— An extended abstract of this work appears in the proceedings of *Symposium on Theory of Computing* - STOC 2012. This is the full version. —

On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption

Adriana López-Alt
New York University

Eran Tromer
Tel Aviv University

Vinod Vaikuntanathan
University of Toronto

Abstract

We propose a new notion of secure multiparty computation aided by a computationally-powerful but untrusted "cloud" server. In this notion that we call *on-the-fly multiparty computation (MPC)*, the cloud can *non-interactively* perform arbitrary, *dynamically chosen* computations on data belonging to arbitrary sets of users chosen *on-the-fly*. All user's input data and intermediate results are protected from snooping by the cloud as well as other users. This extends the standard notion of fully homomorphic encryption (FHE), where users can only enlist the cloud's help in evaluating functions on their own encrypted data.

In on-the-fly MPC, each user is involved only when initially uploading his (encrypted) data to the cloud, and in a final output decryption phase when outputs are revealed; the complexity of both is independent of the function being computed and the total number of users in the system. When users upload their data, they need not decide in advance which function will be computed, nor who they will compute with; they need only retroactively approve the eventually-chosen functions and on whose data the functions were evaluated.

This notion is qualitatively the best possible in minimizing interaction, since the users' interaction in the decryption stage is inevitable: we show that removing it would imply generic program obfuscation and is thus impossible.

Our contributions are two-fold:

1. We show how on-the-fly MPC can be achieved using a new type of encryption scheme that we call *multikey FHE*, which is capable of operating on inputs encrypted under multiple, unrelated keys. A ciphertext resulting from a multikey evaluation can be jointly decrypted using the secret keys of all the users involved in the computation.
2. We construct a multikey FHE scheme based on NTRU, a very efficient public-key encryption scheme proposed in the 1990s. It was previously not known how to make NTRU fully homomorphic even for a single party. We view the construction of (multikey) FHE from NTRU encryption as a main contribution of independent interest. Although the transformation to a fully homomorphic system deteriorates the efficiency of NTRU somewhat, we believe that this system is a leading candidate for a practical FHE scheme.

Contents

1 Introduction **1**
1.1 Our Results and Techniques . . . 4

2 On-the-Fly MPC from Multikey FHE **7**
2.1 Multikey Fully Homomorphic Encryption . . . 8
2.2 The Basic Protocol . . . 10
2.3 Security Against Malicious Adversaries . . . 10

3 Multikey Somewhat Homomorphic Encryption based on NTRU **14**
3.1 Preliminaries for the NTRU Instantiation . . . 14
3.2 The Scheme . . . 16
3.3 Security . . . 17
3.4 Multikey Homomorphism . . . 18

4 From Somewhat to Fully Homomorphic Encryption **22**
4.1 Bootstrapping . . . 23
4.2 Modulus Reduction . . . 24

5 Acknowledgements **28**

A Multikey Properties of Other FHE Schemes **32**
A.1 A Generic Construction . . . 32
A.2 From Ring-LWE . . . 33

B Impossibility of a 2-Round Protocol **33**

C Simulation Extractability **35**

D Proofs **35**
D.1 Security against Semi-Honest Adversaries . . . 35
D.2 Security against Malicious Adversaries . . . 37

1 Introduction

We are fast approaching a new digital era in which we store our data and perform our expensive computations remotely, on powerful servers — the "cloud", in popular parlance. While the cloud offers numerous advantages in costs and functionality, it raises grave questions of confidentiality, since data stored in the cloud could be vulnerable to snooping by the cloud provider or even by other cloud clients [RTSS09]. Since this data often contains sensitive information (e.g., personal conversations, medical information and organizational secrets), it is prudent for the users to encrypt their data before storing it in the cloud. Recent advances in fully homomorphic encryption (FHE) [Gen09b, vDGHV10, BV11b, BV11a, GH11, BGV12] make it possible to perform arbitrary computations on encrypted data, thus enabling the prospect of personal computers and mobile devices as trusted but weak interfaces to a powerful but untrusted cloud on which the bulk of computing is performed.

FHE is only suitable in settings where the computations involve a single user, since it requires inputs to be encrypted under the same key. However, there are many scenarios where users, who have uploaded their large data stores to the cloud in encrypted form, then decide to compute some joint function of their data. For example, they may wish the cloud to compute joint statistical information on their databases, locate common files in their collections, run a computational agent to reach a decision based on their pooled data (without leaking anything but the final decision), or generally, in contexts where multiple (mutually distrusting) users need to pool together their data to achieve a common goal.

The multiparty scenario is significantly more complex, and comes with a set of natural but stringent requirements. First, the participants involved in the computation and the function to be computed may be *dynamically chosen on-the-fly*, well after the data has been encrypted and uploaded to the cloud. Secondly, once the function is chosen, we should not expect the users to be online all the time, and consequently it is imperative that the cloud be able to perform the bulk of this computation (on the encrypted data belonging to the participants) *non-interactively*, without consulting the participants at all. Finally, all the burden of computation should indeed be carried by the cloud: the computational and communication complexity of the users should depend only on the size of the individual inputs and the output, and should be independent of both the complexity of the function computed and the total number of users in the system, both of which could be very large.

On-the-Fly Multiparty Computation. Consider a setting with a large universe of computationally-weak users and a powerful cloud. An on-the-fly multiparty computation protocol proceeds thus:

1. The numerous users each encrypt their data and upload them to the cloud, *unaware of the identity or even the number* of other users in the system. Additional data may arrive directly to the cloud, encrypted under users' public keys (e.g., as encrypted emails arriving to a cloud-based mailbox).

2. The cloud decides to evaluate an arbitrary *dynamically chosen* function on the data of arbitrary subset of users chosen *on-the-fly*. (The choice may be by some users' request, or as a service to compute the function on the data of parties fulfilling some criterion, or by a need autonomously anticipated by the cloud provider, etc.) The cloud can perform this computation *non-interactively*, without any further help from the users. The result is still encrypted.

3. The cloud and the subset of users whose data was used in the computation interact in a decryption phase. At this point the users retroactively approve the choice of function and the choice of peer users on whose data the function was evaluated, and cooperate to retrieve the output.

Crucially, the computation and communication of all the users (including the cloud) in the decryption phase should be *independent of* both the complexity of the function computed, and the size of the universe of parties (both of which can be enormous). Instead, the effort expended by the cloud and the users in this phase should depend only on the size of the output and the number of users who participated in the computation. Also crucially, the users need not be online at all during the bulk of the computation; they need to "wake up" only when it is time to decrypt the output.

We call this an *on-the-fly multiparty computation* (or *on-the-fly MPC* in short) to signify the fact that the functions to be computed on the encrypted data and the participants in the computation are both chosen on-the-fly and dynamically, without possibly even the knowledge of the participants. Protocols following this framework have additional desirable features such as the ability for users to "join" a computation asynchronously.

Possible Approaches (and Why They Do Not Work). The long line of work on secure multiparty computation (MPC) [GMW87, BGW88, CCD88, Yao82] does not seem to help us construct on-the-fly MPC protocols since the computational and communication complexities of *all the parties* in these protocols depends polynomially on the complexity of the function being computed.[1] In contrast, we are dealing with an asymmetric setting where the cloud computes a lot, but the users compute very little. (Nevertheless, we will use the traditional MPC protocols to interactively compute the decryption function at the end.)

Fully homomorphic encryption (FHE) is appropriate in such an asymmetric setting of computing with the cloud. Yet, traditional FHE schemes are *single-key* in the sense that they can perform (arbitrarily complex) computations on inputs encrypted under the same key. In our setting, since the parties do not trust each other, they will most certainly not want to encrypt their inputs using each other's keys. Nevertheless, Gentry [Gen09a] proposed the following way of using single-key FHE schemes in order to do multiparty computation: first, the parties run a (short) MPC protocol to compute a joint public key, where the matching secret key is *secret-shared* among all the parties. The parties then encrypt their inputs under the joint public key and send the ciphertexts to the cloud who then uses the FHE scheme to compute an encryption of the result. Finally, the parties run yet another (short) MPC protocol to recover the result. A recent work by Asharov et al. [AJL$^+$12] extends this schema and makes it efficient in terms of the concrete round, communication and computational complexity.

This line of work does not address the *dynamic* and *non-interactive* nature of on-the-fly MPC. In particular, once a subset of parties and a function are chosen, the protocols of [Gen09a, AJL$^+$12] require the parties to be online and run an interactive MPC protocol to generate a joint public key. In contrast, we require that once the function and a subset of parties is chosen, the cloud performs the (expensive) computations *non-interactively*, without help from any of the users. It would also be unsatisfactory to postpone the (lengthy) computation of the function until the interactive

[1]The works of Damgård et al. [DIK$^+$08, DIK10] are an exception to this claim. However, it is not clear how to build upon these results to address the dynamic and non-interactive nature of on-the-fly MPC.

decryption phase; indeed, we require that once the users "wake up" for the decryption phase, the running time of all parties is independent of the complexity of the function being computed. Thus, even the feasibility of on-the-fly MPC is not addressed by existing techniques.

Our Solution. We present a *new notion* of fully homomorphic encryption (FHE) that we call a multikey FHE that permits computation on data encrypted under multiple unrelated keys; a *new construction* of multikey FHE based on the NTRU encryption scheme (originally proposed by Hoffstein, Pipher and Silverman [HPS98]); and a *new method* of achieving on-the-fly multiparty computation (for any a-priori bounded number of users) using a multikey FHE scheme. Although the number of users involved in any computation has to be bounded in our solution, the total number of users in the system is arbitrary.

1.1 Our Results and Techniques

New Notion: Multikey Homomorphic Encryption. An N-key Fully Homomorphic Encryption scheme is the same as a regular FHE scheme with two changes. First, the homomorphic evaluation algorithm takes in polynomially many ciphertexts encrypted under at most N keys, together with the corresponding evaluation keys, and produces a ciphertext. Second, in order to decrypt the resulting ciphertext, one uses all the involved secret keys.

A multikey FHE scheme is indeed the right tool to perform on-the-fly MPC as shown by the following simple protocol: the users encrypt their inputs using their own public keys and send the ciphertexts to the cloud, the cloud then computes a dynamically chosen function on an arbitrary subset of parties using the multikey property of the FHE scheme, and finally, the users together run an interactive MPC protocol in order to decrypt. Note that the users can be offline during the bulk of the computation, and they need to participate only in the final cheap interactive decryption process. Note also that participants in the protocol need not be aware of the entire universe of users, but only those users that participate in a joint computation. This simple protocol provides us security against a semi-honest collusion of the cloud with an arbitrary subset of parties. We then show how to achieve security against a malicious adversary, using tools such as verifiable computation protocols [GKR08, GGP10, CKV10, AIK10] or succinct argument systems [Kil92, Mic94, BCCT12a, GLR11].

The computation of the decryption function can itself be outsourced to the cloud. In particular, using the cloud-assisted MPC protocol of Asharov et al. [AJL$^+$12] yields a 5-round on-the-fly MPC protocol (one offline round, and four online rounds to perform decryption). As an additional benefit, in the resulting on-the-fly protocol, the parties may communicate with the server *concurrently* at each stage. The only disadvantage of this approach is that it requires a CRS setup. This does not, however, affect the on-the-fly nature of the procotol since only an apriori bound N on the number of computing parties needs to be known when creating the CRS.

(Multikey) Fully Homomorphic Encryption from NTRU. The starting point of our main construction of multikey FHE is the NTRU encryption scheme of Hoffstein, Pipher and Silverman [HPS98] (more precisely, the slightly modified version due to Stehlé and Steinfeld [SS11]). NTRU is one of the earliest lattice-based public-key encryption schemes, together with the Ajtai-Dwork cryptosystem [AD97] and the Goldreich-Goldwasser-Halevi cryptosystem [GGH97]. We first observe that NTRU can be made (single-key) fully homomorphic using the recent techniques of [BV11a, BGV12]. Using some additional tricks, we then show that the scheme is *multikey fully* *homomorphic* for a bounded number of users at essentially the same cost. Previously, it was not even known whether NTRU could be turned into a (regular, single-key) fully homomorphic encryption scheme.[2]

This construction is one of our main contributions and we believe it to be of independent interest. Our construction is particularly interesting since the NTRU scheme was originally proposed as an *efficient* public-key encryption scheme, meant to replace RSA and elliptic curve cryptosystems in applications where computational efficiency is at a premium (for example, applications that run on smart cards and embedded systems). Although the transformation to a fully homomorphic system deteriorates the efficiency of NTRU somewhat, we believe that this system is a leading candidate for a practical FHE scheme. What's more, as we show, the scheme supports homomorphic operations on encryptions under multiple keys.

Theorem 1.1 (Informal)**.** *For every $N \in \mathbb{N}$, there is an N-user multikey fully homomorphic encryption scheme under the assumption that the NTRU encryption scheme (described below) is semantically secure and circular secure. The size of the keys and ciphertexts in the scheme grow polynomially with N.*

We briefly sketch here our variant of the NTRU encryption scheme and the ideas in turning it into a multikey fully homomorphic encryption scheme. The reader is referred to Section 3 and Section 4 for a detailed exposition.

The main differences between the original NTRU scheme and our variant are threefold: (1) Whereas the original NTRU scheme adds a deterministic noise to the ciphertext, the variant considered here adds noise chosen from a distribution with bounded support (specifically, a discrete Gaussian distribution), a modification recently introduced by Stehlé and Steinfeld [SS11]. It seems that this could only improve security; indeed, the purpose of the Stehlé-Steinfeld work was to prove the security of NTRU based on worst-case hardness assumptions on ideal lattices, (2) We do all our operations modulo $x^n + 1$ where n is a power of 2 as in [SS11], as opposed to $x^n - 1$ in the original NTRU. (3) Our parameters are more aggressive than in [HPS98, SS11] to support homomorphisms. As a result, the worst-case to average-case connection shown by [SS11] does not carry over to our setting of parameters.

For security parameter κ, the scheme is parametrized by a prime number $q = q(\kappa)$, and a B-bounded error distribution χ over the ring $R \equiv \mathbb{Z}[x]/\langle x^n + 1\rangle$ (i.e., χ is a distribution over polynomials whose coefficients are all at most $B(\kappa)$ in absolute value). The parameters n, q and χ are public. We show how to encrypt bits using the scheme. All operations in the scheme take place in the ring $R_q \equiv R/qR$.

$\mathsf{Keygen}(1^\kappa)$**:** Sample "bounded" polynomials $f', g \leftarrow \chi$ and set $f := 2f' + 1$ so that $f \equiv 1 \pmod 2$. Set the public key $\mathsf{pk} := h = 2gf^{-1} \in R_q$ and the secret key $\mathsf{sk} = f \in R$. (If f is not invertible over R_q, resample f').

$\mathsf{Enc}(\mathsf{pk}, m)$**:** Sample "bounded" polynomials $s, e \leftarrow \chi$. Output the ciphertext $c := hs + 2e + m \in R_q$.

$\mathsf{Dec}(\mathsf{sk}, c)$**:** Let $\mu = fc \in R_q$. Output $\mu \pmod 2$ as the message.

[2]The observation that NTRU can be made *single-key* fully homomorphic was made concurrently by Gentry et al. [GHL+11].

Decryption works since

$$\begin{aligned} fc \pmod q &= f(hs + 2e + m) \pmod q \\ &= 2(gs + ef) + fm \pmod q \\ &= 2(gs + ef) + fm \end{aligned}$$

where the last equality is true since $|2(gs + ef) + fm| < q/2$.[3] Taking this quantity mod 2 then gives us the message m since $f \equiv 1 \pmod 2$.

The multikey homomorphic properties of the scheme are best seen through the lens of the decryption equation (as in [BV11a, BV11b]). In particular, consider ciphertexts $c_1 = h_1 s_1 + 2e_1 + m_1 \in R_q$ and $c_2 = h_2 s_2 + 2e_2 + m_2 \in R_q$ that encrypt messages m_1 and m_2 under public keys h_1 and h_2 respectively, with noise terms e_1 and e_2. A little algebraic manipulation shows that $c_{\text{add}} = c_1 + c_2$ and $c_{\text{mult}} = c_1 c_2$ are ciphertexts that encrypt the sum and product of m_1 and m_2, respectively, albeit with larger error terms. Namely, decrypting $c_1 + c_2$ and $c_1 c_2$ with the "joint secret key" $f_1 f_2$ (which is simply a product of the two secret keys f_1 and f_2) gives us:

$$\begin{aligned} f_1 f_2 (c_1 + c_2) &= 2(f_1 f_2 (e_1 + e_2) + f_2 g_1 s_1 + f_1 g_2 s_2) + f_1 f_2 (m_1 + m_2) \\ &\triangleq 2E_{\text{add}} + f_1 f_2 (m_1 + m_2) \end{aligned}$$

This shows that decrypting $c_1 + c_2$ using the joint secret key $f_1 f_2$ results in the sum of the two messages, assuming that the error does not grow to be too large. Likewise, we have:

$$\begin{aligned} f_1 f_2 (c_1 c_2) &= 2(2 g_1 g_2 s_1 s_2 + g_1 s_1 f_2 (2e_2 + m_2) + g_2 s_2 f_1 (2e_1 + m_1) + \\ &\qquad f_1 f_2 (e_1 m_2 + e_2 m_1 + 2 e_1 e_2)) + f_1 f_2 (m_1 m_2) \\ &\triangleq 2E_{\text{mult}} + f_1 f_2 (m_1 m_2) \end{aligned}$$

This shows that decrypting $c_1 c_2$ using the joint secret key $f_1 f_2$ results in the product of the two messages, assuming that the error does not grow to be too large.

