non-linear ciphertexts in the WIDESEAS protocol. Indeed, we use a technique called **relinearization** (see § 2.6.4) to reorganize non-linear ciphertexts into linear ones directly after they are formed. More specifically, immediately after multiplying two linear ciphertexts, we relinearize the resulting quadratic ciphertext back down to a linear ciphertext.

2.5.1 Decryption of fresh ciphertexts

A ciphertext is called **fresh** if it is the direct result of the encryption algorithm. Encryption parameters are chosen so that, in particular, the noise $\nu\left(c\left(s\right)\right)$ in a fresh ciphertext $c(X)$ is precisely the least significant digit in $t\tau_q\left(c(s)\right)$, while the message m is precisely the most significant digit in $t\tau_q\left(c(s)\right)$. As homomorphic operations are performed on $c(X)$, the noise tends to increase in magnitude, hence, the noise budget tends to decrease in measure. Eventually, the noise (literally) carries over into the most significant digit in $t\tau_q\left(c(s)\right)$, that is to say, into the message m. This results in a decryption error, since the decryption algorithm simply returns this second digit.

The goal of this section is to bound the noise in a fresh ciphertext. To do this. let us determine under which conditions the decryption algorithm correctly recovers the plaintext underlying a given fresh ciphertext. In this situation,

$$\begin{aligned} c_0 &= \tau_q\left(\left((-b)u + e_1\right) + \Delta m\right) \\ &= \tau_q\left(\left(\left(-\tau_q(as+e)\right)u + e_1\right) + \Delta m\right) \\ &= \tau_q\left(\left(\left(-(as+e)\right)u + e_1\right) + \Delta m\right) \\ &= \tau_q(-asu - eu + e_1 + \Delta m), \end{aligned}$$

from which it follows

$$\begin{aligned}\tau_q(c_0 + c_1 s) &= \tau_q\left(\tau_q(-asu - eu + e_1 + \Delta m) + \tau_q(au + e_2)s\right)\\ &= \tau_q\left((-asu - eu + e_1) + \Delta m + (au + e_2)s\right)\\ &= \tau_q(-asu - eu + e_1 + \Delta m + aus + e_2 s)\\ &= \tau_q(-eu + e_2 s + e_1 + \Delta m)\\ &= \tau_q\left(\tau_q\left(-eu + e_2 s + e_1\right) + \Delta m\right).\end{aligned}$$

Put

$$\nu_0 := \tau_q\left(-eu + e_2 s + e_1\right).$$

Then,

$$\pi_q\left(\tau_q(c_0 + c_1 s)\right) = \pi_q(\nu_0 + \Delta m),$$

which means

$$\tau_q(c_0 + c_1 s) - (\nu_0 + \Delta m) \in \operatorname{Ker}\, \pi_q.$$

Write this difference as $q\delta_0$, with $\delta_0 \in \mathcal{O}$, so that

$$\tau_q(c_0 + c_1 s) = \nu_0 + \Delta m + q\delta_0.$$

Multiply throughout by t,

$$\begin{aligned}t\tau_q(c_0 + c_1 s) &= t\nu_0 + t\Delta m + qt\delta_0\\ &= t\nu_0 + \left(q - r_t(q)\right)m + qt\delta_0\\ &= (t\nu_0 - r_t(q)m) + qm + qt\delta_0\\ &= \nu\left(c\left(s\right)\right) + q\left(m + t\left(\delta_{qt}\left(t\nu_0 - r_t(q)m\right) + \delta_0\right)\right),\end{aligned}$$

where

$$\begin{aligned}\nu\left(c\left(s\right)\right) &= \tau_{qt}\left((t\nu_0 - r_t(q)m) + qt\delta_0\right)\\ &= \tau_{qt}\left((t\nu_0 - r_t(q)m\right)\end{aligned}$$

is the noise in $c\left(X\right)$. Write

$$\nu\left(c\left(s\right)\right) = \tau_q\left(\nu\left(c\left(s\right)\right)\right) + q\tau_t\left(\delta_q\left(\nu\left(c\left(s\right)\right)\right)\right).$$

Deduce, in turn,

$$\begin{aligned}\tau_q\left(\nu\left(c\left(s\right)\right)\right) &= \tau_q\left(t\nu_0 - r_t(q)m\right)\\ \delta_q\left(\nu\left(c\left(s\right)\right)\right) &= \delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right)\\ \tau_t\left(\delta_q\left(\nu\left(c\left(s\right)\right)\right)\right) &= \tau_t\left(\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right)\right)\\ \delta_t\left(\delta_q\left(\nu\left(c\left(s\right)\right)\right)\right) &= \delta_t\left(\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right)\right).\end{aligned}$$

But, we already know

$$\delta_t\left(\delta_q\left(\nu\left(c\left(s\right)\right)\right)\right) = 0_{\mathcal{O}},$$

since the noise lies in Im τ_{qt}. It follows

$$\delta_t\left(\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right)\right) = 0_{\mathcal{O}},$$

hence,

$$\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right) = \tau_t\left(\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right)\right) = \tau_t\left(\delta_q\left(\nu\left(c\left(s\right)\right)\right)\right),$$

hence,

$$\nu\left(c\left(s\right)\right) = \tau_q\left(t\nu_0 - r_t(q)m\right) + q\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right),$$

and, therefore,

$$\begin{aligned} q^{-1} \otimes t\tau_q(c_0 + c_1 s) = {} & q^{-1} \otimes \tau_q\left(t\nu_0 - r_t(q)m\right) \\ & + 1 \otimes \left(\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right) + m + t\left(\delta_{qt}\left(t\nu_0 - r_t(q)m\right) + \delta_0\right)\right). \end{aligned}$$

Rounding all coefficients on either side of the last equation to their nearest integers,

$$\left\lfloor q^{-1} \otimes t\tau_q(c_0 + c_1 s)\right\rceil = \delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right) + m + t\left(\delta_{qt}\left(t\nu_0 - r_t(q)m\right) + \delta_0\right).$$

Finally, reducing modulo t, we obtain

$$\tau_t\left(\left\lfloor q^{-1} \otimes t\tau_q(c_0 + c_1 s)\right\rceil\right) = \delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right) + m$$

as the result of the decryption algorithm. It follows that decryption recovers m if, and only if,

$$\delta_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right) = 0_{\mathcal{O}},$$

or, equivalently,

$$\tau_q\left(\tau_{qt}\left(t\nu_0 - r_t(q)m\right)\right) = \tau_{qt}\left(t\nu_0 - r_t(q)m\right),$$

or, equivalently,

$$\tau_q\left(t\nu_0 - r_t(q)m\right) = \tau_{qt}\left(t\nu_0 - r_t(q)m\right).$$

On the one hand, we have verified (in the special case of a fresh ciphertext) the general result that the decryption algorithm recovers the underlying plaintext if, and only if, the "most significant digit" in the noise vanishes. On the other hand, we have demonstrated (in the special case of a fresh ciphertext) that the noise may be written as

$$\nu\left(c(s)\right) = \tau_{qt}\left(t\nu_0 - r_t(q)m\right),$$

where

$$\nu_0 = \tau_q\left(\tau_q\left(c(s)\right) - \Delta m\right) = \tau_q\left(c(s) - \Delta m\right).$$

In point of fact, no special properties of ν_0 were used to deduce this expression for the noise, and the argument may be easily adapted to show that an arbitrary (possibly unfresh) ciphertext may be likewise expressed.

We now bound the noise in a fresh ciphertext. This analysis will hinge on the special expression for ν_0 in the case of a fresh ciphertext. Since the noise satisfies

$$\nu\left(c\left(s\right)\right)=\tau_{qt}\left(t\nu_0-r_t(q)m\right),$$

it suffices to bound the difference $t\nu_0-r_t(q)m$. In fact,

$$\left\|\tau_{qt}\left(t\nu_0-r_t(q)m\right)\right\|_\infty\leq\left\|t\nu_0-r_t(q)m\right\|_\infty.$$

This follows from the coefficient-wise extension of the following result:

Lemma 4. *Let n, $z\in\mathbb{Z}$, with $n\geq 2$. If $|z|$ is developed to the base n, via*

$$|z|=t_n(|z|)+nd_n(|z|),$$

then $t_n(|z|)\leq|z|$.

Proof. The opposite inequality $t_n(|z|)>|z|$ holds, if, and only if,

$$t_n(|z|)>t_n(|z|)+nd_n(|z|),$$

which, in turn, holds, if, and only if, $0>d_n(|z|)$, which is impossible, as $|z|\geq 0$ forces $d_n(|z|)\geq 0$. $\square$

We deduce

$$\begin{aligned}\left\|\nu\left(c\left(s\right)\right)\right\|_\infty&\leq\left\|t\nu_0-r_t(q)m\right\|_\infty\\&\leq t\left\|\nu_0\right\|_\infty+r_t(q)\left\|m\right\|_\infty.\end{aligned}$$

But, as

$$\nu_0=\tau_q\left(-eu+e_2s+e_1\right),$$

we may deduce from the above lemma that

$$\left\|\nu_0\right\|_\infty\leq\left\|-eu+e_2s+e_1\right\|_\infty.$$

Therefore,

$$\begin{aligned}\left\|\nu\left(c\left(s\right)\right)\right\|_\infty&\leq t\left(\|eu\|_\infty+\|e_2s\|_\infty+\|e_1\|_\infty\right)+r_t(q)\|m\|_\infty\\&<t(2Bd+B)+r_t(q)\left(t2^{-1}\right)\\&=t\left(B(2d+1)+r_t(q)2^{-1}\right).\end{aligned}$$

Thus, if the parameters t, q, d, and B are chosen so that

$$t\left(B(2d+1)+r_t(q)2^{-1}\right)<q2^{-1},\tag{8}$$

then fresh ciphertexts always will decrypt correctly. In the WIDESEAS protocol, these parameters are chosen as follows:

$$\begin{array}{|rcl|}\hline B & = & 20 \\ d & = & 2^{13} \\ q & \approx & 2^{218} \\ t & \approx & 2^{32} \\ r_t(q) & \approx & 2^{29} \\ \hline\end{array}$$

Table 6: WIDESEAS BFV Parameters

In this situation,

$$t\left(B(2d+1) + r_t(q)2^{-1}\right) \approx 2^{63} \ll q2^{-1},$$

which implies that fresh ciphertexts always will decrypt correctly in WIDESEAS. Of course, this upper bound on the ℓ_∞-norm of the noise in a fresh ciphertext is highly pessimistic. Let us compute the expected value of the norm of the noise. Again, it suffices to analyze the pre-image $-eu + e_2 s + e_1$ of the noise under τ_q. First, each coefficient in e_1 is described by a random variable over a discrete Gaussian with mean zero, variance σ^2, and truncation at B. Second, each coefficient in $-eu$ is described by a sum of d independent and identically distributed random variables, each of which is the product of two independent random variables, namely, one over a discrete Gaussian distribution with mean zero, variance σ^2, and truncation at B, and one over a discrete uniform distribution on the set $\{-1, 0, 1\}$, and, thus, with mean zero and variance

$$\frac{\left((1-(-1))+1\right)^2 - 1}{12} = \frac{2}{3}.$$

It follows that each coefficient in $-eu$ is described by a random variable over a Gaussian distribution with mean zero and variance $(2d/3)\sigma^2$. Third, each coefficient in $e_2 s$ is likewise described as those in eu. Finally, since these three Gaussian variates are independent, it follows that their sum is described by a random variable over a Gaussian distribution with mean zero and variance

$$\sigma^2 + (2d/3)\sigma^2 + (2d/3)\sigma^2 = \left(1 + 4d/3\right)\sigma^2.$$

It follows that each coefficient in $t\left(-eu + e_2 s + e_1\right)$ is described by a random variable over a Gaussian distribution with mean zero and variance

$$\left(1 + 4d/3\right)(t\sigma)^2.$$

Assuming that each coefficient of the plaintext m is described by a random variable over a discrete uniform distribution on the set $\tau_t\left(\mathcal{O}\right)$, we deduce that each coefficient in $r_t(q)m$ is described by a random variable over a uniform distribution on $r_t(q)\tau_t\left(\mathcal{O}\right)$ with mean zero and variance

$$\left(r_t(q)\right)^2\left(\frac{\left((t/2)-(-t/2)+1\right)^2 - 1}{12}\right) = \left(r_t(q)\right)^2\left(\frac{(t+1)^2 - 1}{12}\right).$$

In all, each coefficient in $t\nu_0 - r_t(q)m$ is described by a sum of two independent random variables, namely, one over a Gaussian distribution with mean zero and variance $(1 + (4d/3))\,(t\sigma)^2$, and one over a uniform distribution on $r_t(q)\tau_t\left(\mathcal{O}\right)$ with mean zero and variance

$$(r_t(q))^2\left(\frac{(t+1)^2 - 1}{12}\right).$$

In WIDESEAS, the corresponding standard deviations are such that

$$\sqrt{1 + (4d/3)}(t\sigma) \approx 2^{40}$$

and

$$r_t(q)\sqrt{\frac{(t+1)^2 - 1}{12}} \approx 2^{62},$$

with $r_t(q) = 661,215,791 \approx 2^{29}$. The probability mass function of the sum of a Gaussian variate with an independent uniform variate is got by convolving the component mass functions, and its exact expression is given by a scaled difference of (oppositely dilated and identically translated) erf functions. Nevertheless, machine experiments suggest that this mass function does not differ too greatly from that of its Gaussian component. For this reason, we shall ignore the uniform term in our noise analysis. Then, nearly all noise polynomials are seen to (approximately) satisfy

$$\|\nu\left(c\left(s\right)\right)\|_\infty \leq 6 \cdot 2^{40}.$$

It follows that nearly all fresh ciphertexts formed under the WIDESEAS protocol have noise budgets satisfying

$$\begin{aligned}\beta\left(c\left(s\right)\right) &\geq -\log_2\left(2\left(6 \cdot 2^{40}\right)2^{-218}\right)\\ &= 176 - \log_2(3)\\ &\approx 174.\end{aligned}$$

Reassuringly, this number, which we have here deduced from first principles, is, on average, the value of SEAL's estimate for the noise budget in a fresh ciphertext based on the WIDESEAS parameters. For reference, we record the following useful heuristic, which follows easily from the preceding discussion:

Lemma 5. *In the BFV cryptosystem, the noise budget of a fresh ciphertext* $c(X)$ *satisfies*

$$\beta\left(c(s)\right) \approx \log_2\left(qt^{-1}\right) - \log_2\left((12)\sqrt{1 + (4d/3)}\sigma\right).$$

2.6 Noise under homomorphic operations

We have defined the **noise** $\nu\left(c\left(s\right)\right)$ in a ciphertext $c(X)$ (encrypting a plaintext m, via a private key s) as

$$\nu\left(c\left(s\right)\right) = \tau_{qt}\left(t\tau_q\left(c(s)\right) - qm\right) = \tau_{qt}\left(tc(s) - qm\right),$$

and we have established the alternative expression

$$\nu\left(c(s)\right) = \tau_{qt}\left(t\nu_0 - r_t(q)m\right),$$

where

$$\nu_0 = \tau_q\left(\tau_q\left(c(s)\right) - \Delta m\right) = \tau_q\left(c(s) - \Delta m\right).$$

Also, we have shown that $c(X)$ decrypts to m if, and only if,

$$\nu\left(c\left(s\right)\right) = \tau_q\left(tc(s)\right),$$

if, and only if,

$$\|\nu\left(c\left(s\right)\right)\|_\infty \leq \left(q-1\right)2^{-1},$$

if, and only if,

$$\|\nu\left(c\left(s\right)\right)\|_\infty < q2^{-1},$$

if, and only if,

$$-\log_2\left(\left(2\|\nu\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right) > 0,$$

where the quantity

$$\beta\left(c(s)\right) = -\log_2\left(\left(2\|\nu\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right)$$

is the **noise budget** in the ciphertext $c(X)$. Finally, we have indicated that in practice (e.g., in the SEAL library), we actually use the estimates

$$\widetilde{\nu}\left(c(s)\right) := \tau_q\left(\nu\left(c(s)\right)\right)$$

and

$$\widetilde{\beta}\left(c(s)\right) := -\log_2\left(\left(2\|\widetilde{\nu}\left(c\left(s\right)\right)\|_\infty\right)q^{-1}\right)$$

for the noise and budget, respectively,

As mentioned in the previous section, homomorphic operations tend to increase the norm of the noise. The three homomorphic operations used in WIDESEAS are ciphertext addition (i.e., adding together a pair of ciphertexts), plaintext multiplication (i.e., multiplying a ciphertext by a plaintext), and ciphertext multiplication (i.e., multiplying together a pair of ciphertexts). It turns out that the noise in a sum of ciphertexts is roughly the sum of the noises in the summands. Therefore, the noise increase incurred by homomorphic addition usually is low. The noise in a product of a ciphertext by a plaintext constant is roughly the product of the noise of the ciphertext by the size of the support of the plaintext constant. Therefore, the noise increase incurred by plain multiplication usually is moderate. The noise growth in ciphertext multiplication is considerable, and, consequently, the circuit implemented by WIDESEAS is of multiplicative depth only 2. In the following three sections, we make precise this intuition regarding the behavior of the noise under homomorphic operations.

