Information-Theoretic Homomorphic Encryption and 2-Party Computation

Jonathan Trostle

Independent Consultant

Abstract. Homomorphic encryption has been an active area of research since Gentry's breakthrough results on fully homomorphic encryption. We present secret key somewhat homomorphic schemes where client privacy is information-theoretic (server can be computationally unbounded). As the group order in our schemes gets larger, entropy approaches maximal entropy (perfect security). Our basic scheme is additive somewhat homomorphic. In one scheme, the server handles circuit multiplication gates by returning the mulitiplicands to the client which does the multiplication and sends back the encrypted product. We give a 2-party protocol that also incorporates server inputs where the client privacy is information-theoretic. Server privacy is not information-theoretic, but rather depends on hardness of the subset sum problem. Correctness for the server in the malicious model can be verified by a 3rd party where the client and server privacy are information-theoretically protected from the verifier. Scaling the 2PC protocol via separate encryption parameters for smaller subcircuits allows the ciphertext size to grow logarithmically as circuit size grows.

Keywords: Somewhat homomorphic encryption · information-theoretic.

1 Introduction

Homomorphic encryption schemes with respect a single operation (addition or multiplication) include Goldwasser-Micali, Paillier [22],and textbook RSA [24]. Rivest posed the question of homomorphic encryption [23] in 1978. Gentry's breakthrough work [15] presented a fully homomorphic scheme and accelerated the study of homomorphic schemes that can compute arbitrary functions in a model with circuits that have both addition and multiplication gates.

Gentry's scheme along with follow-up work [13], [6], [7], [10], [9], [2], [14] features schemes where security depends on computationally hard problems in lattices or number theory (e.g. lattice SVP, approximate GCD problem). The security of these schemes may be affected by advances in algorithms to more efficiently solve these problems.

In this work, we present somewhat homomorphic schemes (any circuit can be handled by setting scheme parameters to be large enough for the circuit). Our schemes are secret key based and provide security against a computationally unbounded attacker.

Our basic scheme includes a modulus m, base b where $gcd(b, m) = 1$, and random exponents e_i corresponding to each client ciphertext input for the circuit. The ciphertext tuple consists of the vector $(be_1 \bmod m, \ldots, be_c \bmod m)$. For encrypting bits, the parity of each e_i determines the plaintext bit. The size of the $e_i's$ is bounded using either the max norm or L_1 norm so that the client can decrypt the returned result by multiiplying by $b^{-1} \bmod m$. This scheme has two interesting properties:

1. it's additive homomorphic
2. as illustrated in Figure 1, it allows distinct (plaintext key) pairs to map to the same ciphertext. (The key is the pair (b, m).)

The 2nd property gives the information-theoretic security.

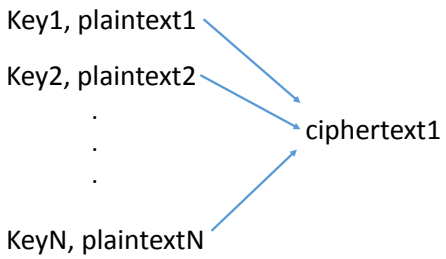

Fig. 1. Distinct Plaintext Key Pairs Map to Same Ciphertext

Supporting multiplication gates is more challenging. If we shrink the exponents further to support multiplication then security is no longer information-theoretic since there are not a sufficient number of vectors.

In our first multiplication scheme, the server returns the multiplicands to the client which decrypts and performs the multiplication. The client sends a new ciphertext element back to the server encrypting the product of the plaintexts. All additions and multiplications are mod 2. (In the general case we replace 2 with N which is power of 2 so addition and multiplication are mod N.)

A second scheme with multiplication includes additional client elements with both odd and even parities. The server computes 2^g results where g is number of multiplication gates. The server assigns one of the additional client elements as the output of each multiplication gate. Thus this scheme can only be practical for a circuit with a small number of multiplication gates.

We also give a 2-party MPC (2PC) scheme where client privacy is information-theoretic (also called statistical or unconditional security) and server privacy is based on the hardness of the subset sum problem. Initially, in a preprocessing

step, the client sends two additional sets of elements to the server; one set has odd parity exponent elements and the other set has even parity exponent elements. When the server has two multiplicands to return to the client, it actually returns four elements where the additional two elements have opposite parity exponents which are obtained by adding odd parity elements to the original multiplicands. Then the client will compute all four products and return four new ciphertext elements to the server. The server discards the three incorrect elements and keeps the fourth element. Thus the server evaluates the circuit obliviously except for knowledge of addition gates, multiplication gates, and its own inputs. Our security argument requires that the client is semi-honest.

There exist 2-party schemes [20], [3] where one party enjoys statistical privacy and the other party is protected from a computationally bounded adversary. To the best of our knowledge, our 2-party scheme is the first one to allow the party receiving output to have statistical privacy (from a computationally unbounded sender) where the sending party has computationally bounded security in a protocol where only one party receives output. The schemes in [20] are in the malicious security model where our 2PC protocol client is assumed to be semi-honest. Receiver privacy is protected unconditionally including if the sender is malicious and sender correctness can be established by a verifier. The client and server privacy is protected unconditionally from the verifier, provided that the verifier does not have access to the client secret key, client plaintext inputs, or server plaintext inputs.

1.1 An Example

Consider an example with parameters $m = 31$, $b = 17$, and $c = 2$. We set $N = 2$ (bit encryption) and $e_1 = 12$, $e_2 = 15$.

The client sends the pair $be_1 \bmod m, be_2 \bmod m$ to the server. This pair encrypts the client input pair $(0, 1)$. This bit input pair is obtained from the parity of e_1 and e_2.

$be_1 \bmod m = 18, be_2 \bmod m = 7$.

Table 1 shows the full set of multiples for $m = 31, c = 2, e_1 = 12, e_2 = 15$.

Given the pair $(18, 7)$, the server cannot deduce the client input pair. For example, if we take $d = 16$, $f_1 = 5$, $f_2 = 14$, then $df_1 = 18$, $df_2 = 7$. Let $d = 7, f_1 = 7, f_2 = 1$. Again $df_1 = 18$, $df_2 = 7$. Thus the pair $(18, 7)$ is consistent with multiple different input exponent pairs; in this case both $(1, 0)$ and $(1, 1)$.

For this example, these exponent pairs all satisfy the relation that the sum of the exponents is less than m. Thus these inputs to the server can be used to evaluate a circuit with one addition gate. The client can decrypt the returned value from the server by multiplying by $b^{-1} \bmod m$ and computing the parity of the result.

Table 2 shows the various possible exponent pairs that are consistent with each bit input pair where the sum of the exponents is less than m. Thus there is a small amount of leakage since the different input pairs have different frequencies given the ciphertext pair. The entropy for this example is $-\sum p_i \log(p_i) \approx 1.89$ which is 0.11 bits less than perfect security (2 bits). We will show in the next

Table 1. Multiples Table for $m = 31$, $c = 2$, $e_1 = 12$, $e_2 = 15$.