Extending this to circuits, we observe that the effective secret key required to decrypt a ciphertext c resulting from evaluating a multivariate polynomial function on the inputs of N users is $\prod_{i=1}^{N} f_i^{d_i}$ where d_i is the degree of the i^{th} variable in the polynomial function. This makes the secret key required to decrypt c *dependent* on the circuit evaluated, which is unacceptable even for somewhat homomorphic encryption. We use the relinearization technique from [BV11a] to transform the ciphertext into one that can be decrypted using the secret key $\prod_{i=1}^{N} f_i$ (namely, reduce all the exponents from d_i to 1), after every operation. In effect, this ensures that the secret key is related to the number of users N involved in the computation, and not to the function being computed. With the use of relinearization, one can show that the scheme is *multikey somewhat homomorphic*, i.e., capable of evaluating circuits of depth $\epsilon \log n$ for some small constant $\epsilon < 1$. (For more details, see Section 3).

To turn this into a fully homomorphic encryption scheme, we use the technique of modulus reduction from the work of Brakerski and Vaikuntanathan [BV11a], later refined in [BGV12]. Modulus reduction shows how to reduce the magnitude of the error (while simultaneously reducing

[3]We associate $\mathbb{Z}_q$ with the set $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$ throughout this work.

the size of the modulus). This technique works transparently in the multikey setting. The bottom line is that we can evaluate functions on N users as long as $N \approx \log q/\text{polylog}(n)$. Put another way, for any number N of users, we get a N-user multikey FHE by setting q to be a large enough function of N. This gives us a leveled multikey FHE scheme. Finally, to turn this into a full-fledged multikey FHE scheme (whose complexity is independent of the complexity of the function being computed), we use (a multikey analog of) Gentry's bootstrapping technique [Gen09b].

Our construction based on the NTRU encryption scheme raises a natural question: can any of the other FHE schemes be made multikey? It turns out that the schemes of [Gen09a, SV10, vDGHV10, BV11b, BV11a] can be made N-key fully homomorphic for a constant N, or sometimes even $N = O(\log n)$. See Appendix A for more details.

Completely Non-Interactive On-the-Fly MPC? Our results raise the natural question of whether the protocols can be made *completely non-interactive*, namely the users do not ever have to talk to each other, even in the decryption phase. We know from [HLP11] that in the non-interactive setting, the server can always evaluate the circuit multiple times, keeping some parties' inputs but plugging in fake inputs of its choosing for the other parties. However, even if we accept this as the ideal functionality, we show that a non-interactive protocol cannot be achieved by drawing on the impossibility of program obfuscation. Thus, our notion is qualitatively "the best possible" in terms of interaction. See Appendix B for a formal theorem statement. [Eran's note: *Verify!*]

Other Related Work. The basic idea of using homomorphic encryption schemes in conjunction with threshold decryption to boost the efficiency of MPC protocols was first noticed by Cramer, Dåmgard and Nielsen [CDN01]. The idea of using a cloud to alleviate the computational efforts of parties was recently explored in the work on "server-aided MPC" by Kamara, Mohassel and Raykova [KMR11]. Their protocols, however, require some of the parties to do a large amount of computation, essentially proportional to the size of the function f being computed. Halevi, Lindell and Pinkas [HLP11] recently considered the model of "secure computation on the web" wherein the goal is to minimize interaction between the parties. While their definition requires absolutely no interaction among the participants in the protocols (the participants interact with the server only), they show that this notion can only be achieved for a small class of functions. Our goal, on the other hand, is to compute arbitary functions with the assistance of a cloud.

Organization. In Section 2 we formally define multikey FHE and on-the-fly MPC, and show our construction of on-the-fly MPC from multikey FHE. In Section 3 we show how to instantiate multikey somewhat homomorphic encryption from the NTRU encryption scheme, and then show how to achieve full homomorphism in Section 4. Finally, in Appendix B, we show the impossibility of a completely non-interactive on-the-fly MPC protocol.

Notation. In the remainder of the paper, we use the following notation. We use κ to denote the security parameter. For an integer n, we use the notation $[n]$ to denote the set $[n] = \{1, \ldots, n\}$. For a randomized function f, we write $f(x; r)$ to denote the unique output of f on input x with random coins r. We write $f(x)$ to denote a random variable for the output of $f(x; r)$ over uniformly random coins r. For a distribution or random variable X, we write $x \leftarrow X$ to denote the operation of sampling a random x according to X. For a set S, we overload notation and use $s \leftarrow S$ to denote sampling s from the uniform distribution over S. We use $y := f(x)$ to denote the *deterministic* evaluation of f on input x with output y. For two distributions D and D', $D \overset{c}{\approx} D'$ denotes computational indistinguishability.

2 On-the-Fly MPC from Multikey FHE

We consider the problem of a *server* or *cloud*, denoted by S, storing the data of U different parties $P_1, \ldots, P_U$. We wish to ensure that the data of each party is kept private, but also allow the server S to compute any joint function of the data of any subset $V \subseteq [U]$ of the parties. We also wish to ensure that the server is able to do this with minimal participation from the parties in V, and no interaction at all from the rest of the parties $[U]\backslash V$. Furthermore, the communication complexity and the computation time of each party P_i should be independent of the complexity of the function since we rely on the computation power of the server, who will carry out the entire computation of the joint function F. The computation should remain secure even if the server or any set of parties are corrupted. We formalize this below.

For a class $\mathcal{C}$ of functions with at most U inputs, an *on-the-fly multiparty protocol* Π for $\mathcal{C}$ is a protocol between $U+1$ interactive Turing Machines $P_1, \ldots, P_U, S$, such that for all inputs $\vec{x} = (x_1, \ldots, x_U)$, all functions $F \in \mathcal{C}$, if F is an N-input function then for all ordered subsets $V \subseteq [U]$ such that $|V| = N$, the output of Π in an execution where P_i is given x_i as input (S does not receive an input), and F, V are chosen for the computation, is $y = F(\{x_i\}_{i \in V})$. An on-the-fly multiparty protocol consists of two phases, an offline phase that is performed before the function $F \in \mathcal{C}$ is chosen, and an online phase that begins once F is chosen together with a subset V of inputs on which F will be evaluated. All parties $P_1, \ldots, P_U, S$ participate in the offline phase, but only the server S and parties in V participate in the online phase. After the function is selected, the server ignores all offline messages from non-computing parties (i.e. those in $[U]\backslash V$).

Unlike in standard MPC, we require the communication complexity of the protocol, as well as the computation time of parties $P_1, \ldots, P_U$ to be *independent* of the complexity of the function F. Furthermore, we let the computation time of parties P_i for $i \in V$ depend on the party's input and the output size of the F but require the computation time of parties P_i for $i \in [U]\backslash V$ to depend only on the size of the party's input and be independent of the output size of F. On the other hand, the computation time of the server S must be *linear* in the circuit-size of F.

Security. We prove security of an on-the-fly MPC protocol in the Ideal/Real paradigm. Let $F(\{x_i\}_{i \in V})$ for $V \subseteq [U]$ be the function to be computed, and let $N = |V|$. For ease of notation, we assume w.l.o.g. that $V = [N]$. In the *ideal world*, the computation of F is performed through a trusted functionality $\mathcal{F}$ that receives input x_i from each party P_i for $i \in [U]$, computes $y = F(x_1, \ldots, x_N)$ (ignoring all inputs x_i for $i \notin V$) and gives y to parties $P_1, \ldots, P_N, S$, while parties P_i for $i \in [U]\backslash V$ do not get an output. Thus, in the ideal world, parties learn nothing more than y. In the *real world*, however, this trusted functionality does not exist and so in order to compute $y = F(x_1, \ldots, x_N)$, parties $P_1, \ldots, P_U, S$ run a protocol Π.

An adversary corrupting a party (resp. the server) receives all messages directed to the corrupted party (resp. the server) and controls the messages that it sends. Since the server ignores messages from parties outside V, we assume w.l.o.g. that an adversary only corrupts computing parties, i.e. parties in V, and possibly the server.

We use $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\vec{x})$ to denote the joint output of the ideal-world adversary $\mathcal{S}$ and the outputs of the server S and the parties $P_1, \ldots, P_N$ in an ideal execution with functionality $\mathcal{F}$ and inputs $\vec{x} =$

$(x_1, \ldots, x_U)$. Similarly, we use $\mathrm{REAL}_{\Pi,\mathcal{A}}(\vec{x})$ to denote the joint output of the real-world adversary $\mathcal{A}$ and the outputs of parties $P_1, \ldots, P_N$ and server S in an execution of protocol Π with inputs $\vec{x} = (x_1, \ldots, x_U)$. We say that a protocol Π *securely realizes* $\mathcal{F}$ if for every real-world adversary $\mathcal{A}$ corrupting any $t < N$ parties (and possibly the server), there exists an ideal-world adversary $\mathcal{S}$ with black-box access to $\mathcal{A}$ such that for all input vectors $\vec{x}$, $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}}(\vec{x}) \overset{c}{\approx} \mathrm{REAL}_{\Pi,\mathcal{A}}(\vec{x})$.

2.1 Multikey Fully Homomorphic Encryption

In this section, we define *multikey fully homomorphic encryption*. Intuitively, multikey FHE allows us to evaluate any circuit on ciphertexts that might be encrypted under *different* public keys. To guarantee semantic security, decryption requires all of the corresponding secret keys.

We introduce a parameter N, which is the number of distinct keys that a scheme can tolerate. We let all algorithms depend polynomially on N. This is similar to the definition of "leveled" FHE from [BGV12]. However, we note that in our definition, the algorithms depend on N but are independent of the depth of circuits that the scheme can evaluate. Thus, we consider schemes that are "leveled" with respect to the number of keys N, but fully homomorphic ("non-leveled") with respect to the circuits that are evaluated. The construction of multikey FHE schemes that are not leveled with respect to the number of keys (i.e., where all algorithms are independent of N) remains an open problem.

We now define multikey FHE as follows, for restricted circuit classes and for arbitrary circuits.

Definition 2.1. (Multikey $\mathcal{C}$-Homomorphic Encryption) *Let $\mathcal{C}$ be a class of circuits. A family $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \quad \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ of algorithms is a* multikey $\mathcal{C}$-homomor-phic encryption scheme family *if for all integers $N > 0$, $\mathcal{E}^{(N)}$ has the following properties:*

- $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Keygen}(1^\kappa)$*, for a security parameter κ, outputs a public key* pk*, a secret key* sk *and a (public) evaluation key* ek.

- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*, given a public key* sk *and message m, outputs a ciphertext c.*

- $m' := \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$*, given N secret keys* sk_i *and a ciphertext c, outputs a message m'.*

- $c^* := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_t, \mathsf{pk}_t, \mathsf{ek}_t))$*, given a (description of) a boolean circuit C along with t tuples $(c_i, \mathsf{pk}_i, \mathsf{ek}_i)$, each comprising of a ciphertext c_i, a public key* pk_i*, and an evaluation key* ek_i*, outputs a ciphertext c^*.*

  We require absence of decryption failures and compactness of ciphertexts. Formally: for every circuit $C \in \mathcal{C}$, all sequences of N key tuples $\{(\mathsf{pk}'_j, \mathsf{sk}'_j, \mathsf{ek}'_j)\}_{j\in[N]}$ each of which is in the support of $\mathsf{Keygen}(1^\kappa)$*, all sequences of t key tuples $\{(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i)\}_{i\in[t]}$ each of which is in $\{(\mathsf{pk}'_j, \mathsf{sk}'_j, \mathsf{ek}'_j)\}_{j\in[N]}$, and all plaintexts $(m_1, \ldots,\ m_t)$ and ciphertexts $(c_1, \ldots, c_t)$ such that c_i is in the support of* $\mathsf{Enc}(\mathsf{pk}_i, m_i)$*,* Eval *satisfies the following properties:*

  Correctness: *Let* $c^* := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_t, \mathsf{pk}_t, \mathsf{ek}_t))$*. Then* $\mathsf{Dec}(\mathsf{sk}'_1, \ldots, \mathsf{sk}'_N, c^*) = C(m_1, \ldots, m_t)$. [4]

[4] Note that correctness still holds even if the circuit C completely ignores all ciphertexts encrypted under a public key $\mathsf{pk}_i{}'$, or if none of the original ciphertexts were encrypted under this key. In other words, using superfluous keys in the decryption process does not affect its correctness (as long as decryption uses at most N keys).

Compactness: *Let* $c^* := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_t, \mathsf{pk}_t, \mathsf{ek}_t))$*. There exists a polynomial* P *such that* $|c^*| \leq P(\kappa, N)$*. In other words, the size of* c^* *is independent of* t *and* $|C|$*. Note, however, that we allow the evaluated ciphertext to depend on the number of keys,* N*.*

Definition 2.2. (MULTIKEY FHE) *A family of encryption schemes* $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ *is* multikey fully homomorphic *if it is multikey* $\mathcal{C}$*-homomorphic for the class* $\mathcal{C}$ *of all circuits.*

Semantic security of a multikey FHE follows directly from the semantic security of the underlying encryption scheme in the presence of the evaluation key ek. This is because given ek, the adversary can compute Eval himself. Note that taking $N = 1$ in Definition 2.1 and Definition 2.2 yield the standard definitions of $\mathcal{C}$-homomorphic and fully homomorphic encryption schemes.

2.2 The Basic Protocol

Let $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ be a multikey fully-homomorphic family of encryption schemes. We construct the following on-the-fly MPC protocol Π^{SH} secure against semi-honest adversaries.

Step 1: For $i \in [U]$, party P_i samples a key tuple

$$(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$$

and encrypts its input x_i under pk_i:

$$c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$$

It sends $(\mathsf{pk}_i, \mathsf{ek}_i, c_i)$ to the server S.

At this point a function F, represented as a circuit C, has been selected on inputs $\{x_i\}_{i \in V}$ for some $V \subseteq U$. Let $N = |V|$. For ease of notation, assume w.l.o.g. that $V = [N]$. The parties proceed as follows.

Step 2: The server S computes

$$c^* := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_N, \mathsf{pk}_N, \mathsf{ek}_N))$$

and broadcasts c^* to parties $P_1, \ldots, P_N$.

Step 3: The parties $P_1, \ldots, P_N$ run a secure MPC protocol $\Pi_{\text{DEC}}{}^{\text{SH}}$ to compute $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c^*)$.

Theorem 2.1. *Let* $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ *be a multikey fully-homomorphic encryption scheme, and let* $\Pi_{\text{DEC}}{}^{\text{SH}}$ *be an* N*-party MPC protocol for computing the decryption function* $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c^*)$*. If* $\mathcal{E}$ *is semantically secure, and* $\Pi_{\text{DEC}}{}^{\text{SH}}$ *is secure against semi-honest adversaries corrupting* $t < N$ *parties, then the above construction is an on-the-fly MPC protocol secure against semi-honest adversaries corrupting* t *parties and possibly the server* S*.*

See Appendix D for the proof of Theorem 2.1.

2.3 Security Against Malicious Adversaries

The protocol described in Section 2.2 is not secure against malicious adversaries. Our first step in handling this type of attack is to replace our decryption protocol with one that is secure against malicious adversaries, which we will denote $\Pi_{\text{DEC}}{}^{\text{MAL}}$. Next, we will apply general MPC techniques to the rest of our protocol (Steps 1 and 2) to ensure parties do not deviate from the protocol. This requires coin-flipping and zero-knowledge proofs. However, there are two subtleties to consider.

1. First, recall that in our model, parties do not communicate with each other until the decryption phase. In particular, parties do not communicate with each other (or even know about the existance of other parties) during Step 1. Therefore, coin-flipping in Step 1 is out of the question. Fortunately, the correctness property (from Definition 2.1) guarantees that the scheme is secure against corrupt parties that follow the protocol in Step 1 but adaptively choose their random coins. This means that parties do not need to coin-flip for each other's random coins. Furthermore, since the server's computation throughout the protocol is deterministic, the parties do not need to coin-flip for the server's random coins.

   We therefore only need to add zero-knowledge proofs of knowledge[5] to ensure that the parties indeed follow the protocol. The intuition behind this is that correctness will guarantee that the simulator can extract the input $\tilde{x}_i$ for a corrupted party and therefore obtain the correct value $\tilde{y}$ from the ideal functionality, regardless of the coins used by the adversary.

2. Second, we wish to ensure that the computation time and communication complexity of the parties is small. This means that the server must be able to prove that he carried out the computation of Eval correctly in such a way that the parties can verify the validity of the proof in time that is much less than linear in the circuit size. To solve this problem, we use techniques from verifiable computation. We offer several solutions, each with its own benefits and drawbacks.

Verification for Small Inputs. We first consider the case where the ciphertexts $(c_1, \ldots, c_N)$ are small enough to be broadcast to the N parties in V (i.e., allowing communication complexity linear in the total input size of the participating parties). In this case, the server needs to convince the participating parties that "$c^* = \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_N, \mathsf{pk}_N, \mathsf{ek}_N))$", i.e., that a deterministic circuit of size $\text{poly}(|C|, \kappa)$ accepts. For any uniform circuit C (i.e., computable by a $\text{poly}(\kappa)$-time Turing machine), the following offer $\text{poly}(\kappa, \log(|C|))$ communcation and verification efficiency.[6]

1. Use the argument system of Kilian [Kil92, Kil95], yielding *interactive* 4-round verification. It relies on expensive PCPs.

[5]There is a subtely here. We assume a *rushing adversary*, that is, we assume that the adversary can choose his messages *adaptively*, depending on the messages from the honest players. Because of this, in the proof the simulator will have to provide simulated proofs for the honest parties and still be able to extract from the proofs created by the adverary on behalf of corrupt players. We therefore need to use *simulation-extractable* ZK proofs (SE-ZK) [Gro06], instead of ordinary ZK proofs of knowledge (ZK-POKs). However, in the interest of clarity, we choose to present the construction above with ZK-POKs instead of SE-ZK. See Appendix C for a precise definition of simulation-extractable proofs.

[6]For any given family of C, $|C| = \text{poly}(\kappa)$, and thus, $\text{poly}(\kappa, \log(|C|)) = \text{poly}(\kappa)$; but the degree of this polynomial depends on the circuit family.