2.6.1 Ciphertext addition

Let $c^{(1)}(X) = c_0^{(1)} + c_1^{(1)}X$ and $c^{(2)}(X) = c_0^{(2)} + c_1^{(2)}X$ be linear ciphertexts encrypting plaintext messages m_1 and m_2, respectively. The operation

$$\left(c^{(1)}(X),\, c^{(2)}(X)\right) \mapsto T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)$$

is called **ciphertext addition**. The ciphertext

$$T_q\left(c^{(1)}(X) + c^{(2)}(X)\right) = \tau_q\left(c_0^{(1)} + c_0^{(2)}\right) + \tau_q\left(c_1^{(1)} + c_1^{(2)}\right)X$$

is called the **ciphertext sum** of $c^{(1)}(X)$ and $c^{(2)}(X)$. It has a natural interpretation as an encryption of the message sum $\tau_t\left(m_1 + m_2\right)$. In fact, from

$$\begin{aligned}
\tau_q\left(T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)(s)\right) &= \tau_q\left(\tau_q\left(c_0^{(1)} + c_0^{(2)}\right) + \tau_q\left(c_1^{(1)} + c_1^{(2)}\right)s\right) \\
&= \tau_q\left(\left(c_0^{(1)} + c_0^{(2)}\right) + \left(c_1^{(1)} + c_1^{(2)}\right)s\right) \\
&= \tau_q\left(\left(c_0^{(1)} + c_1^{(1)}s\right) + \left(c_0^{(2)} + c_1^{(2)}s\right)\right) \\
&= \tau_q\left(c^{(1)}(s) + c^{(2)}(s)\right) \\
&= \tau_q\left(\tau_q\left(c^{(1)}(s)\right) + \tau_q\left(c^{(2)}(s)\right)\right) \\
&= \tau_q\left(c^{(1)}(s)\right) + \tau_q\left(c^{(2)}(s)\right) \\
&\quad - q\delta_q\left(\tau_q\left(c^{(1)}(s)\right) + \tau_q\left(c^{(2)}(s)\right)\right),
\end{aligned}$$

we deduce

$$\tau_{qt}\left(t\tau_q\left(T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)(s)\right)\right) = \tau_{qt}\left(t\tau_q\left(c^{(1)}(s)\right) + t\tau_q\left(c^{(2)}(s)\right)\right).$$

As

$$\tau_{qt}\left(t\tau_q\left(c^{(1)}(s)\right) + t\tau_q\left(c^{(2)}(s)\right)\right) = \tau_{qt}\left(\nu\left(c^{(1)}(s)\right) + \nu\left(c^{(2)}(s)\right) + q\left(m_1 + m_2\right)\right),$$

we deduce that the decryption

$$\tau_t\left(\left\lfloor q^{-1} \otimes \left(t\tau_q\left(T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)(s)\right)\right)\right\rceil\right)$$

of the ciphertext sum $T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)$ is given by

$$\tau_t\left(\left\lfloor q^{-1} \otimes \left(\nu\left(c^{(1)}(s)\right) + \nu\left(c^{(2)}(s)\right)\right)\right\rceil + \left(m_1 + m_2\right)\right).$$

It follows that $T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)$ is an encryption of $\tau_t\left(m_1 + m_2\right)$ with noise

$$\nu\left(T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)(s)\right) = \tau_{qt}\left(\nu\left(c^{(1)}(s)\right) + \nu\left(c^{(2)}(s)\right)\right).$$

Writing

$$\nu\left(c^{(1)}\left(s\right)\right)=\tau_q\left(tc^{(1)}\left(s\right)\right)+q\tau_t\left(\delta_q\left(t\tau_q\left(c^{(1)}(s)\right)\right)-m_1\right)$$

and

$$\nu\left(c^{(2)}\left(s\right)\right)=\tau_q\left(tc^{(2)}\left(s\right)\right)+q\tau_t\left(\delta_q\left(t\tau_q\left(c^{(2)}(s)\right)\right)-m_2\right),$$

we find that $\left\lfloor q^{-1}\otimes\left(\nu\left(c^{(1)}(s)\right)+\nu\left(c^{(2)}(s)\right)\right)\right\rceil$ is congruent modulo t to

$$\begin{aligned}\left\lfloor q^{-1}\otimes\left(\tau_q\left(tc^{(1)}\left(s\right)\right)+\tau_q\left(tc^{(2)}\left(s\right)\right)\right)\right\rceil&+\tau_t\left(\delta_q\left(t\tau_q\left(c^{(1)}(s)\right)\right)-m_1\right)\\&+\tau_t\left(\delta_q\left(t\tau_q\left(c^{(2)}(s)\right)\right)-m_2\right).\end{aligned}$$

We deduce that $T_q\left(c^{(1)}(X)+c^{(2)}(X)\right)$ decrypts to $\tau_t\left(m_1+m_2\right)$, if, and only if, the sum of the "most significant digits" in $\nu\left(c^{(1)}(s)\right)$ and $\nu\left(c^{(2)}(s)\right)$, plus the carry into the "second column", is congruent modulo t to zero. In particular, if $c^{(2)}(X)$ and $c^{(2)}(X)$ are fresh ciphertexts, so that their most significant digits vanish, then the ciphertext sum decrypts to $\tau_t\left(m_1+m_2\right)$, if, and only if, there is no carry into the second column in the addition of the noises. In general, as

$$\nu\left(T_q\left(c^{(1)}(X)+c^{(2)}(X)\right)(s)\right)=\tau_{qt}\left(\nu\left(c^{(1)}(s)\right)+\nu\left(c^{(2)}(s)\right)\right),$$

it follows that the noise budget

$$\beta\left(T_q\left(c^{(1)}(X)+c^{(2)}(X)\right)(s)\right)$$

of the sum of the ciphertexts satisfies

$$-\log_2\left(\left(2\left\|\tau_{qt}\left(\nu\left(c^{(1)}(s)\right)+\nu\left(c^{(2)}(s)\right)\right)\right\|_\infty\right)q^{-1}\right).$$

But, as

$$\begin{aligned}\left\|\tau_{qt}\left(\nu\left(c^{(1)}(s)\right)+\nu\left(c^{(2)}(s)\right)\right)\right\|_\infty&\leq\left\|\nu\left(c^{(1)}(s)\right)+\nu\left(c^{(2)}(s)\right)\right\|_\infty\\&\leq\left\|\nu\left(c^{(1)}(s)\right)\right\|_\infty+\left\|\nu\left(c^{(2)}(s)\right)\right\|_\infty,\end{aligned}$$

it follows that the noise is no smaller than

$$-\log_2\left(\left(\left(2\left\|\nu\left(c^{(1)}(s)\right)\right\|_\infty\right)q^{-1}\right)+\left(\left(2\left\|\nu\left(c^{(2)}(s)\right)\right\|_\infty\right)q^{-1}\right)\right).$$

We may use this lower bound to express the noise budget of the ciphertext sum in terms of the noise budgets of the ciphertext summands, via the following elementary identity, which expresses the logarithm of a sum as a weighted arithmetic mean of the logarithms of the summands (with a correction term coming from entropy):

Lemma 6. *If a and b are positive real numbers, then*

$$\log(a+b) = \left(\frac{a}{a+b}\right)\log(a) + \left(\frac{b}{a+b}\right)\log(b) + H\left(\frac{a}{a+b}, \frac{b}{a+b}\right), \tag{9}$$

where H is the Shannon (binary) entropy function.

Proof. The difference

$$(a+b)\log(a+b) - a\log(a) - b\log(b)$$

may be written as

$$a\log(a+b) + b\log(a+b) - a\log(a) - b\log(b),$$

which, in turn, may be written as

$$(a\log(a+b) - a\log(a)) + (b\log(a+b) - b\log(b)),$$

which, in turn, may be written as

$$a\log\left(\frac{a+b}{a}\right) + b\log\left(\frac{a+b}{b}\right),$$

which, in turn, may be written as

$$-a\log\left(\frac{a}{a+b}\right) - b\log\left(\frac{b}{a+b}\right).$$

Thus,

$$(a+b)\log(a+b) - a\log(a) - b\log(b) = -a\log\left(a/(a+b)\right) - b\log\left(b/(a+b)\right).$$

Scaling by $(a+b)^{-1}$,

$$\log(a+b) - \left(\frac{a}{a+b}\right)\log(a) - \left(\frac{b}{a+b}\right)\log(b) = H\left(\frac{a}{a+b}, \frac{b}{a+b}\right).$$

The result follows by rearranging terms. ☐

Now, taking

$$a = 2\left\|\nu\left(c^{(1)}(s)\right)\right\|_\infty q^{-1}$$

and

$$b = 2\left\|\nu\left(c^{(2)}(s)\right)\right\|_\infty q^{-1}$$

in the lemma, and putting

$$p = a/(a+b),$$

so that

$$b/(a+b) = 1-p,$$

we find, upon multiplying throughout by -1 in (9), that

$$-\log_2\left(\left(2\left\|\nu\left(c^{(1)}(s)\right)\right\|_\infty q^{-1}\right)+\left(2\left\|\nu\left(c^{(2)}\right)\right\|_\infty q^{-1}\right)\right)$$

may be written as

$$p\beta\left(c^{(1)}(s)\right)+(1-p)\beta\left(c^{(2)}(s)\right)-H(p,\,1-p).$$

We deduce

$$\beta\left(T_q\left(c^{(1)}(X)+c^{(2)}(X)\right)(s)\right)\geq p\beta\left(c^{(1)}(s)\right)+(1-p)\beta\left(c^{(2)}(s)\right)-H(p,\,1-p).$$

Thus, we have derived an inequality connecting the noise budget of the sum with the noise budgets of the summands, as promised. In fact, more can be said along these lines. As p and $1-p$ sum to unity,

$$\begin{aligned}\mathrm{Min}\left(\beta\left(c^{(1)}(s)\right),\,\beta\left(c^{(2)}(s)\right)\right)&\leq p\beta\left(c^{(1)}(s)\right)+(1-p)\beta\left(c^{(2)}(s)\right)\\&\leq\mathrm{Max}\left(\beta\left(c^{(1)}(s)\right),\,\beta\left(c^{(2)}(s)\right)\right).\end{aligned}$$

And, as the negative of the binary entropy lies between -1 and 0, inclusive,

$$\begin{aligned}\mathrm{Min}\left(\beta\left(c^{(1)}(s)\right),\,\beta\left(c^{(2)}(s)\right)\right)-1&\leq p\beta\left(c^{(1)}(s)\right)+(1-p)\beta\left(c^{(2)}(s)\right)-H(p,\,1-p)\\&\leq\mathrm{Max}\left(\beta\left(c^{(1)}(s)\right),\,\beta\left(c^{(2)}(s)\right)\right).\end{aligned}$$

In particular,

Lemma 7. *The noise budget of the sum of the ciphertexts* $c^{(1)}(X)$ *and* $c^{(2)}(X)$ *satisfies*

$$\beta\left(T_q\left(c^{(1)}(X)+c^{(2)}(X)\right)(s)\right)\geq\mathrm{Min}\left(\beta\left(c^{(1)}(s)\right),\,\beta\left(c^{(2)}(s)\right)\right)-1.$$

As $H(p,\,1-p)=1$, if, and only if, $c^{(1)}(X)$ and $c^{(2)}(X)$ have identical noise budgets, it follows that this condition is actually necessary for the noise budget of the sum to decrease by an entire bit. This condition is not sufficient, however, since what is really needed is for $\nu\left(c^{(1)}(s)\right)$ and $\nu\left(c^{(2)}(s)\right)$ to interfere constructively.

The entropy-based approach followed above allows for other useful deductions. For example, if

$$\left\|\nu\left(c^{(1)}(s)\right)\right\|_\infty\gg\left\|\nu\left(c^{(2)}(s)\right)\right\|_\infty,$$

then

$$p\approx 1,$$

and

$$1 - p \approx 0,$$

and, therefore,

$$H(p,\, 1-p) \approx 0,$$

so that, in this case,

$$\begin{aligned}\beta\left(T_q\left(c^{(1)}(X) + c^{(2)}(X)\right)(s)\right) &\geq p\beta\left(c^{(1)}(s)\right) + (1-p)\beta\left(c^{(2)}(s)\right) - H(p,\, 1-p)\\ &\approx p\beta\left(c^{(1)}(s)\right).\end{aligned}$$

As another example, we note that the above arguments may easily be extended from two up to any finite number N of ciphertext summands. In fact, let $\left(c^{(i)}(X)\right)_{i=1}^{N}$ be ciphertexts. Put

$$a_i := 2\left\|\nu\left(c^{(i)}(s)\right)\right\|_{\infty} q^{-1},$$

for all i such that $1 \leq i \leq N$. Put

$$A := \sum_{i=1}^{N} a_i,$$

and define

$$p_i := a_i A^{-1},$$

for all i such that $1 \leq i \leq N$. An easy generalization of the identity in the above lemma then reveals

$$\begin{aligned}\beta\left(T_q\left(\sum_{i=1}^{N} c^{(i)}(X)\right)(s)\right) &\geq -\log_2\left(\sum_{i=1}^{N} 2\left\|\nu\left(c^{(i)}(s)\right)\right\|_{\infty} q^{-1}\right)\\ &= \sum_{i=1}^{N} p_i\beta\left(c^{(i)}(s)\right) - H\left((p_i)_{i=1}^{N}\right),\end{aligned}$$

which, again, connects the noise budget of the sum with the noise budgets of the summands. Similar to the case of two summands, we find that, since the $(p_i)_{i=1}^{N}$ sum to unity,

$$\begin{aligned}\mathrm{Min}\left(\left(\beta\left(c^{(i)}(s)\right)\right)_{i=1}^{N}\right) &\leq \sum_{i=1}^{N} p_i\beta\left(c^{(i)}(s)\right)\\ &\leq \mathrm{Max}\left(\left(\beta\left(c^{(i)}(s)\right)\right)_{i=1}^{N}\right).\end{aligned}$$

Furthermore, since the negative of the N-ary entropy lies between $-\log_2(N)$ and 0, inclusive,

$$\begin{aligned}\mathrm{Min}\left(\left(\beta\left(c^{(i)}(s)\right)\right)_{i=1}^{N}\right)-\log_2(N)&\leq\sum_{i=1}^{N}p_i\beta\left(c^{(i)}(s)\right)-H\left((p_i)_{i=1}^{N}\right)\\&\leq\mathrm{Max}\left(\left(\beta\left(c^{(i)}(s)\right)\right)_{i=1}^{N}\right).\end{aligned}$$

In particular,

Lemma 8. *The noise budget of the sum of the ciphertexts* $\left(c^{(i)}(X)\right)_{i=1}^{N}$ *satisfies*

$$\beta\left(T_q\left(\sum_{i=1}^{N}c^{(i)}(X)\right)(s)\right)\geq\mathrm{Min}\left(\left(\beta\left(c^{(i)}(s)\right)\right)_{i=1}^{N}\right)-\log_2(N).$$

Finally, since the N-ary entropy attains its maximum if, and only if, the members of the family of ciphertexts $\left(c^{(i)}(X)\right)_{i=1}^{N}$ have pair-wise identical noise budgets, it follows that this condition is actually necessary for the noise budget to decrease by $\log_2(N)$. This condition is not sufficient, however, since what is really needed is for the $\left(\nu\left(c^{(i)}(s)\right)\right)_{i=1}^{N}$ to interfere constructively.

In the WIDESEAS protocol, the response to a query consists of a sequence of ciphertexts, each of which is the sum of 2^{32} ciphertexts having pair-wise (nearly) identical noise budgets β. It then follows immediately from the preceding result that the noise budget of each of these sums is generally no smaller than

$$\beta-\log_2\left(2^{32}\right)=\beta-32.$$

Thus, we need allot only 32 bits in the budget in order to accommodate the sum. Reassuringly, this result, which we have here derived from first principles, generally agrees with the actual noise budget consumption observed during our experiments with the SEAL library. Note that, since fresh ciphertexts in the WIDESEAS scheme have noise budgets around 174 bits, it follows that we still have 142 bits left to accommodate plaintext and ciphertext multiplications.

Although the preceding discussion has pertained only to the addition of linear ciphertexts, it turns out that ciphertexts of arbitrary degree may be added in the same manner. In the WIDESEAS protocol, we shall only need to add linear ciphertexts.

2.6.2 Plaintext multiplication

Let $c(X)=c_0+c_1X$ be a linear ciphertext encrypting a plaintext message m, and let $p\in\tau_t\left(\mathcal{O}\right)$ be any given plaintext. Write

$$p=\sum_{j=0}^{d-1}a_jx^j+\Phi_{2d}(x)\mathbb{Z}[x],$$

with $(a_j)_{j=0}^{d-1} \in \mathbb{Z}$ satisfying $(1-t)2^{-1} \leq a_j \leq (t-1)2^{-1}$, for each j such that $0 \leq j \leq d-1$. The operation

$$(p,\, c(X)) \mapsto T_q\left(pc(X)\right)$$

is called **plaintext multiplication**. The ciphertext

$$T_q\left(pc(X)\right) = \tau_q\left(pc_0\right) + \tau_q\left(pc_1\right)X$$

is called the **plaintext-ciphertext product** of p with $c(X)$. It has a natural interpretation as an encryption of the message product $\tau_t\left(pm\right)$. In fact, from

$$\begin{aligned}\tau_q\left(T_q\left(pc(X)\right)(s)\right) &= \tau_q\left(\tau_q\left(pc_0\right) + \tau_q\left(pc_1\right)s\right)\\ &= \tau_q\left(\left(pc_0\right) + \left(pc_1\right)s\right)\\ &= \tau_q\left(p\left(c_0 + c_1 s\right)\right)\\ &= \tau_q\left(pc(s)\right)\\ &= \tau_q\left(p\tau_q\left(c(s)\right)\right),\end{aligned}$$

we deduce

$$\begin{aligned}\tau_{qt}\left(tT_q\left(pc(X)\right)(s)\right) &= \tau_{qt}\left(pt\tau_q\left(c(s)\right)\right)\\ &= \tau_{qt}\left(p\nu\left(c(s)\right) + q(pm)\right),\end{aligned}$$

and, therefore, the decryption

$$\tau_t\left(\left\lfloor q^{-1} \otimes \left(t\tau_q\left(T_q\left(pc(X)\right)(s)\right)\right)\right\rceil\right)$$

of the plaintext-ciphertext product $T_q\left(pc(X)\right)$ is given by

$$\tau_t\left(\left\lfloor q^{-1} \otimes \left(p\nu\left(c(s)\right)\right)\right\rceil + pm\right),$$

and, therefore, $T_q\left(pc(X)\right)$ is an encryption of $\tau_t\left(pm\right)$ with noise

$$\nu\left(T_q\left(pc(X)\right)(s)\right) = \tau_{qt}\left(p\nu\left(c(s)\right)\right).$$

Writing

$$\nu\left(c\left(s\right)\right) = \tau_q\left(tc\left(s\right)\right) + q\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right),$$

we find that $\left\lfloor q^{-1} \otimes \left(p\nu\left(c(s)\right)\right)\right\rceil$ is congruent modulo t to

$$\left\lfloor q^{-1} \otimes \left(p\tau_q\left(tc\left(s\right)\right)\right)\right\rceil + p\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right).$$

We deduce that $T_q\left(pc(X)\right)$ decrypts to $\tau_t\left(pm\right)$, if, and only if, the product of the "most significant digit" in $\nu\left(c(s)\right)$ with p, plus the carry into the second column coming from the product of the "least significant digit" in $\nu\left(c(s)\right)$ with p, is congruent modulo t to zero, that is to say,

$$\tau_t\left(\delta_q\left(p\tau_q\left(tc(s)\right)\right) + p\tau_t\left(\delta_q\left(t\tau_q\left(c(s)\right)\right) - m\right)\right) = 0_{\mathcal{O}}.$$

In particular, if $c(X)$ is a fresh ciphertext, so that its most significant digit vanishes, then the plaintext-ciphertext product decrypts to $\tau_t(pm)$, if, and only if, there is no carry into the second column in the product of p with the noise in $c(s)$.

