Optimizing relinearization in circuits for homomorphic encryption

Hao Chen

Microsoft Research, Redmond WA, USA
haoche@microsoft.com

Abstract

Fully homomorphic encryption (FHE) allows an untrusted party to evaluate arithmetic circuits, i.e., perform additions and multiplications on encrypted data, without having the decryption key.

Currently, the most widely known FHE schemes are those based on the hardness of the RLWE problem, and they share some common features. Ciphertext sizes grow after each homomorphic multiplication; multiplication is much more costly than addition, and the cost of homomorphic multiplication scales linearly with the input ciphertext sizes. Furthermore, there is a special *relinearization* operation that reduce the size of a ciphertext, and the cost of relinearization is on the same order of magnitude as homomorpic multiplication. This motivates us to define a discrete optimization problem, which is to decide where (and how much) in a given circuit to relinearize, in order to minimize the total computational cost.

In this paper, we formally define the *relinearize problem.* We prove that the problem is NP-hard. In addition, in the special case where each vertex has at most one outgoing edge, we give a polynomial-time algorithm.

1 Introduction

Fully homomorphic encryption (FHE) is an encryption technique which allows any untrusted party to evaluate functions on encrypted data. As a typical application, FHE allows a client to outsource computation to an untrusted cloud. It has generated great interest in fields such as health and finance, due to the need to analyze sensitive data without having access to the data itself. Since Gentry introduced the first FHE scheme in 2009, there has been a line of works that proposed new FHE schemes with improved efficiency, among which two of the most widely used schemes are [BGV14] and its scale-invariant counterpart [FV12]. Implementations of these schemes include [**?**], [CLP], and [**?**]. There has been numerous work that design applications based on these schemes. Some of them ([**?**, **?**]) evaluate machine learning models on encrypted data. Others use FHE to design secure protocols such as private information retrieval [**?**] and private set intersection [**?**].

Unfortunately, in these schemes homomorphic operations are still several-orders of magnitude slower than performing the same operation on plaintexts. Therefore, any optimization in the computation time has great interest.

In order to use FHE to evaluate a function, one first needs to express the function as an arithmetic circuit. The circuit is represented as a direct acyclic graph with each vertex being either an input, a multiplication, or an addition. In both schemes mentioned above, a fresh ciphertext is a vector of length two. When a homomorphic multiplication is performed, the length of the output ciphertext grows. More precisely, if we denote the length of a ciphertext c by $l(c)$, then

$l(c_1 \otimes c_2) = l(c_1) + l(c_2) - 1$. The length of the result of a homomorphic addition is the maximum length of the two operands, i.e., $l(c_1 \oplus c_2) = \max\{l(c_1), l(c_2)\}$.

Furthermore, the computational cost to perform a homomorphic multiplication scales linearly with its input lengths. In both schemes, we can model the amount of work it takes to perform a homomorphic multiplication between two ciphertexts c_1 and c_2 by

$$k_m(l(c_1) + l(c_2)),$$

where k_m is some scheme-dependent constant. Homomorphic additions, on the other hand, takes much less time to perform compared to multiplication. Hence in this work we will assume that additions are "free". For the same reason, we will adopt the common notation from the FHE literature, and denote by depth of a circuit by the largest number of multiplication vertices contained in a path.

Note that it is undesirable to let the ciphertext sizes grow, since it will increase both the computational cost and the storage burden. To control the ciphertext sizes, both schemes support a special operation called *Relinearization*. Effectively, relinearizing a ciphertext means reducing its length, while keeping the underlying message the same. We can use this operation to reduce the length of a ciphertext to any integer between two and its original length. The cost of relinearization scales linearly with the reduction in ciphertext length. In other words, there exists a constant k_r such that reducing the ciphertext lengths by i takes $i \cdot k_r$ units of work.

Suppose we are given an arithmetic circuit to perform on encrypted inputs. It is now an optimization problem to decide where and how much to relinearize, in order to minimize the total amount of work, consisting of multiplication cost and relinearization cost. Previous works almost always employ a simple strategy, which is to relinearize at every multiplication. In this way, the multiplication costs are kept minimal, since the lengths of inputs to any multiplication vertex are always 2 – the smallest possible.

