Fast Vector Oblivious Linear Evaluation from Ring Learning with Errors

Leo de Castro
Massachusetts Institute of Technology
ldec@mit.edu

Chiraag Juvekar
Analog Devices
chiraag.juvekar@analog.com

Vinod Vaikuntanathan
Massachusetts Institute of Technology
vinodv@mit.edu

ABSTRACT

Oblivious linear evaluation (OLE) is a fundamental building block in multi-party computation protocols. In OLE, a sender holds a description of an affine function $f_{\alpha,\beta}(z) = \alpha z + \beta$, the receiver holds an input x, and gets $\alpha x + \beta$ (where all computations are done over some field, or more generally, a ring). Vector OLE (VOLE) is a generalization where the sender has many affine functions and the receiver learns the evaluation of all of these functions on a single point x.

The state-of-the-art semi-honest VOLE protocols generally fall into two groups. The first group relies on standard assumptions to achieve security but lacks in concrete efficiency. These constructions are mostly based on additively homomorphic encryption (AHE) and are classified as "folklore". The second group relies on less standard assumptions, usually properties of sparse, random linear codes, but they manage to achieve concrete practical efficiency. In this work, we present a conceptually simple VOLE protocol that derives its security from a standard assumption, namely Ring Learning with Errors (RLWE), while still achieving concrete efficiency comparable to the fastest VOLE protocols from non-standard coding assumptions [3, 8, 31]. Furthermore, our protocol admits a natural extension to batch OLE (BOLE), which is yet another variant of OLE that computes many OLEs in parallel.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**;

KEYWORDS

homomorphic encryption, applied cryptography, oblivious linear evaluation

ACM Reference Format:
Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. 2021. Fast Vector Oblivious Linear Evaluation from Ring Learning with Errors. In *Proceedings of the 9th Workshop on Encrypted Computing Applied Homomorphic Cryptography (WAHC '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3474366.3486928

1 INTRODUCTION

Oblivious linear evaluation (OLE), a special case of oblivious polynomial evaluation [28], is a fundamental building block in many secure computation protocols [13, 21]. In an OLE protocol over a ring $\mathbb{Z}_p$, there are two parties, a sender S with values $\alpha, \beta \in \mathbb{Z}_p$ and a receiver R with a value $x \in \mathbb{Z}_p$. At the end of the protocol, R will learn the value $\gamma = \alpha \cdot x + \beta$ while S will learn nothing. In *Vector OLE* (VOLE), the sender has $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^m$, the receiver has $x \in \mathbb{Z}_p$, and obtains $\boldsymbol{\gamma} := \boldsymbol{\alpha} x + \boldsymbol{\beta} \in \mathbb{Z}_p^m$. In other words, vector OLE is running many OLE protocols in parallel where the receiver has the same x. *Batch OLE* (BOLE) can be viewed as many OLE protocols running in parallel where the receiver has a vector $\mathbf{x}$ of values over $\mathbb{Z}_p^m$. In this work, we will primarily focus on VOLE with a simple extension to BOLE.

Our approach to implementing a VOLE protocol is to use the packed additively homomorphic encryption (PAHE) of Brakerski [11] and Fan and Vercuteran [17], henceforth referred to as the BFV scheme. PAHE is a natural choice of primitive to implement such a protocol, especially in the honest-but-curious model. At a high level, our protocol has the receiver R encrypt the value x and send the encryption $[[x]]$ to the sender S. Using the PAHE operations, the sender can then compute the ciphertext $[[\boldsymbol{\gamma}]] = [[\boldsymbol{\alpha} \cdot x + \boldsymbol{\beta}]]$ and return it to the receiver, who can decrypt and learn the VOLE output $\boldsymbol{\gamma}$. As long as the PAHE scheme achieves *circuit privacy* which, in this case, says that the homomorphically evaluated ciphertext $[[\boldsymbol{\gamma}]]$ does not leak information about α and β even to the owner of the secret key, then security is achieved against honest-but-curious adversaries. In the case of VOLE, the circuit is specified by the sender's values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

We note that achieving security in the honest-but-curious setting is sufficient following the work of Hazay, Ishai, Marcedone, and Venkitasubramaniam [21] which instantiates the semi-honest-to-active compiler of Ishai, Prabhakaran, and Sahai [23] using black-box access to an honest-but-curious OLE protocol. In particular, this instantiation enables a semi-honest OLE protocol to be upgraded to an actively secure OLE protocol with roughly a factor of two overhead ([21], section 5.3).

While the high-level story is simple, instantiating a OLE scheme from a PAHE scheme in a *concretely efficient* way is less so. In this work, we present a series of optimizations of the textbook BFV PAHE scheme that results in a vector OLE scheme that outperforms all prior works in parameter regimes of practical interest. At the heart of our optimizations is a new analysis of the modulus reduction operation similar to the BV homomorphic encryption scheme [12], which argues that circuit privacy is achieved when modulus reduction is performed under carefully selected parameters. In addition, we extend the analysis of Halevi, Polyakov, and Shoup [20] to efficiently implement this operation in the double Chinese remainder theorem (DCRT) representation.

WAHC '21, November 15, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8656-2/21/11.
https://doi.org/10.1145/3474366.3486928

1.1 Our Contributions

First, we present a fast, light-weight vector OLE (VOLE) protocol. In fact, our protocol most naturally achieves the more expressive batch OLE (BOLE) functionality at little additional cost. This is in contrast to the LPN-based VOLE protocols of [3] and [31] which, to the best of our knowledge, do not have such a straightforward extension to BOLE.

The main technical idea in our result is a way to achieve circuit privacy in a PAHE scheme which corresponds to sender security in a (V/B)OLE protocol. In slightly more detail, we'd like to ensure that the receiver who has the private key of the PAHE scheme and x, and obtains an evaluated ciphertext $[[\boldsymbol{\gamma}]] = [[\boldsymbol{\alpha} \cdot x + \boldsymbol{\beta}]]$ learns $\boldsymbol{\gamma}$, but no other information about $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The crucial difficulty is that the noise contained in the PAHE evaluated ciphertext $[[\boldsymbol{\gamma}]]$ could reveal information about $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

The folklore idea to achieve circuit privacy is a simple technique called *noise-flooding* [18]. That is, sample from a sufficiently wide distribution, either Gaussian or uniform, so as to flood the noise contained in $[[\boldsymbol{\gamma}]]$. In turn, to achieve sufficient statistical security, this forces us to work with inefficient *multi-precision arithmetic*. For example, the procedure to sample uniformly random numbers from a sufficiently wide interval, using the NTL library [33], will take more than 100% of our sender run-time. Additionally, despite concrete (in)security concerns in practical scenarios [29], homomorphic encryption libraries either do not seem to implement circuit privacy (such as in SEAL [25, 32]) or implement a single-precision version of noise flooding which provides very limited statistical security (such as in PALISADE [30]).

Other approaches to circuit-private homomorphic encryption exist, but are much less efficient than what we can afford for such a lightweight computation as in (V/B)OLE: garbled circuit-based techniques [19] requires computing and communicating a garbled circuit for the PAHE decryption algorithm; FHE bootstrapping-based techniques [15] are even less efficient; and techniques such as [7] only work for the GSW homomorphic encryption scheme which is also too inefficient for our purposes.

We present a simple way to achieve circuit privacy of PAHE, and therefore construct a (V/B)OLE protocol, using a rounding operation that allows greater efficiency than the folklore noise-flooding approach. The rounding operation and its rather simple analysis is inspired by similar techniques in the context of learning with rounding (LWR) [4].

In addition, we give an implementation of this protocol along with performance benchmarks. This implementation aims to be usable as a black-box in larger applications; an earlier version of this implementation was already used in the work of Hazay, Ishai, Marcedone, and Venkitasubramaniam [21] in CCS 2019.

Furthermore, our protocol outperforms recent implementations ([31], CCS 2019) based on the learning parity with noise assumption, as described in [8], for VOLE dimension up to $m = 2^{35}$. In other words, while [8, 31] will outperform our implementation asymptotically for huge m, the crossover point is quite far out. We describe the relation between the two approaches in more detail in Section 1.2, and provide a detailed performance comparison in Section 5.

1.2 Related Work

Recent related work improving VOLE and BOLE efficiency have mainly focused on obtaining optimizations from the Learning Parity with Noise (LPN) assumption [6] and other related coding assumptions. We separate prior work into two main categories based on their communication complexity. The first category consists of protocols that have communication complexity linear or super-linear in the length of the VOLE correlations they generate. More heuristically, these protocols are optimized for smaller VOLE correlations and tend to be very fast for these shorter VOLE lengths. The second category consists of protocols with sublinear communication complexity in the length of the VOLE correlations they generate. Heuristically, these protocols are optimized for larger VOLE lengths and tend to be slower than protocols in the first category for generating shorter VOLE correlations.

Our protocol falls in the first category, so we compare directly with other protocols in this category. While our protocol can be extended to the more general batch-OLE functionality at minimal cost, we focus our comparisons on vector-OLE. Our main point of comparison is the work of Applebaum et al. [3], although we also compare our protocol against the subsequent works of Baum et al. [5], Boyle et al. [10], and Weng et al. [34]. We note that [34] implements the vector-OLE functionality whereas our work as well as [10**?**] implement the more general batch-OLE functionality.

The protocol of Applebaum et al. [3] implements a fast vector-OLE protocol using assumptions over sparse linear codes that are inspired by the LPN assumption but notably are not known to have a reduction from the LPN assumption. Consequently, while this work achieves impressive performance, the security of their protocol is harder to quantify. In contrast, the security of our VOLE protocol is based on the security of the Ring-LWE assumption, which has recently been extensively benchmarked in the context of the NIST post-quantum cryptography standardization process and the homomorphic encryption standardization process [2]. In particular, we base our parameter choices off of the homomorphic encryption standard [2]. This widely endorsed standard significantly strengthens the security claims about the parameter choices for this protocol. In section 5.4, we show that even with these strong security claims, our protocol achieves efficiency that meets or exceeds the efficiency of [3] in many settings. In addition, our VOLE protocol gives a natural extension to BOLE.