Now, for each j, with $0 \leq j \leq d-1$, the multiplication operator on $\mathcal{O}$ induced by $x^j + \Phi_{2d}\mathbb{Z}[x]$ leaves $\tau_n(\mathcal{O})$ set-wise invariant, for all positive integers n. It follows that the induced scalar action of this operator on $\mathcal{O}[X]$ leaves invariant the set of ciphertexts $T_q(\mathcal{O}[X])$. Put

$$c^{(j)}(X) := \left(x^j + \Phi_{2d}\mathbb{Z}[x]\right) \cdot c(X).$$

Then, we also have

$$\left(x^j + \Phi_{2d}\mathbb{Z}[x]\right) \nu\left(c(s)\right) = \nu\left(c^{(j)}(s)\right).$$

It follows that $c(X)$ and $c^{(j)}(X)$ have identical noise budgets. As

$$p\nu\left(c(s)\right) = \sum_{j=0}^{d-1} a_j \nu\left(c^{(j)}(s)\right),$$

and, therefore,

$$\begin{aligned} \nu\left(T_q\left(pc(X)\right)(s)\right) &= \tau_{qt}\left(p\nu\left(c(s)\right)\right) \\ &= \tau_{qt}\left(\sum_{j=0}^{d-1} a_j \nu\left(c(s)\right)^{(j)}\right), \end{aligned}$$

we deduce

$$\beta\left(T_q\left(pc(X)\right)(s)\right) = -\log_2\left(\left(2\left\|\tau_{qt}\left(\sum_{j=0}^{d-1} a_j \nu\left(c^{(j)}(s)\right)\right)\right\|_\infty\right) q^{-1}\right).$$

But, as

$$\begin{aligned} \left\|\tau_{qt}\left(\sum_{j=0}^{d-1} a_j \nu\left(c^{(j)}(s)\right)\right)\right\|_\infty &\leq \left\|\sum_{j=0}^{d-1} a_j \nu\left(c^{(j)}(s)\right)\right\|_\infty \\ &\leq \sum_{j=0}^{d-1} |a_j| \left\|\nu\left(c^{(j)}(s)\right)\right\|_\infty \\ &= \sum_{j=0}^{d-1} |a_j| \left\|\nu\left(c(s)\right)\right\|_\infty \\ &= \left(\sum_{j=0}^{d-1} |a_j|\right) \left\|\nu\left(c(s)\right)\right\|_\infty \\ &\leq \left((t-1)2^{-1}\right) \left|\mathrm{Supp}\left(p\right)\right| \cdot \left\|\nu\left(c(s)\right)\right\|_\infty, \end{aligned}$$

we deduce the following result:

Lemma 9. *The noise budget of a product of a plaintext p with a ciphertext $c(X)$ satisfies*

$$\beta\left(T_q\left(pc(X)\right)(s)\right) \geq -\log_2\left((t-1)2^{-1}\right) - \log_2\left(\left|\mathrm{Supp}\left(p\right)\right|\right) + \beta\left(c(s)\right).$$

In WIDESEAS, plaintext-ciphertext products always feature plaintexts having degrees and support sizes given by divisors of $d = 2^{13}$. Therefore, since $t \approx 2^{32}$, the greatest effect that such a product can have on the noise budget is around

$$\log_2\left((2^{32}-1)2^{-1}\right) + \log_2\left(2^{13}\right) \approx 31 + 13 = 44$$

bits. In actual practice, plaintexts tend to be randomized, in which case the term corresponding to $\log_2\left((t-1)2^{-1}\right)$ may be replaced by a much smaller bound. In WIDESEAS, plaintext-ciphertext multiplications usually consume no more than around 5 bits of the noise budget. Thus, the WIDESEAS protocol consumes around 32 bits of the noise budget doing ciphertext addition, and around 5 bits doing plaintext-ciphertext multiplication. As fresh ciphertexts have noise budgets of around 174 bits, it follows that there still remain around

$$174 - (32 + 5) = 137$$

bits of noise left in the budget, which, as discussed in the proceed two sections, we shall use for performing ciphertext multiplications with follow on relinearizations.

Although the preceding discussion has pertained only to linear ciphertexts, it turns out that ciphertexts of arbitrary degree may be multiplied by plaintexts in the same manner. In the WIDESEAS protocol, we shall not have a need to scale any but linear ciphertexts.

2.6.3 Ciphertext multiplication

Let $c^{(1)}(X) = c_0^{(1)} + c_1^{(1)}X$ and $c^{(2)}(X) = c_0^{(2)} + c_1^{(2)}X$ be linear ciphertexts encrypting plaintext messages m_1 and m_2, respectively. The operation

$$\left(c^{(1)}(X), c^{(2)}(X)\right) \mapsto T_q\left(\left\lfloor q^{-1} \otimes \left(tc^{(1)}(X)c^{(2)}(X)\right)\right\rceil\right)$$

is called **ciphertext multiplication**. Here, $\lfloor - \rceil$ denotes the mapping of

$$\left((qt)^{-2}\mathbb{Z} \otimes_{\mathbb{Z}} \mathcal{O}\right)[X]$$

onto $\mathcal{O}[X]$ induced by the usual mapping of $(qt)^{-2}\mathbb{Z} \otimes_{\mathbb{Z}} \mathcal{O}$ onto $\mathcal{O}$. Put

$$\begin{aligned} c_0 &:= c_0^{(1)}c_0^{(2)} \\ c_1 &:= c_0^{(1)}c_1^{(2)} + c_1^{(1)}c_0^{(2)} \\ c_2 &:= c_1^{(1)}c_1^{(2)}. \end{aligned}$$

The ciphertext

$$\begin{aligned} T_q\left(\left\lfloor q^{-1}\otimes\left(tc^{(1)}(X)c^{(2)}(X)\right)\right\rceil\right) &= \tau_q\left(\left\lfloor q^{-1}\otimes(tc_0)\right\rceil\right)\\ &+\tau_q\left(\left\lfloor q^{-1}\otimes(tc_1)\right\rceil\right)X\\ &+\tau_q\left(\left\lfloor q^{-1}\otimes(tc_2)\right\rceil\right)X^2 \end{aligned}$$

is called the **ciphertext product** of $c^{(1)}(X)$ with $c^{(2)}(X)$. It has a natural interpretation as an encryption of the message product $\tau_t\left(m_1m_2\right)$. In fact,

$$\tau_q\left(T_q\left(\left\lfloor q^{-1}\otimes\left(tc^{(1)}(X)c^{(2)}(X)\right)\right\rceil\right)(s)\right)$$

is equal to

$$\tau_q\left(\tau_q\left(\left\lfloor q^{-1}\otimes(tc_0)\right\rceil\right)+\tau_q\left(\left\lfloor q^{-1}\otimes(tc_1)\right\rceil\right)s+\tau_q\left(\left\lfloor q^{-1}\otimes(tc_2)\right\rceil\right)s^2\right).$$

Write

$$\begin{aligned} \tau_{(qt)^2}\left(t(tc_0)\right) &= a_0+(qt)b_0,\\ \tau_{(qt)^2}\left(t(tc_1)\right) &= a_1+(qt)b_1,\\ \tau_{(qt)^2}\left(t(tc_2)\right) &= a_2+(qt)b_2. \end{aligned}$$

As

$$\tau_{qt}\left(\left\lfloor q^{-1}\otimes(tc_0)\right\rceil\right)=\tau_{qt}\left(\left\lfloor (qt)^{-1}\otimes(t(tc_0))\right\rceil\right)=b_0,$$

$$\tau_{qt}\left(\left\lfloor q^{-1}\otimes(tc_1)\right\rceil\right)=\tau_{qt}\left(\left\lfloor (qt)^{-1}\otimes(t(tc_1))\right\rceil\right)=b_1,$$

$$\tau_{qt}\left(\left\lfloor q^{-1}\otimes(tc_2)\right\rceil\right)=\tau_{qt}\left(\left\lfloor (qt)^{-1}\otimes(t(tc_2))\right\rceil\right)=b_2,$$

it follows that

$$\tau_q\left(\left\lfloor q^{-1}\otimes(tc_0)\right\rceil\right)+\tau_q\left(\left\lfloor q^{-1}\otimes(tc_1)\right\rceil\right)s+\tau_q\left(\left\lfloor q^{-1}\otimes(tc_2)\right\rceil\right)s^2$$

and

$$b_0+b_1s+b_2s^2$$

are congruent modulo q. We deduce that the decryption of the ciphertext product is given by

$$\tau_t\left(\left\lfloor q^{-1}\otimes\left(t\tau_q\left(b_0+b_1s+b_2s^2\right)\right)\right\rceil\right)=\tau_t\left(\left\lfloor q^{-1}\otimes\left(t\left(b_0+b_1s+b_2s^2\right)\right)\right\rceil\right).$$

Furthermore, as

$$q^{-2}\otimes\tau_{(qt)^2}\left(t(tc_0)\right)=q^{-2}\otimes a_0+q^{-1}\otimes(tb_0),$$

$$q^{-2}\otimes\tau_{(qt)^2}\left(t(tc_1)\right)=q^{-2}\otimes a_1+q^{-1}\otimes(tb_1),$$

$$q^{-2} \otimes \tau_{(qt)^2}\left(t(tc_2)\right) = q^{-2} \otimes a_2 + q^{-1} \otimes (tb_2),$$

it follows that

$$q^{-1} \otimes \left(t\left(b_0 + b_1 s + b_2 s^2\right)\right)$$

may be written

$$q^{-2} \otimes \left(\left(\tau_{(qt)^2}\left(t(tc_0)\right) + \tau_{(qt)^2}\left(t(tc_1)\right) s + \tau_{(qt)^2}\left(t(tc_2)\right) s^2\right) - \left(a_0 + a_1 s + a_2 s^2\right)\right).$$

But,

$$\tau_{(qt)^2}\left(t(tc_0)\right) + \tau_{(qt)^2}\left(t(tc_1)\right) s + \tau_{(qt)^2}\left(t(tc_2)\right) s^2$$

is congruent modulo $(qt)^2$ to

$$(t(tc_0)) + (t(tc_1)) s + (t(tc_2)) s^2 = \left(tc^{(1)}(s)\right)\left(tc^{(2)}(s)\right),$$

which, in turn, is congruent modulo $(qt)^2$ to

$$\begin{aligned}&\left(t\tau_q\left(c^{(1)}(s)\right)\right)\left(t\tau_q\left(c^{(2)}(s)\right)\right)\\&= \left(\nu\left(c^{(1)}(s)\right) + qm_1 + (qt)\delta_1\right)\left(\nu\left(c^{(2)}(s)\right) + qm_2 + (qt)\delta_2\right).\end{aligned}$$

Here, we recall that δ_1, $\delta_2 \in \mathcal{O}$ have ℓ_∞-norms at most unity. It follows

$$\tau_{(qt)^2}\left(t(tc_0)\right) + \tau_{(qt)^2}\left(t(tc_1)\right) s + \tau_{(qt)^2}\left(t(tc_2)\right) s^2$$

is congruent modulo $(qt)^2$ to

$$\begin{aligned}\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right) &+ q\left(\nu\left(c^{(1)}(s)\right) m_2 + m_1 \nu\left(c^{(2)}(s)\right)\right)\\&+ (qt)\left(\delta_1 \nu\left(c^{(2)}(s)\right) + \nu\left(c^{(1)}(s)\right)\delta_2\right)\\&+ q^2\left(m_1 m_2\right)\\&+ (q^2 t)\left(\delta_1 m_2 + m_1 \delta_2\right),\end{aligned}$$

and, therefore,

$$q^{-2} \otimes \left(\tau_{(qt)^2}\left(t(tc_0)\right) + \tau_{(qt)^2}\left(t(tc_1)\right) s + \tau_{(qt)^2}\left(t(tc_2)\right) s^2\right)$$

is congruent modulo t^2 to

$$\begin{aligned}q^{-2} \otimes \left(\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right)\right) &+ q^{-1} \otimes \left(\nu\left(c^{(1)}(s)\right) m_2 + m_1 \nu\left(c^{(2)}(s)\right)\right)\\&+ q^{-1} \otimes \left(t\left(\delta_1 \nu\left(c^{(2)}(s)\right) + \nu\left(c^{(1)}(s)\right)\delta_2\right)\right)\\&+ 1 \otimes \left(m_1 m_2\right)\\&+ 1 \otimes \left(t\left(\delta_1 m_2 + m_1 \delta_2\right)\right).\end{aligned}$$

As

$$q^{-1} \otimes \left(t\left(b_0 + b_1 s + b_2 s^2\right)\right)$$

is congruent modulo t to the integer rounding of

$$q^{-2} \otimes \left(\left(\tau_{(qt)^2}\left(t(tc_0)\right) + \tau_{(qt)^2}\left(t(tc_1)\right)s + \tau_{(qt)^2}\left(t(tc_2)\right)s^2\right) - \left(a_0 + a_1 s + a_2 s^2\right)\right),$$

it must also be congruent modulo t to the sum of the product $m_1 m_2$ with the integer part of

$$\begin{aligned} q^{-2} \otimes \left(\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right)\right) &+ q^{-1} \otimes \left(\nu\left(c^{(1)}(s)\right)m_2 + m_1\nu\left(c^{(2)}(s)\right)\right) \\ &+ q^{-1} \otimes \left(t\left(\delta_1\nu\left(c^{(2)}(s)\right) + \nu\left(c^{(1)}(s)\right)\delta_2\right)\right) \\ &- q^{-2} \otimes \left(a_0 + a_1 s + a_2 s^2\right). \end{aligned}$$

We have thus shown:

Lemma 10. *If $c^{(1)}(X), c^{(2)}(X) \in T_q\left(\mathcal{O}[X]\right)$ are ciphertexts encrypting plaintext messages m_1 and m_2, respectively, then their product*

$$T_q\left(\left\lfloor q^{-1} \otimes \left(tc^{(1)}(X)c^{(2)}(X)\right)\right\rceil\right)$$

is an encryption of $\tau_t\left(m_1 m_2\right)$ with noise given by the reduction modulo qt of

$$\begin{aligned} &\left\lfloor q^{-1} \otimes \left(\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right) - \left(a_0 + a_1 s + a_2 s^2\right)\right)\right\rceil \\ &+ \nu\left(c^{(1)}(s)\right)m_2 + m_1\nu\left(c^{(2)}(s)\right) \\ &+ t\left(\delta_1\nu\left(c^{(2)}(s)\right) + \nu\left(c^{(1)}(s)\right)\delta_2\right). \end{aligned}$$

If both ciphertext factors have noise levels beneath the noise floor of $q2^{-1}$, then the noise increase in their product is bounded by a scalar on the order of $2td^2$.

Proof. Only the final claim remains to be shown. Thus,

$$\begin{aligned} \|a_0 + a_1 s + a_2 s^2\|_\infty &\le \|a_0\|_\infty + \|a_1 s\|_\infty + \|a_2 s^2\|_\infty \\ &\le \left((qt)2^{-1}\right) + \left((qt)2^{-1}\right)d + \left((qt)2^{-1}\right)d^2 \\ &= \left((qt)2^{-1}\right)\left(\left(d^3 - 1\right)\left(d - 1\right)^{-1}\right), \end{aligned}$$

hence,

$$\|a_0 + a_1 s + a_2 s^2\|_\infty q^{-1} \le \left(t2^{-1}\right)\left(\left(d^3 - 1\right)\left(d - 1\right)^{-1}\right).$$

Let M denote the maximum value of the noises of the two factors, so that $1 \le M \le (qt - 1)\,2^{-1}$. Then,

$$\left\|\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right)\right\|_\infty \le dM^2,$$

hence,

$$\left\|\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right)\right\|_\infty q^{-1} \leq dM^2q^{-1}.$$

We deduce

$$\begin{aligned}&\left\|\left\lfloor q^{-1}\otimes\left(\nu\left(c^{(1)}(s)\right)\nu\left(c^{(2)}(s)\right)-\left(a_0+a_1s+a_2s^2\right)\right)\right\rceil\right\|_\infty q^{-1}\\ &\leq dM^2q^{-1}+\left(t2^{-1}\right)\left(\left(d^3-1\right)\left(d-1\right)^{-1}\right).\end{aligned}$$

As

$$\begin{aligned}\left\|\nu\left(c^{(1)}(s)\right)m_2+m_1\nu\left(c^{(2)}(s)\right)\right\|_\infty &\leq 2\left(dMt2^{-1}\right)\\ &= dMt\end{aligned}$$

and

$$\begin{aligned}t\left\|\delta_1\nu\left(c^{(2)}(s)\right)+\nu\left(c^{(1)}(s)\right)\delta_2\right\|_\infty &\leq t\left(2\left(dM\right)\right)\\ &= 2dMt,\end{aligned}$$

it follows that the ℓ_∞-norm of the noise of the product is no larger than

$$dM^2q^{-1}+\left(t2^{-1}\right)\left(\left(d^3-1\right)\left(d-1\right)^{-1}\right)+3dMt.$$

In particular, if the maximal noise M is such that $M < q2^{-1}$, so that

$$Mq^{-1} < 2^{-1},$$

then we deduce that the noise of the product is less than

$$\begin{aligned}dM2^{-1}+\left(t2^{-1}\right)\left(\left(d^3-1\right)\left(d-1\right)^{-1}\right)+3dMt &< dM2^{-1}+\left(t2^{-1}\right)\left(3d^2\right)+3dMt\\ &< dM+2td^2+3dMt.\end{aligned}$$

Thus, the ratio of the noise to M is less than

$$d+2td^2M^{-1}+3dt \leq d+2td^2+3dt.$$

It follows that the noise is multiplied by a factor on the order of $2td^2$. □

Interestingly, the previous result shows that the noise does not grow quadratically upon multiplication, but, rather, it is multiplied by a factor on the order of $2td^2$, which is independent of q.