In Section 2, we will formally describe the problem and show why this simple strategy can be sub-optimal. In Section 3, we prove that the relinearize problem is NP-hard by reducing from the knapsack problem. Finally, in Section 4, we restrict to the special case where each vertex in the circuit has at most one outgoing edge, and give a polynomial time algorithm.

1.1 Related work

The work [**?**] is an effort to find a good circuit representation of a function, in order to minimize the total computation time.

Bootstrapping is an operation that refreshes the so-called noise in FHE ciphertexts. It is an essential yet expensive operation. The two papers [LP13] and [BLMZ17] aim at minimizing the total number of bootstrapping operations in a circuit, while keeping the noise from overflowing in order to ensure the final result is correct. In their work, the authors implicitly assume the relinearization is done after every multiplication. Similarly, we will make a simplifying assumption that the boostrapping time is a constant, so that it does not factor into our optimizatoin problem. It will be interesting to combine these works in order to achieve an overall optimization that targets both operations.

Acknowledgement The author thanks Rebecca Hoberg and Mohit Singh for helpful discussions in preparing this work.

2 Problem Description

Now we can formally describe our problem. Suppose the circuit we wish to homomorphically evaluate is represented as a directed acyclic graph (DAG) $G = (V, E)$, where each vertext represents an operation (add/multiply) or input, and each edge represents a wire carrying a ciphertext. We will allow the graph G to have multi-edges, in order to capture the scenario in FHE that sometimes one needs to multiply a ciphertext with itself. We will define the relinearize problem as an integer programming problem. For every vertex i we maintain an integer variable $l^{new}(i)$ (the final length of vertex i during homomorphic evaluation of G), and an integer variable x_i, which indicates the amount of relinearization at i. We will denote the two parents of a vertex i by $p_1(i)$ and $p_2(i)$.

Then the *relinearize problem* is

$$\text{minimize } k_r \sum_{i \in V} x_i + \sum_{i = \otimes} k_m(l^{new}(i) + x_i),$$

s.t.

$$\begin{aligned} l^{new}(i) &\geq 2 && \text{for all } i \\ l^{new}(i) = l^{new}(p_1(i)) + l^{new}&(p_2(i)) - 1 - x_i && \text{if } i = \otimes \\ l^{new}(i) &\geq l^{new}(p_1(i)) - x_i && \text{if } i = \oplus \\ l^{new}(i) &\geq l^{new}(p_2(i)) - x_i && \text{if } i = \oplus \\ x_i, l^{new}(i) &\in \mathbb{Z}_{\geq 0} && \text{for all } i \end{aligned}$$

2.1 An example

We consider the following circuit:

$$\begin{tikzcd}
 & & & & & \otimes & & & & & \\
 & & & \otimes \arrow[urr] & & & & \oplus \arrow[ull] & & & \\
 & \oplus \arrow[urr] & & & & \otimes\, u \arrow[ull] \arrow[urr] & & & & \oplus \arrow[ull] & \\
\bigcirc \arrow[ur] & & \bigcirc \arrow[ul] & & \bigcirc \arrow[ur] & & \bigcirc \arrow[ul] & & \bigcirc \arrow[ur] & & \bigcirc \arrow[ul]
\end{tikzcd}$$

First, we apply the simple strategy and relinearize at every multiplication vertex. Then the total cost equals $12k_m + 3k_r$.

Alternatively, we can choose to only relinearize the vertex u. Then the multiplication cost increases to $14k_m$, while the relinearization cost is k_r, so the total cost is $14k_m + k_r$. Comparing this with the previous cost, we see that as long as $k_r > k_m$, the simple strategy is not optimal.

3 NP-hardness of the Relinearize Problem

In this section we reduce the knapsack problem to the relinearize problem, which establishes that the latter problem is NP-hard. First we recall the definition of knapsack problem. We have positive integers $v_1, \ldots, v_n$ (the values), $w_1, \ldots, w_n$ (the weights), and W (the capacity). The problem is to maximize $\sum v_i x_i$ subject to $x_i \in \{0, 1\}$ and $\sum w_i x_i \leq W$.

For our convenience, we make some modifications to the setting of the relinearize problem. We change the inputs lengths from two to one, and instead of $l(c1 \otimes c2) = l(c1) + l(c2) - 1$, we maintain that $l(c1 \otimes c2) = l(c1) + l(c2)$. Indeed, under this formulation the length of every vertex is smaller by one. Hence it is equivalent to the original problem.

