MULTIKEY FULLY HOMOMORPHIC ENCRYPTION AND APPLICATIONS[*]

ADRIANA LÓPEZ-ALT[†], ERAN TROMER[‡], AND VINOD VAIKUNTANATHAN[§]

Abstract. We propose a new notion of secure multiparty computation aided by a computationally powerful but untrusted "cloud" server. In this notion, *on-the-fly multiparty computation (MPC)*, the cloud can *noninteractively* perform arbitrary *dynamically chosen computations* on data belonging to arbitrary *dynamically chosen sets of users chosen*. All users' 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 define the new notion of *multikey FHE*, an enhanced FHE system which is capable of operating on inputs encrypted under multiple, unrelated keys. A ciphertext resulting from a multikey homomorphic evaluation can be jointly decrypted using the secret keys of all the users involved in the computation. We show how on-the-fly MPC can be achieved using any *multikey FHE* scheme. (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.

Key words. fully homomorphic encryption, multiparty computation, cloud computing, NTRU encryption

AMS subject classifications. 68P25, 94A60

DOI. 10.1137/14100124X

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." 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 [122]. Since this data often contains sensitive information (e.g., personal conversations, medical

[†]New York University, New York, NY 10012 (adrilopo@gmail.com).

[‡]Tel Aviv University, School of Computer Science, Tel Aviv 69978, Israel (tromer@cs.tau.ac.il).

[§]MIT, Cambridge, MA 02139 (vinodv@csail.mit.edu).

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) [71, 132, 30, 29, 72, 27] 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 collective data. For example, they may wish the cloud to compute joint statistical information on their databases, locate common files in their collections, or run a computational agent to reach a decision based on their pooled data (without leaking anything but the final decision). More generally, in contexts where multiple (mutually distrusting) users need to pool together their data to achieve a common goal, FHE is not quite enough.

The multiparty scenario is significantly more complex and comes with a set of natural but stringent requirements. First, the participants involved in a 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. Second, 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) *noninteractively*, 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 the complexity of the function computed (and ideally, also the total number of users in the system) which could be very large.

On-the-fly multiparty computation (MPC). Consider a setting with a large universe of computationally weak users and a powerful cloud. An on-the-fly MPC protocol proceeds as follows.

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 an arbitrary *dynamically chosen subset* of users. (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 noninteractively, 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* (*MPC*) 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.

Motivating scenario: Malware sharing. When malware is detected by a company, it is helpful for its security analysts to know which other companies have observed similar malware. However, it is undesirable for companies to publicize all malware they have detected. Using on-the-fly MPC, each company can build an ever-growing dataset of malware it has detected, stored encrypted on a cloud service: whenever it detects a new sample, the company simply encrypts it with its own public key and sends it to the cloud. The cloud service, on a daily basis, gathers the encrypted datasets of its current subscribers, evaluates a comparison function which scans these datasets for similar malware samples using some executable-code similarity metrics, and sends the short encrypted results to those subscribers, for (interactive but cheap) decryption. Crucially, subscribers may join or leave the service at any time.

Possible approaches (and why they do not work). The long line of work on secure MPC [84, 14, 35, 135] 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.)

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 [70] 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. [6] 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 *noninteractive* nature of on-the-fly MPC. In particular, once a subset of parties and a function are chosen, the protocols of [70, 6] 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 noninteractively, without help from any of the users. It would also be unsatisfactory to postpone the (lengthy) computation of the function until the interactive decryption phase; indeed, we require that once the users wake up for the decryption phase, the running

---

[1] The works of Damgård and co-authors [51, 50] are an exception to this claim. However, it is not clear how to build upon these results to address the dynamic and noninteractive nature of on-the-fly MPC.

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.

1.1. Our results and techniques. We present a *new notion* of 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 [96]), 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.

Multikey FHE. 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. As mentioned above, one of our main contributions is a construction of N-key FHE for any $N \in \mathbb{N}$ from the NTRU encryption scheme. We give an overview of our construction below (in section 1.2) and refer the reader to section 3.3 for more details.

Our NTRU-based construction raises a natural question: can any other FHE schemes be made multikey? We show that *any* FHE scheme is inherently a *multikey FHE* for a constant number of keys (in the security parameter), i.e., it can homomorphically evaluate functions on ciphertexts encrypted under at most a constant number of keys.[2] Furthermore, we show that the ring-LWE based FHE scheme of Brakerski and Vaikuntanathan [30] is multikey homomorphic for a logarithmic number of keys but only for circuits of logarithmic depth. This arises from the fact that when multiple keys are introduced, it is no longer clear how to use relinearization or squashing to go beyond somewhat homomorphism. We refer the reader to section 3.2 for more details.

On-the-fly MPC from multikey FHE. 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 semimalicious collusion [7, 6] of the cloud with an arbitrary subset of parties. We then show how to achieve security against a malicious adversary using zero-knowledge proofs and succinct argument systems [99, 100, 108, 88, 89, 19, 20].