In WIDESEAS, we take $t \approx 2^{32}$ and $d = 2^{13}$, hence, each ciphertext multiplication increases the noise by a factor no greater than around

$$2\cdot 2^{32}\cdot\left(2^{13}\right)^2 = 2^{59}.$$

It follows that each ciphertext multiplication consumes at most 59 bits of the noise budget. Of course, this bound is highly pessimistic. However, even by using the tightest bounds met with in the proof of the above lemma, we can conclude only that the noise consumption is at most around 57 bits. Nevertheless, a more careful statistical analysis of the noise is possible, and such an analysis reveals that ciphertext multiplication very rarely (i.e., "6σ") consumes more than around 45 bits of the noise budget. Reassuringly, this result, which we have derived from first principles, agrees with our observations made while experimenting with the SEAL library. We mention here that WIDESEAS has multiplicative depth equal to 2, hence, we need to allot around $2 \cdot 45 = 90$ bits in our budget to accomodate the two multiplications. In all, the protocol so far consumes around 32 bits of the noise budget doing ciphertext addition, around 5 bits doing plaintext-ciphertext multiplication, and around 90 bits doing ciphertext multiplication. As fresh ciphertexts in WIDESEAS have noise budgets of around 174 bits, it follows that there still remain around

$$174 - (32 + 5 + 90) = 174 - 127 = 47$$

bits of noise left in the budget. In the next section, we shall see that around 10 of these remaining bits will be used to convert quadratic ciphertexts to linear ciphertexts. We will then have around 37 bits left in our budget, which is enough to comfortably cover freak outliers and other random accidents.

Although the preceding discussion has pertained only to linear ciphertexts, it turns out that ciphertexts of arbitrary degree may be multiplied together in the same manner. In the WIDESEAS protocol, we shall not have a need to multiply any but linear ciphertexts.

2.6.4 Relinearization

In the previous section, we saw that, in general, the product of two linear ciphertexts

$$c_0^{(1)} + c_1^{(1)} X$$

and

$$c_0^{(2)} + c_1^{(2)} X$$

is, perhaps unsurprisingly, a quadratic ciphertext

$$c_0 + c_1 X + c_2 X^2.$$

In this section, we discuss a technique called **relinearization**, which transforms a quadratic ciphertext into a linear ciphertext. Our discussion closely follows that given in the SEAL manual [4]. In fact, the manual's description is quite nice and succinct, and we essentially repeat it here using the notation of this paper.

Intuitively, relinearization slices up the coefficient c_2 into multiple parts, each having sufficiently small norm, and then redistributes these parts across the coefficients c_0 and c_1. The slicing aspect is effected by choosing a positive integral base T (independently of the plaintext modulus t) and then developing c_2 to the base T, via

$$c_2 = \sum_{i=0}^{l} T^i c_2^{(i)},$$

where $l := \log_T(q)$ and the family of digits $\left(c_2^{(i)}\right)_{i=0}^{l}$ comes from $\tau_T(\mathcal{O})$. This technique hinges on the use of a **relinearization key** rlk, which consists of masked versions of the $\left(T^i s^2\right)_{i=0}^{l}$. Specifically,

$$\mathrm{rlk} = \left(\left[\mathrm{rlk}_0^{(i)}, \mathrm{rlk}_1^{(i)}\right]\right)_{i=0}^{l},$$

where

$$\mathrm{rlk}_0^{(i)} = \tau_q\left(-\left(a_i s + e_i\right) + T^i s^2\right)$$

and

$$\mathrm{rlk}_1^{(i)} = a_i,$$

for all i such that $0 \leq i \leq l$. Here, the $(a_i)_{i=0}^{l}$ are sampled uniformly at random from $\tau_q(\mathcal{O})$, while the $(e_i)_{i=0}^{l}$ are randomly chosen error polynomials coming from $\mathcal{O}$. As rlk contains masked versions of the $\left(T^i s^2\right)_{i=0}^{l}$, which are neither real samples of the R-LWE distribution, nor real encryptions of the $\left(T^i s^2\right)_{i=0}^{l}$, this technique introduces an extra assumption on the BFV scheme, namely, that the scheme is still secure when the adversary has access to rlk. This property is a form of **weak circular security**.

Now, define

$$c_0' := \tau_q\left(c_0 + \sum_{i=0}^{l} \mathrm{rlk}_0^{(i)} c_2^{(i)}\right)$$

and

$$c_1' := \tau_q\left(c_1 + \sum_{i=0}^{l} \mathrm{rlk}_1^{(i)} c_2^{(i)}\right),$$

and check

$$\tau_q\left(c_0' + c_1' s\right) = \tau_q\left(c_0 + c_1 s + c_2 s^2 - \sum_{i=0}^{l} c_2^{(i)} e_i\right).$$

The foregoing shows that the base T has the following effects:

1. As the size of the relinearization key is $1 + l$, it follows that the sizes of T and rlk vary indirectly.
2. The number of multiplications in the relinearization step is $2l \approx 2\log_T(q)$, where the multiplications are between elements of $\tau_T(\mathcal{O})$ and $\tau_q(\mathcal{O})$.

3. The noise introduced by relinearization is bounded by $(1 + l)BTd2^{-1}$, hence, greater T results in greater noise.

Note that the noise introduced by relinearization is independent of the noise present in the ciphertext being relinearized. Furthermore, we only need to relinearize after a ciphertext multiplication, which, itself, causes the underlying noise to grow. Therefore, we should choose T large enough to ensure that the relinearization noise is of the same order as the noise contained in a ciphertext resulting from multiplying two fresh ciphertexts. In fact, this strategy furnishes us with a minimal value of T that we might consider using. Although this strategy minimizes the relinearization noise, the WIDESEAS protocol uses a different strategy, namely, one that minimizes the relinearization time and space. In this approach, we take T very large, say, $T = \lceil \sqrt{q} \rceil$, since then we only have two slices to take care off. For such large T, the size of the noise after the first relinearization typically will make a huge jump, but all following relinearizations will not cause the noise to increase. As WIDESEAS follows this approach, it turns out that the relinearization done after the first ciphertext multiplication consumes around 10 bits of the noise budget, but the relinearization done after the second ciphertext multiplication usually consumes none of the budget.

In all, the WIDESEAS protocol consumes around 32 bits of the noise budget doing ciphertext addition, around 5 bits doing plaintext-ciphertext multiplication, around 90 bits doing ciphertext multiplication, and around 10 bits doing relinearization. As fresh ciphertexts have noise budgets of around 174 bits, it follows that there still remain around

$$174 - (32 + 5 + 90 + 10) = 174 - 137 = 37$$

bits of noise left in the budget. This “surplus” is enough to cover freak outliers and other random accidents.

2.7 Modulus switching

In this section, we discuss **modulus switching**, which can be used to improve efficiency in both time and space. Our discussion closely follows that given in the documentation to Microsoft’s SEAL library. In fact, the documentation on this topic is quite nice and succinct, and we essentially repeat it here using the notation of this paper.

Modulus switching is a technique for widdling down the encryption parameters along a so-called **modulus switching chain**. The top link in the chain holds the original encryption parameters. As we move down the chain one link at a time, we recursively remove a single prime factor from the current ciphertext modulus, so that the sizes of the ciphertext moduli decrease as we move down along the chain. This continues until the parameter set is no longer valid (e.g., the plaintext modulus t is larger than the current ciphertext modulus).

Freshly encrypted ciphertexts always belong to the highest level of the chain, inasmuch as they correspond to the original (full) ciphertext modulus q. As homomorphic operations are performed, and, consequently, as noise begins to build, we may choose to "switch down" along the modulus chain. Thus, if $q = q_0q_1q_2q_3$ is originally a product of four mutually distinct primes having bit lengths 55, 55, 54, and 54, respectively, and, if $c(X)$ is a ciphertext resulting from one or more homomorphic operations, then the effect of a first modulus switch is to redistribute the modulo q_3 information present in $c(X)$ across the primes q_0, q_1, and q_2, in the same way that this is usually done in Residue Number System (RNS) computations. The resulting ciphertext is then, essentially, encrypted with respect to the ciphertext modulus $q_0q_1q_2$. A second switch redistributes the modulo q_2 information across q_0 and q_1, and, therefore, results in a ciphertext that, essentially, is encrypted with respect to the ciphertext modulus q_0q_1. A third switch redistributes the modulo q_1 information into the q_0 component, and, therefore, results in a ciphertext that, essentially, is encrypted with respect to the ciphertext modulus q_0. This switching process may be visualized thusly:

ciphertext modulus	Level
$q_0q_1q_2q_3$	3 (highest)
$q_0q_1q_2$	2
q_0q_1	1
q_0	0 (lowest)

Figure 1: Modulus switching chain

The SEAL library indexes the prime factors of q in decreasing order of size, and, therefore, each modulus switch always removes the current smallest factor.

In order to understand the potential benefit got from modulus switching, observe that the size of a given ciphertext depends linearly on the number of prime factors in the ciphertext modulus. Thus, if there is no need nor any intention to perform any additional computations on a given ciphertext, then we might as well switch it down to the smallest (i.e., lowest) parameter level in the modulus chain prior to sending it to the private key holder for decryption. Now, at first blush, it might appear that proceeding in this way will sometimes lead to a loss of noise budget. Indeed, as the plaintext modulus t is fixed, it follows that continually decreasing the size of the ciphertext modulus necessarily results in ever lower initial noise budgets. It turns out that this lost noise budget is actually not an issue, at least, that is, if we proceed with care. Thus, let $c(X)$ be a ciphertext with respect to $q = q_0q_1q_2q_3$, and suppose that $c(X)$ is the result of sufficiently many homomorphic operations that its noise budget is just slightly below the level we would have in a fresh ciphertext created after performing a single modulus switch (i.e., a fresh ciphertext with respect to the ciphertext modulus $q_0q_1q_2$). Surprisingly, perhaps, in this situation, modulus switching has no effect at all on the current noise budget of $c(X)$. In general, there is no harm at all in shaving off some factors from the ciphertext modulus after performing enough computations. Indeed, with respect to the switched down modulus, the ciphertext (i) still decrypts correctly (presumably), (ii) is of a smaller size, and (iii) is such that future computations done on it will be more efficient, since it is smaller. We note that, in SEAL's implementation of BFV, the modulus switching technique is not necessary, and, by default, it is switched off. In any case, the programmer chooses when and what to switch.

In WIDESEAS, the initial ciphertext modulus is a product of four mutually distinct primes. The modulus is switched by the Responder a total of three times during the response generation phase: Once after the first ciphertext multiplications, and then twice after performing the column sums. This ensures that the response transmitted to the Querier is as small as possible.

2.8 Batching

When the plaintext modulus t is a prime such that $\pi_{2d}(t) = \pi_{2d}(1)$, the field $\mathbb{F}_t$ has a full complement of $2d$-th roots of unity. Consequently, the cyclotomic polynomial $\Phi_{2d}(x)$ splits completely over $\mathbb{F}_t$, and, therefore, the ring $\mathcal{O}_t$ has a unique decomposition (up to ordering) as a direct sum of $\varphi(2d) = d$ minimal ideals, each isomorphic as an algebra with $\mathbb{F}_t$. The partition of unity in $\mathcal{O}_t$ as a sum of primitive, mutually orthogonal idempotent elements, which effect the projections onto these summands, is termed a **Number Theoretic Transform (NTT)** over $\mathbb{F}_t$. Arithmetic in $\mathcal{O}_t$ is that of modular polynomials when it is seen from the standpoint of the standard primitive element basis. However, the arithmetic appears as point-wise operations when it is viewed from the perspective of the basis of idempotents (i.e., in the **NTT domain**). Consequently, multiplication in the NTT domain is a great deal simpler and more efficient than it is in the primitive element basis. In point of fact, it is more efficient to (i) transform ring elements into their NTT expressions, (ii) multiply these expressions together in a point-wise manner, and then (iii) transform the resulting products back into the primitive element basis, than it is to directly multiply elements in the primitive element basis. The speed up factor is logarithmic in the degree d of $\Phi_{2d}(x)$. The foregoing applies equally well to the ciphertext modulus q.

In WIDESEAS, we take advantage of this algebraic structure for the purpose of **batching** multiple data points together within a single plaintext. For instance, each plaintext in the decrypted response holds one data chunk per targeted selector. Additionally, we choose the ciphertext modulus q as a product of four mutually distinct "NTT-friendly" primes, so that, on the one hand, arithmetic modulo q may be (i) analyzed into mutually orthogonal components with respect to the factors of q, and then (ii) synthesized in the end to obtain results modulo q, and, on the other hand, we may take advantage of the computational gains, with respect to multiplication, attendant with NTTs.

3 WIDESEAS protocol

This section provides a high-level summary of the WIDESEAS protocol for PIR. We begin by presenting the basic (naive) approach, and then we describe various enhancements.

3.1 Naive approach

The naive approach begins with the **query generation phase**. Thus, the Querier encrypts a family of plaintexts $(m_i)_{i=1}^{L}$ coming from $\tau_t(\mathcal{O})$. Here, $L = 2^h$, where h is the hash size in bits, and each plaintext is either $0_{\mathcal{O}}$ or a distinct power $x^j + \Phi_{2d}(x)\mathbb{Z}[x]$, where j is such that $0 \leq j \leq d-1$. The latter case corresponds to **targeted selectors**. In this way, the Querier produces a family $\left(c^{(i)}(X)\right)_{i=1}^{L}$ of linear ciphertexts coming from $\tau_q(\mathcal{O})$. This family is termed the **query**, and each of its members is called a **query element**. The query generation phase closes with the Querier transmitting (e.g., as a flat file) the query to the Responder.

The **response generation phase** begins with the Responder receiving the query from the Querier. Using the query elements, together with the data stream

$$\left(a^{(j)}\right)_j,$$

where, for each j, the "column"

$$a^{(j)} = \left(a_i^{(j)}\right)_{i=0}^{L-1}$$

consists of L-many "chunks" of data such that $a_i^{(j)}$ comes from a database record whose selector hash is i, the Responder computes the "column sums"

$$\left(Y^{(j)}\right)_j,$$

where, for each j,

$$Y^{(j)} := \tau_q\left(\sum_{i=1}^{L} a_i^{(j)} c^{(i)}(X)\right).$$

This involves plaintext-ciphertext multiplication along with ciphertext addition. The family $\left(Y^{(j)}\right)_j$ of column sums obtained in this way is termed the **response**. The response generation phase closes with the Responder transmitting (e.g., as a flat file) the response to the Querier, so that it may be decrypted.

Finally, the **response processing phase** begins with the Querier receiving the response from the Responder. Using the private encryption key, the Querier decrypts each element of the response, and then parses the resulting plaintexts to extract the queried data.

The main drawback to this approach is that the query must be of the same size (i.e., L) as the database, and query sizes are significantly larger with BFV than they are with Paillier, due to the fact that BFV ciphertexts are much larger than Paillier ciphertexts (e.g., 0.5 MiB vs. 768 B).

3.2 Shrinking the query

By using the fully homomorphic properties of BFV it is possible to reduce the query size. In fact, the WIDESEAS protocol uses ciphertext multiplications for the purpose of effecting logical conjunctions of query terms. This has the effect of reducing the query size, but it comes at the cost of requiring the Responder to **expand** the query prior to processing it into the response. In WIDESEAS, the expanded query is formed from four subqueries. We assume throughout that the hash size in bits is $h = 32$, since that is the default setting in WIDESEAS. However, all that is required is for h to be divisible by four, since the unexpanded query will consist of four equally-sized subqueries.

Suppose there are T targeted selectors $(s_m)_{m=0}^{T-1}$, where $1 \leq T \leq d = 2^k$. By replacing T with the smallest divisor of d that is greater than or equal to T, and then padding by zeros as necessary, we may assume that T divides d. For each m such that $0 \leq m \leq T-1$, the selector s_m shall correspond to the slots numbered $m\left(dT^{-1}\right)$ through $(m+1)\left(dT^{-1}\right)-1$ in the plaintext NTT domain. Let

$$\left(a_m 2^{24} + b_m 2^{16} + c_m 2^8 + d_m 2^0\right)_{m=0}^{T-1}$$

be the 32-bit hash values corresponding to the targeted selectors. For simplicity, we assume these hash values are pair-wise distinct. Our query shall consist of the encryptions of the plaintexts in the four lists

$$\begin{aligned} L^{(a)} &= \left(p_i^{(a)}\right)_{i=0}^{2^8-1}, \\ L^{(b)} &= \left(p_j^{(b)}\right)_{j=0}^{2^8-1}, \\ L^{(c)} &= \left(p_k^{(c)}\right)_{k=0}^{2^8-1}, \\ L^{(d)} &= \left(p_l^{(d)}\right)_{l=0}^{2^8-1}. \end{aligned}$$

Here, the list $L^{(a)}$ is defined in such a way that, for each m such that $0 \leq m \leq T-1$, and, for each i such that $0 \leq i \leq 2^8-1$, the slots numbered $m\left(dT^{-1}\right)$ through $(m+1)\left(dT^{-1}\right)-1$ in the NTT domain expression for the plaintext $p_i^{(a)}$ all are occupied either by $0_{\mathbb{F}_t}$ or $1_{\mathbb{F}_t}$, with the latter case holding, if, and only if, $i = a_m$. The three other lists are similarly defined. Next, consider the

Kronecker products

$$L^{(a)} \otimes L^{(b)} = \left(\tau_t \left(p_i^{(a)} p_j^{(b)} \right) \right)_{i,j=0}^{2^8-1}$$

and

$$L^{(c)} \otimes L^{(d)} = \left(\tau_t \left(p_k^{(c)} p_l^{(d)} \right) \right)_{k,l=0}^{2^8-1}.$$

For each $0 \leq m \leq T-1$, the slots numbered $m\left(dT^{-1}\right)$ through $(m+1)\left(dT^{-1}\right)-1$ in the NTT domain expression for the plaintext $\tau_t\left(p_i^{(a)} p_j^{(b)}\right)$ all are occupied either by $0_{\mathbb{F}_t}$ or $1_{\mathbb{F}_t}$, with the latter case holding, if, and only if, $i = a_m$ and $j = b_m$. Similarly, the slots numbered $m\left(dT^{-1}\right)$ through $(m+1)\left(dT^{-1}\right)-1$ in the NTT domain expression for the plaintext $\tau_t\left(p_k^{(c)} p_l^{(d)}\right)$ all are occupied either by $0_{\mathbb{F}_t}$ or $1_{\mathbb{F}_t}$, with the latter case holding, if, and only if, $k = c_m$ and $l = d_m$. Finally, consider the Kronecker product

$$\left(L^{(a)} \otimes L^{(b)}\right) \otimes \left(L^{(c)} \otimes L^{(d)}\right) = \left(\tau_t \left(p_i^{(c)} p_j^{(d)} p_k^{(c)} p_l^{(d)} \right) \right)_{i,j,k,l=0}^{2^8-1}.$$

The slots numbered $m\left(dT^{-1}\right)$ through $(m+1)\left(dT^{-1}\right)-1$ in the NTT domain expression for the plaintext $\tau_t\left(p_i^{(a)} p_j^{(b)} p_k^{(c)} p_l^{(d)}\right)$ all are occupied either by $0_{\mathbb{F}_t}$ or $1_{\mathbb{F}_t}$, with the latter case holding, if, and only if,

$$\begin{aligned} i &= a_m \\ j &= b_m \\ k &= c_m \\ l &= d_m. \end{aligned}$$

Thus, through the use of a circuit having multiplicative depth equal to 2, we have expanded the four smaller lists into the full query.