To compare our protocol against works in the second category, we must first recognize that all protocols in the second category will asymptotically outperform ours, since eventually the linear communication complexity of our protocol will surpass the sublinear communication complexity of protocols in this category. The natural question to ask, then, is at what VOLE length does the sublinear protocol become faster than the linear protocol, and where do the practical applications of VOLE live vis-à-vis the crossover point? When comparing against protocols in this category, we estimate this cross-over point and then show applications where the VOLE lengths are less than this point. Our main point of comparison for protocols in this category is the work of Schoppmann, Gascón, Reichert, and Raykova [31], which is a two-party implementation of the function secret sharing (FSS) based work of Boyle, Couteau, Gilboa, and Ishai [8] that constructs a VOLE protocol with sublinear complexity. In section 5.4, we show that our protocol outperforms [31] for smaller VOLE lengths, which is to be expected. We also show that the expected cross-over point of the performances of this protocol is large enough to allow potential applications that do not require VOLE correlations of greater length. In section 5.6, we discuss one such application: securely evaluating convolutional neural network layers for medical images.

Subsequent Work. Subsequent to the online publication of this work [14], several additional works have been published presenting various VOLE and BOLE protocols. We briefly compare against these works to show that our protocol is still competitive with the state-of-the-art.

The first work we compare against is the BOLE protocol of Baum et al. [5]. This two-party protocol between Alice with a vector input $\boldsymbol{\alpha}$ and Bob with a vector input $\boldsymbol{\beta}$ results in each party holding an additive share of the component-wise product $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$. The main feature of this protocol is that it requires only one round of simultaneous communication; Alice sends a message to Bob and Bob simultaneously sends a message to Alice. Once these messages have been received, the parties locally compute additive shares of the component-wise product. While our protocol is two rounds, we show in section 5.4 that our protocol outperforms this one-round protocol in both total communication and computation time. The intuition for why this is the case is because the one-round BOLE protocol requires *two* iterations of the LPR-style rounding technique, while our approach only performs this operation once.

Next, we compare against a series of works that fall into the category of asymptotically sublinear implementations. These implementations are based on some flavor of the LPN assumption [6]. The majority of these works focus on VOLE, specifically a variant known as *subfield* VOLE, which we denote sVOLE. The sVOLE protocol is defined by a length n, a prime p, and an integer r. The receiver's input is a value $x \in \mathbb{F}_{p^r}$, and the sender's input is the tuple $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathbb{F}_p^n \times \mathbb{F}_{p^r}^n$. Note that the elements of the vector $\boldsymbol{\alpha}$ are drawn from the subfield $\mathbb{F}_p$ rather than the full domain $\mathbb{F}_{p^r}$. The output of sVOLE takes the same format as VOLE, namely the vector $\gamma = \boldsymbol{\alpha} \cdot x + \boldsymbol{\beta} \in \mathbb{F}_{p^r}$. We note that sVOLE can be the same as VOLE when $r = 1$. Building off of the works of Boyle et al. [8, 9], the fastest work (to our knowledge) implementing sVOLE is the work of Weng et al. and we briefly compare against this work in section 5.5.

The subsequent work of Boyle et al. [10] also uses LPN-style assumptions to construct a sublinear protocol for BOLE. While this protocol achieves very low communication, the estimated computational performance is not competitive with out protocol, as we discuss in section 5.5.

1.3 Road Map

In section 2, we give some brief preliminaries and definitions necessary in the explanations below. In section 3, we describe our circuit privacy transformation method and prove its security. In section 4, we give the full VOLE protocol, including a proof of security, and in section 5 we present an implementation of this protocol and analyze the performance.

2 BACKGROUND

In this section, we will briefly review the definitions of the homomorphic encryption scheme used below as well as the mathematics behind residue number system optimizations.

2.1 Notation

The homomorphic encryption scheme used in our work is based off of the Ring Learning with Errors problem [27], which is defined over polynomial rings. In our instantiation, we use the ring

$$\mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \tag{1}$$

For the remainder of this work, n will always be a power of two. For a modulus q, let $\mathcal{R}_q$ be $\mathcal{R}$ with all coefficients mod q.

$$\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

For an integer n, we will denote the set $\{0, 1, 2, \ldots, n-1\}$ as $[n]$. For integers $i \leq j$, we denote the range $\{i, i+1, \ldots, j-1\}$ as $[i : j]$.

We denote the rounding function $\lceil \cdot \rfloor : \mathbb{R} \longrightarrow \mathbb{Z}$ that maps $x_r \in \mathbb{R}$ to the closest integer $x \in \mathbb{Z}$. We denote the flooring function $\lfloor \cdot \rfloor : \mathbb{R} \longrightarrow \mathbb{Z}$ that maps $x_r \in \mathbb{R}$ to the closest integer $x \in \mathbb{Z}$ such that $x \leq x_r$.

For a positive integer b, we write the modular reduction operation $c \equiv a \mod b$ as $c = [a]_b$.

The norm notation $|| \cdot ||$ refers to the ℓ_∞ norm, unless otherwise specified. For a polynomial a with n coefficients each mod q, we have the bound $||a|| \leq q$.

We specify the base-2 logarithm by log.

We call a distribution whose samples have magnitude bounded by B a B-bounded distribution.

We say that a function negl is negligible if for every constant $c > 1$ we have $\mathsf{negl}(n) < 1/n^c$ for all sufficiently large n.

For two vectors v and w of length ℓ, denote the component-wise product of v and w as $v \odot w$.

We will denote sampling from a distribution as $\overset{\$}{\leftarrow}$. We will denote sending or receiving a message from a public channel as $\overset{net}{\longleftarrow}$.

2.2 Circuit Privacy

For brevity, we move the defintion of a leveled homomorphic encryption scheme to appendix A.3

Definition 2.1. *Circuit Privacy ([22] definition 7, [7] definition 5.1)*

Let $\mathcal{E}$ be a leveled homomorphic encryption scheme as defined in definition A.4. Define the following:

$$\begin{aligned}
(\mathsf{sk}, \mathsf{pk}, \mathsf{evk}) &\overset{\$}{\leftarrow} \mathcal{E}.\mathsf{KeyGen}(1^\lambda, 1^L) \\
\mathsf{ct}_i &\overset{\$}{\leftarrow} \mathcal{E}.\mathsf{Encrypt}(\mathsf{pk}, m_i) \quad i \in [1 \ldots k] \\
\mathsf{ct}_i &\overset{\$}{\leftarrow} \mathcal{E}.\mathsf{EncryptSK}(\mathsf{sk}, m_i) \quad i \in [k+1 \ldots n] \\
\langle \mathsf{ct}_i \rangle &:= \{\mathsf{ct}_i\}_{i=1}^n \\
m_{out} &= f(m_1, \ldots, m_n)
\end{aligned}$$

*We say that $\mathcal{E}$ is ϵ-*circuit private *for functions f of depth $\ell \leq L$ if there exists a PPT simulator algorithm* Sim *such that for all PPT* *distinguishing algorithms $\mathcal{D}$ the following holds:*

$$\Big| \Pr\Big[\mathcal{D}\Big(\mathcal{E}.\mathsf{Eval}(\mathsf{pk}, \mathsf{evk}, f, \langle \mathsf{ct}_i \rangle), \langle \mathsf{ct}_i \rangle, \mathsf{sk}, \mathsf{pk}, \mathsf{evk}\Big) = 1\Big] - \Pr\Big[\mathcal{D}\Big(\mathsf{Sim}(1^\ell, \mathsf{pk}, \mathsf{sk}, \mathsf{evk}, m_{out}), \langle \mathsf{ct}_i \rangle, \mathsf{sk}, \mathsf{pk}, \mathsf{evk}\Big) = 1\Big] \Big| \leq \epsilon$$

In words, definition 2.1 says that the output of the evaluation algorithm of a circuit private leveled homomorphic encryption scheme should be indistinguishable from a simulated output, where the simulator is given no information about the function f used to compute the result ciphertext other than the function output. We can view the simulator in definition 2.1 as an alternate encryption procedure that produces a fresh ciphertext that is indistinguishable from the output of the real $\mathcal{E}.\mathsf{Eval}$ algorithm. We only consider functions f that will result in $\mathcal{E}.\mathsf{Eval}$ outputting a correct ciphertext. This can be implemented by having both $\mathcal{E}.\mathsf{Eval}$ and Sim output $\perp$ for functions that are exceed the number of levels supported by $\mathcal{E}$.

We will use the homomorphic encryption scheme of Brakerski, Fan, and Vercuteran [11, 17], denoted as the BFV scheme.

2.3 Ring Expansion Factor

In order to effectively choose parameters for our leveled homomorphic encryption scheme, we must accurately upper bound the noise growth due to homomorphic operations. When multiplying two elements of $\mathcal{R}_q$, we need an upper bound on the norm of the product by a function of the norms of the operands as well as properties of $\mathcal{R}_q$ itself.

Definition 2.2 (Ring Expansion Factor [18, 26]). *The expansion factor $\delta_{\mathcal{R}}$ for a ring $\mathcal{R}$ is defined as follows:*

$$\delta_{\mathcal{R}} = \max_{a,b \in \mathcal{R}} \frac{||a \cdot b||}{||a|| \cdot ||b||}$$

Lemma 2.1 (Ring Expansion Upper Bound). *For a ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$, we can upper bound the ring expansion factor $\delta_{\mathcal{R}}$ for the norm $||\cdot||$ by*

$$\delta_{\mathcal{R}} \leq n$$

Proof. In the worst case for the expansion of the norm $||\cdot||$ is when both polynomials $a, b \in \mathcal{R}$ have coefficients of all the same magnitude. In this case, the maximum coefficient of the product will be n times the product of the maximum coefficient of the operands. Therefore, we have

$$||a|| \cdot ||b|| = n \cdot ||a \cdot b|| = \delta_{\mathcal{R}} \cdot ||a \cdot b||$$

□

Remark 2.1. *The upper bound in lemma 2.1 extends to $\mathcal{R}_q$ for any modulus q.*

3 OUR CIRCUIT PRIVACY APPROACH

In this section, we describe our approach to obtaining circuit privacy. We will begin by describing the main operation of the circuit privacy transformation, which is fundamentally a divide-then-round step. Then, we will describe how this operation is performed efficiently in our implementation. Finally, we describe the full circuit privacy transformation, followed by a security proof.

3.1 The Divide-and-Round Step

We begin by presenting our procedure for performing an efficient divide-and-round operation over ciphertext elements. The benefits of this operation include a decreased overall ciphertext size as well as destroying the information in the ciphertext error term that could leak information about the computation.