2. Use Micali's CS proofs [Mic94]. This reduces interaction to one round, but assumes a random oracle. It also relies on expensive PCPs.

3. Use the succint non-interactive arguments (SNARGs and SNARKs) of Bitansky et al. [BCCT12a, BCCT12b] or Goldwasser at al. [GLR11]. These are 1-round and hold in the standard model, but require a non-falsifiable assumption [Nao03].[7] Some variants rely on PCPs, PIR or FHE.

In case that the evaluation circuit is in logspace-uniform **NC**, we have another alternative:

4. Use the argument system of Goldwasser et al. [GKR08] for a 2-round solution. It relies on PIR.

In the case of arbitrary *nonuniform* poly(κ)-size circuits, one can use the technique of [BSCGT12, Section 5.4]. First, in a preparatory phase, the circuit C is written down and its collision-resistant hash digest d is computed by a trusted party or via an MPC protocol. Then, in step 3, the server proves the NP statement "there exists a circuit $\widetilde{C}$ whose digest is d and $c^* = \mathsf{Eval}(\widetilde{C}, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_N, \mathsf{pk}_N, \mathsf{ek}_N))$". This requires a succint argument system that is *proof of knowledge* and supports *nondeterministic* uniform circuits. This is satisfied by Micali's construction of CS proofs under Valiant's analysis [Mic94, Val08], and by SNARKs [BCCT12a, BCCT12b].

Verification for Large Inputs. We can make communication and verification complexities depend merely polylogarithmically on the size of the relevant inputs $x_1, \ldots, x_N$. In the aforementioned proofs of knowledge for nondeterministic statements [Mic94, Val08, BCCT12a, BCCT12b], the complexity depends polynomially on the size of statement being proven (expressed as a nondeterministic Turing machine and its input), but merely polylogarithmically on the size of the witness for the statement, and in particular, the nondeterministic choices made by the Turing machine. We thus move c_i from the instance into the witness. To recognize the correct c_i, each party P_i remembers the digest of c_i under a collision-resistant hash function family $\mathcal{H} = \{H_{\mathsf{hk}} : \{0,1\}^* \rightarrow \{0,1\}^\kappa\}$.

In the offline stage, every party P_i draws hash key hk_i and computes the digest $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})$, where π_i^{ENC} is the corresponding zero-knowledge proof of plaintext knowledge. P_i then sends $(c_i, \pi_i^{\text{ENC}}, \mathsf{hk}_i, d_i)$ to the cloud. Each party P_i remembers its own (hk_i, d_i) but can forget the potentially long $x_i, c_i, \pi_i^{\text{ENC}}$. In the online stage, the server broadcasts $(\mathsf{hk}_1, d_1), \ldots, (\mathsf{hk}_N, d_N)$ and proves the following NP statement: "there exist $\widetilde{c}_1, \widetilde{\pi}_1^{\text{ENC}}, \ldots, \widetilde{c}_N, \widetilde{\pi}_N^{\text{ENC}}$ such that $d_i = H_{\mathsf{hk}_i}(\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}})$ and $c^* = \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))$ and π_i^{ENC} is a valid proof".

This is secure, since whenever the server convinces the clients, it actually "knows" such $\widetilde{c}_1, \widetilde{\pi}_1^{\text{ENC}}, \ldots, \widetilde{c}_N, \widetilde{\pi}_N^{\text{ENC}}$ which can be efficiently extracted from the server (by the arguments' proof of knowledge property). For an honest party, the extracted $\widetilde{c}_i$ must be the one originally sent by the party (by the collision-resistance of H). For a corrupt party, the extracted $\widetilde{c}_i$ must be a valid ciphertext and its plaintext can be efficiently extracted from $\widetilde{\pi}_i^{\text{ENC}}$ (by the proof of knowledge property of $\widetilde{\pi}_i^{\text{ENC}}$).

We remark that the proceedings version of this work does not include the proof of plaintext knowledge π_i^{ENC} in the digest. Unfortunately, we do not know how to prove the resulting protocol secure if only the ciphertext c_i is included in the digest. This stems from the fact that each party chooses its own digest key hk_i, and thus we cannot guarantee that H_{hk_i} is collision-resistant if P_i

[7] A non-falsifiable assumption is necessary for the argument system to be non-interactive and secure in the standard model [GW11]. Note that we indeed require adaptive security, since the prover (ie. the server) is free to choose the statement to be proven (ie. the function to be computed).

is corrupt. This potentially enables a corrupt server to find a (possibly invalid) ciphertext $\widetilde{c}_i'$ and prove that $d_i = H_{\mathsf{hk}_i}(\widetilde{c}_i')$ and $c^* = \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \dots, (\widetilde{c}_i', \mathsf{pk}_i, \mathsf{ek}_i), \dots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))$. This defeats the purpose of having a proof of plaintext knowledge π_i^{ENC} in the first place, and in short, implies that a malicious party can potentially evaluate the joint function on an unknown input. Including the proof of knowledge π_i^{ENC} guarantees that this attack is not possible (see the security proof in Appendix D for more details).

Protocol for Malicious Adversaries. The protocol for fully malicious adversaries is given below. Let $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ be a multikey fully-homomorphic family of encryption schemes, and let $\mathcal{H} = \{H_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^\kappa\}$ be a family of collision-resistant hash functions. The following construction is an on-the-fly MPC protocol Π^{MAL} secure against malicious adversaries.

Step 1: For $i \in [U]$, party P_i samples a key tuple

$$(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) := \mathsf{Keygen}(1^\kappa \ ; \ r_i)$$

and encrypts its input x_i under pk_i:

$$c_i := \mathsf{Enc}(\mathsf{pk}_i, x_i \ ; \ s_i)$$

It computes zero-knowledge proofs of knowledge π_i^{GEN}, π_i^{ENC} showing it computed these steps correctly. The proofs $\pi_i^{\text{GEN}}, \pi_i^{\text{ENC}}$ attest to these relations.

$$\begin{aligned} R_{\text{GEN}} &= \{ \ (\ (\mathsf{pk}_i, \mathsf{ek}_i) \ , \ (\mathsf{sk}_i, r_i) \ ) \ \mid \ (\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) := \mathsf{Keygen}(1^\kappa \ ; \ r_i) \ \} \\ R_{\text{ENC}} &= \{ \ (\ (\mathsf{pk}_i, c_i) \ , \ (x_i, s_i) \ ) \ \mid \ c_i = \mathsf{Enc}(\mathsf{pk}_i, x_i \ ; \ s_i) \ \} \end{aligned}$$

It also samples a hash key hk_i and computes the digest of the ciphertext and the proof of plaintext knowledge

$$d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})$$

Party P_i sends the tuple $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, \pi_i^{\text{GEN}}, \pi_i^{\text{ENC}}, c_i, d_i)$ to the server S. The server verifies all proofs $\{\pi_i^{\text{GEN}}, \pi_i^{\text{ENC}}\}_{i \in [U]}$.

From this point forward, party P_i can forget its (potentially long) input x_i, ciphertext c_i, and proof π_i^{ENC}. It need only remember the hash key hk_i and digest d_i.

A function F, represented as a circuit C, is now selected on inputs $\{x_i\}_{i \in V}$ for some $V \subseteq U$. Let $N = |V|$. For ease of notation, we assume w.l.o.g. that $V = [N]$.

Step 2: The server S computes

$$c^* := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \dots, (c_N, \mathsf{pk}_N, \mathsf{ek}_N))$$

and a short argument φ proving that

$$\text{“}\exists \ (\widetilde{c}_1, \widetilde{\pi}_1^{\mathsf{Enc}}) \ , \ \dots \ , \ (\widetilde{c}_N, \widetilde{\pi}_N^{\mathsf{Enc}}) \ \text{ s.t. } \ d_i = H_{\mathsf{hk}_i}(\widetilde{c}_i, \widetilde{\pi}_i^{\mathsf{Enc}}) \ \text{ and } \ \mathsf{Verify}^{\text{ENC}}(\ (\mathsf{pk}_i, c_i) \ , \pi_i^{\mathsf{Enc}}) = 1 \\ \text{and } \ c^* = \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \dots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))\text{”}$$

It broadcasts (c^*, φ) to parties $P_1, \dots, P_N$, together with

$$\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}})\}_{i \in [N]}$$

Step 3: The parties $P_1, \ldots, P_N$ verify the argument φ and all proofs $\{\pi_i^{\text{GEN}}\}_{i\in[N]}$. The parties run an MPC protocol $\Pi_{\text{DEC}}{}^{\text{MAL}}$ to compute $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c^*)$.

Theorem 2.2. *Let $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ be a multikey fully-homomorphic encryption scheme, and let $\Pi_{\text{DEC}}{}^{\text{MAL}}$ be an N-party MPC protocol for computing the decryption function $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c^*)$. Let $\mathcal{H} = \{H_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^\kappa\}$ be a family of collision-resistant hash functions. If $\mathcal{E}$ is semantically secure, and $\Pi_{\text{DEC}}{}^{\text{MAL}}$ is secure against malicious adversaries corrupting $t < N$ parties, then the above construction is an on-the-fly MPC protocol secure against malicious adversaries corrupting t parties and possibly the server S.*

See Appendix D for the proof of Theorem 2.2.

3 Multikey Somewhat Homomorphic Encryption based on NTRU

We show how to construct a multikey somewhat homomorphic encryption scheme based on the NTRU encryption system first proposed by Hoffstein, Pipher, and Silverman [HPS98]. More precisely, we rely on a variant of the NTRU scheme proposed by Stehlé and Steinfeld[SS11].

In Section 3.1, we first review definitions and facts from the literature that we use extensively. In Section 3.2, we describe the encryption scheme. In Section 3.3, we discuss its security, and in Section 3.4 show that it is multikey somewhat homomorphic.

3.1 Preliminaries for the NTRU Instantiation

We work over rings $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$ and $R_q \doteq R/qR$ for some degree n integer polynomial $\phi(x) \in \mathbb{Z}[x]$ and a prime integer $q \in \mathbb{Z}$. Note that $R_q \equiv \mathbb{Z}_q[x]/\langle\phi(x)\rangle$, i.e., the ring of degree n polynomials modulo $\phi(x)$ with coefficients in $\mathbb{Z}_q$. Addition in these rings is done component-wise in their coefficients (thus, their additive group is isomorphic to $\mathbb{Z}^n$ and $\mathbb{Z}_q^n$ respectively). Multiplication is simply polynomial multiplication modulo $\phi(x)$ (and also q, in the case of the ring R_q). Thus an element in R (or R_q) can be viewed as a degree n polynomial over $\mathbb{Z}$ (or $\mathbb{Z}_q$). We represent such an element using the vector of its n coefficients, each in the range $\{-\lfloor\frac{q}{2}\rfloor, ..., \lfloor\frac{q}{2}\rfloor\}$. For an element $a(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1} \in R$, we let $\|a\|_\infty = \max|a_i|$ denote its ℓ_∞ norm.

In this work, we set $\phi(x) = x^n + 1$ to be the n^{th} cyclotomic polynomial, where n is a power of two. We use distributions over the ring $\mathbb{Z}[x]/\langle x^n+1\rangle$. To show the homomorphic properties of the scheme, the only property of these distributions we use is the magnitude of the polynomials' coefficients. Hence, we define a B-bounded distribution to be a distribution over R where the ℓ_∞-norm of a sample is bounded.

Definition 3.1. (B-Bounded Polynomial) *A polynomial $e \in R$ is called* B-bounded *if $\|e\|_\infty \leq B$.*

Definition 3.2. (B-Bounded Distribution) *A distribution ensemble $\{\chi_n\}_{n\in N}$, supported over R, is called* B-bounded *(for $B = B(n)$) if for all e in the suport of χ_n, we have $\|e\|_\infty < B$. In other words, a B-bounded distribution over R outputs a B-bounded polynomial.*

We present some elementary facts about the Gaussian distribution and multiplication over the ring $\mathbb{Z}[x]/\langle x^n+1\rangle$. The first fact shows that the discrete Gaussian distribution over $\mathbb{Z}^n$ with standard deviation r, denoted by $D_{\mathbb{Z}^n,r}$, outputs a $(r\sqrt{n})$-bounded polynomial with high probability.

This allows us to define a *truncated* Gaussian distribution that is $(r\sqrt{n})$-bounded and statistically close to the discrete Gaussian. The second says that multiplication in the ring $\mathbb{Z}[x]/\langle x^n+1\rangle$ increases the norm of the constituent elements only by a small amount.

Lemma 3.1 (see [MR07], Theorem 4.4)**.** *Let $n \in \mathbb{N}$. For any real number $r > \omega(\sqrt{\log n})$, we have*

$$\Pr_{x \leftarrow D_{\mathbb{Z}^n,r}}[||x|| > r\sqrt{n}] \leq 2^{-n+1}$$

Define the *truncated* discrete Gaussian distribution to be one that samples a polynomial according to the discrete Gaussian $D_{\mathbb{Z}^n,r}$ and repeat the sampling if the polynomial is not $(r\sqrt{n})$-bounded. By Lemma 3.1, this distribution is statistically close to the discrete Gaussian.

Lemma 3.2. (SEE [LM06, GEN09B]) *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and let $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$. For any $s, t \in R$,*

$$||s \cdot t \pmod{\phi(x)}|| \leq \sqrt{n} \cdot ||s|| \cdot ||t||$$
$$||s \cdot t \pmod{\phi(x)}||_\infty \leq n \cdot ||s||_\infty \cdot ||t||_\infty$$

Lemma 3.2 yields the following corollary.

Corollary 3.3. *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$. Let χ be a B-bounded distribution over the ring R and let $s_1, \ldots, s_k \leftarrow \chi$. Then $\prod_{i=1}^{k} s_i$ is $(n^{k-1}B^k)$-bounded.*

The Ring LWE Assumption. We describe a special case of the "ring learning with errors" assumption of Lyubaskevsky, Peikert and Regev [LPR10], which we will call RLWE. The RLWE assumption is analogous to the standard "learning with errors" (LWE) assumption, first defined by Regev [Reg05, Reg09] (generalizing the learning parity with noise assumption of Blum et al. [BFKL93]).

The $\mathsf{RLWE}_{\phi,q,\chi}$ assumption is that for a random ring element $s \leftarrow R_q$, given any polynomial number of samples of the form $(a_i, b_i = a_i \cdot s + e_i) \in (R_q)^2$, where a_i is uniformly random in R_q and e_i is drawn from the error distribution χ, the b_i's are computationally indistinguishable from uniform in R_q. We use the *Hermite normal form* of the assumption, as in [BV11b], where the secret s is sampled from the noise distribution χ rather than being uniform in R_q. This presentation is more useful for the purposes of this paper and it turns out to be equivalent to the original one up to obtaining one additional sample [ACPS09, LPR10].

Definition 3.3. (THE RLWE ASSUMPTION - HERMITE NORMAL FORM [BV11B, LPR10]) *For all $\kappa \in \mathbb{N}$, let $\phi(x) = \phi_\kappa(x) \in \mathbb{Z}[x]$ be a polynomial of degree $n = n(\kappa)$, let $q = q(\kappa) \in \mathbb{Z}$ be a prime integer, let the ring $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$ and $R_q \doteq R/qR$, and let χ denote a distribution over the ring R.*

The decisional ring LWE assumption $\mathsf{RLWE}_{\phi,q,\chi}$ *states that for any $\ell = \text{poly}(\kappa)$ it holds that*

$$\{(a_i, a_i \cdot s + e_i)\}_{i\in[\ell]} \overset{c}{\approx} \{(a_i, u_i)\}_{i\in[\ell]} \ ,$$

where s is sampled from the noise distribution χ, a_i are uniform in R_q, the "error polynomials" e_i are sampled from the error distribution χ, and finally, the ring elements u_i are uniformly random over R_q.

We use $\mathsf{RLWE}^{(\ell)}_{\phi,q,\chi}$ to denote the assumption when the number of samples ℓ is fixed.

Fact 3.4. *The* $\mathsf{RLWE}^{\ell}_{\phi,q,\chi}$ *assumption implies that,*

$$\{(a_i, a_i \cdot s + 2 \cdot e_i)\}_{i \in [\ell]} \overset{c}{\approx} \{(a_i, u_i)\}_{i \in [\ell]} \ .$$

where a_i, s, e_i and u_i are as in Definition 3.3.

Choice of Parameters. As already stated above, we will rely of the following specific choices of the polynomial $\phi(x)$ and the error distribution χ. For security parameter κ and a dimension parameter $n = n(\kappa)$ which is a power of two:

- We set $\phi(x)$ to be the n^{th} cyclotomic polynomial. This implies that $\phi(x) = x^n + 1$.
- The error distribution χ is the truncated discrete Gaussian distribution $D_{\mathbb{Z}^n,r}$ for standard deviation $r > 0$. A sample from this distribution is a $(r\sqrt{n})$-bounded polynomial $e \in R$.

The Worst-case to Average-case Connection. We state a worst-case to average-case reduction from the shortest vector problem on ideal lattices to the RLWE problem for our setting of parameters. The reduction stated below is a special case of the results of [LPR10].

Theorem 3.5 (A special case of [LPR10])**.** *Let $\Phi_n(x) = x^n + 1$ be the n^{th} cyclotomic polynomial where n is a power of two. Let $r \geq \omega(\sqrt{\log n})$ be a real number, and let $q \equiv 1 \pmod{2n}$ be a prime integer. Let $R \doteq \mathbb{Z}[x]/\langle x^n + 1\rangle$. Then there is a randomized reduction from $2^{\omega(\log n)} \cdot (q/r)$-approximate R-SVP to $\mathsf{RLWE}_{\phi,q,\chi}$ where $\chi = D_{\mathbb{Z}^n,r}$ is the discrete Gaussian distribution. The reduction runs in time* $\mathrm{poly}(n, q)$.