Multiple	*Pair*	*Multiple*	*Pair*
1	(12, 15)	16	(6, 23)
2	(24, 30)	17	(18, 7)
3	(5, 14)	18	(30, 22)
4	(17, 29)	19	(11, 6)
5	(29. 13)	20	(23, 21)
6	(10, 28)	21	(4, 5)
7	(22, 12)	22	(16, 20)
8	(3, 27)	23	(28, 4)
9	(15, 11)	24	(9, 19)
10	(27, 26)	25	(21, 3)
11	(8, 10)	26	(2, 18)
12	(20, 25)	27	(14,2)
13	(1, 9)	28	(26, 17)
14	(13, 24)	29	(7, 1)
15	(25, 28)	30	(19, 16)

section the entropy goes to $c \log(N)$ where $\log(N)$ is the number of plaintext bits for each client input element as $m \to \infty$.

Table 2. Client Exponent Pairs Consistent with Server Received Pair $(18, 7)$ for $m = 31$, $c = 2$.

Bit input pair (0.1)	*Pair* $(1,0)$	*Pair* $(1,1)$	*Pair* $(0,0)$
(12, 15)	(5, 14)	(3, 27)	(8, 10)
(6, 23)	(11, 6))	(15, 11)	(2, 18)
(18, 7)		(1, 9)	(14, 2)
(4, 5)		(9, 19)	
		(21, 3)	
		(7, 1)	

1.2 Our Contributions

We have the following results:

1. We give an additive homomorphic encryption algorithm that provides statistical client privacy with small leakage, and extend this to include multiplication both via a non-interactive protocol and also a more scaleable protocol that relies on client assistance for multiplication operations. The resulting protocols are secret key somewhat homomorphic that protect the client privacy from a computationally unbounded server.

2. We quantify the client's privacy leakage in Theorem 2 and show that leakage goes to 0 as we increase m (the modulus and ciphertext size).
3. Based on the somewhat homomorphic protocol, we construct a 2-party protocol (2PC) where both the client and server provide inputs and the server processes the encrypted inputs through the circuit returning the output to the client. We show that this protocol protects the client privacy, with minimal leakage via a similar bound as in Theorem 2, from the unbounded server. The server is in the malicious model.
4. We show the 2PC protocol protects server privacy assuming hardness of the subset sum assumption given a semi-honest client.
5. The 2PC protocol malicious server's correctness can be verified in the protocol by a verifier entity with access to the protocol transcript, the server computations, and the circuit specification. The client and server privacy is statistically secure from the verifier provided the verifier does not have access to the client secret key, client plaintext inputs, or server plaintext inputs.
6. The 2PC protocol can be scaled by dividing the circuit into smaller subcircuits each with separate encryption parameters. This scaling allows the ciphertext size to grow logarithmically as circuit size grows.

1.3 Related Work

There has been extensive work in homomorphic encryption since the breakthrough work of Gentry [15] (e.g., [16], [8], [5], [17], [13], [6], [7], [10], [9], [2], [14]. These schemes require less rounds than our main scheme with requires interaction with the client for every multiplication gate. Our schemes provide statistical privacy for the client.

Foundational work in multiparty computation includes [25], [18], [4], [12].

It is not possible to protect both parties with statistical security in a 2-party secure computation protocol.

Dakshita and Mughees [20] give 2-party secure protocols where one party is statistically secure and the other party is computationally secure. Their protocols use garbled circuits [25]. Thus our protocol may be more communication bandwidth efficient if the number of multiplication gates is not too large relative to the size of the circuit. Their protocols are secure against a malicious adversary whereas we assume our client, or receiving party, is in the semi honest model (but our receiving party's privacy is protected even if the sender is malicious).

[3] allows the results of [20] to be based on additional assumptions such as CDH. [1] generalizes [20] from 2 parties to n parties; one party can be protected with statistical security and their fallback security provides computational security for the other parties.

Koleskinov [21] gives a 2-party secure protocol where the secrets are assigned to wires. It has information theoretic security if the underlying oblivious transfer (OT) protocol does. OT protocols exist that provide information theoretic security for either the sender or the receiver, but not both [11].

1.4 Terminology and Background

A vector $\mathbf{x}$ of elements in $\mathbb{Z}_m$ (the integers modulo m) will be denoted in boldface. Elements in $\mathbb{Z}_m$ will be taken to be integers between 0 and $m-1$, inclusive.

We will leverage the subset sum problem: given integers $\mathbf{a} = (a_1 \ldots, a_n) \in \mathbb{Z}^n$ and a uniform random vector $\mathbf{s}$ where $s_i \in \{0,1\}$. The adversary is given $(T = \mathbf{a} \cdot \mathbf{s}, \mathbf{a})$ and must find $\mathbf{s}$, where "$\cdot$" denotes the inner product. The subset sum problem is considered hard for a computationally bounded adversary. The best known algorithms are exponential in the smaller of n and T.

For circuits, we let the parameter a be the number of addition gates and g is the number of multiplication gates. For boolean circuits, the multiplication gates could be either AND or NAND gates, and the addition gates are XOR gates.

The notation $s \leftarrow S$ is used when s is randomly selected from S via the uniform distribution.

We follow [19] for our definition of simulation privacy in the semi-honest model, assuming a deterministic functionality f. Our 2PC protocol has server privacy if there is a probabilistic polynomial-time algorithm $\mathcal{S}$ such that

$$\{\mathcal{S}(x, f_1(x,y))\}_{x,y\in\{0,1\}^*} \equiv^c \{VIEW_c^{\Pi}(x,y)\}_{x,y\in\{0,1\}^*}$$

where $VIEW_c$ is the client's view of our protocol Π's execution including it's input, randomness and messages received, $f_1(x,y)$ is the client output, and x, y are the client and server inputs respectively.

2 Somewhat Homomorphic Encryption Schemes

We first present an additive somewhat homomorphic scheme and then schemes with both addition and multiplication. Our schemes are secret key rather than public key.

2.1 Additive Homomorphic Scheme

Definition 1. *Given a server circuit with a addition gates. m, b, e_i, and N are integers where N is a power of 2. A client input instance is*

$$I = (m, b, N, e_1, \ldots, e_c),$$

where $e_i < \frac{m}{a+1}$, $1 \le i \le c$. We assume $\gcd(b, m) = 1$. *More precisely, the e_i satisfy the following:*

1. *the e_i are distinct*
2. *for every pair (e_i, e_j) where $i \neq j$, then if $xe_i = (m-1)/2$ (in $\mathbb{Z}_m$) then $xe_j \neq 1$.*
3. *for all i,, $e_i = a_i N + r_i$ where a_i is random uniform and r_i is the client's plaintext input. $r_i < N$.*

The client ciphertext is obtained by selecting an integer representative v_i, $1 \leq i \leq c$, where $v_i \equiv be_i \bmod m$. Then $Q(I) = (v_1, \ldots, v_c)$ is the client ciphertext request corresponding to the instance.

The client sends the client ciphertext request to the server which uses the input elements as inputs to the circuit. The server returns the output elements to the client.

2.2 Proof of Security

We will make use of the max norm: $\|\mathbf{x}\|_\infty = \max_{\mathbf{i}}\{|\mathbf{x_i}|\}$.

In the following theorem, we assume m is prime, but the theorem can be generalized to composite m.