The foregoing idea can be extended. In WIDESEAS, we take $h = 32 = 2^5$, hence, we might extend the above circuit to one of depth 5. This would result in a query consisting of 32 lists, each of length $2^{32/32} = 2$, rather than 4 lists, each of length $2^{32/4} = 2^8$. However, there are several downsides to using a larger multiplicative depth. In fact, let us regard the dimension d as being fixed. On the one hand, the plaintext modulus t would need to be quite small in order for the noise budget to accomodate the additional ciphertext multiplications and plaintext-ciphertext multiplications. On the other hand, in order for the system to support batching, t must be at least $1 + 2d$, so that it may meet the NTT constraint of $\pi_{2d}(t) = \pi_{2d}(1)$. Also, for a fixed q, the plaintext modulus t should be as large as possible in order to (i) maximize the plaintext processing rate, and (ii) minimize the coefficient-wise expansion rate of qt^{-1}. Thus, the depth 2 circuit employed in WIDESEAS is a kind of compromise. It reduces the query expansion time (on the Responder's side) by reducing the number of ciphertext multiplications required to expand the query, but it does this at the cost of increasing the query size.

3.3 Concrete system parameters

In this section, we give concrete parameters for an instantiation of the WIDESEAS protocol.

There are three main parameters in BFV, namely, the polynomial degree d, the ciphertext modulus q, and the plaintext modulus t. Let us look at each in its turn.

First, consider the degree d. Increasing d has the desirable effect of increasing the security level, while also having the undesirable effect of increasing the size of fresh ciphertexts. The latter effect leads to a diminished overall performance (by slowing down all ciphertext operations).

Second, consider the ciphertext modulus q. Increasing q has the desirable effect of increasing the noise budget in a fresh ciphertext, since, by Lemma 5, the initial noise budget is on the order of $\log_2\left(qt^{-1}d^{-2^{-1}}\right)$ bits, while also having the undesirable effect of decreasing the security level. Thus, if a large noise budget is required for a complicated computation, then a large ciphertext modulus needs to be used, and the reduction in the security level must be countered by simultaneously increasing the degree d, even though this will result in a worse performance.

Finally, consider the plaintext modulus t. Increasing t has the undesirable effect of decreasing the noise budget in fresh ciphertexts, and it also has the undesirable effect of consuming more of the noise budget under homomorphic multiplications. In fact, by Lemma 9, plaintext-ciphertext multiplication consumes up to around $\log_2(dt)$ bits, while, according to Lemma 10, ciphertext multiplication consumes up to around $\log_2\left(d^2t\right)$ bits. Thus, it is important always to choose minimal t.

To make parameter selection easier for the user, the SEAL library has constructed sets of largest safe ciphertext moduli for 128-bit and 192-bit security levels for various choices of the degree d. These default parameters follow the recommendations found in the Security Standard Draft available at `http : //HomomorphicEncryption.org`.

We selected the following parameter set at the 128-bit security level for use in WIDESEAS:

d	2^{13}
q_0	36,028,797,005,856,769 (55 bits)
q_1	36,028,797,001,138,177 (55 bits)
q_2	18,014,398,492,704,769 (54 bits)
q_3	18,014,398,491,918,337 (54 bits)
q	421,249,165,509,532,207,033,449,784,325,084,270,503,814,638,792,416,755,348,439,564,289 (218 bits)
t	4,295,049,217 (33 bits)
$r_t(q)$	661,215,791 (30 bits)
h	2^5
L	2^h
expansion factor	$\log_t(q) \approx 7$

Table 7: BFV Parameters

This parameter set supports a data chunk size of up to

$$d\left(\log_{2^8}(t)\ \text{B}\right) \approx 2^{15}\ \text{B}.$$

It supports a hash size of up to 32 bits (i.e., $h = 2^5$). Each query can support up to $\phi(2d) = d$ targeted selectors at a time. We note that the expansion factor is 3.5 times greater than that of the Paillier-based version of EncryptedQuery. Accordingly, the sizes of the raw (i.e., fully expanded) query and the raw (i.e., not fully modulus switched) response are larger than those in Paillier. In fact, with $h = 32$, the raw query size is around 2 PiB, as compared with 3 TiB for Paillier. However, the raw query size can be reduced using the method from §3.2 based on logical conjunctions, via ciphertext multiplication. This is the approach followed in WIDESEAS, and, for the system parameter set given above, it reduces the query size from 2 PiB down to around 0.5 GiB. This is $6,144$ times smaller than the corresponding query size with Paillier. (Note: The Responder only partially expands the query, resulting in two lists, each consisting of 2^{16}-many linear ciphertexts. This partially expanded query takes up 64 GiB of storage, which still is much better than the 2 PiB required to store the fully expanded query. In point of fact, the Responder can actually expand the query up the point at which there are three lists, two of length 2^8 and one of length 2^{16}. This cuts the necessary storage down to around 32 GiB.) Likewise, the raw response size can be reduced using a technique called modulus switching, which essentially discards all ciphertext information lying below the noise floor. Again, this is the approach followed in WIDESEAS.

The above ciphertext modulus q is SEAL's default modulus for $d = 2^{13}$. This is the largest possible ciphertext modulus at the 128-bit security level. This modulus is the product of 4 primes, namely, two of size 55 bits and two of size 54 bits. As each prime factor is stored in a single 64-bit machine word, it follows that the size of a fresh ciphertext is $(2d)\,(32\ \text{B}) = 0.5$ MB.

We have chosen the plaintext modulus $t = 4,295,049,217$ because it is the smallest NTT-friendly prime greater than 2^{32}, and, as such, it allows us to store four bytes of plaintext per coefficient while also reaping all the benefits of batching.

In the polynomial domain, a plaintext is a polynomial of degree less than d with coefficients from $\mathbb{F}_t$. In the NTT domain, a plaintext is an array of d **slots**, or components, each of which contains an element of $\mathbb{F}_t$ (i.e., $(1 + \lfloor \log_2(t) \rfloor)$ bits). In WIDESEAS, plaintexts usually begin in the NTT domain, and are transformed to the polynomial domain only for operational reasons. Thus, for $d = 8192$, $t = 32$ bits, and a single targeted selector, each plaintext accumulation will process

$$(2^{13}) \times (4\ \text{B}) = 32\ \text{KiB}$$

of data. For larger numbers of selectors, the plaintext may need to be fragmented. For example, if there are 32 targeted selectors, then the plaintext can hold

$$\left(2^{13}/2^5\right)\left(2^2\ \text{B}\right) = 2^{10}\ \text{B}$$

per selector.

3.4 Noise growth

In WIDESEAS, there are four sources of noise: ciphertext multiplication, relinearization, plaintext-ciphertext multiplication, and ciphertext addition. Analyzing the growth of the noise growth during a sequence of operations is non-trivial. On the one hand, there are analytic methods for bounding the noise (e.g., using either the infinity norm or the canonical norm), which we have already discussed. On the other hand, the noise budget can be computed either exactly (from a knowledge of the private key and the underlying plaintext) or approximately (from a knowledge of just the private key). This can be done during the design phase for the purpose of validating parameters. The following table gives mean noise budget approximations (computed from a knowledge of just the private key) as measured along the WIDESEAS circuit:

0.	initialization	174
1.	ciphertext multiplication (level-1)	129
2.	relinearization	119
3.	plaintext-ciphertext multiplication	114
4.	ciphertext addition	98
5.	ciphertext multiplication (level-2)	53
6.	relinearization	53
7.	ciphertext addition	37

Figure 2: Average noise budget consumption along WIDESEAS circuit

Thus, with the given parameters, we find that, on average, ciphertext multiplication consumes around 45 bits of the noise budget, while plaintext multiplication consumes around 3 bits. The first relinearization operation costs around 10 bits, but subsequent relinearizations are essentially free. The ciphertext additions (used in computing the column sums) cost a total of around 32 bits, since there are 2^{32} additions per column sum.

3.5 Query size

Let h be the hash size in bits. An **unexpanded** query consists of d lists, each consisting of $2^{h/d}$ ciphertexts. Such a query is sent by the Querier to the Responder, who, in turn, must expand it by taking the Kronecker product of the d lists. This results in a list of $\left(2^{h/d}\right)^d = 2^h$ ciphertexts, which is the **expanded** query.

For example, if $h = 32$ and $d = 4$, then an unexpanded query consists of 4 lists of 2^8 ciphertexts each, hence, an unexpanded query has a total of $(4)\left(2^8\right) = 1024$ ciphertexts. As each ciphertext has size 0.5 MiB under the standard WIDESEAS parameter set, it follows that, in this case, an unexpanded query has size (1024)(0.5 MiB) = 0.5 GiB. Compared with the size of a fully expanded query, which consists of 2^{32} ciphertexts, this yields a reduction factor of $2^{10}/2^{32} = 2^{-22}$. Again, if $h = 20$ and $d = 4$, then an unexpanded query consists of 4 lists of 2^5 ciphertexts each, hence, an unexpanded query has a total of $(4)(2^5) = 128$ ciphertexts. Assuming each ciphertext has size 0.5 MiB, it follows that, in this case, an unexpanded query has size (128)(0.5 MiB = 64 MiB. Compared with the size of a fully expanded query, which consists of 2^{20} ciphertexts, this yields a reduction factor of $2^7/2^{20} = 2^{-13}$. Let us now compare this last example with the corresponding case using the Paillier-based WIDESKIES protocol. As this protocol does not support ciphertext multiplication (i.e., it is only partially homomorphic), we necessarily have $d = 1$. Typically, we have $h = 20 = h/d$. As each ciphertext is a 6144-bit integer, this yields a query size of 768 MiB, which is then seen to be 12 times larger than the corresponding unexpanded query size in WIDESEAS.

We note that, in WIDESEAS, the query is never actually fully expanded, at least, insofar as the fully expanded query is never expected to reside in either memory or storage at any point during processing. Rather, the Responder expands the query only up to the point at which there are just two lists, each of length $2^{h/2}$. Items coming from these two lists are multiplied as needed, and the resulting products are immediately discarded after being sent to the accumulator. without saving In point of fact, it is actually possible to expand the query only up to the point at which there three lists, two of length $2^{h/4}$ and one of length $2^{h/2}$. In any case, the Responder never needs to store the extremely large (e.g., 2 PiB, if $h = 32$) fully expanded query.

3.6 Response size

In this section, we show that the size of the response is a function of four independent variables, namely, the ciphertext expansion factor F, the bucket expansion factor G, the size D of the database entries, and the number T of query terms.

First, embedding smaller plaintext values into much larger ciphertexts leads to a ciphertext expansion factor, F, which is given by the ratio of the ciphertext size to the plaintext size. In the BGV cryptosystem, the expansion factor is

$$\begin{aligned} F &= 2d\log(q)/(d\log(t)) \\ &= 2\log(q)/\log(t) \\ &= 2\log_t(q). \end{aligned}$$

In WIDESEAS, $q \approx 2^{218}$ and $t \approx 2^{32}$, hence, $F \approx 13.625$. This is much larger than the expansion factor of $\log_N\left(N^2\right) = 2$ for Paillier. Fortunately, modulus switching can be used to reduce the expansion. In fact, q is 218-bit number given by the product of four distinct (approximately) equally-sized primes, and each switching operation removes one of these primes. Thus, each switch removes around 1/4 of the bits in q. In WIDESEAS, the modulus is switched once following the first ciphertext multiplication (during query expansion), and then it is switched twice following the full column sum. As a result of these three switches, the final coefficient modulus is around (1/4)(218 bits) $\approx$ 55 bits, and, therefore, the expansion factor is around

$$2\log_t(2^{55}) \approx 3.4375,$$

which is within a factor of 2 of Paillier's factor 2.

Second, hash collisions lead to a row expansion factor E. Thus, suppose the database has 2^g entries (i.e., records), and let the hash size in bits be h, so that the effective list size is 2^h. On average, each hash value will have

$$E = 2^g/2^h = 2^{g-h}$$

database entries as preimages. For example, if there are 2^{32} database entries and 2^{20} hash values, then each hash value will have 2^{12} entries on average. Using a larger hash size reduces the entry-wise factor E, but it also expands the query size. However, the tradeoff here turns out to be favorable. In fact, consider queries consisting of $d = 4$ lists, with hash size h. The query size is then around $4 \cdot 2^{\frac{h}{4}}$ times the size of a ciphertext. Let $s \in (1/4)\mathbb{N}$. Increasing the hash size by $4s$ is equivalent to scaling the query size by 2^s, since

$$4 \cdot 2^{\frac{4s+h}{4}} = 2^s \left(4 \cdot 2^{\frac{h}{4}}\right).$$

This, in turn, is equivalent to scaling the entry expansion factor E by $1/2^{4s}$, since

$$2^g/2^{4s+h} = \frac{1}{2^{4s}}\left(2^g/2^h\right) = \frac{1}{2^{4s}}E.$$

In particular, taking $s = 1$, we find that doubling the query size has the effect of reducing E by a factor of 2^4. As soon as $s \geq (g-h)/4$, the hash size satisfies $4s + h \geq (g-h) + h = g$. In this situation, E has been reduced by a factor of at least $2^{g-h} = E$, hence, the entry expansion factor is at most unity, and no expansion occurs. Thus, technically, we must define E via

$$E := \max(1, 2^{g-h}).$$

Third, the size of the response scales linearly with the database entry size D. If C is the plaintext chunk size, then a data entry of size D is spread across D/C ciphertexts. In fact, each ciphertext (i.e., column sum) contains exactly one data chunk per targeted selector.

Fourth, if there are T query terms, then the response size will be T times larger than for a single query term. This is to be expected, because supporting multiple query terms only amortizes the cost of query generation and the resulting query size. Note that the data chunk size (in bits) satisfies $C \approx d\log_2(t)T^{-1}$, hence,

$$D/C \approx DT/\left(d\log_2(t)\right).$$

Finally, we combine these four factors in order to estimate the response size under the WIDESEAS protocol:

Lemma 11. *Let h be the hash size in bits. A query made with T targeted selectors on a database consisting of 2^g entries, each of length D bits, results in a response of size*

$$FEDT = 2\log_t(q)\max(1, 2^{g-h})DT \text{ bits.}$$

Proof. There are E hits per hash value, since, for each hash value, there are E database entries whose tag hashes to the given value. Each of these E hits corresponds to D/C ciphertexts, where C is the data chunk size (in bits), since a single data chunk for each targeted selector is stored in each ciphertext. But, $C \approx (d\log_2(t))/T$, since each plaintext holds $d\log_2(t)$ bits in total, and these bits are shared evenly between the T targeted selectors. Thus, each of the E hits per hash value corresponds to $D/\left((d\log_2(t))/T\right) = (DT)/\left(d\log_2(t)\right)$ ciphertexts. This yields a total of

$$E\left((DT)/\left(d\log_2(t)\right)\right) = (EDT)/\left(d\log_2(t)\right)$$

ciphertexts in the response. As each ciphertext has size $2d\log_2(q)$ bits, it follows that the total number of bits in the response is given by

$$\left(2d\log_2(q)\right)\left((EDT)/\left(d\log_2(t)\right)\right).$$

But, as the ciphertext expansion factor F satisfies

$$F = \left(2d\log_2(q)\right)/\left(d\log_2(t)\right),$$

we deduce that the total number of bits in the response may be written as $FEDT$, as claimed. $\square$

For example, if $g = 32 = h$, then the WIDESEAS scheme has a combined FE expansion factor of around 3.44, whereas, in the WIDESKIES scheme, this factor is around

$$2\max(1, 2^{32-20}) = 2^{13},$$

which is more than two thousand times larger.

4 Homomorphic optimizations

In this section, we discuss certain optimizations we have made to the basic WIDESEAS computation described above. These enhancements, all of which are implemented in EncryptedQuery, do not change the underlying algorithm, but they greatly improve its performance.