3.2 The Scheme

We describe the NTRU encryption scheme [HPS98], with the modifications proposed by Stehlé and Steinfeld [SS11]. For a security parameter κ, the scheme is parametrized by the following quantities:

- an integer $n = n(\kappa)$,
- a prime number $q = q(\kappa)$,
- a degree-n polynomial $\phi(x) = \phi_\kappa(x)$,
- a $B(\kappa)$-bounded error distribution $\chi = \chi(\kappa)$ over the ring $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$.

The parameters $n, q, \phi(x)$ and χ are public and we assume that given κ, there are polynomial-time algorithms that output n, q and $\phi(x)$, and sample from the error distribution χ. The message space is $\mathcal{M} = \{0, 1\}$, and all operations on ciphertexts are carried out in the ring R_q (i.e. modulo q and $\phi(x)$).

- $\mathsf{Keygen}(1^\kappa)$: Sample polynomials $f', g \leftarrow \chi$ and set $f := 2f' + 1$ so that $f \equiv 1 \pmod 2$. If f is not invertible in R_q, resample f'. Set

$$\mathsf{pk} := h = 2gf^{-1} \in R_q \quad , \quad \mathsf{sk} := f \in R$$

- $\mathsf{Enc}(\mathsf{pk}, m)$: Sample polynomials $s, e \leftarrow \chi$. Output the ciphertext

$$c := hs + 2e + m \in R_q$$

where all operations are done modulo q and $\phi(x)$.

- $\mathsf{Dec}(\mathsf{sk}, c)$: Let $\mu = fc \in R_q$. Output $m' := \mu \pmod 2$.

It is easily seen that this scheme is correct as long as there is no wrap-around modulo q. To decrypt a ciphertext c, we compute in R_q:

$$fc = fhs + 2fe + fm = 2gs + 2fe + fm$$

If there is no wrap-around modulo q then

$$fc \pmod 2 = 2gs + 2fe + fm \pmod 2 = fm \pmod 2 = m$$

One possible setting which ensures that there is no wrap-around modulo q is to set $\phi(x) = x^n+1$. To see why this helps, notice that since the coefficients of g, s, f, e are all bounded by $2B + 1$.[8] By Corollary 3.3, we know that the coefficients of $gs \pmod{x^n + 1}$ and $fe \pmod{x^n + 1}$ are both bounded by $n(2B+1)^2$. Thus, the coefficients of fc are bounded by $4n(2B+1)^2+B < 36nB^2 < q/2$.

In other words, as long as we set $q > 72nB^2$, a fresh ciphertext generated by Enc is guaranteed to decrypt correctly. From here on, we refer to $\mu = fc \in R_q$ as the "*error* in a ciphertext c".

For the rest of our exposition, we will use $\phi(x) = x^n + 1$ as our modulus polynomial.

3.3 Security

We base the security of the encryption scheme in Section 3.2 on two assumptions – the RLWE assumption described in Section 3.1, as well as a new assumption that we call the (Decisional) Small Polynomial Ratio (DSPR) Assumption. Towards this end, we define the following problem.

Definition 3.4. (Decisional Small Polynomial Ratio ($\mathsf{DSPR}_{\phi,q,\chi}$) Problem) *Let $\phi(x) \in \mathbb{Z}[x]$ be a polynomial of degree n, let $q \in \mathbb{Z}$ be a prime integer, and let χ denote a distribution over the ring $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$. The* (decisional) small polynomial ratio problem *$\mathsf{DSPR}_{\phi,q,\chi}$ is to distinguish between the following two distributions:*

- *a polynomial $h = g/f$, where f and g are sampled from the distribution χ (conditioned on f being invertible over $R_q = R/qR$), and*
- *a polynomial h sampled uniformly at random over R_q.*

Stehlé and Steinfeld [SS11] have shown that the $\mathsf{DSPR}_{\phi,q,\chi}$ problem is hard even for unbounded adversaries (namely, the two distributions above are statistically close) if n is a power of two, $\phi(x) = x^n + 1$ is the n^{th} cyclotomic polynomial, and χ is the discrete Gaussian $D_{\mathbb{Z}^n,r}$ for $r > \sqrt{q}\cdot\mathrm{poly}(n)$. This allowed them to prove semantic security for the modified NTRU scheme described in Section 3.2 under the $\mathsf{RLWE}_{\phi,q,\chi}$ assumption alone.

The security proof follows by a hybrid argument, in two steps.

[8] In fact, the coefficients of g, s and e are bounded by B and that of f is bounded by $2B + 1$.

1. The hardness of $\mathsf{DSPR}_{\phi,q,\chi}$ allows us to change the public key $h = 2g/f$ to $h = 2h'$ for a uniformly sampled h'.

2. Once this is done, we can use $\mathsf{RLWE}_{\phi,q,\chi}$ to change the challenge ciphertext $c^* = hs + 2e + m$ to $c^* = u + m$, where u is uniformly sampled from R_q.

   In this final hybrid, the advantage of the adversary is exactly $1/2$ since c^* is uniform over R_q, independent of the message m.

Unfortunately, their result holds only if $r > \sqrt{q} \cdot \mathrm{poly}(n)$, which is too large to permit even a single homomorphic multiplication. To support homomorphic operations, we need to use a much smaller value of r, for which it is easy to see that the $\mathsf{DSPR}_{\phi,q,\chi}$ assumption does not hold in a statistical sense any more. This makes it necessary for us to assume that the decisional small polynomial ratios problem is hard for our choice of parameters, which we refer to as the $\mathsf{DSPR}_{\phi,q,\chi}$ assumption.

Using the same security proof as in [SS11], we can base the security of the scheme in Section 3.2 on the DSPR assumption and the RLWE assumption. With the choice of parameters stated below, this yields security under the DSPR assumption and the hardness of approximating shortest vectors on ideal lattices to within a factor of 2^{n^ϵ}, which is believed to be hard.

Lemma 3.6. *Let n be a power of two, let $\phi(x) = x^n + 1$, let $q = 2^{n^\epsilon}$ for $\epsilon \in (0, 1)$ and $\chi = D_{\mathbb{Z}^n,r}$ with $r = \mathrm{poly}(n)$. Then, the (modified) NTRU encryption scheme described in Section 3.2 is secure under the $\mathsf{DSPR}_{\phi,q,\chi}$ assumption and the worst-case hardness of approximating shortest vectors on ideal lattices (over the ring $R \doteq \mathbb{Z}[x]/\langle\phi(x)\rangle$) to within a factor of $2^{\Omega(n^\epsilon)}$.*

3.4 Multikey Homomorphism

Let (h_1, f_1) and (h_2, f_2) be two different public/secret key pairs, and let $c_1 = h_1 s_1 + 2e_1 + m_1$ and $c_2 = h_2 s_2 + 2e_2 + m_2$ be two ciphertexts, encrypted under public keys h_1 and h_2, respectively. We show how to compute ciphertexts that decrypt to the sum and the product of m_1 and m_2. In particular, we show that the "ciphertexts" $c_{mult} \doteq c_1 \cdot c_2$ and $c_{add} \doteq c_1 + c_2$ can be decrypted to the product and the sum of m_1 and m_2 respectively, using the secret key $f_{12} \doteq f_1 \cdot f_2$.

To see this, note that as long as there is no wrap-around modulo q,

$$\begin{aligned} f_1 f_2(c_1 + c_2) \pmod 2 &= 2(f_1 f_2(e_1 + e_2) + f_2 g_1 s_1 + f_1 g_2 s_2) + f_1 f_2(m_1 + m_2) \pmod 2 \\ &= m_1 + m_2 \pmod 2 \end{aligned}$$

and

$$\begin{aligned} f_1 f_2(c_1 c_2) \pmod 2 &= 2(2g_1 g_2 s_1 s_2 + g_1 s_1 f_2(2e_2 + m_2) + g_2 s_2 f_1(2e_1 + m_1) + \\ &\quad f_1 f_2(e_1 m_2 + e_2 m_1 + 2e_1 e_2)) + f_1 f_2(m_1 m_2) \pmod 2 \\ &= m_1 m_2 \pmod 2 \end{aligned}$$

since $f_1 \equiv 1 \pmod 2$ and $f_2 \equiv 1 \pmod 2$. In other words, the "joint secret key" $f_{12} \doteq f_1 f_2$ can be used to decrypt $c_{\mathsf{add}} \doteq c_1 + c_2$ and $c_{\mathsf{mult}} \doteq c_1 \cdot c_2$. We can extend this argument to any combination of operations, with ciphertexts encrypted under multiple public keys.

Of course, the error in the ciphertexts will grow with the number of operations performed (as with all known fully homomorphic encryption schemes). Thus, correctness of decryption will only hold for a limited number of operations. We can show that the scheme can correctly evaluate circuits of depth roughly $\epsilon \log(n)$ when $q = 2^{n^\epsilon}$ and $B = \mathrm{poly}(n)$.

However, an astute reader would have observed by now that in order to successfully decrypt a ciphertext that was the result of a homomorphic evaluation, we must know the circuit that was evaluated. For example, to decrypt $c_1^2 + c_2$ we need to multiply it by $f_1^2 f_2$, whereas to decrypt $c_1 + c_2^2$ we need to multiply by $f_1 f_2^2$. This is obviously unsatisfactory.

Furthermore, consider what happens when we add or multiply two ciphertexts c, c' that are themselves a result of homomorphic evaluation. Suppose, for example, that $c = c_1 c_2$ and $c' = c_2 c_3$, where c_i is encrypted under h_i for $i \in \{1, 2, 3\}$. We know c can be decrypted using $f_1 f_2$ and c' can be decrypted using $f_2 f_3$. Thus, we know that

$$f_1 f_2 \cdot c = 2e + m$$
$$f_2 f_3 \cdot c' = 2e' + m'$$

for some messages m and m' and error terms e and e'. Following the discussion above, we can see that $c + c'$ can be decrypted using the key $f_1 f_2 f_3$. In general, given a ciphertext c encrypted under a set of keys K, and c' encrypted under a set of keys K', their sum can be decrypted using the product of the keys in $K \cup K'$. The absolute magnitude of the entries in this product grows exponentially with the number of keys in $K \cup K'$.

Analogously, in the context of homomorphic multiplication, we need $f_1 f_2^2 f_3$ to decrypt $c \cdot c'$. This hints at the fact that the size of the (joint) secret key needed to decrypt an evaluated ciphertext grows *exponentially* with the degree of the evaluated circuit (and not just with the number of parties involved, as in the case of addition). Indeed, after D multiplications, the (joint) secret key needed to decrypt will be the product of D polynomials, and the magnitude of the coefficients in this product will be exponential in D.

It is beneficial, especially for our constructions in Section 4 that we eliminate the exponential dependence (of the magnitude of the coefficients of the joint secret key) on the degree D. We remark that we will not succeed in eliminating the exponential dependence on N, the number of users – indeed, this is the reason why our solution will eventually assume an a-priori upper bound on N.

This motivates our use of *relinearization*, a technique first introduced by Brakerski and Vaikuntanathan [BV11a]. Informally, we will introduce a (public) evaluation key ek that will be output by the Keygen algorithm. Every time we multiply ciphertexts that share a key f_i, we will use the evaluation key to ensure that we only need f_i, and not f_i^2, to decrypt the new ciphertext. This ensures two things.

1. First, it ensures that to decrypt a ciphertext c^*, we only need to multiply it by *one* copy of each secret key, making decryption independent of the circuit used to produce c^*.

2. Second, it ensures that the size of the (joint) secret key needed to decrypt depends only on the number of keys N, and not on the circuit C that was evaluated.

We present below our modified scheme, as well as the Eval algorithm.

- $\mathsf{Keygen}(1^\kappa)$: Sample $f', g \leftarrow \chi$ and set $f := 2f' + 1$ so that $f \equiv 1 \pmod 2$. If f is not invertible in R_q, resample f'. Set

$$\mathsf{pk} := h = 2gf^{-1} \in R_q \quad , \quad \mathsf{sk} := f \in R$$

For all $\tau \in \{0, \ldots, \lfloor \log q \rfloor\}$, sample $s_\tau, e_\tau \leftarrow \chi$ and compute $\gamma_\tau = hs_\tau + 2e_\tau + 2^\tau f \in R_q$. Set

$$\mathsf{ek} = (\gamma_0, \ldots, \gamma_{\lfloor \log q \rfloor}) \in R_q^{\lceil \log q \rceil}$$

- $\mathsf{Enc}(\mathsf{pk}, m)$: Sample $s, e \leftarrow \chi$. Output the ciphertext $c := hs + 2e + m \in R_q$.
- $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$: Parse $\mathsf{sk}_i = f_i$ for $i \in [N]$. Let $\mu = f_1 \cdots f_N \cdot c \in R_q$. Output $m' := \mu$ (mod 2).
- $\mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_t, \mathsf{pk}_t, \mathsf{ek}_t))$: We show how to evaluate a t-input circuit C. To this end, we show how to homomorphically add and multiply two elements in $\{0, 1\}$. Given two ciphertexts c_1, c_2, we assume that we also have a set of distinct public keys associated with each ciphertext.[9] We will denote these lists by K_1, K_2, respectively. The public-key set of a fresh encryption is simply the set $\{\mathsf{pk}\}$ containing the public key under which it was encrypted.

  Given two ciphertexts c_1 and c_2 with corresponding public-key sets K_1 and K_2, output the ciphertext

  $$c_{\mathsf{add}} = c_1 + c_2 \in R_q$$

  as an encryption of the *sum* of the underlying messages. Output the set $K_{\mathsf{add}} = K_1 \cup K_2$ as its corresponding public-key set.

  Given two ciphertexts c_1 and c_2 with corresponding public-key sets K_1 and K_2, compute ciphertext $\widetilde{c}_0 = c_1 \cdot c_2 \in R_q$, and let $K_1 \cap K_2 = \{\mathsf{pk}_{i_1}, \ldots, \mathsf{pk}_{i_r}\}$.

  - If $K_1 \cap K_2 = \emptyset$, let $c_{\mathsf{mult}} = \widetilde{c}_0$.
  - Otherwise, for $j \in [r]$ and $\tau = \{0, \ldots, \lfloor \log q \rfloor\}$, define $\widetilde{c}_{j-1,\tau}$ so that

    $$\widetilde{c}_{j-1} = \sum_{\tau=0}^{\lfloor \log q \rfloor} \widetilde{c}_{j-1,\tau} 2^\tau$$

    is the binary representation of $\widetilde{c}_{j-1}$. Parse

    $$\mathsf{ek}_{i_j} = (\gamma_{i_j,0}, \ldots, \gamma_{i_j,\lfloor \log q \rfloor})$$

    and let

    $$\widetilde{c}_j = \sum_{\tau=0}^{\lfloor \log q \rfloor} \widetilde{c}_{j-1,\tau} \gamma_{i_j,\tau}$$

    At the end of the iteration, let $c_{\mathsf{mult}} = \widetilde{c}_r$.

  In either case, output c_{mult} as an encryption of the *product* of the underlying messages, and output the set $K_{\mathsf{mult}} = K_1 \cup K_2$ as its corresponding public-key set.

We first show that the scheme works correctly as advertised:

Lemma 3.7. *If $q = 2^{n^\epsilon}$ for $\epsilon \in (0, 1)$ and χ is a B-bounded distribution for $B = \mathrm{poly}(n)$, then the (modified) NTRU encryption scheme described above is multikey homomorphic for $N = O\left(n^\delta\right)$ keys and circuits of depth $d < (\epsilon - \delta) \log n - \log \log n - O(1)$.*

[9] That is, we assume each set contains distinct public keys, but the intersection of any two sets might not be empty.

Proof. The main step in the proof of correctness is to show that a homomorphic operation increases the error from η to at most $\eta^2+(B\cdot\text{poly}(n))^{O(N)}$. Putting this together with the fact that successful decryption requires the error to be smaller than $q/2$ gives us the statement of the lemma.

We first compute how much the error in a ciphertext grows with a homomorphic multiplication. Let c and c' be ciphertexts that decrypt to m and m', under two sets of secret keys K and K' respectively. Let f_K (resp. $f_{K'}$) denote the product $\prod_{i\in K} f_i$ (resp. $\prod_{i\in K'} f_i$). Then, we have:

$$\begin{aligned} f_K c &= 2e+m \\ f_{K'}c' &= 2e'+m' \end{aligned}$$

where $|2e+m|\le\eta$ and $|2e'+m'|\le\eta'$. Letting $c_{mult} \doteq cc'$, we have

$$(f_K f_{K'})\cdot c_{mult} = 2(2ee'+em'+e'm)+mm' \tag{1}$$

Thus, the error in the ciphertext c_{mult} is at most $\eta\eta'$, and it decrypts to the product of the two messages mm' as long as the error is not too large.

We now move to analyzing the additional error introduced by relinearization. Let $K\cap K' = \{i_1,\ldots,i_r\}$. Then, the "joint decryption key" $f_K f_{K'}$ contains the term $f_{i_1}^2\ldots f_{i_r}^2$. The goal of relinearization is to convert c_{mult} into a ciphertext that decrypts to the same message under the secret key

$$f_K f_{K'}\bigg(\prod_{j\in K\cap K'} f_j\bigg)^{-1}$$

which replaces the term $f_{i_1}^2\ldots f_{i_r}^2$ with the term $f_{i_1}\ldots f_{i_r}$.

Let $F_0 \doteq f_K f_{K'}$ and let $F_j \doteq F_{j-1}\cdot(f_{i_j})^{-1}$ for $j=1,\ldots,r$. Thus, F_r is a simple product of the secret keys f_i, without any quadratic terms.