Theorem 1. *Given a client ciphertext request $Q(I) = (v_1, \ldots, v_c)$ with N, m, and b as defined above, where $e_i \leq m/(a+1)$, $1 \leq i \leq c$. Let m be prime. Suppose $r_1, \ldots, r_c$ are the client's plaintext values; in other words, $e_i \bmod N = r_i, 1 \leq i \leq c$. Given $\boldsymbol{s} = (s_1, \ldots, s_c)$ as plaintext values corresponding to an arbitrary client ciphertext request.*

1. *Let $\mu = 1/(a+1)^c$. Let T_{m-1} be the number of vectors in the cyclic submodule L of $\mathbb{Z}_m^c$ generated by $(e_1, \ldots e_c)$ that have max norm bounded by $m/(a+1)$. Let k be a positive integer. With probability*

$$P \geq 1 - \frac{k^2(a+1)^c}{m-1}$$

we have

$$\left| \frac{T_{m-1}}{m-1} - \mu \right| \leq \epsilon$$

where $\epsilon = 1/(k(a+1)^c)$. The T_{m-1} vectors correspond to other client ciphertext request instances $\bar{I}$ such that $Q(I) = Q(\bar{I})$.

2. *Let $\mu_2 = 1/((a+1)N)^c$. Let $S_{m-1}^{\boldsymbol{s}}$ be the number of vectors x in the cyclic submodule L of $\mathbb{Z}_m^c$ generated by $(e_1, \ldots e_c)$ that have max norm bounded by $m/(a+1)$ and satisfy $x_i \bmod N = s_i$, $1 \leq i \leq c$. Let k be a positive integer. With probability*

$$P_2 \geq 1 - \frac{k^2((a+1)N)^c}{m-1}$$

we have

$$\left| \frac{S_{m-1}^{\boldsymbol{s}}}{m-1} - \mu_2 \right| \leq \epsilon_2$$

where $\epsilon_2 = 1/(k(N(a+1))^c)$. We call two client ciphertext requests compatible if they satisfy $e_i \bmod N = f_i \bmod N, 1 \leq i \leq c$, where the corresponding client input instances are $I = (m, b, N, e_1, \ldots, e_c)$, and $J = (m, d, N, f_1, \ldots, f_c)$.

We will make use of the following corollary to Chebyshev's inequality:

Lemma 1. *Let X_i be independent, identically distributed random variables with mean μ and variance σ^2. Let $S_n = X_1 + \ldots + X_n$. For any $\epsilon > 0$, we have*

$$P\left(\left|\frac{S_n}{n} - \mu\right| \geq \epsilon\right) \leq \frac{\sigma^2}{n\epsilon^2}.$$

We now give the proof of the theorem:

Proof. (1) The vector $\boldsymbol{e} = (e_1, \ldots, e_c)$ generates a cyclic submodule L of $\mathbb{Z}_m^c$ over the ring $\mathbb{Z}_m$. We may view te_i as t varies over $\mathbb{Z}_m^*$ as a random permutation of $\mathbb{Z}_m^*$. Thus $t\boldsymbol{e}$ is a random permutation over the cross product of the individual coordinate subsets. Thus we may assume that

$$Pr[\|x\|_\infty < m/(a+1) \text{ given } x \in L] = Pr[\|x\|_\infty < m/(a+1)]$$

Also,

$$Pr[\|x\|_\infty < m/(a+1)] = \frac{1}{(a+1)^c}$$

Let $\epsilon = 1/(k(a+1)^c)$. By the lemma above, we have

$$Pr[|T_{m-1}/(m-1) - \mu| \geq \epsilon] < \frac{\sigma^2}{(m-1)\epsilon^2}$$

where $X_i = 1$ if the ith vector in L satisfies

$$\|x\|_\infty < m/(a+1),$$

$T_i = X_1 + X_2 + \ldots + X_i$, with μ as the mean of X_i, and σ^2 is the variance. $\sigma^2 = \mu(1-\mu) < \mu$ since the X_i are Bernoulli random variables.

$\mu = Pr[X_1 = 1] = 1/(a+1)^c$. Thus

$$\frac{\sigma^2}{(m-1)\epsilon^2} < \frac{\mu}{(m-1)\epsilon^2} = \frac{k^2(a+1)^c}{m-1}$$

Thus with probability P where

$$P \geq 1 - \frac{k^2(a+1)^c}{m-1}$$

we have

$$\left|\frac{T_{m-1}}{m-1} - \mu\right| \leq \epsilon$$

It remains to show that the T_{m-1} short vectors in L correspond to other client ciphertext request instances. For each such vector $\mathbf{s}$, we have $t_s e_i \equiv s_i \bmod m$, $1 \leq i \leq c$, for some t_s where $1 \leq t_s \leq m-1$. Let $f \equiv b(t_s)^{-1} \bmod (m)$. Then $fs_i \equiv be_i \bmod (m)$, $1 \leq i \leq c$. Thus $f, s_1, \ldots, s_c$ are part of a client ciphertext request instance $\bar{I}$, such that $Q(I) = Q(\bar{I})$.

(2) Now we prove claim (2) above. Let $Y_i = 1$, if the ith vector in L is short and compatible with $\mathbf{s} = (s_1, \ldots, s_c)$, and let $Y_i = 0$ otherwise. Let $S_i^{\mathbf{S}} = Y_1 + Y_2 + \ldots + Y_i$. Let

$$\epsilon_2 = \frac{1}{k((a+1)N)^c}.$$

From the lemma, we have

$$Pr[|S_{m-1}^{\mathbf{S}}/(m-1) - \mu_2| \geq \epsilon_2] < \frac{\sigma_2^2}{(m-1)\epsilon_2^2}$$

where $\mu_2 = Pr[Y_1 = 1] = 1/((a+1)N)^c)$. Thus

$$\frac{\sigma_2^2}{(m-1)\epsilon_2^2} \leq \frac{k^2((a+1)N)^c}{m-1}$$

Thus with probability P_2 where

$$P_2 \geq 1 - \frac{k^2((a+1)N)^c}{m-1}$$

we have

$$\left| \frac{S_{m-1}^{\mathbf{S}}}{m-1} - \mu_2 \right| \leq \epsilon_2$$

∎

We now show that entropy associated with a client ciphertext request approaches perfect security as m grows.

Theorem 2. *We order the set of plaintext vectors $\pi = (s_1, \ldots, s_c)$ where $s_i < N, 1 \leq i \leq c$. We denote this ordered set as $\{\pi_1, \ldots, \pi_{N^c}\}$ where $\pi_i = (s_1^i, \ldots, s_c^i)$. Given the module L over $\mathbb{Z}_m$ generated by $(e_1, \ldots, e_c)$ associated with client input instance $I = (m, b, N, e_1, \ldots, e_c)$. Let p_i be the probability that π_i is encoded in a vector x in L where $\|x\|_\infty < m/(a+1)$; thus $p_i = S_{m-1}^{\pi_i}/T_{m-1}$. ($T_{m-1}$ and $S_{m-1}^{\pi_i}$ are defined above in Theorem 1.) Then for an integer $k > 0$, we have*

$$\frac{k-1}{k+1} \log \frac{(k-1)N^c}{k+1} \leq \sum_i p_i \log \frac{1}{p_i} \leq \frac{k+1}{k-1} \log \frac{(k+1)N^c}{(k-1)}$$

with probability

$$P \geq \left(1 - \frac{k^2((a+1)N)^c}{m-1}\right)^{N^c+1}$$

Proof. By Theorem 1, we have

$$\left| \frac{S_{m-1}^{\pi_i}}{m-1} - \mu_2 \right| \leq \epsilon_2$$

with probability

$$P_i \geq 1 - \frac{k^2((a+1)N)^c}{m-1}$$

where $\epsilon_2 = 1/k(a+1)^c N^c$, $\mu_2 = 1/((a+1)N)^c$. Similarly, we have

$$\left| \frac{T_{m-1}}{m-1} - \mu \right| \leq \epsilon$$

with probability

$$\overline{P} \geq 1 - \frac{k^2(a+1)^c}{m-1} \geq 1 - \frac{k^2((a+1)N)^c}{m-1},$$

where $\epsilon = 1/k(a+1)^c$, $\mu = 1/(a+1)^c$. We have

$$\left(1 - \frac{k^2((a+1)N)^c}{m-1}\right)^{N^c+1} \approx e^{-k^2((a+1)N)^c(N^c+1)/(m-1)}$$

since we may select m so that $k^2((a+1)N)^c/(m-1)$ is very small ($e^{-x} \approx 1-x$ for very small positive x) and due to independence.