We further remark that 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. [6] yields an on-the-fly MPC protocol with one offline round and 5 online rounds (for decryption).

We give an overview of our construction below (in section 1.3) and refer the reader to section 4 for more details.

---

[2]This construction was originally suggested to us by an anonymous STOC 2012 reviewer; we include it here for completeness.

Completely noninteractive on-the-fly MPC? We know from the work of Halevi, Lindell, and Pinkas [95] that in the noninteractive 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 noninteractive online phase *cannot* be achieved by drawing on the impossibility of general program obfuscation as a virtual black-box with single-bit output [10]. Thus, our notion is qualitatively "the best possible" in terms of interaction. Our techniques in showing this negative result are inspired by those of van Dijk and Juels [133]. We refer the reader to section 4.3 for more details.

1.2. (Multikey) fully homomorphic encryption from NTRU. The starting point for our main construction of multikey FHE is the NTRU encryption scheme of Hoffstein, Pipher, and Silverman [96] with the modifications of Stehlé and Steinfeld [130]. NTRU encryption is one of the earliest lattice-based cryptosystems, together with the Ajtai–Dwork cryptosystem [2] and the Goldreich–Goldwasser–Halevi cryptosystem [83]. One of our most important contributions is to show that NTRU can be made fully homomorphic (for a single key)[3] and, moreover, that the resulting scheme can handle homomorphic evaluations on ciphertexts encrypted under any number of *different and independent* keys.

We find this contribution particularly interesting because NTRU was originally designed to be an *efficient* public-key encryption scheme, meant to replace RSA in applications where computational efficiency is at a premium (e.g., in applications that run on smart cards and embedded systems). Although the transformation to fully homomorphic encryption degrades the efficiency of the scheme, we believe it to be a *leading candidate* for a *practical* FHE scheme. Therefore, we view this as an important contribution of independent interest.

In this section we give an overview of our construction and refer the reader to section 3.3 for more details.

NTRU encryption. We describe the modified NTRU scheme of Stehlé and Steinfeld [130], which is based on the original NTRU cryptosystem [96]. The scheme is parametrized by the ring $R \overset{\mathsf{def}}{=} \mathbb{Z}[x]/\langle x^n + 1\rangle$, where n is a power of two, an odd prime number q, and a B-bounded distribution χ over R for $B \ll q$. By "B-bounded," we mean that the magnitude of the coefficients of a polynomial sampled from χ is guaranteed to be less than B. We define $R_q \overset{\mathsf{def}}{=} R/qR$, and use $[\ \cdot\ ]_q$ to denote coefficient-wise reduction modulo q into the set $\{-\lfloor\frac{q}{2}\rfloor, \ldots, \lfloor\frac{q}{2}\rfloor\}$.

- Keygen(1^κ): Key generation samples "small" polynomials $f', g \leftarrow \chi$ and sets $f \overset{\mathsf{def}}{=} 2f' + 1$ so that $f \pmod 2 = 1$. If f is not invertible in R_q, it resamples f'. Otherwise, it computes the inverse f^{-1} of f in R_q and sets

$$\mathsf{sk} = f \qquad \text{and} \qquad \mathsf{pk} = h \overset{\mathsf{def}}{=} \left[2gf^{-1}\right]_q.$$

- Enc(pk, m): To encrypt a bit $m \in \{0, 1\}$ with public key $\mathsf{pk} = h$, the encryption algorithm samples small polynomials $s, e \leftarrow \chi$ and outputs the ciphertext

$$c = [hs + 2e + m]_q.$$

- Dec(sk, c): To decrypt a ciphertext c, the decryption algorithm computes $\mu = [fc]_q$ and returns $\mu \pmod 2$.

---

[3]The observation that NTRU can be made *single-key* fully homomorphic was made concurrently by Gentry et al. [74].

Correctness follows from a few simple observations. First note that $[fc]_q = [2gs + 2fe + fm]_q$. Furthemore, since the elements g, s, f, e were all sampled from a B-bounded distribution and $B \ll q$, the magnitude of the coefficients in $2gs + 2fe + fm$ is smaller than $q/2$, so there is no reduction modulo q: in other words, $[2gs + 2fe + fm]_q = 2gs + 2fe + fm$. Therefore, $\mu = 2gs + 2fe + fm$. Taking modulo 2 yields the message m since by construction, $f \equiv 1 \pmod 2$.

Multikey homomorphism. We now briefly describe the (multikey) homomorphic properties of the scheme and the challenges encountered when converting it into a fully homomorphic encryption scheme.