We show that the ciphertext $\widetilde{c}_j$ decrypts to the message mm', and has error at most $\eta\eta' + j\cdot(8\log q(nB)^{2N+2})$. The base case (i.e., $j=0$) follows from Equation 1 since $\widetilde{c}_0 = c_{mult}$. In general, we have:

$$\begin{aligned} F_j\widetilde{c}_j &= (f_{i_j}F_j)\cdot(f_{i_j})^{-2}\cdot(f_{i_j}\widetilde{c}_j) \\ &= F_{j-1}\cdot(f_{i_j})^{-2}\cdot(f_{i_j}\widetilde{c}_j) \\ &= F_{j-1}\cdot(f_{i_j})^{-2}\cdot\bigg(\sum_{\tau=0}^{\lfloor\log q\rfloor}\widetilde{c}_{j-1,\tau}(f_{i_j}\gamma_{i_j,\tau})\bigg) \end{aligned} \tag{2}$$

Now,

$$f_{i_j}\gamma_{i_j,\tau} = 2(g_{i_j}s_{i_j,\tau}+f_{i_j}e_{i_j,\tau})+2^\tau f_{i_j}^2 = 2E_\tau + 2^\tau f_{i_j}^2$$

where $|E_\tau| \leq 2nB^2$. Substituting back into Equation 2, we get

$$\begin{aligned}
F_j \widetilde{c}_j &= F_{j-1} \cdot (f_{i_j})^{-2} \cdot \left(\sum_{\tau=0}^{\lfloor \log q \rfloor} \widetilde{c}_{j-1,\tau} (f_{i_j} \gamma_{i_j,\tau}) \right) \\
&= 2 \cdot \left(F_{j-1} f_{i_j}^{-2} \sum_{\tau=0}^{\lfloor \log q \rfloor} \widetilde{c}_{j-1,\tau} E_\tau \right) \\
&\qquad + F_{j-1} \sum_{\tau=0}^{\lfloor \log q \rfloor} \widetilde{c}_{j-1,\tau} 2^\tau \\
&= 2 \cdot \left(F_{j-1} f_{i_j}^{-2} \sum_{\tau=0}^{\lfloor \log q \rfloor} \widetilde{c}_{j-1,\tau} E_\tau \right) \\
&\qquad + F_{j-1} \widetilde{c}_{j-1}
\end{aligned}$$

Since $F_{j-1}\widetilde{c}_{j-1}$ is exactly mm' plus an even error (by the inductive assumption), this shows that $\widetilde{c}_j$ decrypts to mm' as well, under the secret key F_j.

It remains to compute the error in the ciphertext $\widetilde{c}_j$. Since $F_{j-1} f_{i_j}^{-2}$ is a product of at most $2N$ small polynomials, it has ℓ_∞ norm at most $(nB)^{2N}$, by Corollary 3.3. Thus, the error is at most

$$\begin{aligned}
\mathsf{error}_j &\leq 2 \cdot n \cdot (nB)^{2N} \cdot 2n(\lfloor \log q \rfloor + 1)B^2 + \mathsf{error}_{j-1} \\
&\leq 8 \log q \cdot (nB)^{2N+2} + \eta\eta' + (j-1) \cdot 8 \log q (nB)^{2N+2} \\
&\leq \eta\eta' + j \cdot (8 \log q (nB)^{2N+2})
\end{aligned}$$

Thus, the final error after a multiplication and relinearization is at most $\eta\eta' + 8N \log q \cdot (nB)^{2N+2} = \eta\eta' + (B \cdot n)^{O(N)}$, as claimed, for the setting of $q = 2^{n^\epsilon}$.

For a circuit of depth d and an initial error η_0, the error grows to at most

$$(\eta_0 \cdot B \cdot n)^{2^d \cdot O(N)}$$

after homomorphic evaluation. This results in correct decryption as long as $d < \log\log q - (\log\log n + \log N + O(1))$. In particular, for $N = O(n^\delta)$ keys and $q = 2^{n^\epsilon}$, we get $d < (\epsilon - \delta)\log n - \log\log n - O(1)$. $\square$

The main difference between the scheme in Section 3.2 and the one in this section in terms of security is in the evaluation key ek. The evaluation key contains the elements

$$\gamma_\tau \doteq hs_\tau + 2e_\tau + 2^\tau f$$

which can be thought of as "pseudo-encryptions" of multiples of the secret key f under the corresponding public key h. We point out that these are not true encryptions of the "message" $2^\tau f$ since $2^\tau f$ is not a binary polynomial, whereas our scheme is only equipped to correctly encrypt polynomials $m \in \mathbb{Z}_2[x]/\langle \phi(x) \rangle$.

The security of the scheme then relies on a "circular security" assumption which states that semantic security of the scheme is maintained given the evaluation key as constructed above. We remark that this assumption can be avoided at the cost of obtaining a leveled homomorphic encryption scheme (where the size of the evaluation key grows with the depth of the circuits that the scheme supports).

In Section 4 we show how to convert this somewhat homomorphic scheme into a multi-key fully homomorphic one. In the same section, we also discuss any additional assumptions we need to make to maintain security.

4 From Somewhat to Fully Homomorphic Encryption

We use Gentry's bootstrapping theorem [Gen09b, Gen09a] to convert a multikey somewhat homomorphic scheme into a multikey fully homomorphic one. Unfortunately, as we will see, we cannot apply the bootstrapping theorem directly to the somewhat homomorphic encryption scheme from Section 3. We therefore turn to modulus reduction, a technique introduced by [BV11a, BGV12], to convert our somewhat homomorphic scheme into a bootstrappable scheme. We first describe the bootstrapping theorem, and then present the modulus reduction technique and how we can apply it in our case.

4.1 Bootstrapping

We remind the reader of the definition of a bootstrappable encryption scheme and present Gentry's bootstrapping theorem [Gen09b, Gen09a] that states that a bootstrappable scheme can be converted into a fully homomorphic one. We modify the theorem and the corresponding definitions from their original presentation to generalize it to the multikey setting. Taking $N = 1$ yields the theorem and the definitions from [Gen09b, Gen09a].

Definition 4.1. (Bootstrappable Scheme) *Let $\mathcal{E} = \{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ be a family of multikey $\mathcal{C}$-homomorphic encryption schemes, and let f_{add} and f_{mult} be the the augmented decryption functions of the scheme defined as*

$$f_{\mathsf{add}}^{c_1,c_2}(\mathsf{sk}_1, \dots, \mathsf{sk}_N) = \mathsf{Dec}(\mathsf{sk}_1, \dots, \mathsf{sk}_N, c_1) \text{ } XOR \text{ } \mathsf{Dec}(\mathsf{sk}_1, \dots, \mathsf{sk}_N, c_2)$$

and

$$f_{\mathsf{mult}}^{c_1,c_2}(\mathsf{sk}_1, \dots, \mathsf{sk}_N) = \mathsf{Dec}(\mathsf{sk}_1, \dots, \mathsf{sk}_N, c_1) \text{ } AND \text{ } \mathsf{Dec}(\mathsf{sk}_1, \dots, \mathsf{sk}_N, c_2)$$

Then $\mathcal{E}$ is bootstrappable *if*

$$\left\{f_{\mathsf{add}}^{c_1,c_2}, f_{\mathsf{mult}}^{c_1,c_2}\right\}_{c_1,c_2} \subseteq \mathcal{C} \ .$$

Namely, the scheme can homomorphically evaluate f_{add} and f_{mult}.

We first define the notion of weak circular security, and then describe our generalization of Gentry's bootstrapping theorem.

Definition 4.2. (Weak Circular Security) *A public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is weakly circular secure if it is IND-CPA secure even for an adversary with auxiliary information containing encryptions of all secret key bits:* $\{\mathsf{Enc}(\mathsf{pk}, \mathsf{sk}[i])\}_i$.

Namely, no polynomial time adversary can distinguish an encryption of 0 from an encryption of 1 even given the additional information. We now state a generalization of Gentry's bootstrapping theorem to the multi-key setting.

Theorem 4.1. (Multikey Bootstrapping Theorem) *Let $\mathcal{E}$ be a bootstrappable family of multikey homomorphic schemes that is also weakly circular secure. Then there is a multikey fully homomorphic family $\mathcal{E}'$ of encryption schemes.*

The idea behind (multi-key) bootstrapping is that given a *public evaluation key* that consists of encryptions of all bits of all secret keys, $\alpha_{j,i} = \mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_j[i])$, we can evaluate circuits of any depth by periodically "refreshing" the evaluated ciphertext c. Simply evaluate the decryption circuit $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$ homomorphically using the evaluation key $\{\alpha_{j,i} = \mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_j[i])\}$. The result is a ciphertext c^* that encrypts the same plaintext as c and can again be decrypted using $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, but has much smaller noise. Thus, after this refreshing step, we can continue evaluating homomorphically for a few more levels before the noise grows again and we need to apply the refreshing procedure once more.

Unfortunately, the somewhat homomorphic scheme that we described in Section 3 is not bootstrappable. Recall that we can only evaluate circuits of depth less than $\epsilon \log(n)$, where $\epsilon < 1$. However, the shallowest implementation of the decryption circuit we are aware of has depth $c \log N \cdot \log n$ for some constant $c > 1$. We therefore turn to modulus reduction, which will allow us to convert our somewhat homomorphic encryption scheme into a bootstrappable scheme.

4.2 Modulus Reduction

Modulus reduction [BV11a, BGV12] is a noise-management technique that provides an *exponential* gain on the depth of the circuits that can be evaluated, while keeping the depth of the decryption circuit unchanged. Informally, if d_{dec} is the depth of the decryption circuit of the multikey scheme described in Section 3.4, then we modify the scheme so that its decryption circuit is unchanged but the scheme can now evaluate circuits of depth d_{dec}. Hence, the new scheme can evaluate its own decryption circuit, making it bootstrappable. Details follow.

We let $[\ \cdot\ ]_q$ denote the corresponding element in R_q (ie. reducing modulo $\phi(x)$ and q), as in [BGV12]. Then, if (h, f) is a key pair and c is a ciphertext under public key h, $[\ fc\ ]_q$ corresponds to the "noise" in c. Recall that for decryption to be successful, we need $[\ fc\ ]_q$ to be equal to fc over the integers. However, each homomorphic operation increases this noise. Modulus reduction allows us to keep the noise magnitude small by simply scaling the ciphertext after each operation. The key idea is to exploit the difference in how the noise affects security and homomorphisms:

- The growth of noise upon homomorphic multiplication is governed by the *magnitude* of the noise.
- Security is governed by the *ratio* between the magnitude of the noise and the modulus q.

This suggests a method of reducing the magnitude of the noise and the modulus by roughly the same factor, thus preserving security while at the same time making homomorphic multiplications "easier".

In particular, modulus reduction gives us a way to transform a ciphertext $c \in R_q$ into a different ciphertext $c' \in R_p$ (where p is smaller than q) while preserving correctness: for "joint" secret key $f = \prod_{i=1}^{N} f_i$,

$$[\ fc\ ]_p = [\ fc'\ ]_q \pmod 2$$

The transformation from c to c' involves simply scaling by (p/q) and rounding appropriately.

Lemma 4.2 ([BV11a, BGV12]). *Let p and q be two odd moduli, and let $c \in R_q$. Define c' to be the polynomial in R_p closest to $(p/q) \cdot c$ such that $c' \equiv c \pmod 2$. Then, for any f with $\|[\, fc \,]_q\|_\infty < q/2 - (q/p) \cdot \|f\|_1$, we have*

$$[\, fc' \,]_p = [\, fc \,]_q \pmod 2 \qquad \text{and}$$

$$\left\|[\, fc' \,]_p\right\|_\infty < (p/q) \cdot \|[\, fc \,]_q\|_\infty + \|f\|_1$$

where $\|f\|_\infty$ and $\|f\|_1$ are the ℓ_∞ and ℓ_1 norms of f, respectively.

Proof. Let $fc \pmod{x^n + 1} = \sum_{i=0}^{n-1} d_i x^i$, and consider a coefficient d_i. We know that there exists $k \in \mathbb{Z}$ such that:

$$d_i \pmod q = d_i - kq \in \left[-\frac{q}{2} + \frac{q}{p}\|f\|_1, \frac{q}{2} - \frac{q}{p}\|f\|_1\right],$$

so that

$$(p/q) \cdot d_i - kp \in \left[-\frac{p}{2} + \|f\|_1, \frac{p}{2} - \|f\|_1\right]$$

Let $fc' \pmod{x^n + 1} = \sum_{i=0}^{n-1} e_i x^i$. Then $-\|f\|_1 \leq (p/q) \cdot e_i - d_i \leq \|f\|_1$. Therefore,

$$e_i - kp \in \left[-\frac{p}{2}, \frac{p}{2}\right] \quad \text{and} \quad e_i \pmod p = e_i - kp$$

This proves the second part of the lemma. To prove the first part, notice that since p and q are both odd, we know $kp \equiv kq \pmod 2$. Moreover, we chose c' such that $c \equiv c' \pmod 2$. We thus have

$$\begin{gathered} e_i - kp \equiv d_i - kq \pmod 2 \\ (e_i \pmod p) \equiv (d_i \pmod q) \pmod 2 \\ [\, fc' \,]_p \equiv [\, fc \,]_q \pmod 2 \end{gathered}$$

□

The beauty of Lemma 4.2 is that if we know the depth d of the circuit we want to evaluate (in our case, $d = d_{\mathsf{dec}}$, the depth of the augmented decryption circuit), then we can construct a ladder of decreasing moduli $q_0, \ldots, q_d$ and perform modulus reduction after each operation so that at level i all ciphertexts reside in R_{q_i} and the magnitude of the noise at each level is small. In particular, this is true at level d so that once the evaluation is complete, it is possible to decrypt the resulting ciphertext without decryption errors.

Bootstrappable Scheme. We change the scheme so that it uses modulus reduction during the evaluation. Keygen will now sample a ladder of decreasing moduli $q_0, \ldots q_{d_{\mathsf{dec}}}$. We will choose the distribution χ in order to guarantee that any sample is B-bounded, where $B \ll q_{d_{\mathsf{dec}}}$. The modified scheme is as below.

- $\mathsf{Keygen}(1^\kappa)$: For every $i \in \{0, \ldots, d_{\mathsf{dec}}\}$, sample $g^{(i)}, u^{(i)} \leftarrow \chi$ and set $f^{(i)} := 2u^{(i)} + 1$ so that $f^{(i)} \equiv 1 \pmod 2$. If $f^{(i)}$ is not invertible in R_{q_i}, resample $u^{(i)}$. Let $h^{(i)} = 2g^{(i)}(f^{(i)})^{-1} \in R_{q_{i-1}}$, and set

$$\mathsf{pk} := h^{(0)} \in R_{q_0} \quad , \quad \mathsf{sk} := f^{(d_{\mathsf{dec}})} \in R_{q_{d_{\mathsf{dec}}}}$$

For all $i \in [d_{\mathsf{dec}}]$ and $\tau \in \{0, \dots, \lfloor \log q_{i-1} \rfloor\}$, sample $s_\tau^{(i)}, e_\tau^{(i)} \leftarrow \chi$ and compute

$$\gamma_\tau^{(i)} := h^{(i)} s_\tau^{(i)} + 2e_\tau^{(i)} + 2^\tau f^{(i-1)} \in R_{q_{i-1}}$$
$$\zeta_\tau^{(i)} := h^{(i)} s_\tau^{(i)} + 2e_\tau^{(i)} + 2^\tau (f^{(i-1)})^2 \in R_{q_{i-1}}$$

Set

$$\mathsf{ek} := \{\gamma_\tau^{(i)}, \zeta_\tau^{(i)}\}_{i\in[d_{\mathsf{dec}}], \tau \in \{0,\dots,\lfloor \log q_i \rfloor\}}$$

- $\mathsf{Enc}(\mathsf{pk}, m)$: Sample $s, e \leftarrow \chi$. Output the ciphertext $c := hs + 2e + m \in R_{q_0}$.
- $\mathsf{Dec}(\mathsf{sk}_1, \dots, \mathsf{sk}_N, c)$: Assume w.l.o.g. that $c \in R_{q_{d_{\mathsf{dec}}}}$. Parse $\mathsf{sk}_i = f_i$ for $i \in [N]$. Let $\mu := f_1 \cdots f_N \cdot c \in R_{q_{d_{\mathsf{dec}}}}$. Output $m' := \mu \pmod 2$.
- $\mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \dots, (c_t, \mathsf{pk}_t, \mathsf{ek}_t))$: We show how to evaluate a t-input circuit C. We assume w.l.o.g. that the circuit C is leveled in that it is composed of alternating XOR and AND levels.

  We show how to homomorphically add and multiply two elements in $\{0, 1\}$. Given two ciphertexts c_1, c_2, we assume that we also have a set of distinct public keys associated with each ciphertext.[10] We will denote these lists by K_1, K_2, respectively. The public-key set of a fresh encryption is simply the set $\{\mathsf{pk}\}$ containing the public key under which it was encrypted.

  – Given two ciphertexts $c_1, c_2 \in R_{q_i}$ with corresponding public-key sets K_1, K_2, compute $c = c_1 + c_2 \in R_{q_i}$ and let $K_1 \cup K_2 = \{\mathsf{pk}_{j_1}, \dots, \mathsf{pk}_{j_r}\}$. For $\ell = 1, \dots, r$ and $\tau \in \{0, \dots, \lfloor \log q_i \rfloor\}$, define $\widetilde{c}_{\ell-1,\tau}$ so that

  $$\widetilde{c}_{\ell-1} = \sum_{\tau=0}^{\lfloor \log q_i \rfloor} \widetilde{c}_{\ell-1,\tau} 2^\tau$$

  is the binary representation of $\widetilde{c}_{\ell-1}$. Parse

  $$\mathsf{ek}_{j_\ell} = \{\gamma_{j_\ell,\tau}^{(i)}, \zeta_{j_\ell,\tau}^{(i)}\}_{i\in[d_{\mathsf{dec}}], \tau \in \{0,\dots,\lfloor \log q_i \rfloor\}}$$

  Let

  $$\widetilde{c}_\ell := \sum_{\tau=0}^{\lfloor \log q_i \rfloor} \widetilde{c}_{\ell-1} \gamma_{j_\ell,\tau}^{(i)} \in R_{q_i}$$

  Finally, reduce the modulus: let c_{add} be the integer vector closest to $(q_{i+1}/q_i) \cdot \widetilde{c}_r$ such that $c_{\mathsf{add}} \equiv \widetilde{c}_r \pmod 2$. Output $c_{\mathsf{add}} \in R_{q_{i+1}}$ as an encryption of the *sum* of the underlying messages. Output the set $K_{\mathsf{add}} := K_1 \cup K_2$ as its corresponding public-key set.