To assist the proof, we make some definitions.

Definition 1. A circuit of type $\mathcal{L}(k)$ is one that does certain number of multiplications either with itself or with the input, such that if the first non-input vertex length is reduced from 2 to 1, then the length of the final vertex reduces by k.

Figure 1 is an example of $\mathcal{L}(7)$.

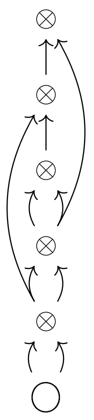

Figure 1: $\mathcal{L}(7)$

Lemma 1. *For all integers $k \geq 1$. $\mathcal{L}(k)$ can be realized with a circuit consisting of $O(\log k)$ verticies, and multiplication cost $k_m \cdot O(k \log k)$.*

Proof. If k is a power of 2, we can realize $\mathcal{L}(k)$ by a circuit that does $O(\log(k))$ consecutive squaring. The total cost of executing the circuit is $k_m \cdot O(k)$. In general, we can start by building the circuit $\mathcal{L}(2^{[\log(k)]})$. for Then every nonzero bit in the binary representation of k, we can handle by adding a multiplication vertex. Since there are at most $\log(k)$ bits, we know the number of vertices is at most $O(\log k)$. As for the cost it is also clear since the length of each vertex is $O(k)$. □

Next we describe some simple ways to construct new circuits from old ones.

Definition 2. (1) The addition of two circuits. Suppose two circuits G_1 and G_2 have unique sink vertices v_1 and v_2. The addition $G_1 \boxplus G_2$ is the circuit that is the union of G_1 and G_2, plus an extra addition vertex that has v_1 and v_2 as parents. See Figure 2 for an example.

(2) The concatenation of two circuits. Let G_1, G_2 be two circuits such that the number of sink vertices of G_1 is equal to the number of inputs of G_2. Then we simply "feed" the outputs of G_1 to inputs of G_2. We denote the resulting circuit by $G_1 \curvearrowright G_2$. See Figure 3 for an example.

(3) The K-repeat of a circuit along a subset of vertices. Let G be a circuit and let $S = \{s_1, \ldots, s_k\}$ be vertices of G. Let K be a positive integer. Then we keep the vertices v_i and all their ancestors, and copy the rest of the circuit K times. The resulting circuit is denoted by $G_S^{(K)}$. See Figure 4 for an example.