Let $c_1 = [h_1 s_1 + 2e_1 + m_1]_q$ and $c_2 = [h_2 s_2 + 2e_2 + m_2]_q$ be ciphertexts under two different keys $h_1 = [2g_1 f_1^{-1}]_q$ and $h_2 = [2g_2 f_2^{-1}]_q$, respectively. We claim that $c_{\mathsf{add}} \stackrel{\mathsf{def}}{=} [c_1 + c_2]_q$ and $c_{\mathsf{mult}} \stackrel{\mathsf{def}}{=} [c_1 c_2]_q$ decrypt to $m_1 + m_2$ and $m_1 m_2$, respectively, under the *joint* secret key $f_1 f_2$. Indeed, notice that

$$f_1 f_2(c_1 + c_2) = 2\left(f_1 f_2 e_1 + f_1 f_2 e_2 + f_2 g_1 s_1 + f_1 g_2 s_2\right) + f_1 f_2(m_1 + m_2)$$
$$= 2e_{\mathsf{add}} + f_1 f_2(m_1 + m_2)$$

for a slightly larger noise element e_{add}. Similarly,

$$f_1 f_2(c_1 c_2) = 2(2g_1 g_2 s_1 s_2 + g_1 s_1 f_2(2e_2 + m_2) + g_2 s_2 f_1(2e_1 + m_1)$$
$$+ f_1 f_2(e_1 m_2 + e_2 m_1 + 2e_1 e_2)) + f_1 f_2(m_1 m_2)$$
$$= 2e_{\mathsf{mult}} + f_1 f_2(m_1 m_2)$$

for slightly larger noise element e_{mult}. This shows that the ciphertexts $c_{\mathsf{add}} \stackrel{\mathsf{def}}{=} [c_1 + c_2]_q$ and $c_{\mathsf{mult}} \stackrel{\mathsf{def}}{=} [c_1 c_2]_q$ can be correctly decrypted to the sum and the product of the underlying messages, respectively, as long as the error does not grow too large.

Extending this to circuits, we notice that the secret key required to decrypt a ciphertext c that is the output of a homomorphic evaluation on ciphertexts encrypted under N different keys is $\prod_{i=1}^N f_i^{d_i}$, where d_i is the degree of the ith variable in the polynomial function computed by the circuit. Thus, decrypting a ciphertext that was the product of a homomorphic evaluation requires knowing the circuit! This is unacceptable even for somewhat homomorphic encryption.

We employ the relinearization technique of Brakerski and Vaikuntanathan [29] to essentially reduce the degree from d_i to 1, so that the key needed to decrypt the evaluated ciphertext is now $\prod_{i=1}^N f_i$. This guarantees that decryption is dependent on the number of keys N but independent of the circuit computed. After using relinearization, we can show that the resulting scheme is multikey somewhat homomorphic for $\approx n^\delta$ keys and circuits of depth $\approx \log \log q - \delta \log n$ for any $\delta \in (0,1)$.

From (multikey) somewhat to fully homomorphic encryption. Once we obtain a (multikey) somewhat homomorphic encryption scheme, we can apply known techniques to convert it into a (multikey) fully homomorphic scheme. In particular, we follow the original template of our work [106] and use modulus reduction [29, 27] to increase the circuit depth that the scheme can handle in homomorphic evaluation. This yields a leveled homomorphic scheme for N keys that can evaluate circuits of depth D as long as $ND \approx \log q$. For any number of keys N and any depth D, we can set q to be large enough to guarantee the successful homomorphic evaluation of depth-D circuits on ciphertexts encrypted under N different keys.

THEOREM 1.1 (informal). *For all $N \in \mathbb{N}$ and $D \in \mathbb{N}$, there exists a leveled homomorphic encryption (HE) scheme that can homomorphically evaluate depth-D circuits*

on ciphertexts encrypted under at most N different keys. The size of the keys and ciphertexts in the scheme grow polynomially with N and D.

Finally, using an analogue of Gentry's bootstrapping theorem [71, 70] for the multikey setting, we can convert the leveled homomorphic encryption scheme into a fully homomorphic encryption scheme, in which the algorithms are independent of the circuit depth D. This requires a "circular security" type assumption which, roughly speaking, asserts that the leveled HE encryption retains semantic security even if the adversary is given an encryption of the bits of the secret key (under the leveled HE scheme). We refer the reader to Definition 2.9 for a precise formulation. Constructing a fully homomorphic encryption scheme, *even in the single key setting*, without such circular security type assumptions is a major open problem (although, see [34] for some recent progress, using the incomparable assumption that indistinguishability obfuscation [10, 66] exists).

On the other hand, even with the circular security assumption, we are unable to remove the dependence on the number of keys N and therefore obtain a scheme that is fully homomorphic with respect to the depth of circuits it can evaluate but "leveled" with respect to the number of different keys it can handle. This state of affairs has since been improved [41, 110]; we refer the reader to section 1.5 for more details.

We remark that using the recent noise-management technique of Brakerski [26], it is possible to obtain a simpler leveled (single-key) homomorphic encryption scheme, based on a weaker security assumption. This was already noted in the follow-up work of Bos et al. [25]. In another recent work, Gentry, Sahai, and Waters [79] show how to remove the required evaluation key, yielding an even simpler scheme.