Let Q be the original ciphertext modulus of our scheme, and let q_i be the primes such that $\prod_i q_i = Q$. Given a ciphertext ct_Q, where the elements are in $\mathcal{R}_Q$ and the decryption operations are performed over $\mathcal{R}_Q$, the goal of this operation is to obtain a ciphertext ct_{q_0} that encrypts the same message. This new ciphertext ct_{q_0} will have elements in $\mathcal{R}_{q_0}$ and the decryption of this ciphertext will be performed over $\mathcal{R}_{q_0}$ as well. Recall that we choose primes q_i to fit in a standard machine word, so decryption of ct_{q_0} is far more efficient than the decryption of ct_Q. In addition, the communication cost of sending ct_{q_0} over a network is significantly less than sending ct_Q. Note that this operation can only be correct if the plaintext modulus p is sufficiently less than q_0.

Let $q_0^* = Q/q_0$. Below, we write the ciphertext ct_Q and expand out the terms to anticipate the division by q_0^*.

$$\begin{aligned}\mathsf{ct}_Q &= \Big(a,\ a \cdot s + \Delta m + e\Big) \\ &= \Big(a' \cdot q_0^* + [a]_{q_0^*},\ (a' \cdot q_0^* + [a]_{q_0^*}) \cdot s + \Delta m + e\Big) \\ &= \Bigg(a' \cdot q_0^* + [a]_{q_0^*},\ (a' \cdot q_0^* + [a]_{q_0^*}) \cdot s + \Delta m + e\Bigg)\end{aligned}$$

The ciphertext ct_{q_0} is obtained by dividing the two components of ct_Q by q_0^*.

$$\begin{aligned}\mathsf{ct}_{q_0} &= \left\lfloor \frac{\mathsf{ct}_Q}{q_0^*} \right\rceil = \left(\left\lfloor \frac{a}{q_0^*} \right\rceil, \left\lfloor \frac{a \cdot s + \Delta m + e}{q_0^*} \right\rceil \right) \\ &= \left(\left\lfloor \frac{a' \cdot q_0^* + [a]_{q_0^*}}{q_0^*} \right\rceil, \left\lfloor \frac{(a' \cdot q_0^* + [a]_{q_0^*}) \cdot s + \Delta m + e}{q_0^*} \right\rceil \right) \\ &= \left(a',\ a' \cdot s + \left\lfloor \frac{[a]_{q_0^*} \cdot s + \Delta m + e}{q_0^*} \right\rceil \right) \\ &= \Big(a',\ a' \cdot s + v\Big)\end{aligned} \tag{2}$$

where

$$v = \left\lfloor \frac{[a]_{q_0^*} \cdot s + \Delta m + e}{q_0^*} \right\rceil \tag{3}$$

is the new term in ct_{q_0} that equals $\Delta' m + v'$ for a scaling factor Δ' and error term v'. Note that we will analyze the whole term v for the remainder of this section to show the security of the circuit privacy procedure, and then we will split v into $\Delta' m + v'$ to show that this procedure produces a correct, decryptable ciphertext.

We will now begin to analyze this term v in terms of the original error term e. Our goal will be to show that the distribution of v is statistically close to a distribution that is independent of e. Once we have shown this, we will be able to construct a simulator that samples an error term that is statistically close to v but still independent of the original error term e.

We begin by defining the event that must occur in order for $\lfloor(a+c)/b\rfloor \neq \lfloor(a+c+e)/b\rfloor$ for a small error term e in terms of a and fixed scalar c. This event is intuitively very similar to the BAD event in the analysis of learning with rounding (LWR) in [4].

Definition 3.1 (Boundary Event). *Let a, b, B be positive integers. Define the event $\mathsf{BAD}(a, b; B)$ to be the following:*

$$\mathsf{BAD}(a, b; B) := \{[a]_b \in [0, B) \cup [b - B, b)\} \tag{4}$$

In words, equation 4 is the event that a is within distance B of a multiple of b.

Lemma 3.1. *Let a, b be positive integers, and let e be an integer such that $|e| \leq B$ where $B < b/2$. Then the following relation holds:*

$$\left\lfloor \frac{a}{b} \right\rfloor \neq \left\lfloor \frac{a+e}{b} \right\rfloor \implies \mathsf{BAD}(a, b; B) \tag{5}$$

Proof. This proof is quite straightforward. If we write $a = kb+r$ where k is an integer and $r \in [b]$, we have that

$$\left\lfloor \frac{a}{b} \right\rfloor = \left\lfloor \frac{kb+r}{b} \right\rfloor = k \tag{6}$$

Since $\lfloor(a+e)/b\rfloor \neq k$, there are two possibilities. The first possibility is that $a + e = (k + 1)b + r'$ where $r' \in [b]$. This occurs when $a \in [(k + 1) \cdot b - e, (k + 1) \cdot b) \subseteq [(k + 1) \cdot b - B, (k + 1) \cdot b)$. The second case is when $a + e = (k - 1)b + r''$. This occurs when $a \in ((k - 1) \cdot b, (k - 1) \cdot b + e] \subseteq ((k - 1) \cdot b, (k - 1) \cdot b + B]$. Both of these cases satisfy the condition of $\mathsf{BAD}(a, b; B)$ occurring. □

Corollary 3.1.1. *Let a, b be positive integers, and let e be an integer such that $|e| \leq B$ and $B < b/2$. Then the following relation holds:*

$$\neg\mathsf{BAD}(a, b; B) \implies \left\lfloor \frac{a}{b} \right\rfloor = \left\lfloor \frac{a+e}{b} \right\rfloor \tag{7}$$

We will now bound the probability that the BAD event occurs for values of a that are uniformly random.

Lemma 3.2 (Boundary Event for Random Values). *Let b and B be positive integers. Let A be a distribution over the integers such that the distribution of $[a]_b$ is uniformly random over $[b]$, where $a \leftarrow A$. We can bound the probability of the BAD event occurring for an output of A as follows:*

$$\Pr_{a \leftarrow A}[\mathsf{BAD}(a, b; B)] \leq \frac{2B}{b} \tag{8}$$

Proof. If we write the value $a \leftarrow A$ as $a = k \cdot b + r$, we are given that the distribution of $r = [a]_b$ is uniformly random over $[b]$. From definition 3.1, the BAD event occurs when r falls into the range $[0, B) \cup [b - B, b)$. This range is of size $2B$, so this occurs with probability $2B/b$. □

We will now extend the definition of the boundary event to elements over $\mathcal{R}_q$.

Definition 3.2 (Ring Boundary Event). *Let a be an element of the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, let b be a scalar in $\mathbb{Z}_q$, and let B be a positive integer. We will define the boundary case for an element of $\mathcal{R}_q$ to occur if the boundary case for any of its coefficients occurs. Denote the i^{th} coefficient of a as a_i.*

$$\mathsf{BAD}(a, b; B) = \{\exists i \in [n] \text{ such that } \mathsf{BAD}(a_i, b; B)\} \tag{9}$$

Lemma 3.3. *Let a be an element of the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, let b be a scalar in $\mathbb{Z}_q$, and let B be a positive integer. We can bound the ring boundary event from definition 3.2 by the following:*

$$\Pr[\mathsf{BAD}(a, b; B)] \leq \sum_{i \in [n]} \Pr[\mathsf{BAD}(a_i, b; B)] \tag{10}$$

Proof. This follows directly from taking the union bound over the events corresponding to the coefficients of a. □

Corollary 3.3.1. *For a polynomial a that is sampled over $\mathcal{R}_q$ such that each coefficient a_i is uniform modulo b, we can bound the probability of the BAD event occurring by the following:*

$$\Pr[\mathsf{BAD}(a, b; B)] \leq \sum_{i \in [n]} \Pr[\mathsf{BAD}(a_i, b; B)] = 2n\frac{B}{b} \tag{11}$$

Corollary 3.3.2. *For a polynomial a that is sampled uniformly over $\mathcal{R}_Q$ for $Q = q_0^* \cdot q_0$ and a small polynomial error term e such that $||e|| \leq B$ for a positive integer $B < q_0^*/2$, we can bound the following probability:*

$$\Pr_{a \leftarrow \mathcal{R}_Q}\left[\left\lfloor \frac{a+e}{q_0^*} \right\rfloor \neq \left\lfloor \frac{a}{q_0^*} \right\rfloor\right] \leq \Pr[\mathsf{BAD}(a, q_0^*; B)] \leq 2n\frac{B}{q_0^*}$$

We now return to equation 3 for the compressed term v and give a lower bound the probability that this term hides the original error term e.

We begin by examining the numerator of this function and observe that if the secret s is invertible modulo q_0^*, then the term $[a]_{q_0^*} \cdot s$ would define a bijection with domain and range $[q_0^*]$. From this, we have the simple observation that a uniformly random input to this bijection gives a uniformly random output over the same range.

However, we cannot guarantee that s is invertible, and we cannot restrict to the case where s is invertible while still appealing to the security of RLWE. We handle this by observing that while the polynomial element $a \cdot s$ may not be random, the marginal distribution of each of the individual coefficients of $a \cdot s$ are still random. This is formalized in the following lemma.

Lemma 3.4. *Fix an integer $Q = q_0^* \cdot q_0$. Fix a non-zero $s \in \mathcal{R}_Q = \mathbb{Z}_Q[x]/(x^n + 1)$ and an index $i \in [n]$. Define the distribution A_i to produce samples by first sampling a truly random $a \leftarrow \mathcal{R}_Q$, then computing the product $a \cdot s$ modulo q_0^*, then outputing the i^{th} coefficient of this product. The output of this distribution is statistically close to uniform over $[q_0^*]$.*

Proof. We can write the i^{th} coefficient of $a \cdot s$ as the inner product the coefficients of s with a rotation of the coefficients of a. Since the coefficients of a are uniformly random modulo q_0^*, and s is non-zero, this inner product is statistically close to uniform over $[q_0^*]$. □

Corollary 3.4.1. *Fix an integer $Q = q_0^* \cdot q_0$. Fix a non-zero $s \in \mathcal{R}_Q = \mathbb{Z}_Q[x]/(x^n + 1)$ and a positive integer B. For a distribution of polynomials sampled uniformly over $\mathcal{R}_Q$, we can upper bound the probability of the BAD event for the product $a \cdot s$ by the following:*

$$\Pr_{a \leftarrow \mathcal{R}_Q}\left[\mathsf{BAD}(a \cdot s, q_0^*; B)\right] \leq 2n\frac{B}{q_0^*} \tag{12}$$

We now complete this analysis by applying corollary 3.4.1 to equation 3 for the final term v to show that v hides e. We will define one distribution that outputs v as described in equation 3 and one distribution that leaves out the error term e. The next lemma will give an upper bound that these two distributions output different values.