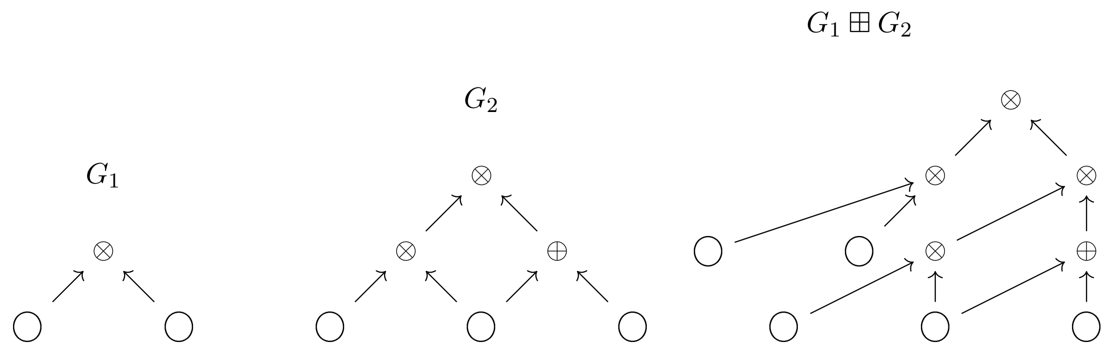

Figure 2: Example of $G_1 \boxplus G_2$

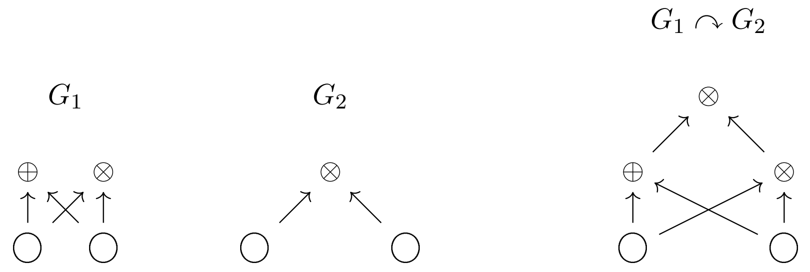

Figure 3: Example of $G_1 \curvearrowright G_2$

(4) The gluing of two circuits along a subset of vertices. Let G_1 and G_2 be two circuits and S_1, S_2 be subsets of their vertices, such that the subgraph of G_1 consisting of ancestors of S_1 (including vertices in S_1) is isomorphic to the corresponding subgraph in G_2. Then the gluing of G_1 and G_2 along S_1, S_2 is the circuit that contains the common subgraph and the disjoint union of the rest of the two graphs. We denote the new circuit by $G_1 \star_{S_1} G_2$ when S_2 and the isomorphism is clear from context. See Figure 5 for an example. Note that (3) is a special case of (4).

Now we are ready to state and prove our main theorem. Consider a knapsack problem with parameters $v_i (1 \leq i \leq n), w_i (1 \leq i \leq n), W$.

Theorem 1. *There exists a circuit $G = G(v_i, w_i, W)$, and integers k_m, k_r such that*

(1) G has $O(polylog(v_i, w_i, W)) \cdot poly(n))$ vertices.

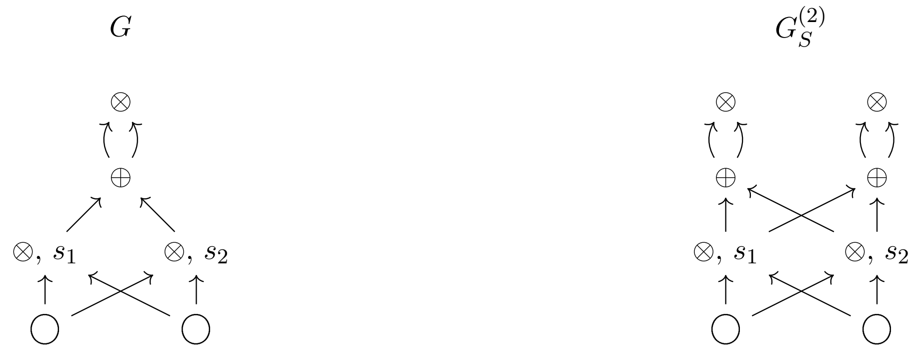

Figure 4: Example of $G_S^{(K)}$ for $K = 2$ and $S = \{s_1, s_2\}$

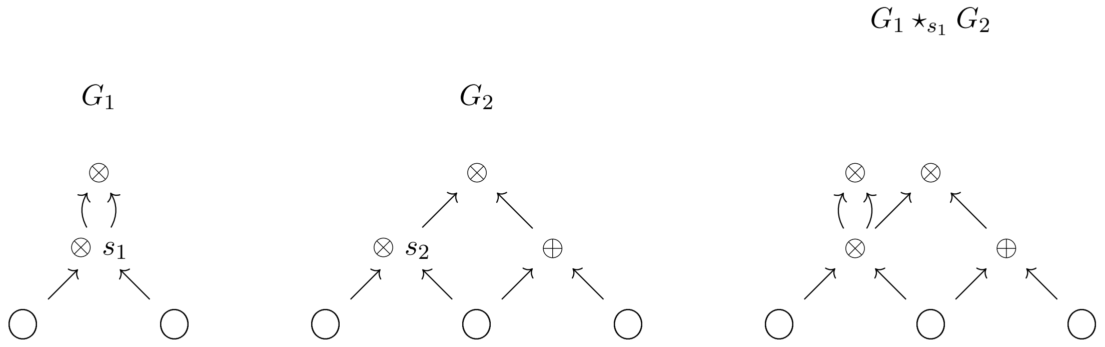

Figure 5: Example of $G_1 \star_{s_1} G_2$

(2) $k_m, k_r = O(poly(v_i, w_i, W, n))$.