Security. Stehlé and Steinfeld [130] showed that the security of the modified NTRU encryption scheme can be based on the ring-LWE assumption of Lyubashevsky, Peikert, and Regev, which can be reduced to worst-case hard problems in ideal lattices [107]. To prove the security of NTRU, Stehlé and Steinfeld first show that the public key $h = [2gf^{-1}]_q$ is *statistically* close to uniform over the ring R if f' and g are sampled from a discrete Gaussian with standard deviation $\mathsf{poly}(n)\sqrt{q}$ (which can be shown to be a $\mathsf{poly}(n)\sqrt{q}$-bounded distribution). Unfortunately, if we sample f' and g from this distribution, the error in a *single* homomorphic multiplication would grow large enough to cause decryption failures. We must therefore make the *assumption* that the public key $h = [2gf^{-1}]_q$ is *computationally indistinguishable*[4] from uniform over R when f' and g are sampled from a discrete Gaussian that is B-bounded for $B \ll q$.

Ultimately, we arrive at the following theorem.

THEOREM 1.2 (informal). *For all $N \in \mathbb{N}$, there exists a fully homomorphic encryption scheme that can perform homomorphic evaluation on ciphertexts encrypted under at most N different keys. The size of the keys and ciphertexts in the scheme grow polynomially with N. The security of the scheme is based on the ring-LWE assumption, the assumption that the public key is pseudorandom, and the assumption that the scheme is weakly circular secure.*

In a follow-up work, Bos et al. [25] show how to apply Brakerski's techniques [26] to maintain the fully homomorphic properties of the scheme while sampling the elements f' and g from a discrete Gaussian with standard deviation $\mathsf{poly}(n)\sqrt{q}$, as in the work of Stehlé and Steinfeld [130]. This yields an NTRU-based (single-key)

---

[4]It is not difficult to see that with our setting of parameters, the distribution of the public key is not statistically close to uniform. We rely on *computational* indistinguishability.

FHE scheme that is secure under the RLWE assumption alone. However, their scheme is not a multikey FHE scheme. In a nutshell, this is because decrypting a multikey evaluated ciphertext in the setting of [25] would entail multiplication by the product of all keys that were involved in the generation of the ciphertext. With the parameter settings required by Stehlé and Steinfeld, namely, when f' and g are sampled from a discrete Gaussian with standard deviation $\mathsf{poly}(n)\sqrt{q}$, multiplying by a product of only two keys would already lead to a noise overflow, making it impossible to decrypt correctly. Nevertheless, their scheme can be made into a multikey FHE scheme for a constant number of parties using the general transformation described in section 3.2.1. We refer the reader to [25] for a more detailed description of their scheme.

1.3. On-the-fly MPC from multikey FHE. Once we have constructed multikey FHE for any number of keys, we can construct on-the-fly MPC. The following gives an informal outline of our protocol.

Offline phase: The clients sample independent key pairs $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i)$, encrypt their input under their corresponding public key: $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$, and send this ciphertext to the server along with the public and evaluation keys $(\mathsf{pk}_i, \mathsf{ek}_i)$.

Online phase: Once a function has been chosen, together with a corresponding subset of computing parties V:

  Step 1. The server performs the *multikey homomorphic evaluation* of the desired circuit on the corresponding ciphertexts and broadcasts the evaluated ciphertext to all computing parties (i.e., all parties in V).

  Step 2. The computing parties (i.e., parties in V) run a generic MPC protocol to decrypt the evaluated ciphertext using their individual secret keys sk_i.

Observe that the computation of the decryption function in Step 2 of the online phase can itself be delegated to the server. In particular, if we instantiate the decryption protocol using the cloud-assisted MPC protocol of Asharov et al. [7, 6] we obtain a round-efficient solution: the overall protocol has an online phase of only 5 rounds.

1.3.1. Protocol security. We show that the above protocol is secure against *semimalicious* adversaries [7, 6], who follow the protocol specifications (like semihonest adversaries) but choose their random coins from an arbitrary distribution (like malicious adversaries). We then show how to modify the protocol to achieve security against malicious adversaries. To this end, we make three modifications, described below.

Modifying the decryption protocol. The first modification we make is to change the decryption protocol in Step 2 of the online phase to first check that the secret key being used is a *valid* secret key for the corresponding public and evaluation keys. This ensures that if decryption is successful, then in particular, a corrupted party knows a valid secret key $\widetilde{\mathsf{sk}}_i$. This secret key *binds* the corrupted party to the input $\widetilde{x}_i = \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$, which by semantic security of the FHE, must be independent of the honest inputs.

Once again, we note that the computation of this function can be delegated to the server using the cloud-assisted protocol of Asharov et al. [7, 6], yielding a 5-round online phase.

Adding zero-knowledge proofs. We further require that in the offline phase, each party create a noninteractive zero-knowledge proof π_i^{ENC} showing that the ciphertext c_i is well-formed (i.e., that there exists plaintext x_i and randomness s_i such that $c_i = \mathsf{Enc}(\mathsf{pk}_i, x_i \; ; \; s_i)$). This guarantees that for a corrupted party, $\mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i) \neq \perp$ and thus the party really "knows" an input $\widetilde{x}_i$. Furthermore, it guarantees that the ciphertexts c_i are *fresh* encryptions, which is important in our setting of fully homo-

morphic encryption where we must ensure that the error stays low in a homomorphic evaluation.