Lemma 3.5. *If $||e|| \le B$, the final term v from equation 3 hides the original error term e with probability $2n\frac{B}{q_0^*}$, where n is the degree of the polynomial modulus $x^n + 1$.*

Proof. Define two distributions in terms of s, Δm, and e. Distribution D_1 produces samples by first sampling a uniformly random $a \xleftarrow{\$} \mathcal{R}_Q$ for $Q = q_0^* \cdot q$, then producing a sample of the form:

$$d_1 = \left\lfloor \frac{[a]_{q_0^*} \cdot s + \Delta m + e}{q_0^*} \right\rceil$$

This is identical to the real distribution of v. The second distribution D_2 produces samples by first sampling a uniformly random $a \xleftarrow{\$} \mathcal{R}_Q$ and produces a sample of the following form:

$$d_2 = \left\lfloor \frac{[a]_{q_0^*} \cdot s + \Delta m}{q_0^*} \right\rceil$$

We can bound the probability that any distinguisher can distinguish between the outputs of D_1 and D_2 by upper bounding the probability that these outputs are different. By corollary 3.4.1, we can say that the marginal distribution of each coefficient of the polynomial $[a]_{q_0^*} \cdot s + \Delta m$ modulo q_0^* is uniformly random, so this is upper bounded by $2n\frac{B}{q_0^*}$. □

3.2 Avoiding Multi-Precision Arithmetic

The naïve approach to the above operation is to take an integer x in DCRT representation $\{x_0, \ldots, x_{k-1}\}$, recombine into a multi-precision integer, then perform a multi-precision divide and floor by q_0^*, then take the result mod q_0. In total, this requires $2k$ integer multiplications, k integer additions, 1 multi-precision divide-and-floor, and $k + 1$ modular reductions for each coefficient.

To avoid multi-precision arithmetic required when operating over elements modulo the full ciphertext modulus, we make use of the linearity of the DCRT recombination. Let $Q = \prod_i q_i$, let $q_i^* = Q/q_i$, and let $\tilde{q}_i = q_i^* \pmod{q_i}$. From this equation, we have the following expression for division by q_0^*:

$$\left\lfloor \frac{x}{q_0^*} \right\rfloor_{q_0} = \left[\frac{1}{q_0^*} \left(\left(\sum_{i=0}^{k-1} [x_i \cdot \tilde{q}_i]_{q_i} \cdot q_i^* \right) - v \cdot q \right) \right]_{q_0} \tag{13}$$

$$= \left[\frac{1}{q_0^*} \left(\sum_{i=0}^{k-1} [x_i \cdot \tilde{q}_i]_{q_i} \cdot q_i^* \right) \right]_{q_0} \tag{14}$$

$$= \left[\sum_{i=0}^{k-1} [x_i \cdot \tilde{q}_i]_{q_i} \cdot \frac{q_0}{q_i} \right]_{q_0} \tag{15}$$

From equation 15 above, we have that only the terms $\tilde{q}_i$ and q_0/q_i for all $i \in [k]$ are needed to compute the desired term $\lfloor x/q_0^* \rfloor_{q_0}$. Since these values only depend on the choice of factors of the ciphertext modulus, we can easily precompute them, which allows us to perform ciphertext compression with k integer multiplications, k floating point multiplications, k floating point additions, and $k + 1$ modular reductions. Comparing with the naïve approach, this technique replaces the entire multi-precision divide-and-floor operation with only the marginal cost increase of k floating point addition and multiplication operations versus k integer addition and multiplication operations. We refer the reader to [20] for further analysis of the benefits of the DCRT representations.

3.3 Circuit Privacy Operation

We now give the full circuit privacy procedure and prove that it's secure given a sufficiently small error term relative to the divisor q_0^*. The procedure, given in algorithm 1, is very simple: we add an encryption of zero and then perform the divide-and-round operation described in section 3.1.

Algorithm 1 Circuit Privacy Procedure, denoted CP

```
Input: Ciphertext ct, Public Key pk
  $\mathsf{ct}_{zero} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, 0)$
  $\mathsf{ct}_+ = \mathsf{EvalAdd}(\mathsf{ct}_{zero}, \mathsf{ct})$
  $\mathsf{ct}_{cp} = \left\lfloor \frac{\mathsf{ct}_+}{q_0^*} \right\rceil$
Output: $\mathsf{ct}_{cp}$
```

Algorithm 2 defines the full circuit-private evaluation algorithm in terms of algorithm 1 and the original BFV Eval procedure.

Algorithm 2 Circuit Private Eval Algorithm, denoted $\mathsf{Eval_{CP}}$

```
Input: Ciphertexts $\langle \mathsf{ct}_i \rangle = \{\mathsf{ct}_i\}_{i=0}^n$, Public Key pk, Function $f$
  $\mathsf{ct}_{in} \leftarrow \mathsf{BFV.Eval}(\mathsf{evk}, f, \langle \mathsf{ct}_i \rangle)$
  $\mathsf{ct}_{cp} \xleftarrow{\$} \mathsf{CP}(\mathsf{ct}_{in}, \mathsf{pk})$
Output: $\mathsf{ct}_{cp}$
```

Finally we define BFV-CP to be a leveled homomorphic encryption scheme specified by the tuple (BFV.KeyGen, BFV.Encrypt, BFV.EncryptSK, BFV.Decrypt, $\mathsf{Eval_{CP}}$).

We will now argue that if the noise level of the ciphertext generated by BFV.Eval subroutine within the $\mathsf{Eval_{CP}}$ does not exceed a pre-specified bound, BFV-CP is circuit private (Definition 2.1). The high-level proof strategy is to define three hybrids ($\mathsf{Hybrid_{Real}}$, $\mathsf{Hybrid_{NoiseFree}}$, $\mathsf{Hybrid_{Ideal}}$) and show that they are indistinguishable from the point of view of a computationally bounded distinguisher.

Define $\mathcal{D}(\mathsf{ct}', \langle \mathsf{ct}_i \rangle, \mathsf{BFV.pk}, \mathsf{BFV.sk}, \mathsf{BFV.evk}) \to \{0, 1\}$ to be a PPT distinguisher, where ct_i are valid BFV encryptions of messages m_i respectively. The domain of the first argument (ct') is the set of BFV ciphertexts.

- $\mathsf{Hybrid_{Real}}$
  This hybrid corresponds to the real-world where the adversary tries to learn the function being computed from the ciphertext that is generated by the evaluator. Concretely, we define $\mathsf{ct}' \leftarrow \mathsf{Eval_{CP}}(\langle \mathsf{ct}_i \rangle, \mathsf{BFV.pk}, f)$.
  By the correctness of BFV, the intermediate ciphertext (ct_{in}) computed in $\mathsf{Eval_{CP}}$ is a valid encryption of $f(m_1, \ldots m_n)$. Represent this as $\mathsf{ct}_{in} = (a_{in}, a_{in} \cdot s + \Delta m + e_{in})$. Additionally,

represent $\mathsf{pk} = (a', a's + e')$.[1] The encryption of zero has the form,

$$\mathsf{ct}_{zero} = (a'u + e'', a'su + e'u + e''')$$

where $u, e'', e''' \leftarrow \chi$.
Hence, the ciphertext ct_+ has the form,

$$\begin{aligned}\mathsf{ct}_+[0] &= a_{in} + a'u + e'' = a_+\\ \mathsf{ct}_+[1] &= a_{in} \cdot s + \Delta m + e_{in} + a'su + e'u + e'''\\ &= a_+ \cdot s + \Delta m + e_+\end{aligned}$$

where $e_+ = e_{in} + e'u + e''' - s \cdot e''$.
Performing the divide-and-floor operation during the CP procedure on ct_+ results in a new ciphertext $\mathsf{ct}_{quot} = \lfloor \mathsf{ct}_+/q_0^* \rfloor$, which simplifies to,

$$\mathsf{ct}_{quot}[0] = \left\lfloor \frac{a_+}{q_0^*} \right\rfloor = a'_+ \tag{16}$$

$$\mathsf{ct}_{quot}[1] = \left\lfloor \frac{a_+ \cdot s + \Delta m + e_+}{q_0^*} \right\rfloor \tag{17}$$

$$= a'_+ \cdot s + \left\lfloor \frac{[a_+]_{q_0^*} \cdot s + \Delta m + e_+}{q_0^*} \right\rfloor \tag{18}$$

where $a_+ = a'_+ \cdot q_0^* + [a_+]_{q_0^*}$.
Note that the ciphertext $\mathsf{ct}' := \mathsf{ct}_{quot}$.

- $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$

  In this hybrid, we define a new simulator that outputs a ciphertext for the distinguisher $\mathcal{D}$, that is very similar to the $\mathsf{Hybrid}_{\mathsf{Real}}$. The main difference is that this simulator tweaks the computation of $\mathsf{ct}_{quot}[1]$ to set the e_+ term to zero. Concretely, we define a simulator $\mathsf{Sim}(1^l, \mathsf{pk}, \mathsf{evk}, \mathsf{sk}, f, \langle ct_i \rangle) \rightarrow \mathsf{ct}'$, which recomputes all the quantities computed in the $\mathsf{Hybrid}_{\mathsf{Real}}$. It then uses the secret key sk to compute e_{in}. Subsequently it computes e_+ and which can be subtracted from the $\mathsf{ct}_+[1]$.
  Thus, the output of this simulator ct' simplifies to,

$$\mathsf{ct}'[0] = \left\lfloor \frac{a_+}{q_0^*} \right\rfloor = a'_+ \tag{19}$$

$$\mathsf{ct}'[1] = a'_+ \cdot s + \left\lfloor \frac{[a_+]_{q_0^*} \cdot s + \Delta m}{q_0^*} \right\rfloor \tag{20}$$

- $\mathsf{Hybrid}_{\mathsf{Ideal}}$
  This hybrid corresponds to the ideal world where the adversary only has access to the function output $m = f(m_1 \dots m_n)$ but has no access to the function f. We define a new simulator $\mathsf{Sim}(1^l, \mathsf{pk}, \mathsf{evk}, \mathsf{sk}, m) \rightarrow \mathsf{ct}'$ whose output will be sent to the distinguisher $\mathcal{D}$. Note that this is the same simulator required by definition 2.1.
  The simulator samples a uniformly polynomial r from $\mathcal{R}_q$. It then outputs,

$$\mathsf{ct}'[0] = \left\lfloor \frac{r}{q_0^*} \right\rfloor = r' \tag{21}$$

$$\mathsf{ct}'[1] = r' \cdot s + \left\lfloor \frac{[r]_{q_0^*} \cdot s + \Delta m}{q_0^*} \right\rfloor \tag{22}$$

[1]Since we are performing a simple homomorphic operation without homomorphic multiplications or rotations, it is sufficient for evk to simply be the public key pk.