Thus with probability P we have

$$\left| \frac{T_{m-1}}{m-1} - \mu \right| \leq \epsilon$$

and

$$\left| \frac{S^{\pi_i}_{m-1}}{m-1} - \mu_2 \right| \leq \epsilon_2$$

for all π_i.

We have $\mu_2 + \epsilon_2 = (k+1)/(k((a+1)N)^c)$, $\mu_2 - \epsilon_2 = (k-1)/(k((a+1)N)^c)$, $\mu+\epsilon = (k+1)/(k(a+1)^c)$, and $\mu-\epsilon = (k-1)/(k(a+1)^c)$. Thus with probability P we have

$$\frac{S^{\pi_i}_{m-1}}{T_{m-1}} \leq \frac{k+1}{(k-1)N^c}$$

and

$$\frac{S^{\pi_i}_{m-1}}{T_{m-1}} \geq \frac{k-1}{(k+1)N^c}$$

Thus

$$\begin{aligned}\sum_i p_i \log \frac{1}{p_i} &= \sum_i \frac{S^{\pi_i}_{m-1}}{T_{m-1}} \log \frac{T_{m-1}}{S^{\pi_i}_{m-1}} \leq \sum_i \frac{k+1}{(k-1)N^c} \log \frac{(k+1)N^c}{k-1} \\ &= \frac{k+1}{k-1} \log \frac{(k+1)N^c}{k-1}\end{aligned}$$

and

$$\begin{aligned}\sum_i p_i \log \frac{1}{p_i} &= \sum_i \frac{S^{\pi_i}_{m-1}}{T_{m-1}} \log \frac{T_{m-1}}{S^{\pi_i}_{m-1}} \geq \sum_i \frac{k-1}{(k+1)N^c} \log \frac{(k-1)N^c}{k+1} \\ &= \frac{k-1}{k+1} \log \frac{(k-1)N^c}{k+1}\end{aligned}$$

2.3 L_1 norm

The max norm in our theorems above corresponds to a circuit that has $(a+1)\max e_i < m$ where a is the number of addition gates. Alternatively, if we let a_2 be the maximum number of times any element is added to itself in the circuit, we have $(a_2+1)\sum e_i < m$ and $\sum e_i < m/(a_2+1)$ which gives rise to the L_1 norm.

We now give a basic combinatorial lemma which is needed to establish the analogs of the above theorems for the L_1 norm.

Lemma 2.

$$S = |\{\boldsymbol{x} : \|\boldsymbol{x}\|_1 \leq B \text{ where } \boldsymbol{x} \in \mathbb{Z}_m^c;\ 1 \leq x_i \leq B-1, 1 \leq i \leq c\}| = \binom{B}{c}$$

Proof. If $c = 2$, we have $1+2+\ldots+B-1 = (B-1)B/2 = \binom{B}{2}$. Using induction, we have

$$\binom{B-1}{c-1} + \binom{B-2}{c-1} + \ldots + \binom{c-1}{c-1} = |S|$$

We show the left side of this equality is equal to $\binom{B}{c}$. Choose any element x in a set of B elements. $\binom{B-1}{c-1}$ is the number of subsets with x in them. Now remove x. Select y in the set. $\binom{B-2}{c-1}$ is the number of subsets of size c with y (but not x.) Remove y now. Iterate this step until only c elements are left. This is the last subset with c elements. ∎

We omit the proofs of the following two theorems since they follow the proofs of Theorem 1 and Theorem 2 above while leveraging Lemma 2 and Stirling's formula.

Theorem 3. *Given a client ciphertext request $Q(I) = (v_1, \ldots, v_c)$ with N, m, and b as defined above. Suppose $r_1, \ldots, r_c$ are the client's plaintext values; thus $e_i \bmod N = r_i, 1 \leq i \leq c$. Given $s = (s_1, \ldots, s_c)$ as plaintext values corresponding to an arbitrary client ciphertext request. λ is a constant between $1/2$ and 2; a_2 bounds the number of times any element is added to itself in the circuit.*

1. *Let $\mu = \lambda/c!(a_2+1)^c$. Let T_{m-1} be the number of vectors in the cyclic submodule L of $\mathbb{Z}_m^c$ generated by $(e_1, \ldots e_c)$ that have L_1 norm bounded by $m/(a_2+1)$. Let k be a positive integer. With probability*

$$P \geq 1 - \frac{k^2 c!(a_2+1)^c}{(m-1)\lambda}$$

we have

$$\left| \frac{T_{m-1}}{m-1} - \mu \right| \leq \epsilon$$

where $\epsilon = \lambda/(kc!(a_2+1)^c)$.

2. *Let* $\mu_2 = \lambda/(c!((a_2+1)N)^c)$. *Let* S^s_{m-1} *be the number of vectors* x *in the cyclic submodule* L *of* $\mathbb{Z}^c_m$ *generated by* $(e_1, \ldots e_c)$ *that have* L_1 *norm bounded by* $m/(a_2+1)$ *and satisfy* $x_i \bmod N = s_i$, $1 \leq i \leq c$, *Let* k *be a positive integer. With probability*

$$P \geq 1 - \frac{k^2 c!((a_2+1)N)^c}{\lambda(m-1)}$$

we have

$$\left| \frac{S^s_{m-1}}{m-1} - \mu_2 \right| \leq \epsilon_2$$

where $\epsilon_2 = \lambda/(kc!(N(a_2+1))^c)$. *We call two client ciphertext requests compatible if they satisfy* $e_i \bmod N = f_i \bmod N, 1 \leq i \leq c$, *where the corresponding client input instances are* $I = (m, b, N, e_1, \ldots, e_c)$, *and* $J = (m, d, N, f_1, \ldots, f_c)$.