4.1 Caching subproducts

We recall the response generation phase. First, the Responder receives the unexpanded query

$$\left((c_{n,m})_{m=0}^{2^8-1} \right)_{n=1}^{4}$$

from the Querier. Second, the Responder expands the query by computing and storing the products

$$(c_{1,i} c_{2,j} c_{3,k} c_{4,l})_{i,j,k,l=0}^{2^8-1} .$$

In order to minimize the multiplicative depth of the circuit, and, thereby, minimize the noise budget consumption, these products are computed along a binary tree. In order to minimize the size of the ciphertexts, relinearization and modulus switching are performed after each multiplication. Third, the Responder processes one or more columns of data chunks drawn from a database. With each such column, the data is initially encoded as plaintexts $(p_{i,j,k,l})_{i,j,k,l=0}^{2^8-1}$ in the NTT domain, which then are transformed by the inverse NTT to yield $(\tilde{p}_{i,j,k,l})_{i,j,k,l=0}^{2^8-1}$. Here, $\tilde{p}$ is the inverse NTT encoding of the plaintext data p. Finally, the column sum

$$C = \sum_{i,j,k,l} \tilde{p}_{i,j,k,l} c_{1,i} c_{2,j} c_{3,k} c_{4,l}, \tag{10}$$

is computed. The collection of column sums thus computed constitutes the Response, which the Responder transmits to the Querier for processing.

The most expensive part (i.e., the bottleneck) of the response generation is the ciphertext multiplication. It would be convenient to hold the family of products

$$(c_{1,i} c_{2,j} c_{3,k} c_{4,l})_{i,j,k,l=0}^{2^8-1}$$

in memory. However, this turns out to be infeasible. In fact, since each of the four component families

$$\left((c_{n,m})_{m=0}^{2^8-1} \right)_{n=1}^{4}$$

has 2^8 members, it follows that the product family has $\left(2^8\right)^4 = 2^{32}$ members. Since each member of the product family is a single (relinearized) ciphertext, it follows that each such member has size

$$(2^1)(2^{13})(2^5 \text{ B}) = 2^{19} \text{ B}.$$

Therefore, the product family has size

$$(2^{32})(2^{19} \text{ B}) = 2^{51} \text{ B} = 2 \text{ PiB}.$$

This is too large to hold in memory. Instead, the Responder forms two tables

$$\left(c_{12,i+j2^8}\right)_{i,j=0}^{2^{16}-1} := (c_{1,i} c_{2,j})_{i,j=0}^{2^8-1}$$

and

$$\left(c_{34,k+l2^8}\right)_{k,l=0}^{2^8-1} := \left(c_{3,k}c_{4,l}\right)_{k,l=0}^{2^8-1}$$

of relinearized subproducts. Each of these tables has 2^{16} entries, hence, each has size 32 GiB. After forming these lists, the Responder then switches the modulus by one prime factor. This reduces the modulus from four prime factors to just three prime factors, hence, it reduces the size of each list by a factor of 3/4, so that each list then has size 24 GiB. This is small enough to fit the inner list $\left(c_{34,k+l2^8}\right)_{k,l=0}^{2^8-1}$ on a single multi-core node. The outer list $\left(c_{12,i+j2^8}\right)_{i,j=0}^{2^8-1}$ can be computed once and shared across columns, or, it can be computed on the fly, since it is not the bottleneck.

4.2 Summing products

Using the two expanded lists, the column computation becomes

$$C = \sum_{i,j=0}^{2^{16}-1} \tilde{p}_{i,j} c_{12,i} c_{34,j}. \tag{11}$$

This sum has 2^{32} terms. If it is computed naively, then it requires 2^{32} ciphertext multiplications and 2^{32} plaintext multiplications. Now, it turns out that ciphertext multiplication is around 7 times slower than plaintext multiplication. In fact, with the given system parameters, each ciphertext multiplication can be performed in around 21ms, whereas each plaintext multiplication can be performed in around 3ms. It follows that 2^{32} ciphertext multiplications corresponds to around $7 \cdot 2^{32} \approx 2^{35}$ plaintext multiplications. Thus, ciphertext multiplication is the computational bottleneck.

We reduce the number of ciphertext multiplications, and, thereby, move the bottleneck on to the plaintext multiplications, by using the distributive law to rewrite the above sum as

$$C = \sum_{i=0}^{2^{16}-1} c_{12,i} \sum_{j=0}^{2^{16}-1} \tilde{p}_{i,j} c_{34,j}. \tag{12}$$

Computed according to this form, the sum still performs 2^{32} plaintext multiplications, but now it performs just 2^{16} ciphertext multiplications, which corresponds to around $7 \cdot 2^{16} \approx 2^{19}$ plaintext multiplications. It follows that plaintext multiplication is now the bottleneck. In the next two sections, we present techniques for speeding up this operation.

4.3 NTT caching

In Equation 12, the ciphertexts $(c_{34,j})_{j=0}^{2^{16}-1}$ can be stored in the NTT domain. In fact, the algorithm for plaintext multiplication involves operation (e.g., rounding) that fails to commute with the CRT. Thus, the inner loop computation consists of NTT transforming the plaintext, performing a pointwise product between the transformed plaintext and the NTT-domain ciphertext, and then accumulating the result. Unfortunately, the outer loop computation cannot be performed in the same way. More specifically, the ciphertexts $(c_{12,i})_{i=0}^{2^{16}-1}$ cannot be stored in the NTT domain. This is because the algorithm for ciphertext multiplication requires that the CRT basis be extended by several supplemental primes, so that, effectively, the multiplication can be performed over the integers, and then scaled by t/q and rounded.

4.4 Lazy reduction

The multiply-accumulate operation can (and does) use lazy reduction. To see this, we start by observing that this operation is actually performed separately modulo each of the (four) CRT primes, and each such prime fits entirely within a single machine word (i.e., 64 bits), since the primes are at most 56 bits long. Fix any one of the four primes. We deduce that the product of any two residues modulo this prime fits comfortably within two machine words. Now, the SEAL library uses generic Barrett reduction to reduce the result of the multiplication. Barrett reduction (or Montgomery reduction) converts the modular operation into several multiplication operations along with shifting operations. These multiplications can be amortized by summing multiple products and then performing a single reduction. The GCC C extension `__int128` type makes this operation simple to implement, and the compiler generates fast multiplication and addition-with-carry instructions.

4.5 Periodic NTT's

During the response generation phase of the WIDESEAS protocol, data chunks drawn from database records are embedded into ciphertexts, via plaintext-ciphertext multiplication, as follows.

First, each data chunk is inserted into a plaintext in the NTT domain. Suppose there are T targeted selectors, where, without loss of generality, T is a divisor of d. Each selector is allotted d/T NTT slots, hence, each data chunk consists of d/T four byte values (since t is just over four bytes in length). As the Responder generally does not know whether or not a given data chunk actually corresponds to one of the targeted selectors, it follows that the Responder must attempt to embed each data chunk into its corresponding query element. Just in case a given data chunk does, in fact, correspond to a targeted selector, the Responder must embed it into each and every (d/T)-long window of NTT slots. These T-many slots correspond in some order to the T-many targeted selectors. In this way, the Responder covers all possibilities. In detail, consider the data chunk $(a_i)_{i=0}^{d/T}$, where each a_i is a four byte value embedded in $\mathbb{F}_t$, The Responder repeats this data chunk T times, so that it then covers all d NTT slots. The plaintext $\widetilde{p}$ thus formed has period d/T and frequency T (its period repeats T times). The Responder then applies the inverse NTT to this plaintext, resulting in the polynomial plaintext p. Since the NTT domain expression has frequency T, it follows from standard properties of the NTT that p has its support contained within $T\mathbb{Z}/d\mathbb{Z}$. In fact, the d-point inverse NTT applied to the periodic extension of the data chunk is equivalent to the (d/T)-point inverse NTT applied to just the data chunk itself. Thus, let $(d_j)_{j=0}^{(d/T)-1} \in \mathbb{F}_t$ be a data chunk consisting of (d/T)-many four byte values. Periodically extend this sequence so that its frequency is T, and embed the resulting sequence in an NTT domain plaintext

$$\tilde{p} := (e_i)_{i=0}^{d-1}.$$

For each j such that $0 \leq j \leq (d/T)-1$, and, for each k such that $0 \leq k \leq T-1$, we have $e_{k(d/T)+j} = d_j$. Let m be a natural number such that $0 \leq m \leq d-1$. Denote by $\omega_{2d} \in \mathbb{F}_t$ a primitive $(2d)$-th root of unity. If the inverse NTT is applied to $\tilde{p}$, then the coefficient of x^g, for any g such that $0 \leq g \leq d-1$, is

$$\begin{aligned}
d^{-1}\sum_{m=0}^{d-1} e_m \left(\omega_{2d}^{2m+1}\right)^g &= d^{-1}\sum_{k=0}^{T-1}\sum_{j=0}^{(d/T)-1} e_{k(d/T)+j}\left(\omega_{2d}^{2(k(d/T)+j)+1}\right)^g \\
&= d^{-1}\sum_{k=0}^{T-1}\sum_{j=0}^{(d/T)-1} d_j \left(\omega_T^k \omega_d^j \omega_{2d}\right)^g \\
&= d^{-1}\left(\sum_{k=0}^{T-1}\left(\omega_T^g\right)^k \sum_{j=0}^{(d/T)-1} d_j \left(\omega_d^g\right)^j\right)\omega_{2d}^g \\
&= d^{-1}\left(\sum_{k=0}^{T-1}\left(\omega_T^g\right)^k\right)\left(\sum_{j=0}^{(d/T)-1} d_j \left(\omega_d^g\right)^j\right)\omega_{2d}^g \\
&= \begin{cases} \left((d/T)^{-1}\sum_{j=0}^{(d/T)-1} d_j \left(\omega_d^g\right)^j\right)\omega_{2d}^g & \text{if } g \equiv_T 0 \\ 0 & \text{otherwise.}\end{cases}
\end{aligned}$$

It follows that only $\left(x^{lT}\right)_{l=0}^{(d/T)-1}$ have coefficients that are possibly non-zero, so that the support is contained within $T\mathbb{Z}/d\mathbb{Z}$, as claimed. Moreover, for each l such that $0 \leq l \leq (d/T)-1$, the coefficient of x^{lT} is given by

$$(d/T)^{-1}\sum_{j=0}^{(d/T)-1} d_j \left(\omega_{2(d/T)}^{2j+1}\right)^l.$$

It follows that the d-point inverse NTT is equivalent to a (d/T)-point inverse NTT, as claimed. Now, the approach based on d points requires around $d\log(d)$ operations. The approach based on (d/T) points requires around $(d/T)\log(d/T)$ operations. Thus, in case T is a proper divisor of d, the (d/T)-point approach is faster than the d-point approach by a factor of

$$T\log_{d/T}(d) = T\left(1-\log_d(T)\right)^{-1}.$$

In case $T = d$, so that $1 = d/T$, the (d/T)-point transform is trivial, hence, it is infinitely faster than the d-point approach. In the situation of WIDESEAS, we have the following speed ups coming from this optimization:

T	$\left\lfloor T\left(1 - \log_d(T)\right)^{-1}\right\rceil$
2^0	1
2^1	2
2^2	5
2^3	10
2^4	23
2^5	52
2^6	119
2^7	277
2^8	666
2^9	1,664
2^{10}	4,437
2^{11}	13,312
2^{12}	53,248
2^{13}	∞

Table 8: Speed up factors for $d = 2^{13}$

Of course, this same optimization applies dually to the forward NTT. This is quite convenient for us, since, after inverse transforming the periodic plaintext data, the very next step in the WIDESEAS protocol is to compute the forward NTT of the reductions modulo the prime factors of the ciphertext modulus q. Each of these forward NTTs can be performed as a (d/T)-point forward NTT, which is faster than a d-point forward NTT, unless $T = 1$. In WIDESEAS, q has four prime factors, hence, this savings is fourfold.

We remark that the periodicity-related property exploited above is actually a well-known property of the Discrete Fourier Transform (DFT). In fact, the d-point NTT is a twisted version (by ω_{2d}) of the d-point DFT, and, a function $f : \mathbb{Z}/d\mathbb{Z} \to \mathbb{F}_t$ has its support contained within a subgroup $H \leq \mathbb{Z}/d\mathbb{Z}$, if, and only if, its Fourier transform $\hat{f} : \widehat{\mathbb{Z}/d\mathbb{Z}} \to \mathbb{F}_t$ is constant on the cosets of the annihilator $\mathrm{Ann}(H)$ of H in the dual group $\widehat{\mathbb{Z}/d\mathbb{Z}}$.

5 Performance numbers

In this section, we present results from some experiments we have run with our implementation of WIDESEAS in EncryptedQuery. Among other things, the results highlight the reciprocal relationship between the number of targeted selectors and the response generate rate, along with the direct relationships between the number of targeted selectors and the query and response generation times. In all, we summarize (in Table 9) the results of fourteen different experiments. These experiments correspond to the fourteen possible choices for the number of targeted selectors, T. For each experiment, the values of d, q, and t are as in §3.3, but the hash size h is 20 bits, rather than 32 bits, as it is in §3.3. The database always has size 1 TB, and it always consists of 2^{20}-many records. Each record splits into 2^5T-many data chunks, and each data chunk has size $(4(d/T))$ B. In each of the fourteen different experiments, the query has size 64 MiB, and the response has size 4 MiB. The reported timings were extrapolated from timings on single columns.

T	query gen. time (s)	resp. gen. time (days)	resp. gen. rate (MiB/s)
2^0	0.41	0.51	23.67
2^1	0.51	0.64	19.01
2^2	0.49	0.89	13.67
2^3	0.56	1.36	8.93
2^4	0.66	2.40	5.06
2^5	0.97	4.42	2.75
2^6	1.39	8.64	1.41
2^7	2.22	17.22	0.71
2^8	3.73	34.53	0.35
2^9	7.39	65.56	0.19
2^{10}	17.66	131.41	0.09
2^{11}	31.66	255.35	0.05
2^{12}	54.80	527.93	0.02
2^{13}	109.59	1059.15	0.01

Table 9: Experimental results

These measurements were done using gcc-7 on an Oracle VM VirtualBox running Ubuntu 18.04.1 LTS (Bionic Beaver) on a MacBook Pro laptop running MacOS Sierra using a 2.8 GHz Intel core i7 and with Turbo Boost disabled.

5.1 A comparison with SEAL PIR

We first recall the basic PIR protocol followed by Microsoft in their SEAL PIR paper [5].

The Querier chooses a single targeted selector $0 \leq j < d$, and then encrypts the plaintext $\tau_t\left(x^j + \Phi_{2d}(x)\mathbb{Z}[x]\right)$. This sole ciphertext constitutes the entirety of the query. The Querier then sends this query to the Responder, who proceeds to expand it. The effect of the expansion is to create d-many ciphertexts from the query ciphertext, with the i-th such ciphertext encrypting either zero or one, where the latter case holds, if, and only if, $i = j$. Now, in order to explain how query expansion is performed, we must first recall several mathematical results.

Let $K = \mathbb{Q}[x]/\Phi_{2d}(x)\mathbb{Q}[x]$ be the $2d$-th cyclotomic extension of $\mathbb{Q}$. Let

$$f(x) + \Phi_{2d}(x)\mathbb{Q}[x] = \sum_{k=0}^{d-1} a_k x^k + \Phi_{2d}(x)\mathbb{Q}[x]$$

be an element of K. Let $G = \langle g \rangle$ be a cyclic group of order d, and denote by $K[G]$ the group algebra of G over K. Recall that $K[G]$ consists precisely of the K-valued functions on G. Write

$$\operatorname{Gal}(K/\mathbb{Q}) = (\gamma_r)_{r=0}^{d-1},$$

where, for each r,

$$\gamma_r : f(x) + \Phi_{2d}(x)\mathbb{Q}[x] \mapsto f\left(x^{2r+1}\right) + \Phi_{2d}(x)\mathbb{Q}[x].$$

Consider the K-valued function F on G mapping $g^r \in G$ to the r-th Galois conjugate of f, that is to say,

$$F\left(g^r\right) = f\left(x^{2r+1}\right) + \Phi_{2d}(x)\mathbb{Q}[x].$$

In terms of the standard basis for $K[G]$,

$$\begin{aligned} F &= \sum_{r=0}^{d-1} F\left(\gamma_r\right) \delta_G\left(g^r\right) \\ &= \sum_{r=0}^{d-1} \left(f\left(x^{2r+1}\right) + \Phi_{2d}(x)\mathbb{Q}[x]\right) \delta_G\left(g^r\right). \end{aligned}$$

As $\omega_d = x^2 + \Phi_{2d}(x)\mathbb{Q}[x]$ is in K, it follows that K is a splitting field for G. Therefore, we may compute the Fourier transform of F. Thus, let $\widehat{G} = (\chi_k)_{k=0}^{d-1}$ be the group of simple K-valued characters on G, where, for each k,

$$\chi_k : g \mapsto \omega_d^k = x^{2k} + \Phi_{2d}(x)\mathbb{Q}[x].$$

The function F has the Fourier expansion

$$F = \sum_{k=0}^{d-1} \widehat{F}\left(\chi_k^{-1}\right) d^{-1} \chi_k,$$

where, for each k,

$$\begin{aligned} \widehat{F}\left(\chi_k^{-1}\right) &= \sum_{r=0}^{d-1} F\left(g^r\right) \chi_k^{-1}\left(g^r\right) \\ &= \sum_{r=0}^{d-1} f\left(x^{2r+1}\right)\left(\omega_d^{-k}\right)^r + \Phi_{2d}(x)\mathbb{Q}[x] \\ &= \sum_{r=0}^{d-1} f\left(x^{2r+1}\right)\left(x^{-2k}\right)^r + \Phi_{2d}(x)\mathbb{Q}[x]. \end{aligned} \tag{13}$$

The following lemma shall be used to simplify these coefficients.