Given a Hybrid, let $P[\mathsf{Hybrid}]$ denote a probability defined as follows,

$$\Pr[\mathsf{Hybrid}] = \Pr\left[\mathcal{D}\left(\cdot, \langle \mathsf{ct}_i \rangle, \mathsf{sk}, \mathsf{pk}, \mathsf{evk}\right) = 1\right]$$

.

Lemma 3.6. *If* $n\frac{||e_+||}{q_0^*} \leq 2^{-\lambda}$ *for security parameter* λ*, then no PPT distinguisher can distinguish* Hybrid_0 *and* Hybrid_1 *with probability better than* $2^{-\lambda}$.

$$|\Pr[\mathsf{Hybrid}_{\mathsf{Real}}] - \Pr[\mathsf{Hybrid}_{\mathsf{NoiseFree}}]| \leq 2^{-\lambda}$$

Proof. We can upper bound the success of any PPT distinguisher D from distinguishing the outputs of $\mathsf{Hybrid}_{\mathsf{Real}}$ and $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$ by the probability that the outputs of these hybrids are different. The only place where these hybrids differ is in the error term that is added to the numerator. In equation 18 in $\mathsf{Hybrid}_{\mathsf{Real}}$, there is an e_+ term that is added in the flooring numerator, while in equation 20 in hybrid $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$, this term is omitted.

To invoke lemma 3.5, we must show that each coefficient of a_+ is uniformly random modulo q_0^*. This can be seen through a straightforward application of the same argument used in the proof of lemma 3.5 on the coefficients of the product $a' \cdot u$. Once we show that these coefficients are random, the fixed offsets defined by the coefficients of the rest of the terms in a_+ do not change this fact. Therefore, we can invoke lemma 3.5 on the v term in hybrid $\mathsf{Hybrid}_{\mathsf{Real}}$ to show that this term hides v and is statistically close to the distribution of the v terms in $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$. □

Lemma 3.7. *Assuming the hardness of* $\mathsf{RLWE}_{n,q,\chi}$*, no PPT distinguisher* $\mathcal{D}$ *can distinguish* $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$ *from* $\mathsf{Hybrid}_{\mathsf{Ideal}}$ *with non-negligible advantage.*

$$|\Pr[\mathsf{Hybrid}_{\mathsf{NoiseFree}}] - \Pr[\mathsf{Hybrid}_{\mathsf{Ideal}}]| \leq \mathsf{negl}(\lambda)$$

Proof. Assume a distinguisher $\mathcal{D}$ exists that can distinguish $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$ from $\mathsf{Hybrid}_{\mathsf{Ideal}}$ with probability at least δ. We can use $\mathcal{D}$ to construct an adversary $\mathcal{A}$ that breaks the RLWE problem with advantage δ. Given an RLWE sample (a, b), we can construct a key-pair and ciphertext such that the ciphertext will have the form of a $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$ sample if $b = a \cdot u + e$ and the form of a $\mathsf{Hybrid}_{\mathsf{Ideal}}$ sample if b is uniform over $\mathcal{R}_q$.

Given an RLWE sample (a, b), begin by sampling $s', e' \xleftarrow{\$} \chi$ and set $\mathsf{sk} = s'$ and $\mathsf{pk} = (a, as' + e') = (\mathsf{pk}_0, \mathsf{pk}_1)$. Next, set $r' = b$ and compute ct_{quot} as in equation 22. If $b = au + e$ then r' is identically distributed to a_+ in $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$ and if b is uniform then r' is identically distributed to $\mathsf{Hybrid}_{\mathsf{Ideal}}$. Therefore, any PPT distinguisher $\mathcal{D}$ that can distinguish $\mathsf{Hybrid}_{\mathsf{NoiseFree}}$ from $\mathsf{Hybrid}_{\mathsf{Ideal}}$ can solve decisional RLWE with the same probability. □

Theorem 3.8 (Circuit Privacy). *Given that the input ciphertext and public key are well-formed, the input ciphertext error has magnitude bounded by* B' *and the error distribution* χ *is* B*-bounded, and for a security parameter* λ *we have*

$$2n\frac{2nB^2 + B + B'}{q_0^*} \leq 2^{-\lambda} \tag{23}$$

then algorithm 2 achieves $2^{-\lambda}$*-circuit privacy as defined in definition 2.1.*

Proof. This follows by combining lemmas 3.6 and 3.7, since $\mathsf{Hybrid}_{\mathsf{Real}}$ is equivalent to the distribution of outputs of algorithm 2 and $\mathsf{Hybrid}_{\mathsf{Ideal}}$ is a distribution that is independent of the circuit used to compute the final message. The magnitude of e_+ in lemma 3.6 is B' plus the magnitude of the error of a public key encryption scheme using error distribution χ. The bound in equation 23 follows. Therefore, algorithm 2 achieves $2^{-\lambda}$-circuit privacy as defined in definition 2.1. □

3.4 Correctness

In this subsection, we give the correctness of our circuit privacy procedure. Note that since we've already shown the security of this operation, we don't have to worry about equivalent expressions leaking information about the original error term.

We begin with the expression for the ciphertext quotient.

$$\begin{aligned}
\mathsf{ct} &= \left(a',\ a'\cdot s+v\right) = \left(a',\ a'\cdot s + \left\lfloor \frac{[a]_{q_0^*}\cdot s+\Delta m+e}{q_0^*}\right\rceil\right)\\
&= \left(a',\ a'\cdot s + \left\lfloor \frac{[a]_{q_0^*}\cdot s+e}{q_0^*} + \frac{1}{q_0^*}(\frac{Q}{p}-\frac{[Q]_p}{p})m\right\rceil\right)\\
&= \left(a',\ a'\cdot s + \left\lfloor \frac{[a]_{q_0^*}\cdot s+e}{q_0^*} + (\frac{q_0}{p}-\frac{[Q]_p}{q_0^*\cdot p})m\right\rceil\right) \qquad (24)\\
&= \left(a',\ a'\cdot s + \left\lfloor \frac{q_0}{p}\right\rfloor m + e_f + \left\lfloor \frac{[a]_{q_0^*}\cdot s - \frac{[Q]_p}{p}m+e}{q_0^*}\right\rceil\right)\\
&= \left(a',\ a'\cdot s+\Delta' m + e_f + v'\right)
\end{aligned}$$

where $\Delta' = \lfloor q_0/p \rfloor$ and e_f is the small error term ($||e_f|| \leq 1$) introduced by removing $\lfloor q_0/p\rfloor m$ from the flooring term. In section 4, we will give concrete parameters that result in sufficiently small errors to allow for decryption correctness, but at a high level this ciphertext is decryptable if

$$\Delta'/2 > ||v' + e_f||$$

4 OUR VOLE PROTOCOL

In this section, we give our VOLE protocol. Let m be the length of the VOLE vectors and let n be the dimension of the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n+1)$. Our VOLE protocol splits the vectors in $\mathbb{Z}_p^m$ over $\tau = \lceil m/n \rceil$ elements of $\mathcal{R}_p$, padding with zeros when necessary. The sender then uses homomorphic encryption operations to compute the VOLE result, which is decrypted by the receiver.

In more detail, the receiver begins the protocol by encoding the input $x \in \mathbb{Z}_p$ in a single BFV ciphertext ct_x, which it sends to the sender. The sender's inputs $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^m$ are split into τ elements $(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(2)}, \dots, \boldsymbol{\alpha}^{(\tau)})$ and $(\boldsymbol{\beta}^{(1)}, \boldsymbol{\beta}^{(2)}, \dots, \boldsymbol{\beta}^{(\tau)})$. For each of the τ blocks, the sender computes a ciphertext

$$\mathsf{ct}^{(i)} = \mathsf{EvalAddPlain}(\mathsf{EvalMultPlain}(\mathsf{ct}_x, \boldsymbol{\alpha}^{(i)}), \boldsymbol{\beta}^{(i)})$$

The sender then performs the circuit privacy procedure from section 3 and returns each of the resulting ciphertexts to the receiver. The receiver then decrypts the τ blocks to obtain the VOLE result.

The pseudo-code for the sender and receiver roles are given in algorithms 3 and 4.

Algorithm 3 Receiver VOLE Protocol

```
Input: $x \in \mathbb{Z}_p$, Secret Key $\mathsf{sk} \in \mathcal{R}_q$
    $\mathsf{ct}_{in} \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{sk}, x)$
    Sender $\xleftarrow{net} \mathsf{ct}_{in}$
    $\{\mathsf{ct}_{res}^{(i)}\}_{i=1}^{\tau} \xleftarrow{net}$ Sender
    $\{\boldsymbol{\gamma}^{(i)}\}_{i=1}^{\tau} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}_{res}^{(i)})$
Output: $\boldsymbol{\gamma} = (\boldsymbol{\gamma}^{(1)}, \boldsymbol{\gamma}^{(2)}, \dots, \boldsymbol{\gamma}^{(\tau)})$
```

Algorithm 4 Sender VOLE Protocol

```
Input: VOLE inputs $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}_p^m$, Public Key pk
    $\mathsf{ct}_{in} \xleftarrow{net}$ Receiver
    for $i$ from 1 to $\tau$ do
        $\mathsf{ct}_{ole}^{(i)} \leftarrow \mathsf{EvalAddPlain}(\mathsf{EvalMultPlain}(\mathsf{ct}_{in}, \boldsymbol{\alpha}^{(i)}), \boldsymbol{\beta}^{(i)})$
        $\mathsf{ct}_{cp}^{(i)} \leftarrow \mathsf{CP}(\mathsf{ct}_{ole}^{(i)}, \mathsf{pk})$
    end for
    Receiver $\xleftarrow{net} \mathsf{ct}_{cp} = \{\mathsf{ct}_{cp}^{(i)}\}_{i=1}^{\tau}$
Output: $\perp$
```

In sections 4.1 and 4.2, we show the security and correctness of the VOLE protocol defined by algorithms 3 and 4.