Theorem 4. *We order the set of plaintext vectors* $\pi = (s_1, \ldots, s_c)$ *where* $s_i < N, 1 \leq i \leq c$. *We denote this ordered set as* $\{\pi_1, \ldots, \pi_{N^c}\}$ *where* $\pi_i = (s^i_1, \ldots, s^i_c)$. *Given the module* L *over* $\mathbb{Z}_m$ *generated by* $(e_1, \ldots, e_c)$ *associated with client input instance* $I = (m, b, N, e_1, \ldots, e_c)$. *Let* p_i *be the probability that* π_i *is encoded in a vector* x *in* L *where* $\|x\|_1 < m/(a_2+1)$; *thus* $p_i = S^{\pi_i}_{m-1}/T_{m-1}$. *(*$T_{m-1}$ *and* $S^{\pi_i}_{m-1}$ *are defined above in Theorem 3.) Then for an integer* $k > 0$, *we have*

$$\frac{k-1}{k+1} \log \frac{(k-1)N^c}{k+1} \leq \sum_i p_i \log \frac{1}{p_i} \leq \frac{k+1}{k-1} \log \frac{(k+1)N^c}{(k-1)}$$

with probability

$$P_2 \geq \left(1 - \frac{k^2 c!((a_2+1)N)^c}{\lambda(m-1)}\right)^{N^c+1}$$

2.4 Incorporating Multiplication

The multiplication of $be_1 \bmod m$ and $be_2 \bmod m$ where $e_i \bmod N = r_i$, $i = 1, 2$ is defined to be $be \bmod m$ for some e where $e < m/(a+1)$ and $e \bmod N = r_1 r_2 \bmod N$. [1]

Non-Interactive Scheme Our first scheme requires the client to send N elements $be_1 \bmod m, \ldots, be_N \bmod m$ for each multiplication gate where $e_i mod N = r_i$ and $r_1, \ldots, r_N$ are the N possible client plaintexts. The server computes all N^g possible results where g is the number of multiplication gates. The server selects one possible multiplication result, $be_i \bmod m$, for each multiplication gate. For each of the N^g possible results, the server returns the 2 multiplicand elements and the selected multiplication result for each gate, plus the output. Thus the

[1] For $N = 2$, (a boolean circuit), a multiplication gate is an AND gate. Alternatively, we could replace the AND gate with a NAND gate in our descriptions below.

server returns $3g+1$ elements for each of the N^g selection tuples. Note that the server does not know which of the N^g results contains correct multiplications. Only one of the results has all correct multiplications.

The client, upon receiving the server results, performs the following steps:

1. The client loops through all N^g results. Each result is a tuple of g triples:

$$(x_1, y_1, z_1), \ldots, (x_g, y_g, z_g)$$

2. For each tuple, the client computes $r_1 = b^{-1}x_1 \bmod m \bmod N$ and $r_2 = b^{-1}y_1 \bmod m \bmod N$ and checks if $r_1 r_2 \bmod N = b^{-1}z_1 \bmod m \bmod N$. If not, then the tuple is discarded.
3. Only 1 tuple will not be discarded and the output is the output element from this tuple.

This scheme can only be practical for a circuit with a small number of multiplication gates since the work scales exponentially with the number of multiplication gates.

Interactive Scheme To preserve information-theoretic security, the client is assumed to know the number of multiplication gates in the circuit. Each multiplication gates adds 1 to the c parameter in the information-theoretic security analysis above.

The interactive scheme works as follows. Initially, the client sends c_1 elements $be_1 \bmod m, \ldots, be_{c_1} \bmod m$ to the server, one element for each of the client's inputs to the circuit. The server can process addition gates on its own. The server leverages help from the client to process multiplication gates:

1. When the server has two multiplicands for a multiplication gate, it returns both elements, $be_1 \bmod m$ and $be_2 \bmod m$ to the client.
2. The client decrypts by multiplying by $b^{-1} \bmod m$ to obtain $e_1 \bmod m$ and $e_2 \bmod m$. The client then computes $r_1 = e_1 \bmod m \bmod N$ and $r_2 = e_2 \bmod m \bmod N$.
3. The client computes $e = aN + r$ where a is random uniform and $r = r_1 r_2 \bmod N$; we take $r < N$.
4. The client sends $be \bmod m$ to the server.
5. The server lets $be \bmod m$ be the output of the multiplication gate.

The server returns the final output to the client.

3 Two-Party Computation (2PC)

In this section, we consider the case where the server also has inputs for the circuit. As in Section 2, the client provides its (encrypted) inputs to the server, the server processes the client and server inputs through the circuit, and it returns the output to the client. Optionally, the client may share the decrypted output with the server.

For multiplication, we assume the interactive scheme is being used. Client privacy is information-theoretic per the results in Section 2; client privacy holds even if the server is fully malicious and computationally unbounded, except possibly if the client shares the decrypted output with the server. (A malicious server could learn additional information about client input values if the client returns the decrypted output to the server since the server could have returned as output the sum of one of its encrypted input values with a client encrypted input value.) If the client receives notification from a verifier that verifies the correctness of server's processing after the output is delivered to the client (see Section 3 for details), then the client can deliver an output to the server and be assured that client privacy is not impacted by a malicious server.

Server privacy depends on the hardness of the subset sum problem and our assumption that the client is semi-honest. Correctness also assumes the client is semi-honest. A verifier can inspect the protocol transcript along with the specification of the circuit to verify that the server's actions are correct. If the secret key and parities of the server inputs are not shared with the verifier, then client and server privacy is statistically protected from the verifier.

Our security proof assumes $N = 2$.

Our protocol includes a preprocessing step that occurs prior to client and server inputs and subsequent processing. The client creates a small number of elements of the form $be \bmod m$ following the same algorithm as in creating client input elements. The client creates client input elements, server input elements, and noise elements. The client sends these elements to the server along with the parities of the exponents for the server input elements and the noise elements. The server input elements include an even and an odd parity exponent element for each server input. (Alternatively, the client creates the encrypted server inputs and the parties use an Oblivious Transfer protocol with sender statistical privacy to transfer 1 out of 2 strings to the server for each server input. The advantage is reducing $2c_s$ to c_s in the sum for the c parameter at the cost of some public key operations and another computational assumption.)

For processing a multiplication gate, the server computes random subset sums of the noise elements and adds them to each multiplicand. Four elements are sent to the client.[2] Each multiplicand is added to both an odd sum and an even subset sum prior to both of the resulting elements being sent to the client. We define an even (odd) sum as a sum where the sum of the exponents is even (odd). Each pair of elements is randomly ordered prior to being sent to the client. The client is able to obtain the least significant bit after multiplying by $b^{-1} \bmod m$ for all four of the elements and return four products. Three of the products will be discarded by the server and the client does not know which of the products is the correct one (see Figure 2).

[2] For optimization, only two elements have to be sent; one element from each pair is sent. The client knows the other values after decrypting these two elements.

Algorithm 1 Preprocessing Steps for 2-Party Protocol with Information-Theoretic Security for Client, Steps Occur Prior to Introduction of Data

2-Party Protocol with Information-Theoretic Privacy for Client: Preprocessing

1: The client selects the c parameter: $c = c_c + 2c_s + c_1 + 4g$ where g is the number of multiplication gates in the circuit, c_c is the number of client inputs, c_s is the number of server inputs, and c_1 is a small integer (e.g., $c_1 = 8$) for the number of base elements used to create noise elements. The client selects m such that m and c satisfy the Theorem 2 bound.

2: $b \leftarrow \mathbb{Z}_m$ where $gcd(b, m) = 1$. The secret key is (b, m).

3: The client creates the set C of c_1 elements (the noise generator elements) where these c_1 elements are part of the larger set of c elements. t is an integer parameter which will limit leakage of subset sums to the client which will be described in our privacy proof. (As above, $e = aN + r$ where a is random uniform and r is 0 or 1, $N = 2$.) Roughly half of the elements are even:
$e_i \leftarrow \mathbb{Z}_m$, where $2 \leq e_i \leq m/(t(a+1)), 1 \leq i \leq c_1$.