Lemma 12. *(Simpson's multi-section formula [6]) Let* $g(y) = \sum_i b_i y^i$ *be any formal series in* y *over a commutative ring* R*. Assume* $\omega_m \in R$ *is a primitive* m*-th root of unity. If* $0 \leq k < m$*, then*

$$m^{-1} \sum_{r=0}^{m-1} g\left(\omega_m^r y\right)\left(\omega_m^{-k}\right)^r = \sum_{i \equiv_m k} b_i y^i.$$

Proof. We compute

$$\begin{aligned} m^{-1} \sum_{r=0}^{m-1} g\left(\omega_m^r y\right)\left(\omega_m^{-k}\right)^r &= m^{-1} \sum_{r=0}^{m-1} \left(\sum_i b_i \left(\omega_m^r y\right)^i\right)\left(\omega_m^{-k}\right)^r \\ &= m^{-1} \sum_{r=0}^{m-1} \sum_i \omega_m^{ri-rk} b_i y^i \\ &= \sum_i m^{-1} \left(\sum_{r=0}^{m-1} \left(\omega_m^{i-k}\right)^r\right) b_i y^i \\ &= \sum_{i \equiv_m k} b_i y^i, \end{aligned}$$

where the last equality follows from the fact that $\sum_r \left(\omega_m^{i-k}\right)^r$ is either zero or m, with the latter case holding, if, and only if, $i-k$ is congruent to zero modulo m. ☐

Specializing this result to our situation, and fixing $0 \leq k < d$, we find, working modulo $\Phi_{2d}(x)$,

$$\begin{aligned} a_k x^k &= \sum_{i \equiv_d k} a_i x^i \\ &= d^{-1} \sum_{r=0}^{d-1} f\left(\omega_d^r x\right)\left(\omega_d^{-k}\right)^r \\ &= d^{-1} \sum_{r=0}^{d-1} f\left(x^{2r} x\right)\left(x^{-2k}\right)^r \\ &= d^{-1} \sum_{r=0}^{d-1} f\left(x^{2r+1}\right)\left(x^{-2k}\right)^r, \end{aligned}$$

which recovers the frequency-k Fourier coefficient of F computed above in (13), up to scalar multiplication by d^{-1}. We deduce that the Fourier expansion of F

may be written as

$$\begin{aligned} F &= \sum_{k=0}^{d-1} \left(d^{-1} \sum_{r=0}^{d-1} f\left(x^{2r+1}\right) \left(x^{-2k}\right)^{r} + \Phi_{2d}(x)\mathbb{Q}[x] \right) \chi_k \\ &= \sum_{k=0}^{d-1} \left(a_k x^k + \Phi_{2d}(x)\mathbb{Q}[x]\right) \chi_k. \end{aligned}$$

Effectively, this procedure extracts the terms of $f(x)$, which, essentially, is the same thing as extracting the coefficients of $f(x)$. It follows that this procedure is akin to the "divided power" derivations introduced by Teichmüller [7]. We record this result in the following lemma:

Lemma 13. *Let* $K = \mathbb{Q}[x]/\Phi_{2d}(x)\mathbb{Q}[x]$ *be the* $2d$*-th cyclotomic extension of* $\mathbb{Q}$*. Let*

$$f(x) + \Phi_{2d}(x)\mathbb{Q}[x] = \sum_{k=0}^{d-1} a_k x^k + \Phi_{2d}(x)\mathbb{Q}[x]$$

be an element of K*. Let* $G = \langle g \rangle$ *be a cyclic group of order* d*, and let* $K[G]$ *be the group algebra of* G *over* K*. Let* $F \in K[G]$ *be the* K*-valued function on* G *mapping* $g^r \in G$ *to the* r*-th Galois conjugate of* f*, that is to say,*

$$F\left(g^r\right) = f\left(x^{2r+1}\right) + \Phi_{2d}(x)\mathbb{Q}[x],$$

for all $0 \leq r < d$*. The Fourier expansion of* F *is given by*

$$F = \sum_{k=0}^{d-1} \left(a_k x^k + \Phi_{2d}(x)\mathbb{Q}[x]\right) \chi_k.$$

Now, in case $f(x)$ corresponds to the distinguished "targeted selector" monomial x^j, we deduce from Simpson's formula

$$\widehat{F}\left(\chi_k^{-1}\right) \equiv_{\Phi_{2d}(x)} \begin{cases} 0 & \text{if } k \not\equiv_d j \\ x^j & \text{if } k \equiv_d j. \end{cases}.$$

The SEAL PIR query expansion algorithm computes this Fourier transform homomorphically. Moreover, the k-th Fourier coefficient is scaled (via plaintext-ciphertext multiplication) by x^{-k}, yielding, for general $f(x)$,

$$\widehat{F}\left(\chi_k^{-1}\right) x^{-k} \equiv_{\Phi_{2d}(x)} a_k,$$

which, in case $f(x) = x^j$, gives

$$\widehat{F}\left(\chi_k^{-1}\right) x^{-k} \equiv_{\Phi_{2d}(x)} \begin{cases} 0 & \text{if } k \not\equiv_d j \\ 1 & \text{if } k \equiv_d j. \end{cases}.$$

The k-th ciphertext thus obtained encrypts either zero or one, with the latter case holding, if, and only if, k and j are congruent modulo d. We note that this final normalization is not essential, insofar as the Querier may alternatively handle the wrap around during extraction. Nevertheless, scaling by x^{-k} is inexpensive, inasmuch as it is effected by rotations with possible sign changes.

Upon expanding the query, the Responder proceeds to use plaintext-ciphertext multiplication to embed data into each element of the expanded query and then the Responder sums these products. By construction, the resulting sum encrypts the data corresponding to the targeted selector. This sum constitutes the response, which is then sent to the Querier for decryption and data extraction. Using Microsoft's reference implementation[3], we obtained the following performance numbers.

entries	entry (B)	degree	$\log_2(t)$ (bits)	DB dim.	query (MiB)	response (MiB)	query gen. (s)	response gen. (days)
2^{20}	2^{10}	2^{12}	12	2	0.125	640	0.002	0.66
2^{20}	2^{10}	2^{13}	12	2	0.0625	1280	0.003	0.55
2^{20}	2^{8}	2^{11}	12	2	0.0625	1280	0.001	0.54
2^{20}	2^{8}	2^{12}	12	2	0.125	2560	0.004	1.65
2^{20}	2^{8}	2^{13}	12	2	0.25	5120	0.004	1.54

Table 10: SEAL PIR results

By comparison, WIDESEAS has query and response sizes of 64 MiB and 4 MiB, respectively. Thus, the query size in SEAL PIR is 1.6 to 6.6 times smaller than that in WIDESEAS, while the response size in WIDESEAS is 160 to 1280 times smaller than that in SEAL PIR. The response generation in WIDESEAS is 1.06 to 3.24 times faster than that in SEAL PIR. In order to accomodate multiple selectors, Microsoft's team uses probabilistic batch codes (PBC). However, Microsoft's implementation of SEAL PIR does not include this capability. Alternatively, we can generalize the single selector approach to multiple selectors in a straightforward fashion. Thus, we simply have the query be an encryption of the sum of all the powers of x that correspond to selectors, that is to say, we essentially encrypt the characteristic function of the set of targeted selectors. The Fourier transform then splits this into encryptions of the individual powers, which is exactly what we want. For example, if the targeted selectors correspond to indices 3, 11, 30 and 55, then we simply encrypt the plaintext polynomial $x^3 + x^{11} + x^{30} + x^{55}$. When we apply the Fourier transform to this ciphertext, we obtain a list of encrypted Fourier coefficients, all of which encrypt zero, except for the frequency-3, 11, 30 and 55 coefficients, which encrypt x^3, x^{11}, x^{30} and x^{55}, respectively. We can, therefore, plaintext multiply all coefficients by their corresponding data chunks, and then perform the column sum, as usual.

---

[3] `https://github.com/microsoft/SealPIR`

6 Conclusion

This report has introduced the WIDESEAS protocol for lattice-based Private Information Retrieval (PIR), and we have given performance numbers for its recent implementation in the EncryptedQuery open-source PIR software. This protocol uses the fully homomorphic Brakerski–Fan–Vercauteren (BFV) encryption scheme, as opposed to the Paillier scheme, which is used in all earlier versions of EncryptedQuery. We have shown that the homomorphic capabilities of BFV result in smaller query sizes (due to a query-shrinking technique based on batching and ciphertext multiplication), higher response generation rates (due to using relinearization to keep ciphertexts small; due to caching certain products of query elements in the NTT domain; due to using the distributive law to achieve a quadratic reduction in the total number of ciphertext multiplications; due to using lazy reduction to speed up modular multiplies; and, due to exploiting properties of inverse NTTs over periodic data, and forward NTTs over sparse data, for the purpose of accelerating plain multiplications), and comparable response sizes (due to using modulus switching to discard redundant ciphertext information prior to transmitting the response). For instance, running a single thread (with Turbo Boost disabled) on a MacBook Pro equipped with a 2.8 GHz Intel core i7, and using a 20-bit hash and a 2^{15}-byte data chunk size (which allows us to search for a single targeted selector), our implementation can (i) generate a query of size 64 MiB in around 0.41 seconds, (ii) process a query against a 1 TiB database (comprised of 2^{20}-many 1 MiB records) at a rate of 23.67 MiB/s (which is at least two orders of magnitude faster than the Paillier-based version of EncryptedQuery), and (iii) generate a response of size 4 MiB in around 0.51 days. We expect a speed up on server class machines. Our implementation uses the Microsoft SEAL library along with a small amount of custom code.

A Group algebras and Fourier analysis

In this section, we recall the notion of Fourier analysis over finite cyclic groups.

Let G be a finite cyclic group of order $|G|$. Let $\mathbb{F}$ be a field of characteristic prime to $|G|$, and assume that $\mathbb{F}$ contains a primitive $|G|$-th root of unity $\omega_{|G|}$. The group algebra, $\mathbb{F}[G]$, of G over $\mathbb{F}$ is defined as the set of all $\mathbb{F}$-valued functions on G under point-wise addition (i.e., $f_1 + f_2 : g \mapsto f_1(g) + f_2(g)$) and convolutional multiplication (i.e., $f_1 * f_2 : g \mapsto \sum_{ab=g} f_1(a) f_2(b)$). The **standard** basis for the free $\mathbb{F}$-module $\mathbb{F}[G]$ is the set of characteristic functions $(\delta_G(g))_{g \in G}$ on the singleton subsets of G. The coefficients of $f \in \mathbb{F}[G]$ with respect to the standard basis are given by the values of f on G, via

$$f = \sum_{g \in G} f(g) \delta_G(g).$$

Fix any generator $g_0 \in G$, and let $\lambda_{g_0} : G \to \mathbb{Z}/|G|\mathbb{Z}$ be the (discrete) logarithm to the base g_0. Also, let $\overline{\eta} : \mathbb{Z}/|G|\mathbb{Z} \to \langle \omega_{|G|} \rangle$ denote the isomorphism of groups given by the reduction of the epimorphism $\eta : \mathbb{Z} \to \langle \omega_{|G|} \rangle$ determined by the assignment $\eta : 1 \mapsto \omega_{|G|}$. Finally, let $\varepsilon \in \mathrm{End}_{\mathbb{Z}} \left(\langle \omega_{|G|} \rangle \right)$ be the map determined by the prescription $\varepsilon : \omega_{|G|} \mapsto \omega_{|G|}$. The simple $\mathbb{F}$-valued characters on G are given by the family of group homomorphisms $(\chi_k : G \to \mathbb{F})_{k \in \mathbb{Z}/|G|\mathbb{Z}}$, where

$$\chi_k := \varepsilon^k \circ \overline{\eta} \circ \lambda_{g_0},$$

for each $k \in \mathbb{Z}/|G|\mathbb{Z}$. Here, ε^k denotes the order k point-wise power of ε. The set of these characters is, itself, a group, $\widehat{G}$, under point-wise multiplication of functions. As each character is, in particular, an $\mathbb{F}$-valued on G, it follows $\widehat{G} \subset \mathbb{F}[G]$. In point of fact, this subset is actually a basis (called the **Fourier** basis) for $\mathbb{F}[G]$ over $\mathbb{F}$. The *scaled* Fourier basis $\left(\chi |G|^{-1}\right)_{\chi \in \widehat{G}}$ consists of orthogonal primitive idempotents, hence, both addition and multiplication of functions are performed in a point-wise (i.e., coefficient-wise) manner when functions are written in this basis. The change of coordinates map from the standard basis to the scaled Fourier basis is termed a (discrete) Fourier Transform (DFT), and the coefficients of a given function $f \in \mathbb{F}[G]$ with respect to the scaled Fourier basis are termed its **Fourier coefficients**. By the orthogonality and idempotency of the characters,

$$\begin{aligned}
f &= \sum_{\chi \in \widehat{G}} f * \chi |G|^{-1} \\
&= \sum_{\chi \in \widehat{G}} \left(\sum_{g \in G} \left(\sum_{x \in G} f(x) \chi(x^{-1} g) |G|^{-1} \right) \delta(g) \right) \\
&= \sum_{\chi \in \widehat{G}} \left(\sum_{g \in G} \left(\sum_{x \in G} f(x) \chi^{-1}(x) \right) \chi(g) \delta(g) \right) |G|^{-1} \\
&= \sum_{\chi \in \widehat{G}} \left(\sum_{x \in G} f(x) \chi^{-1}(x) \right) \left(\sum_{g \in G} \chi(g) \delta(g) \right) |G|^{-1} \\
&= \sum_{\chi \in \widehat{G}} \widehat{f}\left(\chi^{-1}\right) \chi |G|^{-1},
\end{aligned}$$

where

$$\widehat{f}\left(\chi^{-1}\right) := \sum_{x \in G} f(x) \chi^{-1}(x),$$

for all $\chi \in \widehat{G}$.

B Fourier transform

In this section, we take three progressively lower-level views of the notion of the Fourier transform on the finite cyclic group G of order d. Let $\mathbb{F}$ be a splitting field for G, and write $\widehat{G}$ for the group of simple $\mathbb{F}$-valued characters on G. We identify G with its double dual using the natural isomorphism between them. The reader will observe that, for each $f \in \mathbb{F}[G]$, and, for each $\chi \in \widehat{G}$, we denote by $\widehat{f}(\chi^{-1})$ the Fourier coefficient $\sum_{x\in G} f(x)\chi^{-1}(x)$ corresponding to the character (i.e., "frequency") χ. In the literature, this same coefficient is more commonly (and more confusingly) denoted by $\widehat{f}(\chi)$.

High-level view. By the universal property of free modules, the natural inclusions of G and $\widehat{G}$ into each other's group algebras give rise to the following commutative diagram:

$$\begin{tikzcd}
 & \mathbb{F}[G] \arrow[dd, dashed, tail, bend right=50, "\widehat{\epsilon_G}"'] \arrow[dd, dashed, tail, bend left=15, "\widehat{\delta_{\widehat{G}}}"] & \\
G \arrow[ur, hook, "\delta_G"] \arrow[dr, hook', "\epsilon_G"'] & & \widehat{G} \arrow[ul, hook', "\epsilon_{\widehat{G}}"'] \arrow[dl, hook, "\delta_{\widehat{G}}"] \\
 & \mathbb{F}[\widehat{G}] \arrow[uu, dashed, tail, bend left=15, "\widehat{\delta_G}"] \arrow[uu, dashed, tail, bend right=50, "\widehat{\epsilon_{\widehat{G}}}"'] &
\end{tikzcd}$$

The map $\widehat{\epsilon_G} : \mathbb{F}[G] \to \mathbb{F}[\widehat{G}]$ is called the *Fourier transform on G*. It makes the following assignments:

$$\begin{aligned}
\delta_G(x) &\mapsto (\epsilon_G(x) : \chi \mapsto \chi(x))\\
d^{-1}\chi &\mapsto \delta_{\widehat{G}}(\chi^{-1})\\
f &\mapsto \sum_{\chi\in\widehat{G}} \widehat{f}(\chi)\delta_{\widehat{G}}(\chi) = \widehat{f}\\
\widehat{\widehat{\widehat{f}}} &\mapsto d(\widehat{f} \circ \wedge_{\widehat{G}}^{-1}) = \widehat{\widehat{\widehat{\widehat{f}}}}.
\end{aligned}$$

The map $\widehat{\epsilon_{\widehat{G}}} : \mathbb{F}[\widehat{G}] \to \mathbb{F}[G]$ is called the *Fourier transform on $\widehat{G}$*. It makes the following assignments:

$$\begin{aligned}
\delta_{\widehat{G}}(\chi) &\mapsto \chi\\
d^{-1}\epsilon_G(x) &\mapsto \delta_G(x^{-1})\\
\widehat{f} &\mapsto \sum_{x\in G} \widehat{\widehat{f}}\left(\epsilon_G(x)\right)\delta_G(x) = \widehat{\widehat{f}} = d\left(f \circ \wedge_G^{-1}\right)\\
\widehat{\widehat{\widehat{f}}} &\mapsto \widehat{\widehat{\widehat{\widehat{f}}}}.
\end{aligned}$$

The composite map $\widehat{\epsilon_{\widehat{G}}} \circ \widehat{\epsilon_G} \in \mathrm{End}_{\mathbb{F}}(\mathbb{F}[G])$ satisfies

$$\widehat{\epsilon_{\widehat{G}}} \circ \widehat{\epsilon_G} : f \mapsto d\left(f \circ \wedge_G^{-1}\right) = \widehat{\widehat{f}}.$$

Now, on the one hand, the square of this composite map takes f to $\hat{\hat{\hat{\hat{f}}}}$. On the other hand, we find, by inspection, that the square of this composite map takes f to $d^2 f$. We deduce

$$\hat{\hat{\hat{\hat{f}}}} = d^2 f.$$

The map $\widehat{\delta_G} : \mathbb{F}[\widehat{G}] \to \mathbb{F}[G]$ is called the *inverse Fourier transform on* G, although it is defined on $\mathbb{F}[\widehat{G}]$ and not on $\mathbb{F}[G]$. It makes the following assignments:

$$\begin{aligned} \epsilon_G(x) &\mapsto \delta_G(x) \\ \delta_{\widehat{G}}(\chi) &\mapsto d^{-1}(\chi \circ \wedge_G^{-1}) \\ \widehat{f} &\mapsto f \end{aligned}$$

The map $\widehat{\delta_{\widehat{G}}} : \mathbb{F}[G] \to \mathbb{F}[\widehat{G}]$ is called the *inverse Fourier transform on* $\widehat{G}$, although it is defined on $\mathbb{F}[G]$ and not on $\mathbb{F}[\widehat{G}]$. It makes the following assignments:

$$\begin{aligned} \delta_G(x) &\mapsto d^{-1}\epsilon_G(x^{-1}) \\ d^{-1}\chi &\mapsto d^{-1}\delta_{\widehat{G}}(\chi) \\ f &\mapsto d^{-1}\left(\widehat{f} \circ \wedge_{\widehat{G}}^{-1}\right) = \sum_{\chi} \widehat{f}(\chi^{-1}) d^{-1}\delta_{\widehat{G}}(\chi) \\ \hat{\hat{f}} &\mapsto \widehat{f}. \end{aligned}$$

The composite map $\widehat{\delta_G} \circ \widehat{\delta_{\widehat{G}}} \in \mathrm{End}_{\mathbb{F}}(\mathbb{F}[G])$ satisfies

$$\widehat{\delta_G} \circ \widehat{\delta_{\widehat{G}}} : f \mapsto d^{-1}(f \circ \wedge_G^{-1}) = d^{-2}\hat{\hat{f}}.$$

Now, on the one hand, the square of this composite map takes f to $d^{-4}\hat{\hat{\hat{\hat{f}}}}$. On the other hand, we find, by inspection, that the square of this composite map takes f to $d^{-2}f$. We recover the result

$$d^{-4}\hat{\hat{\hat{\hat{f}}}} = d^{-2} f,$$

that is,

$$\hat{\hat{\hat{\hat{f}}}} = d^2 f.$$

Finally, it follows that the composite map

$$\widehat{\delta_G} \circ \widehat{\delta_{\widehat{G}}} \circ \widehat{\epsilon_{\widehat{G}}} \circ \widehat{\epsilon_G} \in \mathrm{End}_{\mathbb{F}}(\mathbb{F}[G])$$

is the identity map, since it takes f onto $d^{-2}\hat{\hat{\hat{\hat{f}}}} = f$. In some approaches to Fourier analysis, the transform is viewed as an endomorphism on $\mathbb{F}[G]$ by fixing an isomorphism between G and $\widehat{G}$. Under this approach, the preceding result effectively shows that the transform always has order 4, hence, that its eigenvalues always are 4-th roots of unity, independent of the order d of the cyclic group G.