4.1 Security

We will now argue the security of the VOLE protocol.

Lemma 4.1 (Security Against Sender). *If the parameters of the homomorphic encryption scheme are chose to satisfy semantic security, algorithm 3 achieves security against an honest-but-curious sender.*

Proof. This follows directly from the semantic security of the encryption scheme. □

The security against a semi-honest receiver is dependent on the choosing parameters that satisfy the constraints given in theorem 3.8.

Lemma 4.2 (VOLE Circuit Privacy Parameter). *Let χ be a B-bounded error distribution. For error distribution χ, polynomial degree n, plaintext modulus p, ciphertext modulus $q = q_0 \cdot q_0^*$, and security parameter λ, the following parameter bound will result in the homomorphic evaluation in algorithm 4 to maintain $2^{-\lambda}$-circuit privacy.*

$$2n\frac{2nB^2+B+npB}{q_0^*} \leq 2^{-\lambda} \qquad (25)$$

Proof. This follows from theorem 3.8 if we show that the input to the circuit privacy procedure has an error term with magnitude bounded by npB. This follows directly from the analysis of the error growth of the EvalMultPlain operation. The encrypted input to the EvalMultPlain function in algorithm 4 is a fresh encryption of the receiver's input which has an error term with magnitude upper bounded by B, so the magnitude of the resulting error term is no more than npB. The bound in equation 25 follows from theorem 3.8 where $B' = npB$. □

Lemma 4.3 (Security Against Receiver). *Given that the parameters of the homomorphic encryption scheme are chosen to satisfy the circuit privacy constraint in lemma 4.2, algorithm 4 achieves security against an honest-but-curious receiver.*

Proof. The proof of this lemma relies on the circuit privacy result of theorem 3.8. Note that we crucially rely on the receiver's initial ciphertext being well-formed. If this is the case, then a simulator $\mathcal{S}$ can use the simulator from theorem 3.8 to return a ciphertext to the receiver that is independent of the circuit used to compute the final message $\boldsymbol{\gamma}$ by sampling a ciphertext as in Hybrid_2 from section 3. Since $\mathcal{S}$ knows the correct output message $\boldsymbol{\gamma}$, this ciphertext is indistinguishable from the output of the circuit privacy procedure in the real algorithm 4. Therefore, by theorem 3.8, the message produced by algorithm 4 is indistinguishable from the output produced by the simulator that samples a ciphertext from Hybrid_2. □

Theorem 4.4 (Secure VOLE Protocol). *Algorithms 3 and 4 define of a secure* VOLE *protocol.*

Proof. This follows from combining lemmas 4.1 and 4.3. □

4.2 Correctness

We will now give parameters for which the VOLE protocol is correct.

Lemma 4.5 (Correctness). *Let χ be a B-bounded error distribution. For error distribution χ, polynomial degree n, plaintext modulus p, and ciphertext modulus $q = q_0 \cdot q_0^*$. Algorithms 3 and 4 achieve the correct* VOLE *functionality as defined in definition A.1 if q_0^* is chosen to satisfy the circuit privacy constraint from lemma 4.2 and $q_0 > 2pnB + 2p + [q_0]_p$, where v is the compressed error term from equation 3.*

Proof. This follows directly from the BFV scheme. □

4.3 Simple Extension to Batched OLE

The extension of this VOLE protocol to a BOLE protocol is straightforward and preserves the security proofs of section 4.1. In addition, this BOLE extension maintains the exact communication complexity of the VOLE protocol. Only the receiver's role is modified in this extension. We show the receiver's BOLE procedure in algorithm 5. The receiver begins with a vector $\mathbf{x}$ of values, which she encodes as a polynomial as described in the full version of this paper. The operations performed by the sender are identical. To obtain the BOLE output, the receiver must decode the decryption result by evaluating the resulting polynomial at the roots of unity determined by the encoding NTT parameters.

Algorithm 5 Receiver BOLE Protocol

```
Input: $\mathbf{x} \in \mathbb{Z}_p^m$, Secret Key $\mathsf{sk} \in \mathcal{R}_q$
  Let $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(\tau)}) \leftarrow \mathbf{x}$, where $\tau = \lceil m/n \rceil$
  for $i$ from 1 to $\tau$ do
    $\mathsf{ct}_{in}^{(i)} \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{sk}, \mathbf{x}^{(i)})$
  end for
  Sender $\xleftarrow{net} \{\mathsf{ct}_{in}^{(i)}\}_{i=1}^{\tau}$
  $\{\mathsf{ct}_{res}^{(i)}\}_{i=1}^{\tau} \xleftarrow{net}$ Sender
  $\{\boldsymbol{\gamma}^{(i)}\}_{i=1}^{\tau} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}_{res}^{(i)})$
Output: $\boldsymbol{\gamma} = (\boldsymbol{\gamma}^{(1)}, \boldsymbol{\gamma}^{(2)}, \ldots, \boldsymbol{\gamma}^{(\tau)})$
```

5 IMPLEMENTATION & PERFORMANCE

In this section, we give an implementation of the VOLE and BOLE protocols presented in section 4.

5.1 Parameter Selection

In both of these protocols, there are four parameters that must selected: the magnitude bound B of the B-bounded error distribution χ, the plaintext modulus p, the ciphertext modulus q, and the ring dimension n. These parameter sets must satisfy both the computational security requirements of the RLWE problem as well as the security requirements of the circuit-privacy procedure. In the VOLE protocol, there are no restrictions on the plaintext modulus p other than its size, so we opt for powers of two for ease of comparison. In the BOLE protocol, the plaintext modulus must support the NTT operation to encode the vectors as polynomials for component-wise operations. This requires that $\mathbb{Z}_p$ contain a $2n^{\text{th}}$ root of unity.

From the previous section, we have several constraints on the parameters for the protocol to satisfy both security and correctness.

(1) For circuit privacy parameter λ_{cp}, lemma 4.2 gives the following lower bound on q_0^*.

$$1 - n\frac{2nB^2 + B + npB}{q_0^*} \geq 1 - 2^{-\lambda_{cp}}$$

$$q_0^* \geq 2^{\lambda_{cp}} \cdot n(2nB^2 + B + npB)$$

(2) For correctness, lemma 4.5 gives the following lower bound on q_0.

$$q_0 > 2pnB$$

(3) Finally, for semantic security, we must choose n and q to satisfy the desired security level. These parameters are given in table 1 in the Homomorphic Encryption Standard [2]. In particular, this table gives a maximum size of q for a given value of n. We must select an n that is large enough for our choice of q.

Since we have competing constraints on q, it is not immediately clear that we can pick parameters that will be able to satisfy all these constraints. Luckily, there are many parameter sets that satisfy these conditions.

Our implementation of the discrete Gaussian sampling algorithm follows [16], which takes advantage of the small standard deviation to simply compute the probability of sampling all of the most common integers. This truncated distribution is statistically close to the real discrete Gaussian distribution while still allowing us to bound the maximum size of a sampled term. Following the homomorphic encryption standard, we use a standard deviation of $\sigma = 3.2$. Including the sign bit, no term sampled from χ is greater than five bits, so $B = 32$ for all our parameter sets.

In most of the parameter sets in this section, each limb of the ciphertext modulus will be 55 bits. For a circuit privacy parameter of $\lambda_{cp} = 80$, we will use a ciphertext modulus consisting of four limbs for a total of 220 bits. This is the maximum size of the modulus for

$n = 2^{13}$ in table 1 of the HE standard [2]. For a plaintext modulus $p = 2^{32}$, these parameters satisfy the three constraints given above.

5.2 Extending to Larger Moduli

There are some applications which require moduli that are larger than can be supported by a final q_0 that fits in a standard machine word. We handle these moduli by using the same DCRT representation as the ciphertext modulus. In words, for a plaintext modulus p that is too large for a standard machine word, we represent p as a product of primes $p = \prod_{i=0}^{\ell} p_i$ where each p_i is small enough to be supported by a q_0 that fits in a standard machine word for the given circuit privacy parameter. This allows us to extend the support of the protocol in section 4 to larger moduli with a factor of ℓ overhead.

5.3 Optimizing for Low-Bandwidth Networks

In section 5.4, we will benchmark our implementation in an environment with low network bandwidth. We define this as a setting where the network bandwidth is the bottleneck of the protocol. To reduce the overall communication of our protocol, we include in our implementation a common optimization to reduce the size of a fresh Ring LWE encryption by a factor of two. The idea is simple: instead of sampling a random a polynomial for a ciphertext (a, b), sample a random PRG seed σ and generate $a \leftarrow PRG(\sigma)$ to be the output of the PRG. The remainder of the ciphertext is generated as if a were sampled normally, but when it comes time to send the ciphertext we only need to send the pair (σ, b), replacing the element of $\mathcal{R}_q$ with a 16 byte PRG seed.

5.4 Experimental Setup & Results

We implement[2] the VOLE and BOLE protocols from section 4 on top of an implementation of the BFV [11, 17] homomorphic encryption scheme based on the BFV-RNS implementation of Halevi, Polyakov, and Shoup [20] in addition to code from the work of Juvekar, Vaikuntanathan, and Chandrakasan [24]. We use an NTT implementation based off of the NFLlib library of Aguilar-Melchor, Barrier, Guelton, Guinet, Killijian, and Lepoint [1].

Asymptotically Linear Comparisons. As mentioned in section 1.2, we separate prior work into two categories and compare to each category separately. The first category consists of protocols with (asymptotically) linear communication complexity, and, more heuristically, protocols that are optimized for small VOLE lengths. The fastest protocol in this category that we are aware of is the work of Applebaum, Dåmgard, Ishai, Nielsen, and Zichron [3]. To compare against [3], we ran our protocol on two AWS `m5.2xlarge` instances, each having 8 vCPUs at 3.1 GHz and 32 GB of RAM. These instances were in the same geographic region (US-East) and were connected by a network with a bandwidth of 500 MB/sec. This setup was intended to replicate the conditions of the experiments of [3] (see section 6.5). Our network bandwidth is much lower than in their setup which favors [3] as their communication complexity does not exceed the bandwidth of the network we used. We employ the communication optimization described in section 5.3. We also note that the clock speed of the CPU of our machine is slightly slower than the machine used for the benchmarks in [3], but this is not considered in our comparisons (all numbers are reported without scaling from [3]).