4: The client creates uniform random noise elements from the c_1 elements discarding any duplicates until the client has n new elements. n is approximately $\log(m^{1/2})$. The random noise elements have the form $a_1e_1 + a_2e_2$ where a_1 and a_2 are integers and e_1, e_2 are relatively prime integers in the set of c_1 elements. Let $h_1, \ldots, h_n$ be the set of noise element exponents. The client ensures that these values are positive and $\sum_i h_i < m/2$:
$i = 0$
while $i < n$ **do**
  $h_i \leftarrow a_{i1}e_1 + a_{i2}e_2$; $a_{i1}, a_{i2} \leftarrow Z$, $e_1, e_2 \in C$.
  if $h_i = h_j$ for some $j < i$ or $h_i > m/2n$ **then**
    discard h_i
  else
    store h_i, parity(h_i)
    i = i + 1
  endif
endwhile

5: The client sends the noise elements to the server along with information to identify the parity of the exponents for each element:
send $z_i = bh_i \bmod m$, parity(h_i) to server, $1 \leq i \leq n$.
server stores z_i, parity(h_i), $1 \leq i \leq n$.

Algorithm 2 Input Processing Steps for 2-Party Protocol with Information-Theoretic Security for Client

2-Party Protocol with Information-Theoretic Privacy for Client: Processing Inputs Through Circuit

1: The client creates $2c_s$ server input elements each of the form $be \bmod m$ where these $2c_s$ elements are part of the larger set of c elements:
for $i = 0$ to c_s **do**
$e_i \leftarrow Z_m$ where $2 < e_i < m/(t(a+1))$
if $parity(e_i) = 1$ **then**
$e_i = e_i - 1$
endif
$y_i = be_i \bmod m$
end for
for $i = c_s + 1$ to $2c_s$ **do**
$e_i \leftarrow Z_m$ where $2 < e_i < m/(t(a+1))$
if $parity(e_i) = 0$ **then**
$e_i = e_i + 1$
endif
$y_i = be_i \bmod m$
end for

2: Client randomizes the order of $y_i, 1 \leq i \leq c_s$, stores the y_i exponent parities separately, sends the y_i to the server, and sends a separate message with y_i exponent parities to the server. (The exponent parities will not be shared with a verifier.) Alternatively, oblivious transfer is used to send c_s elements to the server.

3: The client creates c_c input elements each of the form $be \bmod m$ where these c_c elements are part of the larger set of c elements:
for $i = 0$ to c_c **do**
$e_i \leftarrow Z_m$ where $2 < e_i < m/(t(a+1))$
if $input_i = 0$ **then**
if $parity(e_i) = 1$ **then**
$e_i = e_i - 1$
endif
endif
if $input_i = 1$ **then**
if $parity(e_i) = 0$ **then**
$e_i = e_i + 1$
endif
endif
$w_i = be_i \bmod m$
end for

4: Client sends w_i to server, $1 \leq i \leq c_c$.

5: The server can now begin processing through the circuit given the client and server input elements.

Algorithm 3 Steps for 2-Party Protocol with Information-Theoretic Security for Client: Multiplication, Addition Gates and Output

2-Party Protocol with Information-Theoretic Privacy for Client: Processing Multiplication, Addition, and Output

1: When the server needs to multiply be_1 and be_2, it creates four elements (in $\mathbb{Z}_m$) to send to the client: be_1 plus an even parity subset sum element (using the n h_i elements), be_1 plus an odd parity subset sum element, be_2 plus plus an even parity subset sum element, and be_2 plus an odd parity subset sum element. The 1st and 2nd elements are randomly ordered and the 3rd and 4th elements are randomly ordered:
   for $i = 0$ to 3 **do**
   $\mathbf{z} = (z_1, \ldots, z_n)$
   $\mathbf{s} \leftarrow \{0,1\}^n$
   $sum_i \leftarrow \mathbf{s} \cdot \mathbf{z} \bmod m$
   define parity$(sum_i) = \sum_{s_j=1} parity(h_j)$
   do while $parity(sum_i) \neq i \bmod 2$
   select random j where $s_j = 1, sum_i = sum_i - z_j$
   select random j where $s_j = 0, sum_i = sum_i + z_j$
   end do while
   end for
   $x_0 = be_1 + sum_0 \bmod m$.
   $x_1 = be_1 + sum_1 \bmod m$.
   $x_2 = be_2 + sum_2 \bmod m$.
   $x_3 = be_2 + sum_3 \bmod m$.
2: x_0 and x_1 are randomly ordered and sent to the client.
3: x_2 and x_3 are randomly ordered and sent to the client.
4: Relabel x_0, x_1, x_2, x_3 as u_0, u_1, u_2, u_3 where u_0, u_1, u_2, u_3 is the receiving order.
5: Client computes:
   for $i = 0$ to 3 **do**
   $v_i = u_i b^{-1} \bmod m$
   if $parity(v_i) = 1$ **then**
   $r_i = 1$
   else
   $r_i = 0$
   endif
   end for
6: Client generates f_0, f_1, f_2, f_3 as in step 3 in Algorithm 2 where $parity(f_0) = r_0 r_2$, $parity(f_1) = r_0 r_3$, $parity(f_2) = r_1 r_2$, and $parity(f_3) = r_1 r_3$.
7: The client computes $bf_0 \bmod m$, $bf_1 \bmod m$, $bf_2 \bmod m$, and $bf_3 \bmod m$, and sends these elements to the server in this order.
8: The server keeps the element corresponding the correct product and discards the other 3 elements. The kept element is the output of the multiplication gate.
9: Addition gates are processed by the server without client interaction.
10: An output o is summed with an even parity subset sum element s to get $s + o$ which is sent to the client. The client computes the plaintext output as $parity(b^{-1}(s + o) \bmod m)$.

Remark 1. One question is how does server privacy hold up given a client in the malicious model. During preprocessing, the client can give incorrect parity

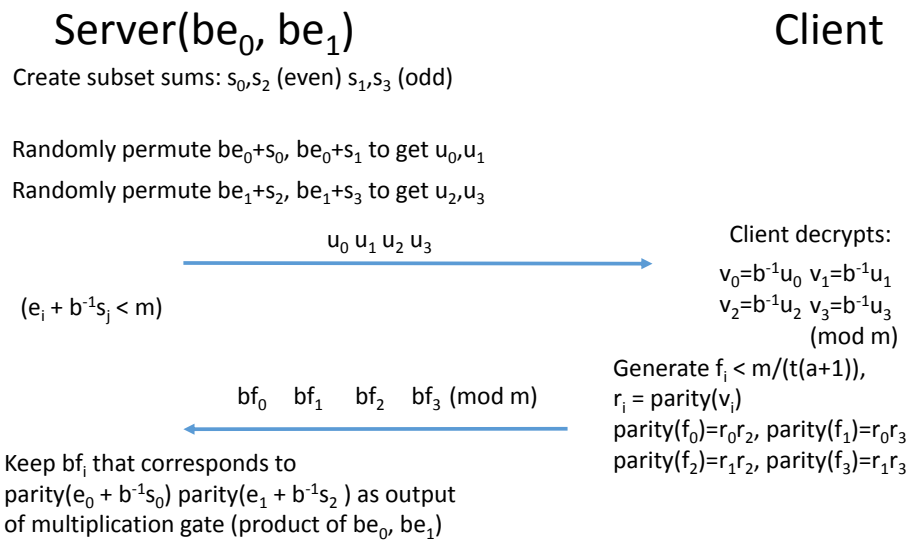

Fig. 2. Multiplication Gate Protocol

results to the server. We could construct a protocol where the server sends subset sums of its server elements to the client and the client responds with the parity for each of multiple queries. Given the hardness of subset sum, a malicious client could be detected (the client has to return the correct parity or guess, while the server knows what the parity should be based on the client's parity statements during preprocessing.) A similar protocol could detect a client that sends server elements with exponents that are incorrectly sized. A malicious client could return incorrect results for multiplication queries from the server resulting in leakage of server inputs or send initial client input elements and server elements (c_c and $2c_s$ elements respectively) that are not independent from each other which could lead to leakage of server inputs.