While constructions of NIZK arguments are known for all of NP [92, 93], using these constructions requires expensive NP reductions. To avoid this, in section 4.2.3 we show how to construct an efficient NIZK argument system, secure in the random oracle model, for proving the well-formedness of a ciphertext in the NTRU-based multikey FHE scheme (the scheme we use to instantiate the generic multikey FHE scheme in our on-the-fly MPC construction).

Adding verification of computation. Finally, we must also rely on a succinct argument system [99, 100, 108, 88, 89, 19, 20] to ensure that the server performs the homomorphic computation correctly. Due to the dynamic nature of our on-the-fly model, we are unable to use verifiable computation protocols in the preprocessing model [68, 40, 5] or succinct arguments with a reference string that depends on the circuit being computed [90, 103, 69, 115, 104]. These would require the clients to perform some precomputation dependent on the circuit to be computed *before* knowing the circuit or to interact with the server after a function has been selected and compute in time proportional to the circuit-size of the function. Indeed, the beauty of our on-the-fly MPC model is that the server can choose *any* function *dynamically*, on-the-fly, and homomorphically compute this function *without* interacting with the clients, who additionally, compute in time only polylogarithmically in the size of any function being computed.

We show how to guarantee verification of computation in two different cases.

Verification for small inputs: When the total size of the inputs (and therefore the ciphertexts) is small enough to be broadcasted to all parties, it suffices for the server to use any of the succinct arguments of [99, 100, 108, 88, 89, 19, 20] to prove that it carried out the computation correctly as specified. Along with this argument, the server broadcasts the ciphertexts c_i and public and evaluations keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ for all parties in V. With this information, the computing parties can verify the argument before engaging in the decryption protocol.

Verification for large inputs: In the case when the total size of the inputs (and therefore the ciphertexts) is too large to be broadcasted to all parties, then we additionally require the parties to sample a hash key hk_i for a collision-resistant hash function and compute a digest d_i of the ciphertext c_i. Each party then sends the tuple $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\mathrm{ENC}}, \mathsf{hk}_i, d_i)$ to the server in the offline phase. It is then sufficient for the server to broadcast the tuples $(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)$ and a succinct argument for the NP language:

there exist $\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{ENC}}, \ldots, \widetilde{c}_N, \widetilde{\pi}_N^{\mathrm{ENC}}$ such that $d_i = H_{\mathsf{hk}_i}(\widetilde{c}_i)$ 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 $\widetilde{\pi}_i^{\mathrm{ENC}}$ is a valid proof.

If the succinct argument is additionally a *proof of knowledge*, as in the case of CS proofs [108] under Valiant's analysis [131], and the SNARKs of Bitansky et al. [19, 20], then we are guaranteed that the server actually "knows" such $\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{ENC}}, \ldots, \widetilde{c}_N, \widetilde{\pi}_N^{\mathrm{ENC}}$ whenever it successfully convinces the clients.

Putting everything together, we arrive at the following theorem.

THEOREM 1.3 (informal). *There exists an on-the-fly MPC protocol with the following properties:*

- *It achieves security against malicious corruptions of an arbitrary subset of clients and possibly the server, under the ring-LWE assumption, the assump-*

tion that the public key in the (modified) NTRU cryptosystem [96, 130] *is pseudorandom for a special setting of parameters, and the existence of zero-knowledge proofs and a secure succinct argument system.*

- *The offline phase runs in one (asynchronous) round of unidirectional communication from the parties to the server. The online phase runs in* 5 *rounds.*
- *The communication complexity of the online phase and the computation time of the computing parties therein is polylogarithmic in the size of the computation and the total size of the inputs and linear in the size of their own input and the size of the output.*
- *The computation time of the server is polynomial in the size of the circuit.*

We remark that our MPC protocol inherits any setup assumptions (such as the existence of a common random or reference string or the use of random oracles) from the underlying primitives, namely, zero-knowledge proofs and succinct argument systems. In the common random string model, the prover and the verifier in a proof system are given access to a uniformly random string chosen by a trusted process that neither entity has control over. The common reference string model is the same, except that the string is chosen from some fixed (not necessarily uniform) distribution. In the random oracle model [13], all parties (including the adversary) are assumed to have access to a random function that they can access as an oracle. Sections 2.2 and 2.3 discuss the state of the art in the construction of these primitives (including the setup assumptions).

1.4. Related work. We briefly survey related works in the areas of fully homomorphic encryption, MPC from homomorphic encryption, and MPC with the aid of a cloud server.

Fully homomorphic encryption. The notion of fully homomorphic encryption was first proposed by Rivest, Adleman, and Dertouzos [123] and was constructed in the groundbreaking result of Gentry [71, 70]. In subsequent years, many improvements and new constructions have appeared in the literature, including [132, 30, 72, 29, 27, 26, 25, 79, 31, 38].