[2]https://github.com/leodec/ole_wahc

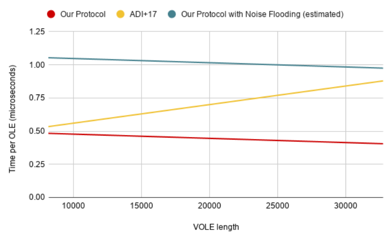

Figure 1: Time per OLE multiplication vs. VOLE length compared to [3] (denoted ADI+17). Times are measured in microseconds, and the VOLE protocol is over a 32-bit field. The runtime for the noise-flooding variant of our protocol was estimated by implementing the noise sampling function using NTL [33] and adding this sampling time to our protocol time.

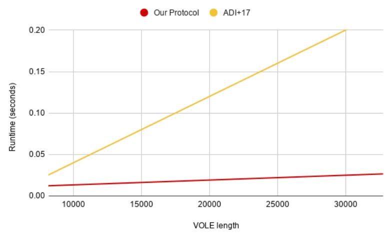

Figure 2: Time for a single run of the VOLE protocol vs. VOLE length compared to [3] (denoted ADI+17). Times are measured in seconds, and the VOLE protocol is over a 32-bit field.

Using this setup, we took two types of benchmarks. The first benchmark is of the protocol running between the two machines using all of the threads. Based on this runtime, we divided by the length of the VOLE protocol to get the time per OLE multiplication. We then ran the protocol once using a single thread per machine to measure the latency, the time that a higher level application must wait for the protocol to complete. We compare these results against the numbers given in [3] (see tables 2 and 3 in [3]) for their optimized 32 bit protocol. These comparisons are given in figures 1 and 2. We note that the trend lines for the [3] protocol do not consider the increase in communication of the protocol, which could result in further slowdown in settings where the bandwidth is constrained. We also note that this graph includes an estimated benchmark for the time per OLE for our protocol if the circuit privacy analysis from section 3 is not considered and noise flooding is used to achieve circuit privacy. This runtime is estimated by implementing the sampling function for the noise flooding term in NTL [33], then adding the time for this sampling algorithm to our runtime.

Finally, we measured the communication complexity of a single run of our VOLE protocol and compared it to estimates of the communication complexity of [3] based on the reported consumed bandwidth for their protocol. This comparison is displayed in figure

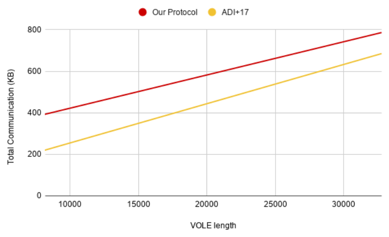

Figure 3: Total communication in kilobytes for a single run of the VOLE protocol vs. VOLE length compared to [3] (denoted ADI+17). The VOLE is over a 32-bit field. Note that the slope of the ADI+17 trend line is greater than the slope of our protocol's trend line. We estimate the cross-over point in the communication complexity to be a VOLE length of around 2^{16}.

3. For small VOLE, the protocol of [3] achieves better communication complexity than our protocol for VOLE lengths less than 2^{16}, although beyond this length our protocol achieves better communication complexity.

Asymptotically Sublinear Comparisons. The second category of prior art consists of protocols with sub-linear communication complexity. These protocols are designed to be optimized for large VOLE lengths, and due the sub-linear communication they will always be asymptotically faster than our protocol. Despite this, our protocol is faster for smaller VOLE lengths, and our comparison aims to determine VOLE length at which one protocol becomes faster than the other. By determining this point, we are able to discuss applications for which our protocol is more efficient and when it makes sense to switch to a protocol optimized for larger VOLE sizes. To our knowledge, the fastest protocol in this category is the work of Schoppmann, Gascón, Reichert, and Raykova [31], which is an optimized two-party implementation of the FSS-based sub-linear vector VOLE [8].

Since the authors of [31] included their code, we were able to run side-by-side comparisons between our protocol and theirs. Following the setup from their work, we ran each protocol on a single thread on two machines connected by a network with 500 MB/sec bandwidth. The results of these comparisons are given in figures 4 and 5. By extrapolating the data in figure 4, we estimate the the cross-over point is near a VOLE length of 2^{35}. Extrapolating the communication complexity trends in figure 5 gives an approximate cross-over point near a VOLE length of 2^{32}. In section 5.6, we give some applications that require VOLE protocols of width less than this cross-over point.

We note briefly that the protocol in [31] requires running a smaller VOLE protocol that it then expands using an LPN code. When generating an exceptionally long VOLE correlation, one could run a protocol that recursively calls [31] until it reaches a point where our protocol is faster, then runs our protocol as the base case. It is an interesting future direction to combine our protocol with [31] to improve the performance of VOLE with sub-linear communication.

We also note that both protocols in this section focus on optimizing the generation of random VOLE correlations while our protocol

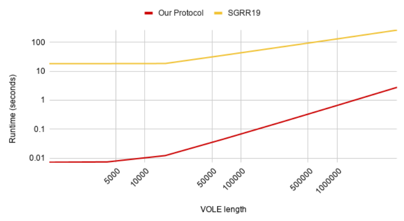

Figure 4: Time for a single run of the VOLE protocol vs VOLE length compared to [31] (denoted SGRR19). The runtime is measured in seconds, and the VOLE is over a 32-bit field. Both axes are shown in log scale to better display the trends. Based on extrapolation, we estimate the intersection point in the runtimes to be near a VOLE length of 2^{35}.

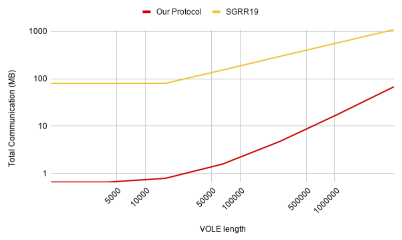

Figure 5: Total communication for a single run of the VOLE protocol vs VOLE length compared to [31] (denoted SGRR19). The communication is measured in megabytes, and the VOLE is over a 32-bit field. Both axes are shown in log scale to better display the trends. Based on extrapolation, we estimate the intersection point in the communication to be near a VOLE length of 2^{32}.

is able to generate arbitrary VOLE correlations directly. While there is a simple and secure reduction from random VOLE to arbitrary VOLE, it is an interesting future direction to try to further optimize our protocol for generating random VOLE correlations.

5.5 Comparison Against Subsequent Works

In this section, we briefly compare against works subsequent to the online publication of this work.

Comparisons to Asymptotically Linear Protocols. We compare against the one-round BOLE protocol of Baum et al. [5]. This work presents several different BOLE protocols, and we compare against the version the authors chose to benchmark. This protocol assumes that the two parties each begin with the public key of the other party, and we do not count this key exchange as part of the communication of this protocol. Aside from the round complexity, our BOLE protocol outperforms the one-round protocol in both communication and computation. To compare this one-round protocol to our C++ implementation, we implemented the one-round BOLE protocol using the same underlying RLWE library as our BOLE protocol. Based on these comparisons, our protocol is a 20-30× faster computationally in experiments where each party is running in a single thread connecting over `localhost`. Our benchmarks for the one-round protocol for a 60-bit plaintext was less than one second, faster than Golang benchmark given in Baum et al. [5] with the same parameters, which was roughly 1.5 seconds. Our total communication is strictly less than the one-round protocol, since one of their ciphertexts requires essentially the same LPR overhead as ours, while the other ciphertext that they send requires double the LPR overhead.

Comparisons to Asymptotically Sublinear Protocols. To compare to the subsequent VOLE protocol of Weng et al. [34], we note that this work claims a roughly 20× computational speedup over Schoppmann et al. [31]. We can extrapolate this comparison to estimate the crossover point for the Weng et al. protocol, which is roughly a VOLE length of 2^{28}. For communication, the Weng et al. protocol a setup cost of 1.1 MB and subsequent communication of 0.42 bits per VOLE correlation puts the crossover point with our protocol to be near a VOLE length of 2^{15}.

To compare against the BOLE protocol of Boyle et al. [10], we first note that the work does not give the benchmarks of an actual implementation, only estimates of the computation time. They give various parameter sets to allow for a communication-computation trade-off. For 128 bits of security, The estimates they provide for generating 2^{20} batch OLE correlations give performance that ranges from roughly 13 seconds with nearly 18 MB of communication to nearly 20 seconds with communication roughly 1.5 MB. The plaintext modulus of these BOLE correlations was roughly 2^{124}, and while our BOLE protocol is not optimized for these large moduli, we can estimate the performance for a plaintext modulus of this size by generating four 32-bit BOLE correlations for each 124-correlation in [10] (this simulates a CRT splitting of a 128-bit plaintext modulus). Our benchmarks for 2^{22} BOLE correlations with a 32-bit plaintext modulus give a computation time of 3 seconds and a total communication of about 200 MB.

5.6 Application: Secure Image Convolution

There are many applications for a VOLE protocol, including many that do not require VOLE correlations of length greater than the cross-over points discussed in section 5.4. One concrete example is image convolution for secure neural network evaluation. The canonical image convolution operations takes in a two-dimensional image and a small two-dimensional kernel and produces an output image where each output pixel is the sum of the component-wise product of the kernel and a region of input image. While one can describe this operation using two-dimensional Fourier transforms, one of the most efficient implementations of secure convolution is to use VOLE. In particular, one can define a VOLE correlation for each element in the kernel and then perform a scalar-vector product with the elements of the image multiplied by the specific kernel element. Even for large neural networks such as networks that process mammograms for cancer diagnosis [35], the size of a convolution does not exceed the size of the input image, which is $2028 \times 2028 = 2^{22}$ pixels. Based on the benchmarks in section 5.4, we would outperform all prior works for the generation of these VOLE correlations.

ACKNOWLEDGMENTS

We would like to thank Michael Nielsen for providing the code for [3] and Yuriy Polyakov for answering our questions about the PALISADE library. We gratefully acknowledge funding from the Jameel-Clinic and from the National Science Foundation.

REFERENCES

[1] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. 2016. NFLlib: NTT-Based Fast Lattice Library. In *Topics in Cryptology - CT-RSA 2016*, Kazue Sako (Ed.). Springer International Publishing, Cham, 341–356.