  – Given two ciphertexts $c_1, c_2 \in R_{q_i}$ with corresponding public-key sets K_1, K_2, compute ciphertext $\widetilde{c}_0 = c_1 \cdot c_2 \in R_{q_i}$, and let $K_1 \cup K_2 = \{\mathsf{pk}_{j_1}, \dots, \mathsf{pk}_{j_r}\}$. For $\ell = 1, \dots, r$ and $\tau \in \{0, \dots, \lfloor \log q_i \rfloor\}$, define $\widetilde{c}_{\ell-1,\tau}$ so that

  $$\widetilde{c}_{\ell-1} = \sum_{\tau=0}^{\lfloor \log q_i \rfloor} \widetilde{c}_{\ell-1,\tau} 2^\tau$$

[10] That is, we assume each set contains distinct public keys, but the intersection of any two sets might not be empty.

is the binary representation of $\widetilde{c}_{\ell-1}$. Parse

$$\mathsf{ek}_{j_\ell} = \{\gamma^{(i)}_{j_\ell,\tau}, \zeta^{(i)}_{j_\ell,\tau}\}_{i\in[d_{\mathsf{dec}}],\tau\in\{0,\dots,\lfloor\log q_i\rfloor\}}$$

If $\mathsf{pk}_{j_\ell} \in K_1 \cap K_2$, let

$$\widetilde{c}_\ell := \sum_{\tau=0}^{\lfloor\log q_i\rfloor} \widetilde{c}_{\ell-1}\zeta^{(i+1)}_{j_\ell,\tau} \in R_{q_i}$$

Otherwise, let

$$\widetilde{c}_\ell := \sum_{\tau=0}^{\lfloor\log q_i\rfloor} \widetilde{c}_{\ell-1}\gamma^{(i+1)}_{j_\ell,\tau} \in R_{q_i}$$

Finally, reduce the modulus. Let c_{mult} be the integer vector closest to $(q_{i+1}/q_i)\cdot\widetilde{c}_r$ such that $c_{\mathsf{mult}} \equiv \widetilde{c}_r \pmod 2$. Output $c_{\mathsf{mult}} \in R_{q_{i+1}}$ as an encryption of the *product* of the underlying messages, and output the set $K_{\mathsf{mult}} := K_1 \cup K_2$ as its corresponding public-key set.

We can now show the following lemma, whose proof is deferred to the full version.

Lemma 4.3. *If $q = 2^{n^\epsilon}$ for $\epsilon \in (0,1)$ and χ is a B-bounded distribution for $B = \mathrm{poly}(n)$, then the (modified) NTRU encryption scheme described above is multikey homomorphic for N keys and circuits of depth d as long as $Nd = O(n^\epsilon/\log n)$.*

Multikey Fully Homomorphic Encryption. To turn this into a fully homomorphic encryption scheme, we use the (multi-key) bootstrapping theorem (Theorem 4.1), but first, we show an upper bound on the depth of the decryption circuit.

Lemma 4.4. *The decryption circuit for the scheme above for N keys can be implemented as a polynomial-size arithmetic circuit over $GF(2)$ of depth $O(\log N(\log\log q + \log n))$.*

Proof. The decryption circuit for a ciphertext encrypted under N keys can be written as

$$\mathsf{Dec}_{f_1,\dots,f_N}(c) = c \cdot \prod_{i=1}^{N} f_i$$

Multiplying two polynomials over R_q can be done using a polynomial-size Boolean circuit of depth $O(\log\log q + \log n)$ (see, e.g., [BV11a, Lemma 4.5] for a proof). Using a binary tree of polynomial multiplications, the decryption operation above can then be done in depth $O(\log N(\log\log q + \log n))$, as claimed. □

This means that the modified scheme is bootstrappable, and therefore applying the bootstrapping theorem gives us:

Theorem 4.5. *For every $N \in \mathbb{N}$, let $B = \mathrm{poly}(n)$, χ to be a B-bounded distribution, and $q = 2^{\Omega(N\log N\cdot\log^2 n)}$. Then, there exists a multikey fully homomorphic encryption scheme for N keys, secure under the $\mathsf{DSPR}_{\phi,q,\chi}$ and $\mathsf{RLWE}_{\phi,q,\chi}$ assumptions.*

Proof. To apply Theorem 4.1, we require that the depth of the decryption circuit is smaller than the depth of the circuits that the scheme can evaluate. That is,

$$\log N \cdot (\log\log q + \log n) < C \cdot \frac{\log q}{N \cdot \log n}$$

for some universal constant $C > 0$, which holds for the parameter settings in the statement of the theorem. $\square$

In particular, this scheme maintains security as long as $q = 2^{n^{1-\delta}}$ for some $\delta > 0$, thus supporting as many as $N = n^{1-\delta}/\log^{O(1)} n$ users.

Finally, we remark that bootstrapping (and therefore assuming weak circular security) can be avoided at the cost of obtaining a leveled homomorphic encryption scheme (where the size of the evaluation key grows with the depth of the circuits that the scheme supports).

5 Acknowledgements

This work was supported by the Check Point Institute for Information Security, the Israeli Centers of Research Excellence (I-CORE) program (center No. 4/11), an NSERC Discovery Grant, and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

We thank Craig Gentry for pointing out an error in our initial construction. We also thank an anonymous STOC reviewer for suggesting the construction described in Appendix A of multikey FHE for $N = O(1)$ parties from FHE .

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2010.

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multi-party computation with low communication, computation and interaction from threshold fhe. In *EUROCRYPT*, 2012.

[BCCT12a] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.

[BCCT12b] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. *Cryptology ePrint Archive: Report 2012/095*, 2012.

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BSCGT12] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fully homomorphic encryption without modulus switching from classical gapsvp. *Cryptology ePrint Archive: Report 2012/078*, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In Ostrovsky [Ost11], pages 97–106.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Rogaway [Rog11], pages 505–524.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001.

[CKV10] Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Rabin [Rab10], pages 483–501.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010.

[DIK$^+$08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, 2008.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Gilbert [Gil10], pages 445–465.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 1997.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Rabin [Rab10], pages 465–482.

[GH11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [Ost11], pages 107–109.

[GHL$^+$11] Craig Gentry, Shai Halevi, Vadim Lyubashevsky, Christopher Peikert, Joseph Silverman, and Nigel Smart. Personal communication, 2011.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *STOC*, pages 113–122. ACM, 2008.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier cs-proofs. *Cryptology ePrint Archive: Report 2011/456*, 2011.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[Gro06] Jens Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 99–108. ACM, 2011.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Rogaway [Rog11], pages 132–150.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992.

[Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 1995.

[KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. `http://eprint.iacr.org/`.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2006.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23.

[Mic94] Silvio Micali. Cs proofs (extended abstracts). In *FOCS*, pages 436–453. IEEE, 1994.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[Ost11] Rafail Ostrovsky, editor. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011.* IEEE, 2011.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[Rog11] Phillip Rogaway, editor. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*. Springer, 2011.

[RTSS09] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, pages 199–212, 2009.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[SS11] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47. Springer, 2011.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography*, pages 420–443, 2010.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43.

[vDJ10] Marten van Dijk and Ari Juels. On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Proceedings of the 5th USENIX conference on Hot topics in security*, HotSec'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

A Multikey Properties of Other FHE Schemes

A.1 A Generic Construction

It turns out that any FHE scheme is inherently multikey for a constant number of parties, $N = O(1)$. This can be seen from the following construction.

Let $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme with plaintext space $\{0,1\}$, and ciphertext space $\{0,1\}^{p(\kappa)}$ for some polynomial $p(\cdot)$. For $c \in \{0,1\}^*$, we overload notation and let $\mathsf{Enc}(\mathsf{pk}, c)$ denote the bit-wise encryption of c. Let $c_1 = \mathsf{Enc}(\mathsf{pk}_1, m_1), \ldots, c_N = \mathsf{Enc}(\mathsf{pk}_N, m_N)$ be a set of ciphertexts, each encrypting a bit message under a different key. From $c_1, \ldots, c_N$, it is possible to construct the following ciphetexts using Enc and Eval:

$$c'_1 = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{Enc}(\mathsf{pk}_2, \mathsf{Enc}(\ldots, \mathsf{Enc}(\mathsf{pk}_N, m_1))\ldots))$$

$$c'_2 = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{Enc}(\mathsf{pk}_2, \mathsf{Enc}(\ldots, \mathsf{Enc}(\mathsf{pk}_N, m_2))\ldots))$$

$$\vdots$$

$$c'_N = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{Enc}(\mathsf{pk}_2, \mathsf{Enc}(\ldots, \mathsf{Enc}(\mathsf{pk}_N, m_N))\ldots))$$

Using Eval once again, it is possible to obtain the ciphertext

$$c* = \mathsf{Enc}(\mathsf{pk}_1, \mathsf{Enc}(\mathsf{pk}_2, \mathsf{Enc}(\ldots, \mathsf{Enc}(\mathsf{pk}_N, C(m_1, \ldots, m_N)))\ldots))$$

It is then possible to decrypt c^* using $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$. However, note that the size of c'_i is $p(\kappa)^N$. This means that we must have $N = O(1)$, and thus can only obtain this generic construction of multikey FHE from (regular) FHE for a constant number of parties.

A.2 From Ring-LWE

The FHE scheme of Brakerski and Vaikuntanathan [BV11b] based on the Ring-LWE assumption can be made multikey for $N = O(\log(\kappa))$ parties by using relinearization and modulus reduction [BV11a, BGV12]). Indeed, it is possible to show that in this new scheme, the size of the ciphertext (as number of elements in R_q) increases to at most 2^N when evaluating on ciphertexts encrypted under at most N keys. Thus, it is able to handle a logarithmic number of parties, $N = O(\log(\kappa))$.

B Impossibility of a 2-Round Protocol

In this section, we prove the impossibility of a 2-round on-the-fly MPC protocol. We show that if such a protocol securely computes a function f, then a certain class of functions can be obfuscated. This has a similiar flavor to a recent result of Van Dijk and Jules [vDJ10]. We review the definition of obfuscation from [BGI$^+$01].

Definition B.1. *A probabilistic algorithm $\mathcal{O}$ is a* (circuit) obfuscator *if the following three conditions hold:*

Functionality: *For every circuit C, the string $\mathcal{O}(C)$ describes a circuit that computes the same function as C.*

Polynomial Slowdown: *There is a polynomial p such that for every circuit C, $|\mathcal{O}(C)| \le p(|C|)$.*

"Virtual Black-Box" Property: *For any PPT adversary $\mathcal{A}$, there is a PPT simulator $\mathcal{S}$ such that for all circuits C*

$$\left|\Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - \Pr[\mathcal{S}^C(1^{|C|}) = 1]\right| \le \text{negl}(|C|)$$

Barak et. al. [BGI$^+$01] show that assuming one-way functions exist, there does not exist any algorithm $\mathcal{O}$ satisfying Definition B.1, even if we do not require that $\mathcal{O}$ run in polynomial time . Thus, we conclude that assuming one-way functions exist, a 2-round on-the-fly MPC protocol is impossible.

We now show the connection between on-the-fly MPC and obfuscation. For ease of notation, w.l.o.g. assume that the N parties whose input will be used in the computation are parties $\{1, \ldots, N\}$. Such a protocol can be modeled by $U + N + 1$ (possibly randomized) algorithms: $\mathsf{In}_1, \ldots, \mathsf{In}_U, \mathsf{ComputeF}, \mathsf{Out}_1, \ldots, \mathsf{Out}_N$, where:

- $(d_i, c_i) \leftarrow \mathsf{In}_i(x_i)$: On input x_i, the algorithm In_i outputs two elements, c_i to be sent to the server S and d_i to be kept by party P_i.
- $(z_1, \ldots, z_N) \leftarrow \mathsf{ComputeF}(c_1, \ldots, c_N)$: On input $c_1, \ldots, c_N$, which are the messages the server received from parties $P_1, \ldots, P_N$, $\mathsf{ComputeF}$ outputs N elements $z_1, \ldots, z_N$. The server sends back z_i to party P_i.
- $y_i \leftarrow \mathsf{Out}_i(z_i, d_i)$: On input z_i which was received from the server, and the auxiliary information d_i output by In_i, Out_i computes the output y_i.

In what follows, we use the following notation. For a set T and a vector $\vec{x} = (x_1, \ldots, x_n)$, we let $\vec{x}_T = (x_i)_{i \in T}$.

Theorem B.1. *Let f be an N-input function. If a 2-round on-the-fly MPC protocol* $\Pi = (\mathsf{In}_1, \ldots, \mathsf{In}_U, \mathsf{ComputeF}, \mathsf{Out}_1, \ldots, \mathsf{Out}_N)$ *securely computes f against a semi-honest adversary corrupting $t < N$ parties, then for any $\vec{x} \in (\{0,1\}^*)^N$ and any $H \subset [N]$ such that $|H| = N - t$, there exists an obfuscator $\mathcal{O}_{\vec{x}_H}$ for $f_{\vec{x}_H}$, where $f_{\vec{x}_H}$ is the t-input function defined by f with fixed inputs x_i for $i \in H$, and restricted to outputs y_j for $j \in [N] \backslash H$.*

Proof of Theorem B.1: Fix $\vec{x} \in (\{0,1\}^*)^N$ and $H \subset [N]$ such that $|H| = N - t$. Consider an execution of Π where $\vec{x}$ is the input vector. We can build an obfuscator $\mathcal{O}_{\vec{x}_H}$ for $f_{\vec{x}_H}$ as follows: compute $(d_i, c_i) \leftarrow \mathsf{In}_i(x_i)$ for $i \in H$ and sample random coins $r_S, \{r_i, s_i\}_{i \in [N] \backslash H}$. $\mathcal{O}_{\vec{x}_H}$ outputs the circuit that on input $\vec{x}_C$:

- For $j \in C$, computes $(c_j, d_j) := \mathsf{In}_j(x_j\ ; r_j)$.
- Computes $(z_1, \ldots, z_N) := \mathsf{ComputeF}(c_1, \ldots, c_N\ ; r_S)$
- For $j \in C$, computes $y_j := \mathsf{Out}_j(z_j\ ; s_j)$.
- Outputs $(y_j)_{j \in C}$.

Notice that $\mathcal{O}_{\vec{x}_H}$ can be considered as a semi-honest adversary $\mathcal{A}_\Pi$ attacking Π by corrupting the server S and the parties in $C = [N] \backslash H$ (ie. the parties in H are honest). For the sake of simplicity, assume $\mathcal{A}_\Pi$ outputs its entire view. Then by the security of Π, we know that the output of $\mathcal{A}_\Pi$ is computationally indistinguishable from the output of an adversary $\mathcal{S}_\Pi$ corrupting the same parties in the ideal model. More formally, for all $\vec{x}$ and any predicate D, we have

$$|\Pr[D(\mathrm{REAL}_{\Pi,\mathcal{A}}(\vec{x})) = 1] - \Pr[D(\mathrm{IDEAL}_{f,\mathcal{S}_\Pi}(\vec{x})) = 1]| = \mathrm{negl}(\kappa)$$

In our case, $\mathrm{REAL}_{\Pi,\mathcal{A}}(\vec{x}) = (\vec{x}_C, \mathcal{O}_{\vec{x}_H}(\vec{x}_C))$ and $\mathrm{IDEAL}_{f,\mathcal{S}_\Pi}(\vec{x}) = \mathcal{S}(\vec{x}_C, \vec{y}_C)$, so we have:

$$|\Pr[D(\vec{x}_C, \mathcal{O}_{\vec{x}_H}(\vec{x}_C)) = 1] - \Pr[D(\mathcal{S}(\vec{x}_C, \vec{y}_C)) = 1]| = \mathrm{negl}(\kappa) \tag{3}$$

Because this holds for all $\vec{x}$ (and in particular for all $\vec{x}_C$), then (3) holds even if D is allowed to chose $\vec{x}_C$. Now consider an adversary $\mathcal{A}_\mathcal{O}$ attacking the obfuscation of $f_{\vec{x}_H}(\cdot)$. Then we can substitute D in (3) with $\mathcal{A}_\mathcal{O}$, and because $\mathcal{A}_\mathcal{O}$ can only run $\mathcal{O}_{\vec{x}_H}(\cdot)$ on a polynomial number of inputs, then by a hybrid argument we have that for all $\mathcal{A}_\mathcal{O}$ there exists $\mathcal{S}_\mathcal{O}$ (namely $\mathcal{A}_\mathcal{O}(\mathcal{S}(\cdot))$ such that:

$$|\Pr[\mathcal{A}_\mathcal{O}(\mathcal{O}_{\vec{x}_H}(\cdot)) = 1] - \Pr[\mathcal{S}_\mathcal{O}^{f_{\vec{x}_H}(\cdot)}(1^\kappa) = 1]| = \mathrm{negl}(\kappa)$$

□

C Simulation Extractability

For completeness, we include the definition of simulation-extractability [Gro06].