(3) There exists a set of n *vertices* $s_1, \ldots, s_n$ *in* G*, such that if the length* $l^*_{new}(i)$ *is the length of* s_i *in the optimal solution to the relinearize problem on* G*. Then* $l^*_{new}(i)(1 \leq i \leq n)$ *is an optimal solution to*

$$\max \sum v_i l_i, \text{ s.t. } l_i \in \{1,2\}, \sum w_i l_i \leq W + \sum w_i,$$

Hence $l^*_{new}(i) - 1(1 \leq i \leq n)$ *s an optimal solution to the original knapsack problem.*

Since our proof is long, we will break it into several parts. First, let K, T be positive integers whose values will be determined later. We define a circuit

$$G^0 := \{(\mathcal{L}(w_1) \boxplus \cdots \boxplus \mathcal{L}(w_n) \boxplus \mathcal{L}(W')) \curvearrowright \mathcal{L}(T)\}^{(K)}_S$$

Here $W' = W + \sum_i w_i$, and $S = \{s_1, \ldots, s_n\}$, where s_i is the first non-input vertex in the circuit $\mathcal{L}(w_i)$. In particular, with no relinearization the length of s_i is equal to 2. Consider the relinearize problem on the circuit G^0 and let l_i be the new lengths of s_i. Without loss of generality, we assume that $w_i \leq W$ for all i (if $w_i > W$, then any optimal solution of the knapsack problem always have $x_i = 0$, and we can reduce the dimension of the problem by one).

Lemma 2. *Suppose*

$$k_r > T \log T + W' \log W',$$

and $k_m = 1$*. Then for any optimal solution to the relinearize problem on* G^0*, the only vertices that could have nonzero relinearization are the* s_i*.*

Proof. We proved that the total cost of evaluating $\mathcal{L}(T)$ is bounded by $T \log T$, hence relinearizing any single vertex in this circuit has benefit bounded by $T \log T$. The situation is similar for $\mathcal{L}(W')$. Note that relinearizing verteices in $\mathcal{L}(W')$ could reduce the length of vertices in $\mathcal{L}(T)$, but the total cost is still bounded above by $T \log T + W' \log W'$. For the same reason, the benefit of relinearizing any vertex in any of the K copies of $\mathcal{L}(w_i)$ is bounded by $T \log T + w_i \log w_i$. Since $w_i \leq W'$, this completes the proof. □

Lemma 3. *Suppose* $KT > k_r$ *and* $k_m = 1$*. Then for any optimal solution to the relinearization problem on* G^0 *we must have*

$$\sum l_i w_i \leq W' := W + \sum_{i=1}^n w_i.$$

Here again we recall that $l_i \in \{1, 2\}$ *denote the length of* s_i *in an optimal solution.*

Proof. We prove by contradiction. Suppose the claim is false. Then there exists i such that $l_i = 2$. We relinearize the vertex s_i, which reduces the length of the final output in each copy of $\mathcal{L}(w_i)$ by w_i, and the final result of their sum by w_i. Since the claim is false, we have $\sum l_i w_i > W'$. So the length of the input vertex in each $\mathcal{L}(T)$ is reduced by at least 1, and the reduction in cost to each $\mathcal{L}(T)$ is at least T. Hence we the benefit we collect from changing l_i from 2 to 1 is at least KT, whereas the cost is k_r. Since we assumed $KT > k_r$, we know relinearizing the vertex s_i reduces the total cost. This is a contradiction. □

Proof. (of Theorem 1) We start by taking $T = \lceil W' \log W' \rceil$, $k_r = \lceil 2.5 T \log T \rceil$, $k_m = 1$ and $K = \lceil 3 \log T \rceil$. Note that K, T, k_r are of size polynomial in W, w_i, v_i. By lemma **??**, we have obtained the correct constraint in Theorem 1. However, the costs are wrong: the total cost of evaluating the circuit G^0 is given by

$$K(\sum r_i l_i) + const1 + const2 + \sum_i k_r (2 - l_i).$$

where as we proved in a previous lemma, $r_i = O(w_i \log w_i)$. Const1 is the cost of evaluating the $\mathcal{L}(T)$ circuit (it is a constant because in the optimal solution we always have the input sizes equal to W'), and const2 is the cost of evaluating all the $\mathcal{L}(W')$ circuits. It is a constant by Lemma **??**. Note that the coefficient before l_i is equal to $Kr_i - k_r$, and we want to modify this coefficient to $-v_i$. We do this in two steps:

First, note that $Kr_i - k_r < 0$ by assumption. By scaling up k_r and K by a sufficiently large factor, we can assume $Kr_i - k_r \le v_i$ for all i.