Gentry's first construction [71, 70] followed the following blueprint: first, he constructed a *somewhat homomorphic* encryption scheme working over ideal lattices that was able to perform a limited number of evaluations. He then proved a *bootstrapping theorem*, showing that if a somewhat homomorphic scheme can homomorphically evaluate its own decryption circuit (plus one more gate), then (a) it can be converted into a *leveled fully homomorphic* scheme, namely, a scheme that can evaluate circuits of any a priori bounded (polynomial) depth, under the same assumption, and (b) it can be converted into a *fully homomorphic* scheme that can evaluate circuits of arbitrary depth, assuming in addition that the somewhat homomorphic scheme is "circular secure." Unfortunately, Gentry's somewhat homomorphic scheme cannot evaluate its own decryption circuit and is therefore not bootstrappable. Nevertheless, he was able to construct a bootstrappable scheme by *squashing* the decryption circuit sufficiently for the scheme to be able to homorphically evaluate it. Using this squashing technique required making an additional security assumption, namely, the sparse subset sum (SSS) assumption.

Van Dijk et al. [132] subsequently showed how to construct a different FHE scheme "over the integers" under the approximate-GCD assumption, and Brakerski and Vaikuntanathan [30] showed how to construct FHE from the ring-LWE assumption of Lyubashevsky, Peikert, and Regev [107]. Both of these works use squashing and bootstrapping, as in Gentry's original blueprint (and thus, require the accompanying

additional assumptions). Gentry and Halevi [72] showed how to use depth-3 arithmetic circuits and a hybrid of somewhat homomorphic encryption and multiplicatively homomorphic encryption (e.g., Elgamal encryption [63]) to construct FHE without the use of squashing and therefore without assuming the hardness of the SSS problem.

The second generation of FHE schemes arrived with the work of Brakerski and Vaikuntanathan, who showed how to construct a leveled FHE scheme from Regev's (standard) learning with errors (LWE) assumption [120, 121] alone. In particular, they introduced the techniques of *relinearization* and *modulus reduction* which have been instrumental in many subsequent FHE constructions. Brakerski, Gentry, and Vaikuntanathan [27] subsequently refined and generalized these techniques into *key-switching* and *modulus switching* and showed how to build a leveled FHE scheme *directly*, without the use of squashing or bootstrapping. Formally, they show that for every $D \in \mathbb{N}$, there exists a homomorphic scheme $\mathcal{E}^{(D)}$ that is able to homomorphically evaluate circuits of depth D. Their technique involves switching to a smaller modulus after every level in a homomorphic computation, therefore requiring a fairly large modulus at the start of the computation. This required basing security of their scheme on the hardness of solving approximate-SVP to within subexponential factors.

In work subsequent to ours, Brakerski [26] showed a new noise-management technique that forwent the modulus switching step, allowing the use of a single modulus that is much smaller than the one needed in the BGV scheme. The security of Brakerski's scheme can be based on the hardness of solving approximate-SVP to within quasi-polynomial factors, a much weaker assumption. Bos et al. [25] show how to apply Brakerki's noise-management technique to the (multikey) FHE described in this paper [106], based on the NTRU encryption scheme of Hofftein, Pipher, and Silverman [96] with the modifications of Stehlé and Steinfeld [130]. They further show that using these techniques, one can base security of the resulting FHE scheme on the ring-LWE assumption alone, by using Stehlé and Steinfeld's original analysis. Their construction, however, is multikey for only a constant number of keys, which we show can be achieved from any generic FHE scheme already.

Finally, in another work subsequent to ours, Gentry, Sahai, and Waters [79] came up with a third generation of FHE schemes: they constructed a leveled homomorphic scheme that does not require the use an evaluation key to perform homomorphic computation, as all previous schemes do. Brakerski and Vaikuntanathan [31] show how to leverage the techniques of Gentry, Sahai, and Waters [79] to build a leveled homomorphic scheme that is as secure as standard (nonhomomorphic) LWE-based public-key encryption.

Many other works study the efficiency of the schemes described above and present several optimizations [128, 127, 73, 43, 75, 44, 76, 77, 78, 36, 42, 129].

MPC from homomorphic encryption. The basic idea of using threshold homomorphic encryption (e.g., Paillier encryption [114]) to boost the efficiency of MPC protocols was first presented by Cramer, Damgård, and Nielsen [46], predating the existence of fully homomorphic encryption (first showed by Gentry in 2009 [71, 70]). They show that if the parties have access to a public key for an additively homomorphic encryption scheme, and if they also have a corresponding secret key secret-shared among them, then they can evaluate any boolean circuit "under the covers" of the encryption. Using the homomorphic properties of the scheme, the parties can locally evaluate all addition gates. Cramer and co-authors additionally show a short, interactive subprotocol for evaluating multiplication gates. After showing the first construction of fully homomorphic encryption, Gentry used the same template to show a generic MPC construction from any FHE [70].