Definition C.1 (Simulation Extractability [Gro06])**.** *Let* $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a proof system for a relation R, satisfying the completeness, soundness and zero-knowledge properties. We say that* Π *is* simulation-extractable *if:*

- Setup *also outputs an extraction key* xk.
- *There exists a PPT algorithm* $\mathsf{Ext}(y, \varphi, \mathsf{xk})$ *such that for all P^* we have the probability that P^* wins in the following game in negligible in κ:*
  1. **Key Generation:** *The challenger runs* $(\mathsf{pp}, \mathsf{xk}) \leftarrow \mathsf{Setup}(1^\kappa)$ *and gives* pp *to* P^*.
  2. **Simulation queries:** ${P^*}^{\mathcal{S}(\cdot)}$ *is given access to the simulation oracle $\mathcal{S}(\cdot)$, which it can adaptively access.*
  3. **Adversary Output:** P^* *outputs a tuple* (y^*, φ^*).
  4. **Extraction:** *The challenger runs* $x^* \leftarrow \mathtt{Ext}(y^*, \varphi^*, \mathsf{ek})$.
  5. P^* *wins if (a) y^* was not part of a simulator query, (b)* $\mathtt{Verify}(y^*, \varphi^*) = 1$*, and (c)* $R(y, x^*) = 0$.

Katz and Vaikuntanathan [KV09] implicitly construct simulation-extractable proofs for all of NP from CPA encryption and simulation-sound proofs [Sah99]. See [DHLW10] for a more detailed exposition.

D Proofs

D.1 Security against Semi-Honest Adversaries

Proof of Theorem 2.1: We prove that the protocol is correct and secure, and that it satisfies the performance requirements of an *on-the-fly* protocol.

Correctness: Correctness follows directly from the correctness properties of homomorphic evaluation and the decryption MPC protocol.

Performance: By compactness of evaluation, we know that c is independent of $|C|$. This means that the communication complexity and the computation time of the parties is independent of $|C|$.

Security: Recall that for security, we only need to consider adversaries corrupting a subset T of the parties $P_1, \ldots, P_N$ involved in the computation. We let $\overline{T} = [N] \backslash T$. For a semi-honest adversary $\mathcal{A}^{\text{SH}}$ corrupting $t < N$ parties, we construct a simulator $\mathcal{S}^{\text{SH}}$ corrupting the same parties in the ideal world. We show the case when the server S is corrupted (the other case is analogous).

Simulator $\mathcal{S}^{\mathbf{sh}}$: Runs $\mathcal{A}^{\text{SH}}$ on input $\{x_i\}_{i \in T}$.

Step 1: For non-computing parties $i \in \{N+1, \ldots, U\}$ and for honest parties $i \in \overline{T}$, $\mathcal{S}^{\text{SH}}$ computes $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ honestly and computes $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0)$. For each party P_i, $\mathcal{S}^{\text{SH}}$ sends $(c_i, \mathsf{pk}_i, \mathsf{ek}_i)$ to $\mathcal{A}^{\text{SH}}$ on behalf of P_i.

Step 2: Receive c^* from $\mathcal{A}^{\text{SH}}$.

Step 3: Gives $\{x_i\}_{i \in T}$ to the ideal functionality and receives $y = f(x_1, \ldots, x_N)$. Runs the simulator $\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}$ (interacting with $\mathcal{A}^{\text{SH}}$) on input y.

$\mathcal{S}^{\text{SH}}$ outputs whatever $\mathcal{A}^{\text{SH}}$ outputs.

We prove that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{SH}}}(\vec{x}) \overset{c}{\approx} \text{REAL}_{\Pi^{\text{SH}}, \mathcal{A}^{\text{SH}}}(\vec{x})$. Since $\mathcal{A}^{\text{SH}}$ is semi-honest, correctness guarantees that the outputs of the honest parties does not change. Furthermore, we prove that the view created by $\mathcal{S}^{\text{SH}}$ for $\mathcal{A}^{\text{SH}}$ is indistinguishable from the view of $\mathcal{A}^{\text{SH}}$ in a real world execution, and thus its output is indistinguishable in both cases. We prove this using a hybrid argument.

Hybrid 0: This is the view of $\mathcal{A}^{\text{SH}}$ in a real-world execution. We have:

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\Pi_{\text{DEC}}}(\mathcal{A}^{\text{SH}})$$

Hybrid 1: We change how Step 3 is performed. Instead of executing Π_{DEC}, we run the simulator $\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}$ (interacting with $\mathcal{A}^{\text{SH}}$) on input $y = f(x_1, \ldots, x_N)$. We now have:

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}(y)}(\mathcal{A}^{\text{SH}})$$

We claim that the view of $\mathcal{A}^{\text{SH}}$ in Hybrid 0 is computationally indistinguishable from its view in Hybrid 1 by the *security of* Π_{DEC}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 0 and 1. We construct an adversary $\mathcal{B}$ that breaks the security of Π_{DEC}. The adversary $\mathcal{B}$ works as follows:

1. For all $i \in \overline{T}$, sample $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and compute $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$ honestly. For all $i \in \overline{T}$, give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i)$ to $\mathcal{A}^{\text{SH}}$ on behalf of P_i for all $i \in \overline{T}$. Receive c^* from $\mathcal{A}^{\text{SH}}$.
2. Receive the challenge view $\mathsf{View}^*(\mathcal{A}^{\text{SH}})$, and send the entire view of $\mathcal{A}^{\text{SH}}$ to $\mathcal{D}$.
3. Output the bit output by $\mathcal{D}$.

When $\mathsf{View}^*(\mathcal{A}^{\text{SH}}) = \mathsf{View}_{\Pi_{\text{DEC}}}(\mathcal{A}^{\text{SH}})$, $\mathcal{B}$ perfectly emulates Hybrid 0, whereas if $\mathsf{View}^*(\mathcal{A}^{\text{SH}}) = \mathsf{View}_{\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}(y)}(\mathcal{A}^{\text{SH}})$, $\mathcal{B}$ perfectly emulates Hybrid 1. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 0 and 1, then $\mathcal{B}$ can distinguish between $\mathsf{View}_{\Pi_{\text{DEC}}}(\mathcal{A}^{\text{SH}})$ and $\mathsf{View}_{\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}(y)}(\mathcal{A}^{\text{SH}})$, contradicting the security of Π_{DEC}.

Hybrids $2.k$ **for** $k = 1, \ldots, N-t$**:** Let $\overline{T} = \{i_1, \ldots, i_{N-t}\}$. In Hybrid $2.k$ we change c_{i_k} so that instead of encrypting x_{i_k} it now encrypts 0. More formally, in Hybrid $2.k$ we have $c_{i_j} = \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)$ for $j \leq k$ and $c_{i_j} = \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})$ for $j > k$.

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}(y)}(\mathcal{A}^{\text{SH}})$$
$$\{c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)\}_{j \leq k} \quad , \quad \{c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})\}_{j > k}$$

For ease of notation we let Hybrid 1 be Hybrid 2.0. We claim that the view of $\mathcal{A}^{\text{SH}}$ in Hybrid $2.k$ is indistinguishable from its view in Hybrid $2.(k-1)$ by the *semantic security of* $\mathcal{E}$ under public key pk_{i_k}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids $2.k$ and $2.(k-1)$. We construct an adversary $\mathcal{B}$ that breaks the semantic security of $\mathcal{E}$ under public key pk_{i_k}. The adversary $\mathcal{B}$ works as follows:

1. Receive $(\mathsf{pk}, \mathsf{ek})$ from the semantic security challenger and set $\mathsf{pk}_{i_k} = \mathsf{pk}$ and $\mathsf{ek}_{i_k} = \mathsf{ek}$. Give $m_0 = 0$ and $m_1 = x_{i_k}$ to the challenger and receive $c = \mathsf{Enc}(\mathsf{pk}, m_b)$. Set $c_{i_k} = c$. For all $i \in \overline{T}, i \neq i_k$, compute $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ honestly. For $j < k$, compute $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)$ and for $j > k$, compute $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})$. For all $i \in \overline{T}$ give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i)$ to $\mathcal{A}^{\text{SH}}$ on behalf of P_i. Receive c^* from $\mathcal{A}^{\text{SH}}$.
2. Obtain y from the ideal functionality, run the simulator $\mathcal{S}^{\text{SH}}_{\Pi_{\text{DEC}}}(y)(\mathcal{A}^{\text{SH}})$. Give $\mathcal{D}$ the resulting view.
3. Output the bit output by $\mathcal{D}$.

When $b = 0$, $\mathcal{B}$ perfectly emulates Hybrid $2.k$, whereas if $b = 1$, $\mathcal{B}$ perfectly emulates Hybrid $2.(k-1)$. Therefore, if $\mathcal{D}$ can distinguish between Hybrids $2.k$ and $2.(k-1)$, then $\mathcal{B}$ can distinguish between an encryption of m_0 and an encryption of m_1, contradicting the semantic security of $\mathcal{E}$.

We have proved that the view of $\mathcal{A}^{\text{SH}}$ in Hybrid 0 is computationally indistinguishable from the view of $\mathcal{A}^{\text{SH}}$ in Hybrid $2.(N-t)$. But notice that the view of $\mathcal{A}^{\text{SH}}$ in Hybrid $2.(N-t)$ is precisely the simulated view created by $\mathcal{S}^{\text{SH}}$. We conclude that the view created by $\mathcal{S}^{\text{SH}}$ for $\mathcal{A}^{\text{SH}}$ is indistinguishable from the view of $\mathcal{A}^{\text{SH}}$ in a real world execution, as desired.

$\square$

D.2 Security against Malicious Adversaries

Proof of Theorem 2.2: We prove that the protocol is correct and secure, and that it satisfies the performance requirements of an *on-the-fly* protocol.

Correctness: Correctness follows directly from the correctness properties of homomorphic evaluation and the decryption MPC protocol.

Performance: By compactness of evaluation, we know that c^* is independent of $|C|$. Also, all zero-knowledge proofs are independent of C. Furthermore, we know that the proof φ has size polylogarithmic in $|C|$. Furthermore, the complexity of φ and the time needed to verify depend only polylogarithmically on the total size of the ciphertexts c_i. This means they also only depend polylogarithmically on the total size of the inputs x_i. Thus, the computation time and communication complexity party P_i is at most polylogarithmic in $|C|$ and the total size of inputs, and polynomial in y and its input x_i.

Security: Recall that for security, we only need to consider adversaries corrupting a subset T of the parties $P_1, \ldots, P_N$ involved in the computation. We let $\overline{T} = [N]\backslash T$. For a malicious adversary $\mathcal{A}^{\mathrm{MAL}}$ corrupting $t < N$ parties, we construct a simulator $\mathcal{S}^{\mathrm{MAL}}$ corrupting the same parties in the ideal world. We show the case when the server S is corrupted (the other case is analogous).

Simulator $\mathcal{S}^{\mathbf{mal}}$: Runs $\mathcal{A}^{\mathrm{MAL}}$ on input $\{x_i\}_{i \in T}$.

Step 1: For non-computing parties $i \in \{N+1, \ldots, U\}$ and for honest parties $i \in \overline{T}$, $\mathcal{S}^{\mathrm{MAL}}$ computes $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and samples hk_i honestly. It computes $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0)$ and simulated proofs π_i^{GEN} and π_i^{ENC} using the ZK simulator. It also computes $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})$. For each party P_i, $\mathcal{S}^{\mathrm{MAL}}$ sends $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}}, \pi_i^{\mathrm{ENC}})$ to $\mathcal{A}^{\mathrm{MAL}}$ on behalf of P_i. Receives (c^*, φ), $\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}})\}_{i \in [N]}$ from $\mathcal{A}^{\mathrm{MAL}}$.

Step 2: Receive (c^*, φ) from $\mathcal{A}^{\mathrm{MAL}}$, together with $\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}})\}_{i \in [N]}$. Verify φ and $\{\pi_i^{\mathrm{GEN}})\}_{i \in [T]}$ and use the extractor to extract witness $\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in T}$ from φ. Use the extractor to extract witness $\widetilde{x}_i$ from $\widetilde{\pi}_i^{\mathrm{ENC}}$ for all $i \in T$.

Step 3: Gives $\{\widetilde{x}_i\}_{i \in T}$ to the ideal functionality and receives $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where $\widetilde{x}_j = x_j$ for honest parties $j \in \overline{T}$. Runs the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ (interacting with $\mathcal{A}^{\mathrm{MAL}}$) on input $\widetilde{y}$.

$\mathcal{S}^{\mathrm{MAL}}$ outputs whatever $\mathcal{A}^{\mathrm{MAL}}$ outputs.

We prove that $\mathrm{IDEAL}_{\mathcal{F}, \mathcal{S}^{\mathrm{MAL}}}(\vec{x}) \overset{c}{\approx} \mathrm{REAL}_{\Pi^{\mathrm{MAL}}, \mathcal{A}^{\mathrm{MAL}}}(\vec{x})$. We prove that the view created by $\mathcal{S}^{\mathrm{MAL}}$ for $\mathcal{A}^{\mathrm{MAL}}$ is indistinguishable from the view of $\mathcal{A}^{\mathrm{MAL}}$ in a real world execution, and thus its output is indistinguishable in both cases. We prove this using a hybrid argument. We give a sketch of each hybrid.

Hybrid 0: This is the view of $\mathcal{A}^{\mathrm{MAL}}$ in a real-world execution. We have:

$$\begin{gathered}
\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Prove}^{\mathrm{GEN}}(\cdots)\}_{i \in \overline{T}} \\
\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Prove}^{\mathrm{ENC}}(\cdots)\}_{i \in \overline{T}} \\
\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\Pi_{\mathrm{DEC}}}(\mathcal{A}^{\mathrm{MAL}})
\end{gathered}$$

Hybrid 1: We change how we compute the proofs π_i^{GEN}; instead of computing real proofs, we use the ZK simulator.

$$\begin{gathered}
\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Sim}^{\mathrm{GEN}}(\cdots)\}_{i \in \overline{T}} \\
\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Prove}^{\mathrm{ENC}}(\cdots)\}_{i \in \overline{T}} \\
\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\Pi_{\mathrm{DEC}}}(\mathcal{A}^{\mathrm{MAL}})
\end{gathered}$$

We claim that the view of $\mathcal{A}^{\mathrm{MAL}}$ in Hybrid 1 is computationally indistinguishable from its view in Hybrid 0 by the *(adaptive unbounded) zero-knowledge* property of the proof system for relation R^{GEN}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 0 and 1. We construct an adversary $\mathcal{B}$ that breaks zero-knowledge. The adversary $\mathcal{B}$ works as follows:

1. For all $i \in \overline{T}$, sample $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and hk_i honestly, and compute $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$ and $\pi_i^{\text{ENC}} \leftarrow \mathsf{Prove}^{\text{ENC}}(\cdots)$. Obtain π_i^{GEN} from a query to the ZK challenger. For all $i \in \overline{T}$, compute $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})$ and give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}}, \pi_i^{\text{ENC}})$ to $\mathcal{A}^{\text{MAL}}$ on behalf of P_i. Receive $(c^*, \varphi), \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}})\}_{i\in[N]}$ from $\mathcal{A}^{\text{MAL}}$.
2. Verify φ and $\{\pi_i^{\text{GEN}}\}_{i\in T}$ and run the protocol $\Pi_{\text{DEC}}(c^*)$ (interacting with $\mathcal{A}^{\text{MAL}}$). Give $\mathcal{D}$ the resulting view.
3. Output the bit output by $\mathcal{D}$.

When $\pi_i^{\text{GEN}} \leftarrow \mathsf{Prove}^{\text{GEN}}(\cdots)$ for all $i \in \overline{T}$, $\mathcal{B}$ perfectly emulates Hybrid 0, whereas if $\pi_i^{\text{GEN}} \leftarrow \mathsf{Sim}^{\text{GEN}}(\cdots)$ for all $i \in \overline{T}$, $\mathcal{B}$ perfectly emulates Hybrid 1. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 0 and 1, then $\mathcal{B}$ can distinguish between real proofs and simulated proofs, contradicting the (adaptive unbounded) zero-knowledge property of the proof system for relation R^{GEN}.

Hybrids 2: We change how we compute the proofs π_i^{ENC}. Instead of computing real proofs, we use the ZK simulator.

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i\in\overline{T}} \quad , \quad \{\pi_i^{\text{GEN}} \leftarrow \mathsf{Sim}^{\text{GEN}}(\cdots)\}_{i\in\overline{T}}$$
$$\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i\in\overline{T}} \quad , \quad \{\pi_i^{\text{ENC}} \leftarrow \mathsf{Sim}^{\text{ENC}}(\cdots)\}_{i\in\overline{T}}$$
$$\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})\}_{i\in\overline{T}} \quad , \quad \mathsf{View}_{\Pi_{\text{DEC}}}(\mathcal{A}^{\text{MAL}})$$

We claim that the view of $\mathcal{A}^{\text{MAL}}$ in Hybrid 2 is computationally indistinguishable from its view in Hybrid 1 by the *(adaptive unbounded) zero-knowledge property* of the proof system for relation R^{ENC}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 1 and 2. We construct an adversary $\mathcal{B}$ that breaks zero-knowledge. The adversary $\mathcal{B}$ works as follows:

1. For all $i \in \overline{T}$, sample $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and hk_i honestly, and compute $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$ and $\pi_i^{\text{GEN}} \leftarrow \mathsf{Sim}^{\text{ENC}}(\cdots)$. Obtain π_i^{ENC} from a query to the ZK challenger. For all $i \in \overline{T}$, compute $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})$ and give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}}, \pi_i^{\text{ENC}})$ to $\mathcal{A}^{\text{MAL}}$ on behalf of P_i. Receive $(c^*, \varphi), \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}})\}_{i\in[N]}$ from $\mathcal{A}^{\text{MAL}}$.
2. Verify φ and $\{\pi_i^{\text{GEN}}\}_{i\in T}$ and run the protocol $\Pi_{\text{DEC}}(c^*)$ (interacting with $\mathcal{A}^{\text{MAL}}$). Give $\mathcal{D}$ the resulting view.
3. Output the bit output by $\mathcal{D}$.