3.1 Proof of Security

Definition 2. *(Linear Subset Sum Problem): Given a circuit with a addition gates. For server security, we depend on the subset sum problem: we have a uniform random vector* $\boldsymbol{s}$ *where* $s_i \in \{0,1\}$, $T = \boldsymbol{s} \cdot b\boldsymbol{e} \bmod m$, $\boldsymbol{e} = (e_1, \ldots, e_n)$, $2 \leq e_i \leq m/(2n)$, $1 \leq i \leq n$. *The adversary is given* T, m, $\boldsymbol{e}$, *and* b, *and must find* $\boldsymbol{s}$.

Theorem 5. *Suppose the adversary has a polynomial time algorithm for the linear subset sum problem defined above. Then the adversary can use this algorithm as a subroutine to solve an integer subset sum problem in polynomial time.*

Proof. We are given integers $a_1, \ldots, a_n$, and $T = \sum_{i=1}^{n} s_i a_i$ where $\mathbf{s} = (s_1, \ldots, s_n)$, $s_i \in \{0, 1\}$, $1 \leq i \leq n$. The adversary constructs the linear subset sum problem as follows: We select m such that $a_1, \ldots, a_n < m/(2n)$, $b \in \mathbb{Z}_m$ such that $gcd(b, m) = 1$. Then let $T_2 = bT \bmod m$. Thus T_2, $(a_1, \ldots, a_n)$, b is an instance of a linear subset sum problem and the adversary can obtain $\mathbf{s}$ in polynomial time. ∎

Theorem 6. *The 2-party protocol of Algorithms 1, 2, and 3 has server privacy given the hardness of the subset sum problem and information-theoretic client privacy where server privacy also assumes the client is semi-honest. Server privacy has leakage roughly bounded by $8g/(t+2)$ plus additional leakage due to subset sum noise distribution leakage of the multiplicand. Both leakages approach zero as t grows. We assume that $c_s \leq \log(m^{1/4})$ if using oblivious transfer to send server input elements from the client, or $c_s \leq \log(m^{1/4})/2$ otherwise. The number of subset sum noise elements is $n \approx \log(m^{1/2})$.*

Proof. Theorem 2 establishes information-theoretic client privacy where we replace $a+1$ with $t(a+1)$ in the statement of the theorem and in the probability bound, given a malicious model server that does not receive decrypted output from the client. If the malicious server is to receive decrypted output from the client, the client first checks with the verifier (see below) to confirm that the server's protocol actions are correct. If so, then the client can send the decrypted output to the server.

Given that the client decrypts elements it receives from the server, collisions do not affect server privacy. The client ensures that both its client input elements and multiplication query response elements do not intersect with either the server input elements or the noise generator and noise elements. Then Theorem 2 bounds leakage of the client parities (the server's knowledge of its own element parities does not affect the possible parities for the client elements).

We use a simulator argument for server privacy. The simulator is given the client inputs, output, and the security parameter m.. It selects uniform random b such that $gcd(b, m) = 1$. It encrypts the client inputs $i_i, i_2, \ldots, i_{c_c} \in \{0, 1\}$ by selecting uniform random $a_1, \ldots, a_{c_c}$ so that $e_j = 2a_j + i_j$, $1 \leq j \leq c_c$, where $e_j <= m/(t(a+1))$. It creates the noise elements as described in Algorithm 1.

It then selects c_s server input elements of the same form $be \bmod m$ as described in Algorithm 2. The (trial) server plaintext bits are random.

The simulator processes the plaintext client and trial server inputs through the circuit and obtains the trial client output.

If the trial client output is not equal to the actual client output, the simulator will change one or more of the random trial server inputs and recheck the resulting trial client output as follows: The simulator can mark all of the inputs to the gates in the circuit as not modifiable, for the inputs that derive solely from the client inputs. The simulator then works backwards through the circuit starting with flipping one of the bits that is an input to the output gate. This forces changes to each of the gate inputs on a path from the output gate to an input gate. If the path dead ends prior to reaching an input gate, the simulator

has to backtrack and push the changes up another path to an input gate of the circuit. The process completes when the simulator reaches an input gate and flips a server input bit. The resulting modified server input bits are now used as the server input bits. The simulator can now encrypt these bits as in Algorithm 2.

For multiplication gates and output, subset sums are added to the multiplicands and output (the target point). When the sum of the noise and target point is between $m/2$ and $m/2 + m/t$ or less than m/t, we have potential leakage to the client. We roughly bound this leakage as $2mt/(m/2 + m/t) = 4/(t + 2)$ per element resulting in $8g/(t + 2)$ total leakage for the circuit.

A random subset sum added to two distinct target points will result in slightly different probability distributions. The maximal difference will be bounded by a zero and a $m/(t(a + 1))$ target point distributions (the approximately normal curve of the latter will be slid to the right by $m/(t(a + 1))$.

Both leakages go to zero as t grows.

∎

A verifier without access to the client secret key and parities of the client, server inputs but with access to the rest of the server's data and calculations, client's protocol messages, and circuit specification can verify that the server's computations are correct:

Theorem 7. *A verifier with access to a run of the 2-party protocol of Algorithms 1, 2, and 3 can verify that the server's computations are correct. Access to the run is defined as:*

1. *access to the server's internal data and computations other than the server input parity information,*
2. *access to the client's protocol record of messages sent and received by the client, and*
3. *access to the circuit specification.*

If the verifier does not have access to additional data (client secret key, input parities, and server's input parities), then client and server privacy is protected information-theoretically from a computationally unbounded verifier. If the verifier is correct then a malicious unbounded server that deviates from the protocol will be detected.

4 Scaling the Homomorphic and 2PC Schemes

For the 2PC scheme, the client will need to create $4g$ additional elements of the form $be \bmod m$, where g is the number of multiplication gates in the circuit. We recall $c = c_c + c_s + c_1 + 4g$. Thus m grows exponentially as the number of multiplication gates grows (see Theorem 2).

In order to limit the growth of m, the circuit can be partitioned into subcircuits so that each subcircuit has only a small number of multiplication gates. The outputs of these gates can be viewed as subcircuit outputs. The client can

take the subcircuit outputs that are not final circuit outputs and submit them as inputs into one of the other subcircuits, where the inputs are re-encrypted under a new secret key and possibly a new modulus.

This technique allows us to keep the m values growing very slowly (if we double the circuit size we expect the bit length of m grows by 1) while processing aribitrarily large circuits.

Figure 3 gives an example of dividing a circuit into two subcircuits C_1 and C_2. The input gates are circles with an i character inside. The inputs to C_1 are A, B, C, D, E, and F. The inputs to C_2 are G, H, I, J, K, and L. Note that G and H are outputs of multiplication gates in C_1, so they are re-encrypted with the new secret key for subcircuit C_2. More precisely, all four of the client responses are returned to the server and the server selects the correct product for each of the gates. The correct products are the inputs G and H to subcircuit C_2.