Medium-level view. The group algebra $\mathbb{F}[G]$ is a free $\mathbb{F}$-module on the set G. By the universal property of free modules, each $f \in \mathbb{F}[G]$ lifts uniquely to an $\mathbb{F}$-linear form $\widehat{f}$ on $\mathbb{F}[G]$, and, conversely, each linear form $\lambda \in (\mathbb{F}[G])^*$ is the lift of a unique $\mathbb{F}$-valued function on G, namely, the function uniquely determined by the restriction to $\delta_G(G)$ of λ, so that

$$\lambda = \widehat{\lambda \circ \nu_{\delta_G(G)} \circ \delta_G},$$

where

$$\nu_{\delta_G(G)} : \delta_G(G) \to \mathbb{F}[G]$$

is the natural inclusion map. As the mapping $f \mapsto \widehat{f}$ of $\mathbb{F}[G]$ onto its dual is $\mathbb{F}$-linear, it follows that this mapping is a non-singular bilinear form on $\mathbb{F}[G]$. Furthermore, as

$$\widehat{f_1}(f_2) = \widehat{f_2}(f_1),$$

for all f_1,$f_2 \in \mathbb{F}[G]$, this form is symmetric, so that, in particular, any matrix representing this mapping with respect to dual bases is a symmetric matrix. The basis $\left(\widehat{\delta(x)}\right)_{x \in G}$ in $(\mathbb{F}[G])^*$ is dual to the standard basis $(\delta(x))_{x \in G}$ in $\mathbb{F}[G]$, since

$$\widehat{\delta(x_1)}(\delta(x_2)) = (\delta(x_1))\,(x_2) = \begin{cases} 1 & x_2 = x_1 \\ 0 & x_2 \neq x_1 \end{cases}.$$

Similarly, the basis $(\widehat{\chi})_{\chi \in \widehat{G}}$ in $(\mathbb{F}[G])^*$ is dual to the basis of primitive orthogonal idempotents $\left(d^{-1}\chi\right)_{\chi \in \widehat{G}}$ in $\mathbb{F}[G]$, since

$$\widehat{\chi_1^{-1}}(d^{-1}\chi_2) = d^{-1} \sum_{x \in G} \chi_2(x)\chi_1^{-1}(x) = d^{-1} \sum_{x \in G} \left(\chi_2 \cdot \chi_1^{-1}\right)(x) = \begin{cases} 1 & \chi_2 = \chi_1 \\ 0 & \chi_2 \neq \chi_1 \end{cases}.$$

It follows that for each $f \in \mathbb{F}[G]$,

$$\sum_{x \in G} \widehat{\delta(x)}(f)\delta(x) = f = \sum_{\chi \in \widehat{G}} \widehat{\chi^{-1}}(f) d^{-1}\chi$$

and

$$\sum_{x \in G} \widehat{f}\left(\delta(x)\right) \widehat{\delta(x)} = \widehat{f} = \sum_{\chi \in \widehat{G}} \widehat{f}\left(d^{-1}\chi^{-1}\right) \widehat{\chi}.$$

Using symmetry and bilinearity, we may write these identities as

$$\sum_{x \in G} f(x)\delta(x) = f = \sum_{\chi \in \widehat{G}} \widehat{f}(\chi^{-1}) d^{-1}\chi$$

and

$$\sum_{x \in G} f(x)\widehat{\delta(x)} = \widehat{f} = \sum_{\chi \in \widehat{G}} \widehat{f}\left(\chi^{-1}\right) d^{-1}\widehat{\chi}.$$

Now, the restriction to $\widehat{G}$ of each $\widehat{f} \in (\mathbb{F}[G])^*$ uniquely defines an element of the group algebra $\mathbb{F}[\widehat{G}]$, and, conversely, each function $\varphi \in \mathbb{F}[\widehat{G}]$ is the restriction to $\widehat{G}$ of a unique $\mathbb{F}$-linear form λ on $\mathbb{F}[G]$, namely,

$$\lambda = \sum_{\chi \in \widehat{G}} \varphi(\chi^{-1}) d^{-1} \widehat{\chi},$$

since the restriction to $\widehat{G}$ of $d^{-1}\widehat{\chi}$ is $\delta_{\widehat{G}}(\chi^{-1})$. It follows that the map taking $\widehat{f} \in (\mathbb{F}[G])^*$ onto its restriction to $\widehat{G}$ in $\mathbb{F}[\widehat{G}]$ is an isomorphism of $\mathbb{F}$-modules. Therefore, the mapping obtained by pre-composing this map with the above bilinear form on $\mathbb{F}[G]$ is an $\mathbb{F}$-module isomorphism of $\mathbb{F}[G]$ onto $\mathbb{F}[\widehat{G}]$. In point of fact,

Lemma 14. *The map taking each $f \in \mathbb{F}[G]$ onto the restriction to $\widehat{G}$ of its lift $\widehat{f}$ to $(\mathbb{F}[G])^*$ is an isomorphism of $\mathbb{F}$-algebras, where the multiplication law on $\mathbb{F}[G]$ is given by functional convolution, and where the multiplication law on $\mathbb{F}[\widehat{G}]$ is given by point-wise products.*

Proof. The character group $\widehat{G}$ of G consists of all homomorphisms taking G into the group of units $\mathbb{F}^\times$ of the $\mathbb{F}$-algebra $\mathbb{F}$. By the universal property of group algebras, the lift $\widehat{\chi}$ of each $\chi \in \widehat{G}$ is an algebra homomorphism of $\mathbb{F}[G]$ into $\mathbb{F}$, that is to say, each $\widehat{\chi}$ is a multiplicative linear form on $\mathbb{F}[G]$, and these are precisely the non-zero multiplicative linear forms on $\mathbb{F}[G]$. It follows that for each f_1,$f_2 \in \mathbb{F}[G]$,

$$\begin{aligned}
\widehat{f_1 * f_2} &= \sum_{\chi \in \widehat{G}} \widehat{f_1 * f_2}(\chi^{-1}) d^{-1} \widehat{\chi} \\
&= \sum_{\chi \in \widehat{G}} \widehat{\chi^{-1}}(f_1 * f_2) d^{-1} \widehat{\chi} \\
&= \sum_{\chi \in \widehat{G}} \widehat{\chi^{-1}}(f_1) \widehat{\chi^{-1}}(f_2) d^{-1} \widehat{\chi} \\
&= \sum_{\chi \in \widehat{G}} \widehat{f_1}(\chi^{-1}) \widehat{f_2}(\chi^{-1}) d^{-1} \widehat{\chi}.
\end{aligned}$$

Restricting to $\widehat{G}$, and recalling that $d^{-1}\widehat{\chi}$ restricts to $\delta(\chi^{-1}) \in \mathbb{F}[\widehat{G}]$, it follows

$$f_1 * f_2 \mapsto \sum_{\chi \in \widehat{G}} \widehat{f_1}(\chi^{-1}) \widehat{f_2}(\chi^{-1}) \delta(\chi^{-1}) = \sum_{\chi \in \widehat{G}} \widehat{f_1}(\chi) \widehat{f_2}(\chi) \delta(\chi),$$

which is the point-wise product of the images of f_1 and f_2. $\square$

The map figuring in the above lemma is called the **Fourier transform** on G.

Low-level view. Let

$$\nu_{\widehat{G}} : \widehat{G} \to \mathbb{F}[G]$$

be the natural set-theoretic inclusion map. Applying the left exact contravariant functor

$$\mathcal{F} = \mathrm{Hom}_{\mathbf{Sets}}(-,\, \mathbb{F})$$

to the exact sequence

$$0 \longrightarrow \widehat{G} \xrightarrow{\nu_{\widehat{G}}} \mathbb{F}[G]$$

over the category **Sets** of sets yields the exact sequence

$$\mathrm{Hom}_{\mathbf{Sets}}(\mathbb{F}[G],\, \mathbb{F}) \xrightarrow{\mathcal{F}(\nu_{\widehat{G}})} \mathbb{F}[\widehat{G}] \longrightarrow 0,$$

over the category of $\mathbb{F}$-algebras. Here, the addition and multiplication operations on $\mathrm{Hom}_{\mathbf{Sets}}(\mathbb{F}[G],\, \mathbb{F})$ are point-wise defined, and the $\mathbb{F}$-algebra map $\mathcal{F}(\nu_{\widehat{G}})$ takes each $\mathbb{F}$-valued function on $\mathbb{F}[G]$ to its restriction to $\widehat{G}$. The kernel of $\mathcal{F}(\nu_{\widehat{G}})$ is the annihilator of $\widehat{G}$ in $\mathrm{Hom}_{\mathbf{Sets}}(\mathbb{F}[G],\, \mathbb{F})$. Each $F \in \mathrm{Hom}_{\mathbf{Sets}}(\mathbb{F}[G],\, \mathbb{F})$ is congruent modulo Ker $\mathcal{F}\left(\nu_{\widehat{G}}\right)$ to at least one linear form on $\mathbb{F}[G]$. In fact, F is congruent to the linear form $\lambda(F) \in (\mathbb{F}[G])^*$ given by

$$\lambda(F) = \sum_{\chi \in \widehat{G}} F(\chi) d^{-1} \widehat{\chi^{-1}},$$

since, for each $\chi_0 \in \widehat{G}$,

$$\left(\lambda(F)\right)(\chi_0) = \sum_{\chi \in \widehat{G}} F(\chi) d^{-1} \widehat{\chi^{-1}}(\chi_0) = F(\chi_0) d^{-1} \widehat{\chi_0^{-1}}(\chi_0) = F(\chi_0),$$

which shows that F and $\lambda(F)$ have identical restrictions to $\widehat{G}$. Conversely, distinct linear forms on $\mathbb{F}[G]$ cannot be congruent modulo Ker $\nu_{\widehat{G}}^{\mathrm{T}}$, since each such form is completely determined by its values on the linear basis $\widehat{G}$ of $\mathbb{F}[G]$. It follows that $\mathcal{F}(\nu_{\widehat{G}})$ induces a linear isomorphism

$$\overline{\nu^{\mathrm{T}}} : (\mathbb{F}[G])^* \to \mathbb{F}[\widehat{G}],$$

since there exists a unique $\mathbb{F}$-linear form on $\mathbb{F}[G]$ having a prescribed restriction to $\widehat{G}$. Now, we may naturally identify G with its double dual $\widehat{\widehat{G}}$, via the group isomorphism

$$x \mapsto \epsilon(x) : \chi \mapsto \chi(x).$$

This map, together with the trivial map on $\mathbb{F}$, induces an isomorphism of algebras

$$\mathbb{F}[G] \mapsto \mathbb{F}[\widehat{\widehat{G}}].$$

But, as $\widehat{G}$ forms a basis for $\mathbb{F}[G]$, it follows that the set of linear extensions of the elements of $\mathbb{F}[\widehat{\widehat{G}}]$ is precisely the dual space $(\mathbb{F}[G])^*$ of $\mathbb{F}[G]$. Thus, we have a natural isomorphism

$$\mathbb{F}[\widehat{\widehat{G}}] \to (\mathbb{F}[G])^* .$$

Composing the three preceding isomorphisms, we obtain a natural map

$$\Phi : \mathbb{F}[G] \to \mathbb{F}[\widehat{\widehat{G}}] \to (\mathbb{F}[G])^* \to \mathbb{F}[\widehat{G}].$$

The map Φ is such that

$$\Phi : \delta(x) \mapsto \Phi\left(\delta(x)\right) : \chi \mapsto \chi(x).$$

Thus,

$$\Phi(f) : \chi \mapsto \sum_{x \in G} f(x)\chi(x) = \widehat{f}(\chi),$$

and, therefore, with respect to the standard basis in $\mathbb{F}[\widehat{G}]$,

$$\begin{aligned} \Phi(f) &= \sum_{\chi \in \widehat{G}} \left(\left(\mathcal{F}(f)\right)(\chi)\right)\delta(\chi) \\ &= \sum_{\chi \in \widehat{G}} \widehat{f}(\chi)\delta(\chi) \\ &= \widehat{f}. \end{aligned}$$

We have shown that there is a natural isomorphism of algebras

$$\Phi : \mathbb{F}[G] \to \mathbb{F}[\widehat{G}]$$

mapping $f \in \mathbb{F}[G]$ onto $\widehat{f} \in \mathbb{F}[\widehat{G}]$, where

$$\widehat{f}(\chi) := \sum_{x \in G} f(x)\chi(x).$$

The map Φ is called the **Fourier transform** on G over $\mathbb{F}$. The coefficients of $f \in \mathbb{F}[G]$ with respect to the Fourier basis are given (up to scaling) by the Fourier transform, via

$$f = \sum_{k=1}^{d} \widehat{f}(\chi) d^{-1} \chi^{-1}.$$

C NTT's as twisted Fourier transforms

Let G be a finite cyclic group of order d. Let $\mathbb{F}$ be a field of characteristic prime to d. Assume $\mathbb{F}$ contains the d-th roots of unity. The d-point NTT over $\mathbb{F}$ used in

BFV is a twisted version of the Fourier transform on G over $\mathbb{F}$. In fact, consider the mapping

$$\varepsilon : \mathbb{F}[x] \to \mathbb{F}[G]$$

determined by the identity map on $\mathbb{F}$ along with the assignment

$$x \mapsto \omega\delta(g).$$

This mapping is surjective with kernel $\Phi_{2d}(x)\mathbb{F}[x]$, so there is an induced algebra isomorphism

$$\overline{\varepsilon} : \overline{\mathcal{O}} \to \mathbb{F}[G]$$

taking

$$x + \Phi_{2d}(x)\mathbb{F}[x] \mapsto \omega\delta(g).$$

Here, $\overline{\mathcal{O}}$ denotes the reduction $\mathbb{F} \otimes_{\mathbb{Z}} \mathcal{O}$. Under this isomorphism, the standard basis in $\mathbb{F}[G]$ corresponds to the monomial basis in $\overline{\mathcal{O}}$, while the Fourier basis in $\mathbb{F}[G]$ corresponds to the **Number Theoretic Transform (NTT)** basis $\left(d^{-1}\overline{\varepsilon}^{-1}(\chi_k)\right)_{k=1}^{d}$ in $\overline{\mathcal{O}}$. It follows that the NTT basis consists of orthogonal primitive idempotents, and

$$\overline{\mathcal{O}} = \bigoplus_{k=1}^{d} \mathbb{F} d^{-1}\overline{\varepsilon}^{-1}(\chi_k)$$

as an internal direct sum. The transformation on $\overline{\mathcal{O}}$ effecting the change from the monomial basis to the NTT basis is termed a Number Theoretic Transform, and the coefficients of a given element $a \in \overline{\mathcal{O}}$ with respect to the NTT basis are termed the element's **NTT coefficients**. These coefficients correspond under $\overline{\varepsilon}$ with the Fourier coefficients of $\overline{\varepsilon}(a) \in \mathbb{F}[G]$. In fact,

$$\overline{\varepsilon}^{-1} : f = \sum_{k=1}^{d} \widehat{f}(\chi_k^{-1}) d^{-1} \chi_k \mapsto \sum_{k=1}^{d} \widehat{f}(\chi_k^{-1}) d^{-1} \overline{\varepsilon}^{-1}(\chi_k).$$

Consequently, NTTs may be computed efficiently using any of the various fast Fourier transform algorithms.

References

[1] E. A. Williams *et al.*, "WIDESKIES: SCALABLE PRIVATE INFORMATION RETRIEVAL." http://homes.sice.indiana.edu/henry/courses/b609/s17/cache/pirk.incubator.apache.org/papers/wideskies_paper.pdf. Online; accessed June 30, 2019.

[2] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding* [2], pp. 223–238.

[3] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[4] Kim Laine, "Simple Encrypted Arithmetic Library 2.3.1." `https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf`. Online; accessed May 14, 2019.

[5] S. Angel, H. Chen, K. Laine, and S. Setty, "Pir with compressed queries and amortized query processing," *IEEE SP*, vol. May, pp. 962–979, 2018.

[6] T. Simpson, "Ciii. the invention of a general method for determining the sum of every 2d, 3d, 4th, or 5th, *et cetera* term of a series, taken in order; the sum of the whole series being known.," *Philosophical Transactions of the Royal Society of London*, vol. 51, pp. 757–759, 1757.

[7] O. Teichmüller, "Differentialrechnung bei charakteristik p," *J. Reine Angew. Math.*, vol. 175, pp. 89–99, 1936.