In a work concurrent to ours, Myers, Sergi, and Shelat [111] show a *black-box* construction of MPC from any threshold FHE scheme. Their main hurdle is devising a way for parties to prove plaintext knowledge of a ciphertext. To this end, they present a 2-round protocol for proving plaintext knowledge, which they construct from any circuit-private FHE scheme. Their protocol is not zero-knowledge [85], but it conserves the semantic security of the ciphertext in question. They also show how to construct threshold FHE using the scheme of van Dijk et al. [132] over the integers. While the communication of their protocol is independent of the circuit-size of the function being computed, their protocol is not computation-efficient: parties compute proportional to the complexity of the function.

Other works by Damgård and co-authors [18, 53, 52] build MPC from "semi-homomorphic" and somewhat homomorphic encryption. Their protocols require all parties to compute proportional to the complexity of the function at hand and require interaction between parties at every gate. However, they display very good concrete efficiency. A work of Choudhury et al. [39] shows how to trade computation efficiency for communication efficiency. Their protocol is parametrized by an integer L. Setting $L = 2$ yields a classic MPC protocol, in which interaction is required for computing every gate. As L increases, interaction is required less frequently and only to "refresh" the computation after an increasing number of steps. Thus, at their heart of their construction lies an interactive bootstrapping protocol that refreshes ciphertexts during the evaluation.

Finally, a recent work by Garg et al. [65] shows how to achieve 2-round MPC in the common random string (CRS) model from indistinguishability obfuscation ($i\mathcal{O}$) [11]. As an optimization, they use multikey FHE (as defined in this work) to construct 2-round MPC with communication complexity that is independent of the circuit being computed. Though an efficient construction of $i\mathcal{O}$ is known for all circuits [66], its security is based on assumptions on multilinear maps [64] that are not very well understood yet.

MPC on the cloud. The idea of using a powerful cloud server to alleviate the computational efforts of parties in an MPC protocol was recently explored in the work on "server-aided MPC" by Kamara, Mohassel, and Raykova [98]. Their protocols, however, require some of the parties to do a large amount of work, essentially proportional to the size of the computation.

Halevi, Lindell, and Pinkas [95] 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 of the protocol (they only interact with the server), they show that this notion can only be achieved for a small class of functions. Our goal, on the other hand, is to construct MPC protocols for *arbitrary functions*.

1.5. Subsequent work. Garg and Polychroniadou [67] showed a construction of two-round MPC from indistinguishability obfuscation. Their MPC protocol achieves the stronger notion of adaptive security [32].

Clear and McGoldrick [41] constructed a multikey leveled fully homomorphic encryption scheme based on the LWE problem. (This can be bootstrapped into a nonleveled scheme assuming the circular security of the underlying leveled scheme.) Subsequently, Mukherjee and Wichs [110] significantly simplified their construction and showed how to use it to construct a two-round MPC protocol in the CRS model from LWE. These results improve on our work in two significant ways: first, they rely on a much more standard assumption, namely, LWE; and secondly, their multi-

key FHE schemes work for an a priori unbounded number of parties N, whereas our schemes require N to be known in advance of choosing the parameters; and finally, the techniques, based on [79], are very different and truly novel. We refer the reader to [41, 110] for more details.

Dodis et al. [55] showed how to extend the multikey FHE scheme of [41, 110] into a primitive that they call a spooky encryption scheme. In a spooky encryption scheme, one can start from $\mathcal{E}_{\mathsf{pk}_1}(x_1)$ and $\mathcal{E}_{\mathsf{pk}_2}(x_2)$ and compute two ciphertexts c_1 and c_2 such that $\mathsf{Dec}_{\mathsf{sk}_1}(c_1) \oplus \mathsf{Dec}_{\mathsf{sk}_2}(c_2) = f(x_1, x_2)$ for any function f of the evaluator's choice. This results in a particularly simple two-round MPC protocol where the parties can locally compute an additive secret sharing of the output at the end of the first round. It also gives a strong counterexample to a method proposed by Aiello et al. [1] to construct succinct arguments for NP using homomorphic encryption.

Finally, Brakerski and Perlman [28] and Peikert and Shiehian [117], concurrently and using different methods, showed how to make the scheme of [41, 110] into a multihop multikey FHE scheme; that is, while the [41, 110] scheme required the evaluator to know all the keys involved in the computation in advance, the scheme of [117] does not.

1.6. Roadmap. We have given a high-level overview of our results. Detailed descriptions of all the results highlighted in this introduction can be found in the corresponding sections.

In section 2 we present preliminaries, definitions, and technical tools used throughout the remaining chapters.

In section 3, we define multikey FHE and describe several constructions. In particular, we show that any FHE is inherently multikey for a constant number of keys and that the ring-based FHE scheme of Brakerski and Vaikuntanathan is somewhat homomorphic for a logarithmic number of keys. More importantly, we show that the NTRU encryption scheme can be made multikey fully homomorphic for any number of keys.