[2] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. 2018. *Homomorphic Encryption Security Standard.* Technical Report. HomomorphicEncryption.org, Toronto, Canada.

[3] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. 2017. Secure Arithmetic Computation with Constant Computational Overhead. In *Advances in Cryptology – CRYPTO 2017*, Jonathan Katz and Hovav Shacham (Eds.). Springer International Publishing, Cham, 223–254.

[4] Abhishek Banerjee, Chris Peikert, and Alon Rosen. 2012. Pseudorandom Functions and Lattices. In *Advances in Cryptology – EUROCRYPT 2012*, David Pointcheval and Thomas Johansson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 719–737.

[5] Carsten Baum, Daniel Escudero, Alberto Pedrouzo-Ulloa, Peter Scholl, and Juan Ramón Troncoso-Pastoriza. 2020. Efficient Protocols for Oblivious Linear Function Evaluation from Ring-LWE. In *Security and Cryptography for Networks*, Clemente Galdi and Vladimir Kolesnikov (Eds.). Springer International Publishing, Cham, 130–149.

[6] Avrim Blum, Adam Tauman Kalai, and Hal Wasserman. 2003. Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model. *Journal of the ACM (JACM)* 50, 4 (July 2003), 506–519. https://www.microsoft.com/en-us/research/publication/noise-tolerant-learning-parity-problem-statistical-query-model/

[7] Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. 2016. FHE Circuit Privacy Almost for Free. In *Advances in Cryptology – CRYPTO 2016*, Matthew Robshaw and Jonathan Katz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–89.

[8] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. 2018. Compressing Vector OLE. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 896–912. https://doi.org/10.1145/3243734.3243868

[9] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. 2019. Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 291–308. https://doi.org/10.1145/3319535.3354255

[10] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. 2020. Efficient Pseudorandom Correlation Generators from Ring-LPN. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing, Cham, 387–416.

[11] Zvika Brakerski. 2012. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. *Proceedings of Advances in Cryptology-Crypto* 7417 (08 2012). https://doi.org/10.1007/978-3-642-32009-5_50

[12] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Efficient Fully Homomorphic Encryption from (Standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, Rafail Ostrovsky (Ed.). IEEE Computer Society, 97–106. https://doi.org/10.1109/FOCS.2011.12

[13] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. 2019. Reusable Non-Interactive Secure Computation. In *Advances in Cryptology – CRYPTO 2019*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer International Publishing, Cham, 462–488.

[14] Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. 2020. Fast Vector Oblivious Linear Evaluation from Ring Learning with Errors. Cryptology ePrint Archive, Report 2020/685. https://ia.cr/2020/685.

[15] Léo Ducas and Damien Stehlé. 2016. Sanitization of FHE Ciphertexts. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9665)*, Marc Fischlin and Jean-Sébastien Coron (Eds.). Springer, 294–310. https://doi.org/10.1007/978-3-662-49890-3_12

[16] Nagarjun Dwarakanath and Steven Galbraith. 2014. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing* 25 (06 2014). https://doi.org/10.1007/s00200-014-0218-3

[17] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. https://eprint.iacr.org/2012/144.pdf
[18] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. *STOC.*
[19] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. *i*-Hop Homomorphic Encryption and Rerandomizable Yao Circuits. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6223)*, Tal Rabin (Ed.). Springer, 155–172. https://doi.org/10.1007/978-3-642-14623-7_9
[20] Shai Halevi, Yuriy Polyakov, and Victor Shoup. 2019. An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. In *Topics in Cryptology – CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, Proceedings (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics))*, Mitsuru Matsui (Ed.). Springer-Verlag, 83–105. https://doi.org/10.1007/978-3-030-12612-4_5
[21] Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. 2019. LevioSA: Lightweight Secure Arithmetic Computation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 327–344. https://doi.org/10.1145/3319535.3354258
[22] Yuval Ishai and Anat Paskin. 2007. Evaluating Branching Programs on Encrypted Data. In *Theory of Cryptography*, Salil P. Vadhan (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 575–594.
[23] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. 2008. Founding Cryptography on Oblivious Transfer – Efficiently. 572–591. https://doi.org/10.1007/978-3-540-85174-5_32
[24] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1651–1669. https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar
[25] Kim Laine. 2019. https://github.com/microsoft/SEAL/issues/89.
[26] Vadim Lyubashevsky and Daniele Micciancio. 2006. Generalized Compact Knapsacks Are Collision Resistant. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 144–155.
[27] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors Over Rings. *EUROCRYPT* (2010).
[28] Moni Naor and Benny Pinkas. 2006. Oblivious Polynomial Evaluation. *SIAM J. Comput.* 35 (01 2006), 1254–1281. https://doi.org/10.1137/S0097539704383633
[29] Zhiniang Peng. 2019. Danger of using fully homomorphic encryption: A look at Microsoft SEAL. *CoRR* abs/1906.07127 (2019). arXiv:1906.07127 http://arxiv.org/abs/1906.07127
[30] Y. Polyakov, K. Rohloff, and G.W Ryan. [n.d.]. PALISADE lattice cryptography library. https://git.njit.edu/palisade/PALISADE.
[31] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. 2019. Distributed Vector-OLE: Improved Constructions and Implementation. https://doi.org/10.1145/3319535.3363228
[32] SEAL 2020. Microsoft SEAL (release 3.5). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA.
[33] Victor Shoup. [n.d.]. NTL: A Library for doing Number Theory. https://shoup.net/ntl/.
[34] C. Weng, K. Yang, J. Katz, and X. Wang. 2021. Wolverine: Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits. In *2021 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1074–1091. https://doi.org/10.1109/SP40001.2021.00056
[35] Adam Yala, Constance Lehman, Tal Schuster, Tally Portnoi, and Regina Barzilay. 2019. A Deep Learning Mammography-based Model for Improved Breast Cancer Risk Prediction. *Radiology* 292 (05 2019), 182716. https://doi.org/10.1148/radiol.2019182716

A FURTHER BACKGROUND

In this appendix, we give further background omitted from section 2 due to space constraints.

A.1 Oblivious Linear Evaluation

We now give formal definitions of vector OLE and batch OLE, along with security definitions.

Definition A.1 (Vector Oblivious Linear Evaluation). *For an integer p and n, define the protocol* $\mathsf{VOLE}^{(p,n)}$ *to have the following ideal functionality. Let* $\Xi_S = \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, *let* $\Omega_S = \emptyset$, *let* $\Xi_R = \mathbb{Z}_p$, *and let* $\Omega_R = \mathbb{Z}_p^n$. *The function* $f_S(\xi_S, \xi_R) = \perp$ *for all* $(\xi_S, \xi_R) \in \Xi_S \times \Xi_R$. *For* $\xi_S = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ *and* $\xi_R = x$, *define* $f_R(\xi_S, \xi_R) = \boldsymbol{\alpha} \cdot x + \boldsymbol{\beta} \mod p$.

Definition A.2 (Batch Oblivious Linear Evaluation). *For an integer p and n, define the protocol* $\mathsf{BOLE}^{(p,n)}$ *to have the following ideal functionality. Let* $\Xi_S = \mathbb{Z}_p^n \times \mathbb{Z}_p^n$, *let* $\Omega_S = \emptyset$, *let* $\Xi_R = \mathbb{Z}_p^n$, *and let* $\Omega_R = \mathbb{Z}_p^n$. *The function* $f_S(\xi_S, \xi_R) = \perp$ *for all* $(\xi_S, \xi_R) \in \Xi_S \times \Xi_R$. *For* $\xi_S = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ *and* $\xi_R = \mathbf{x}$, *define* $f_R(\xi_S, \xi_R) = \boldsymbol{\alpha} \odot \mathbf{x} + \boldsymbol{\beta} \mod p$.

A.2 Ring Learning with Errors

In this section, we define the decisional Ring Learning with Errors (RLWE) [27] problem.

Definition A.3 (Decisional Ring Learning with Errors [27]). *For integers n and q, a ring* $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$, *and an error distribution* χ *over* $\mathcal{R}_q$, *the decisional Ring Learning with Errors problem* $\mathsf{RLWE}_{n,q,\chi}$ *is to distinguish between samples of the form* $(a, as + e)$ *and* (a, u) *where* $a, u \xleftarrow{\$} \mathcal{R}_q$ *and* $s, e \xleftarrow{\$} \chi$.

In this work, we consider χ to be the discrete, zero-centered Gaussian distribution over $\mathcal{R}_q$. This follows the homomorphic encryption security standard [2] from which we get the concrete parameters used in the implementation in section 5.

A.3 Homomorphic Encryption

In this section, we give high level definitions for the algorithms comprising a leveled homomorphic encryption scheme as well as necessary security definitions.

Definition A.4 (Leveled Homomorphic Encryption). *A leveled homomorphic encryption scheme*

$$\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Eval}, \mathsf{Decrypt})$$

is a set of PPT algorithms defined as follows:

- $\mathsf{KeyGen}(1^\lambda, 1^L) \to (\mathsf{sk}, \mathsf{pk}, \mathsf{evk})$
Given the security parameter λ *and a maximum circuit depth L, outputs a key pair consisting of a public encryption key* pk, *a secret decryption key* sk, *and an evaluation key* evk.
- $\mathsf{Encrypt}(\mathsf{pk}, m) \to \mathsf{ct}$
Given a message $m \in \mathcal{M}$ *and an encryption key* pk, *outputs a ciphertext* ct.
- $\mathsf{Eval}(\mathsf{evk}, f, \mathsf{ct}_1, \mathsf{ct}_2, \ldots, \mathsf{ct}_n) \to \mathsf{ct}'$
Given the evaluation key, a description of a function $f: \mathcal{M}^n \to \mathcal{M}$ *with multiplicative depth at most L, and n ciphertexts encrypting messages* $m_1, \ldots, m_n$, *outputs the result ciphertext* ct' *encrypting* $m' = f(m_1, \ldots, m_n)$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) = m$ *Given the secret decryption key and a ciphertext* ct *encrypting m, outputs m.*

Optionally the scheme $\mathcal{E}$ may be extended with a PPT algorithm $\mathsf{EncryptSK}(\mathsf{sk}, m) \to \mathsf{ct}$ which uses the secret key sk rather than the public key pk, to compute the ciphertext ct from the message m.

When returning a ciphertext output by the Eval function, it is often desirable for this ciphertext to hide the function f that was used to produce it. This property of the scheme is called circuit privacy, which we formally define below.