Next let $\lambda_i = k_r - Kr_i - v_i \in \mathbb{Z}_{\ge 0}$. We claim that there exists a circuit $\mathcal{L}'(\lambda_i)$ of such that relinearizing its second vertex has reduces the total multiplication cost by λ_i. We omit the details of construction of $\mathcal{L}'$ since it is similar to that of $\mathcal{L}$. We then let

$$G^1 = G^0 \star_{s_1} \mathcal{L}(\lambda_1), \ldots, G^n = G^{n-1} \star_{s_n} \mathcal{L}(\lambda_n)$$

and set $G = G^n$. Since $\lambda_i < k_r$, we know that in any optimal solution of the relinearize problem on G, the vertices on $\mathcal{L}(\lambda_i)$ have zero relinearization. Then, the relinearize problem on G is equivalent to

$$\min \sum -v_i l_i + const, \text{ s.t. } l_i \in \{1, 2\}, \sum w_i l_i \le W + \sum w_i,$$

which is equivalent to

$$\max \sum v_i l_i, \text{ s.t. } l_i \in \{1, 2\}, \sum w_i l_i \le W + \sum w_i.$$

This proves (3). (1) is clear since the number of vertices in G is $O(\log(T) + \log(W') + K \sum_{i=1}^{n} \log(w_i) + \log(\lambda_i)$. Hence it is logarithm in the parameters v_i, w_i, W and linear in the number of variables n. For (2), note that we set $k_m = 1$, so it suffices to show for k_r. By construction, k_r is also bounded by a polynomial in v_i, w_i, W. This completes the proof. □

Corollary 1. *The relinearize problem is NP-hard.*

4 An Simple Case

Assume we are in the situation where each non-input vertex in the circuit has two inputs and at most one output. In this case, we have a polynomial time algorithm for the relinearize problem.

For a vertex i, define $M(i, \ell)$ to be the minimal cost to compute the circuit up to vertex i, so that the (new?) length of i is ℓ.

Recall that $p_1(i)$ and $p_2(i)$ denote the parents of i. If i is a multiplicative vertex, we have

$$M(i, \ell) = \min_{\ell_1, \ell_2}\{M(p_1(i), \ell_1) + M(p_2(i), \ell_2) + k_r(\ell_1 + \ell_2 - \ell) + k_m(\ell_1 + \ell_2)\}.$$

If i is an addition vertex, we have

$$M(i, \ell) = \min_{\ell_1, \ell_2}\{M(p_1(i), \ell_1) + M(p_2(i), \ell_2) + k_r(\max\{\ell_1, \ell_2\} - \ell)\}.$$

Claim 1. *Suppose $N = |V| \geq 2$. Then in the above formulae, it suffices to take the minimum over range $2 \leq \ell_1, \ell_2 \leq N$.*

Proof. For the input vertices, the lengths is at most 2. For any non-input vertex v, we prove inductively that its length cannot exceed its number of ancestors. The length is at most $l(p_1(v)) + l(p_2(v)) - 1$, and by inductive hypothesis, both $l(p_1(v))$ and $l(p_1(v))$ are at most their number of ancestors (or plus one if it happens to be an input vertex). That is, $l(p_1(v)) + l(p_2(v)) - 1 \leq n_1 + n_2 + 1 = n$. Here n_1, n_2, n denote the number of ancestors for $p_1(v), p_2(v), v$, respectively. $\square$

Now our algorithm proceeds as follows. We traverse the N vertices. At each vertex, we compute $M(i, l)$ for $O(N)$ values of l, and each computation requires $O(N^2)$ operations. Thus the total running time is $O(N^4)$. Finally, the optimal cost is given by $\min_{2 \leq l \leq N} M(v, l)$, where v is the sink of the graph G.

References

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

[BLMZ17] Fabrice Benhamouda, Tancrède Lepoint, Claire Mathieu, and Hang Zhou. Optimization of bootstrapping in circuits. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2423–2433. SIAM, 2017.

[CLP] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-SEAL v2.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[LP13] Tancrède Lepoint and Pascal Paillier. On the minimal number of bootstrappings in homomorphic circuits. In *International Conference on Financial Cryptography and Data Security*, pages 189–200. Springer, 2013.