When $\pi_i^{\text{ENC}} \leftarrow \mathsf{Prove}^{\text{ENC}}(\cdots)$ for all $i \in \overline{T}$, $\mathcal{B}$ perfectly emulates Hybrid 0, whereas if $\pi_i^{\text{ENC}} \leftarrow \mathsf{Sim}^{\text{ENC}}(\cdots)$ for all $i \in \overline{T}$, $\mathcal{B}$ perfectly emulates Hybrid 1. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 0 and 1, then $\mathcal{B}$ can distinguish between real proofs and simulated proofs, contradicting the (adaptive unbounded) zero-knowledge property of the proof system for relation R^{ENC}.

Hybrid 3: Hybrid 3 is the same as hybrid 2 except that we use the extractor for φ to extract $\{(\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}})\}_{i\in[N]} \leftarrow \mathsf{Ext}(\varphi)$ and compute $\widetilde{c} := \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))$. The simulator outputs $\perp$ if verification fails for any $\widetilde{\pi}_i^{\text{ENC}}$, if $d_i \neq H_{\mathsf{hk}_i}(\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}})$, or if $c \neq c^*$, where c^* is the evaluated ciphertext provided by $\mathcal{A}^{\text{MAL}}$. By the *proof of knowledge* property of φ, we know that this event happens with negligible probability. The simulator also outputs $\perp$ if $\widetilde{c}_i \neq c_i$ for any $i \in \overline{T}$. By the *collision-resistance* property of $H_{\mathsf{hk}_i}(\cdot)$ (since we sampled hk_i correctly for $i \in \overline{T}$), we know this happens with only negligible probability. Thus, hybrids 2 and 3 are statistically close.

Hybrid 4: Hybrid 4 is the same as hybrid 3 except that in addition, we extract $\{\widetilde{\mathsf{sk}_i} \leftarrow \mathsf{Ext}^{\text{GEN}}(\pi_i^{\text{GEN}})\}_{i\in T}$, and run $\widetilde{y} := \mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c^*)$, where $\widetilde{\mathsf{sk}}_i = \mathsf{sk}_i$ for $i \in \overline{T}$. If $\widetilde{y} \neq y^*$, where y^* is the output of the protocol $\Pi_{\text{DEC}}(c^*)$, then the simulator outputs $\perp$. By correctness of decryption and of the protocol Π_{DEC}, we know that as long as $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i)$ is in the support of Keygen for all i, this event happens with negligible probability. This is indeed true for $i \in \overline{T}$ since we sample the key tuple honestly in that case. Furthermore, by the *simulation-extractability* of the proof system for relation R^{GEN}, we know that except with negligible probability, $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i)$ is indeed in the support of Keygen for $i \in T$. Thus, by a union bound over all $i \in T$, we know that all of $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i)$ are in the support of Keygen, except with negligible probability. Thus, hybrids 3 and 4 are statistically close.

Hybrid 5: Instead of running the protocol Π_{DEC} and outputting $\mathsf{View}_{\Pi_{\text{DEC}}}(\mathcal{A}^{\text{MAL}})$, we run the simulator $\mathcal{S}_{\text{DEC}}^{\text{MAL}}$ and ouput $\mathsf{View}_{\mathcal{S}_{\text{DEC}}^{\text{MAL}}(\widetilde{y})}(\mathcal{A}^{\text{MAL}})$, where $\widetilde{y} := \mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c^*)$.

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i\in\overline{T}} \quad , \quad \{\pi_i^{\text{GEN}} \leftarrow \mathsf{Sim}^{\text{GEN}}(\cdots)\}_{i\in\overline{T}}$$
$$\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i\in\overline{T}} \quad , \quad \{\pi_i^{\text{ENC}} \leftarrow \mathsf{Sim}^{\text{ENC}}(\cdots)\}_{i\in\overline{T}}$$
$$\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})\}_{i\in\overline{T}} \quad , \quad \mathsf{View}_{\mathcal{S}_{\text{DEC}}^{\text{MAL}}(\widetilde{y})}(\mathcal{A}^{\text{MAL}})$$

where

$$\{(\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}})\}_{i\in[N]} \leftarrow \mathsf{Ext}(\varphi) \quad , \quad \{\widetilde{\mathsf{sk}_i} \leftarrow \mathsf{Ext}^{\text{GEN}}(\pi_i^{\text{GEN}})\}_{i\in T}$$
$$\widetilde{y} := \mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c^*) \text{ where } \widetilde{\mathsf{sk}}_j = \mathsf{sk}_j \text{ for } j \in \overline{T}$$

We claim that the view of $\mathcal{A}^{\text{MAL}}$ in Hybrid 4 is computationally indistinguishable from its view in Hybrid 5 by the *security* of the protocol Π_{DEC}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 4 and 5. We construct an adversary $\mathcal{B}$ that breaks the security of Π_{DEC}. The adversary $\mathcal{B}$ works as follows:

1. For all $i \in \overline{T}$, sample $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and hk_i honestly, and compute $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i), \pi_i^{\text{GEN}} \leftarrow \mathsf{Sim}^{\text{ENC}}(\cdots)$, and $\pi_i^{\text{ENC}} \leftarrow \mathsf{Sim}^{\text{ENC}}(\cdots)$. Compute $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\text{ENC}})$ and give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}}, \pi_i^{\text{ENC}})$ to $\mathcal{A}^{\text{MAL}}$ on behalf of P_i. Receive $(c^*, \varphi), \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\text{GEN}})\}_{i\in[N]}$ from $\mathcal{A}^{\text{MAL}}$.
2. Verify φ and $\{\pi_i^{\text{GEN}}\}_{i\in T}$. Receive the challenge view $\mathsf{View}^*(\mathcal{A}^{\text{MAL}})$, which is either $\mathsf{View}_{\Pi_{\text{DEC}}}(\mathcal{A}^{\text{MAL}})$ or $\mathsf{View}_{\mathcal{S}_{\text{DEC}}^{\text{MAL}}(y^*)}(\mathcal{A}^{\text{MAL}})$, and send the entire view of $\mathcal{A}^{\text{SH}}$ to $\mathcal{D}$.
3. Output the bit output by $\mathcal{D}$.

Recall that unless the simulator outputs $\bot$, we are guaranteed that $\widetilde{y} = y^*$, where y^* is the output of the protocol $\Pi_{\mathrm{DEC}}(c^*)$. Thus, when $\mathsf{View}^*(\mathcal{A}^{\mathrm{MAL}}) = \mathsf{View}_{\Pi_{\mathrm{DEC}}}(\mathcal{A}^{\mathrm{MAL}})$, $\mathcal{B}$ perfectly emulates Hybrid 4, whereas if $\mathsf{View}^*(\mathcal{A}^{\mathrm{MAL}}) = \mathsf{View}_{\mathcal{S}^{\mathrm{MAL}}_{\mathrm{DEC}}(\widetilde{y})}(\mathcal{A}^{\mathrm{MAL}})$, $\mathcal{B}$ perfectly emulates Hybrid 5. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 4 and 5, then $\mathcal{B}$ can distinguish between a real execution of Π_{DEC} and a simulated view, contradicting the security of the protocol Π_{DEC}.

Hybrid 6: Instead of computing $\widetilde{y} := \mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \dots, \widetilde{\mathsf{sk}}_N, c^*)$, we decrypt $\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}_i}, \widetilde{c}_i)$ and obtain $\widetilde{y} = f(\widetilde{x}_1, \dots, \widetilde{x}_N)$, where $\widetilde{x}_j = x_j$ for $j \in \overline{T}$.

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Sim}^{\mathrm{GEN}}(\cdots)\}_{i \in \overline{T}}$$
$$\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}^{\mathrm{ENC}}(\cdots)\}_{i \in \overline{T}}$$
$$\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\mathcal{S}^{\mathrm{MAL}}_{\mathrm{DEC}}(\widetilde{y})}(\mathcal{A}^{\mathrm{MAL}})$$

where

$$\{(\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}})\}_{i \in [N]} \leftarrow \mathsf{Ext}(\varphi) \quad , \quad \{\widetilde{\mathsf{sk}_i} \leftarrow \mathsf{Ext}^{\mathrm{GEN}}(\pi_i^{\mathrm{GEN}})\}_{i \in T} \quad , \quad \{\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}_i}, \widetilde{c}_i)\}_{i \in T}$$
$$\widetilde{y} = f(\widetilde{x}_1, \dots, \widetilde{x}_N) \text{ where } \widetilde{x}_j = x_j \text{ for } j \in \overline{T}$$

This is purely a syntactical change, since by *correctness* of the encryption scheme $\mathcal{E}$, we are guaranteed that $\widetilde{y}$ is the same in both hybrids. Thus, hybrids 5 and 6 are identical.

Hybrid 7: We change how we extract $\widetilde{x}_i$. Instead of decrypting $\widetilde{c}_i$ with $\widetilde{\mathsf{sk}}_i$, we extract $\widetilde{x}_i$ from $\widetilde{\pi}_i^{\mathrm{ENC}}$.

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Sim}^{\mathrm{GEN}}(\cdots)\}_{i \in \overline{T}}$$
$$\{c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}^{\mathrm{ENC}}(\cdots)\}_{i \in \overline{T}}$$
$$\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\mathcal{S}^{\mathrm{MAL}}_{\mathrm{DEC}}(\widetilde{y})}(\mathcal{A}^{\mathrm{MAL}})$$

where

$$\{(\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}})\}_{i \in [N]} \leftarrow \mathsf{Ext}(\varphi) \quad , \quad \{\widetilde{x}_{i_j} \leftarrow \mathsf{Ext}(\widetilde{\pi}_{i_j}^{\mathrm{ENC}})\}_{i \in T}$$
$$\widetilde{y} = f(\widetilde{x}_1, \dots, \widetilde{x}_N) \text{ where } \widetilde{x}_j = x_j \text{ for } j \in \overline{T}$$

We claim that the view of $\mathcal{A}^{\mathrm{MAL}}$ in Hybrid 6 is computationally indistinguishable from its view in Hybrid 7 by the *simulation-extractability* of the proof system for relation R^{ENC}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 6 and 7. We construct an adversary $\mathcal{B}$ that breaks simulation-extractability. The adversary $\mathcal{B}$ works as follows:

1. For all $i \in \overline{T}$, sample $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$, compute $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$, $\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Sim}^{\mathrm{GEN}}(\cdots)$, and obtain $\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}^{\mathrm{ENC}}(\cdots)$ from a query to the simulation-extractability challenger. Sample hk_i honestly, compute $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})$, and give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}}, \pi_i^{\mathrm{ENC}})$ to $\mathcal{A}^{\mathrm{MAL}}$ on behalf of P_i. Receive $(c^*, \varphi), \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}})\}_{i \in [N]}$ from $\mathcal{A}^{\mathrm{MAL}}$.
2. Verify φ and $\{\pi_i^{\mathrm{GEN}}\}_{i \in T}$ and run $\{(\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}})\}_{i \in [N]} \leftarrow \mathsf{Ext}(\varphi)$. Sample $b \leftarrow \{0,1\}$. If $b = 0$, then for $i \in T$, run the extractor $\widetilde{\mathsf{sk}}_i \leftarrow \mathsf{Ext}^{\mathrm{GEN}}(\pi_i^{\mathrm{GEN}})$ and decrypt $\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$. If $b = 1$, run the extractor $\widetilde{x}_i \leftarrow \mathsf{Ext}(\widetilde{\pi}_i^{\mathrm{ENC}})$. Run the simulator $\mathcal{S}^{\mathrm{MAL}}_{\mathrm{DEC}}$ on input $\widetilde{y}$ (interacting with $\mathcal{A}^{\mathrm{MAL}}$), where $\widetilde{y} = f(\widetilde{x}_1, \dots, \widetilde{x}_N)$ and $\widetilde{x}_j = x_j$ for $j \in \overline{T}$. Give $\mathcal{D}$ the resulting view.

3. If $\mathcal{D}$ outputs $b' = b$, then guess a random $i^* \in T$ and output $((\mathsf{pk}_{i^*}, c_{i^*}), \widetilde{\pi}_{i^*}^{\mathrm{ENC}})$.

Let $z^{(k)}$ denote the variable z in hybrid k. If $\mathcal{D}$ distinguishes between hybrids 6 and 7 then we must have that $\widetilde{y}^{(6)} \neq \widetilde{y}^{(7)}$, since the views are identical up to the point where $\widetilde{y}$ is computed. Thus, there exists $i^* \in T$ such that $\widetilde{x}_{i^*}^{(6)} \neq \widetilde{x}_{i^*}^{(7)}$. Furthermore, we are guaranteed that if $\mathcal{D}$ distinguishes between hybrids 6 and 7, $\widetilde{\pi}_i^{\mathrm{ENC}}$ is a valid proof for every $i \in T$ (since otherwise, the simulator outputs $\perp$ in both hybrids). Thus, $\widetilde{\pi}_{i^*}^{\mathrm{ENC}}$ is a valid proof but the extractor fails to extract a valid witness (the only valid witness is $\widetilde{x}_{i^*}^{(6)}$ and we know that $\widetilde{x}_{i^*}^{(6)} \neq \widetilde{x}_{i^*}^{(7)}$), and $\mathcal{B}$ will guess this index i^* with probability $1/t$. Therefore, if $\mathcal{D}$ distinguishes hybrids 6 and 7 with non-negligible probability ϵ, then $\mathcal{B}$ breaks the simulation-extractability of the proof system for R^{ENC} with probability ϵ/t, which is also non-negligible.

Hybrids $8.k$ **for** $k = 1, \ldots, N-t$**:** Let $\overline{T} = \{i_1, \ldots, i_{N-t}\}$. In Hybrid $8.k$ we change c_{i_k} so that instead of encrypting x_{i_k} it now encrypts 0. More formally, in Hybrid $8.k$ we have $c_{i_j} = \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)$ for $j \leq k$ and $c_{i_j} = \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})$ for $j > k$.

$$\{(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)\}_{i \in \overline{T}} \quad , \quad \{\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Sim}^{\mathrm{GEN}}(\cdots)\}_{i \in \overline{T}}$$

$$\{c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)\}_{j \leq k} \quad , \quad \{c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})\}_{j > k} \quad , \quad \{\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}^{\mathrm{ENC}}(\cdots)\}_{i \in \overline{T}}$$

$$\{d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})\}_{i \in \overline{T}} \quad , \quad \mathsf{View}_{\mathcal{S}_{\mathrm{DEC}}^{\mathrm{MAL}}(\widetilde{y})}(\mathcal{A}^{\mathrm{MAL}})$$

where

$$\{(\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}})\}_{i \in [N]} \leftarrow \mathsf{Ext}(\varphi) \quad , \quad \{\widetilde{x}_i \leftarrow \mathsf{Ext}(\widetilde{\pi}_i^{\mathrm{ENC}})\}_{i \in T}$$

$$\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N) \text{ where } \widetilde{x}_j = x_j \text{ for } j \in \overline{T}$$

For ease of notation we let Hybrid $7.(N-t)$ be Hybrid 6.0. We claim that the view of $\mathcal{A}^{\mathrm{MAL}}$ in Hybrid $8.k$ is computationally indistinguishable from its view in Hybrid $8.(k-1)$ by the *semantic security* of $\mathcal{E}$ under public key pk_{i_j}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids $8.k$ and $8.(k-1)$. We construct an adversary $\mathcal{B}$ that breaks the semantic security of $\mathcal{E}$. The adversary $\mathcal{B}$ works as follows:

1. Receive $(\mathsf{pk}, \mathsf{ek})$ from the semantic security challenger and set $\mathsf{pk}_{i_k} = \mathsf{pk}$ and $\mathsf{ek}_{i_k} = \mathsf{ek}$. Give $m_0 = 0$ and $m_1 = x_{i_k}$ to the challenger and receive $c = \mathsf{Enc}(\mathsf{pk}, m_b)$. Set $c_{i_k} = c$. For all $i \in \overline{T}, i \neq i_k$, sample $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ honestly. For $j < k$, compute $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)$ and for $j > k$, compute $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})$. For all $i \in \overline{T}$, sample hk_i honestly, compute $\pi_i^{\mathrm{GEN}} \leftarrow \mathsf{Sim}^{\mathrm{GEN}}(\cdots), \pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}^{\mathrm{ENC}}(\cdots)$ and $d_i = H_{\mathsf{hk}_i}(c_i, \pi_i^{\mathrm{ENC}})$, and give $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}}, \pi_i^{\mathrm{ENC}})$ to $\mathcal{A}^{\mathrm{SH}}$ on behalf of P_i. Receive $(c^*, \varphi), \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i, \pi_i^{\mathrm{GEN}})\}_{i \in [N]}$ from $\mathcal{A}^{\mathrm{SH}}$.
2. Verify φ and $\{\pi_i^{\mathrm{GEN}}\}_{i \in T}$. Obtain y from the ideal functionality, run the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{SH}}(y)(\mathcal{A}^{\mathrm{SH}})$ and output the view $\mathsf{View}_{\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{SH}}(y)}(\mathcal{A}^{\mathrm{SH}})$.

When $b = 0$, $\mathcal{B}$ perfectly emulates Hybrid $8.k$, whereas if $b = 1$, $\mathcal{B}$ perfectly emulates Hybrid $8.(k-1)$. Therefore, if $\mathcal{D}$ can distinguish between Hybrids $8.k$ and $8.(k-1)$, then $\mathcal{B}$ can distinguish between an encryption of m_0 and an encryption of m_1, contradicting the semantic security of $\mathcal{E}$.

We have proved that the view of $\mathcal{A}^{\text{MAL}}$ in Hybrid 0 is computationally indistinguishable from the view of $\mathcal{A}^{\text{MAL}}$ in Hybrid 8.$(N-t)$. But notice that the view of $\mathcal{A}^{\text{MAL}}$ in Hybrid 8.$(N-t)$ is precisely the simulated view created by $\mathcal{S}^{\text{MAL}}$. We conclude that the view created by $\mathcal{S}^{\text{MAL}}$ for $\mathcal{A}^{\text{MAL}}$ is indistinguishable from the view of $\mathcal{A}^{\text{MAL}}$ in a real world execution, as desired.

$\square$