Definition 3. *(Subcircuits): Given boolean circuit C. A subcircuit C_1 is a connected subgraph of C such that the output of every gate in C_1 can be computed using only the wires and gates of C_1. A terminal gate of C_1 is a gate G such that all of the output wires of G lead to gates outside of C_1, and G is not an output gate of C. These outputs are the terminal outputs of C_1 and are $C_1's$ inputs to other subcircuits of C. If subcircuit C_1 uses secret key b and modulus m, then the terminal outputs are encrypted in keys that correspond to the subcircuit for which they are inputs. The other subcircuits have distinct secret keys. The other subcircuits also use separate sets of noise elements (see Algorithm 1). The non-terminal output inputs to the other subcircuits are computed as in Algorithm 2.*

The probability bound of Theorem 2 is slightly affected:

$$P \geq \left(1 - \frac{wk^2(t(a+1)N)^c}{m-1}\right)^{N^c+1}$$

is the approximate bound for subdividing into w subcircuits, each using m as the modulus or other modulus values of similar size.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Acharya, A., Hazay, C., Poburinnaya, O., Venkitasubramaniam, M.: Best of both worlds: Revisiting the spymasters double agent problem. In: Advances in Cryptology âĂŞ CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20âĂŞ24, 2023, Proceedings, Part I. p. 328âĂŞ359. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-3-031-38557-5_11, https://doi.org/10.1007/978-3-031-38557-5_11
2. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. Cryptology ePrint Archive, Paper 2014/094 (2014), https://eprint.iacr.org/2014/094, https://eprint.iacr.org/2014/094

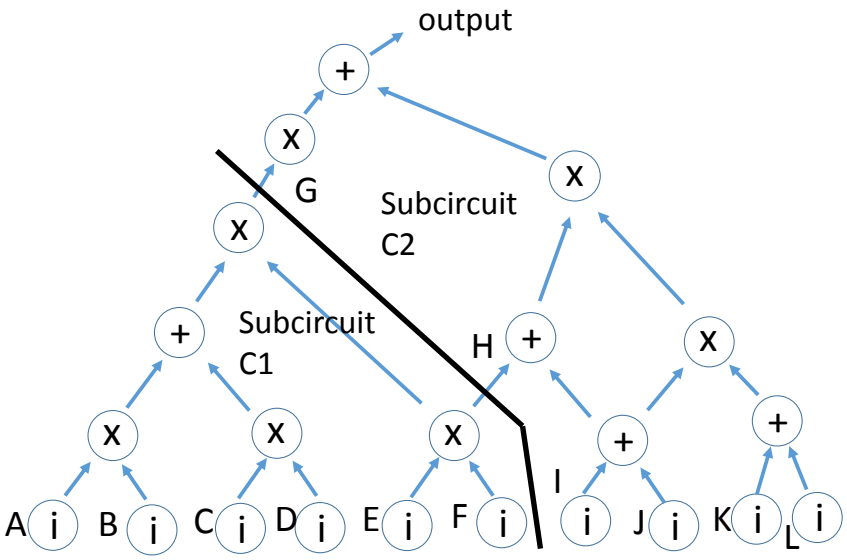

Fig. 3. An Example of Scaling the 2PC Protocol with Subcircuits

3. Badrinarayanan, S., Patranabis, S., Sarkar, P.: Statistical security in two-party computation revisited. In: Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7âĂŞ10, 2022, Proceedings, Part II. p. 181âĂŞ210. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-22365-5_7, https://doi.org/10.1007/978-3-031-22365-5_7
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. p. 1âĂŞ10. STOC '88, Association for Computing Machinery, New York, NY, USA (1988). https://doi.org/10.1145/62212.62213, https://doi.org/10.1145/62212.62213
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. Cryptology ePrint Archive, Paper 2012/078 (2012), https://eprint.iacr.org/2012/078, https://eprint.iacr.org/2012/078
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Paper 2011/277 (2011), https://eprint.iacr.org/2011/277, https://eprint.iacr.org/2011/277
7. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. pp. 97–106 (2011). https://doi.org/10.1109/FOCS.2011.12
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. Cryptology ePrint Archive, Paper 2011/344 (2011), https://eprint.iacr.org/2011/344, https://eprint.iacr.org/2011/344
9. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-lwe and security for key dependent messages. In: Rogaway, P. (ed.) Advances in Cryp-

tology – CRYPTO 2011. pp. 505–524. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
10. Brakerski, Z., Vaikuntanathan, V.: Lattice-based fhe as secure as pke. In: Proceedings of the 5th Conference on Innovations in Theoretical Computer Science. pp. 1–12. ITCS '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2554797.2554799, https://doi.org/10.1145/2554797.2554799
11. Branco, P., Döttling, N., Srinivasan, A.: A framework for statistically sender private ot with optimal rate. In: Advances in Cryptology âĂŞ CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20âĂŞ24, 2023, Proceedings, Part I. p. 548âĂŞ576. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-3-031-38557-5_18, https://doi.org/10.1007/978-3-031-38557-5_18
12. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. p. 11âĂŞ19. STOC '88, Association for Computing Machinery, New York, NY, USA (1988). https://doi.org/10.1145/62212.62214, https://doi.org/10.1145/62212.62214
13. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Paper 2009/616 (2009), https://eprint.iacr.org/2009/616, https://eprint.iacr.org/2009/616
14. Ducas, L., Micciancio, D.: Fhew: Bootstrapping homomorphic encryption in less than a second. Cryptology ePrint Archive, Paper 2014/816 (2014), https://eprint.iacr.org/2014/816, https://eprint.iacr.org/2014/816
15. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. pp. 169–178. Association for Computing Machinery, New York, NY, USA (2009), https://doi.org/10.1145/1536414.1536440
16. Gentry, C., Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. Cryptology ePrint Archive, Paper 2010/520 (2010), https://eprint.iacr.org/2010/520, https://eprint.iacr.org/2010/520
17. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. Cryptology ePrint Archive, Paper 2011/566 (2011), https://eprint.iacr.org/2011/566, https://eprint.iacr.org/2011/566
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. p. 218âĂŞ229. STOC '87, Association for Computing Machinery, New York, NY, USA (1987). https://doi.org/10.1145/28395.28420, https://doi.org/10.1145/28395.28420
19. Goldreich, O.: Foundations of cryptography volume ii basic applications. Cambridge University Press (2004)
20. Khurana, D., Mughees, M.H.: On statistical security in two-party computation. In: Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16âĂŞ19, 2020, Proceedings, Part II. p. 532âĂŞ561. Springer-Verlag, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-64378-2_19, https://doi.org/10.1007/978-3-030-64378-2_19
21. Kolesnikov, V.: Gate evaluation secret sharing and secure one-round two-party computation. In: Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3788,

pp. 136–155. Springer (2005). https://doi.org/10.1007/11593447_8, https://iacr.org/archive/asiacrypt2005/134/134.pdf
22. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques. pp. 223–238. EUROCRYPT'99, Springer-Verlag, Berlin, Heidelberg (1999)
23. Rivest, R.L., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation. pp. 169–180 (1978)
24. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (feb 1978). https://doi.org/10.1145/359340.359342, https://doi.org/10.1145/359340.359342
25. Yao, A.C.C.: How to generate and exchange secrets. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science. p. 162âĂŞ167. SFCS '86, IEEE Computer Society, USA (1986). https://doi.org/10.1109/SFCS.1986.25, https://doi.org/10.1109/SFCS.1986.25