In section 4 we show how to construct on-the-fly MPC from multikey FHE. We show a basic protocol that is secure against semimalicious corruptions and then describe how to modify it to achieve security against malicious adversaries. We also show how to construct efficient NIZKs (in the random oracle model) for proving plaintext knowledge for the NTRU-based FHE scheme described in section 3. Finally, we show that a completely noninteractive solution is impossible.

2. Definitions and preliminaries.

2.1. Notation. In this work, 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 $[n] \stackrel{\mathsf{def}}{=} \{1, \ldots, n\}$. 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, X and Y, we use $X \stackrel{c}{\approx} Y$ to mean that X and Y are computationally indistinguishable and $X \stackrel{s}{\approx} Y$ to mean that they are statistically close.

2.2. Σ-protocols and zero-knowledge proofs.

Σ-protocols. We recall the notion of *gap* Σ-protocols [7], a weaker version of Σ-protocols [47], where honest-verifier zero-knowledge (HVZK) holds for all statements in some NP relation R_{zk} but soundness only holds w.r.t. $R_{\mathsf{sound}} \supseteq R_{\mathsf{zk}}$. In other words,

zero-knowledge is guaranteed for an honest prover holding a statement in R_{zk}, but an honest verifier is only convinced that the statement is in a larger set $R_{\mathsf{sound}} \supseteq R_{\mathsf{zk}}$.

DEFINITION 2.1 (gap Σ-protocol). *Let R_{zk} and R_{sound} be two NP relations such that $R_{\mathsf{zk}} \subseteq R_{\mathsf{sound}} \subseteq \{0,1\}^* \times \{0,1\}^*$, and let L_{zk} and L_{sound} be their corresponding NP languages. A gap Σ-protocol for $(R_{\mathsf{zk}}, R_{\mathsf{sound}})$ is a 3-step interactive protocol $\langle P, V \rangle$ between a prover $P = (P_1, P_2)$ and a verifier $V = (V_1, V_2)$ with the following syntax:*
- *$(a, st) \leftarrow P_1(x, w)$: Given a statement and witness pair (x, w), outputs a message a and a state string st.*
- *$c \leftarrow V_1(x, a)$: Given a statement x and message a, outputs a random challenge c from a challenge space $\mathcal{C}$.*
- *$z \leftarrow P_2(st, c)$: Given a state string st and a challenge c, outputs an answer z.*
- *$b \leftarrow V_2(x, a, c, z)$: Given a statement x, a message a, a challenge c, and an answer z, outputs a bit b, i.e., either accepts or rejects the transcript (a, c, z) for statement x.*

We require that the following three properties hold:

Completeness: *For any $(x, w) \in R_{\mathsf{zk}}$,*

$$\Pr\left[V_2(x, a, c, z) = 1 \;\middle|\; \begin{array}{c} (a, st) \leftarrow P_1(x, w) \\ c \leftarrow V_1(x, a) \\ z \leftarrow P_2(st, c) \end{array} \right] = 1.$$

Special soundness: *There exists an "extractor" such that for any two accepting transcripts (a, c, z) and (a, c', z') for the same statement x with $c \neq c'$, the extractor outputs a valid witness for $x \in R_{\mathsf{sound}}$. Formally, there exists a PPT algorithm Ext such that for all x and all (a, c, z) and (a, c', z') such that $c \neq c'$ and $V_2(x, a, c, z) = V_2(x, a, c', z') = 1$:*

$$\Pr\left[(x, w) \notin R_{\mathsf{sound}} \;\middle|\; w \leftarrow \mathsf{Ext}(x, a, c, z, c', z') \right] = 1.$$

Honest-verifier zero knowledge: *There exists a PPT simulator Sim that "simulates" valid transcripts without knowing a witness if it sees the challenge beforehand. Formally, there exists PPT algorithm Sim such that for all $(x, w) \in R_{\mathsf{zk}}$ and all $c \in \mathcal{C}$, we have*

$$\left[(a, c, z) \;\middle|\; \begin{array}{c} (a, st) \leftarrow P_1(x, w) \\ z \leftarrow P_2(st, c) \end{array} \right] \stackrel{s}{\approx} \left[(a', c, z') \;\middle|\; (a', z') \leftarrow \mathsf{Sim}(x, c) \right].$$

For an NP relation R with corresponding language L, a well-known construction using Σ-protocols allows a prover to show that either $x_0 \in L$ or $x_1 \in L$ *without revealing which one holds*. Suppose $\langle P, V \rangle$ is a Σ-protocol for R such that the challenge space $\mathcal{C}$ is a finite additive group. We construct a new protocol for proving that either $x_0 \in L$ or $x_1 \in L$. Let b be such that $(x_b, w_b) \in R$ for some witness w_b known to the prover. The prover chooses c_{1-b} at random from the challenge space $\mathcal{C}$ and runs $(a_b, st) \leftarrow P_1(x_b, w_b)$ and $(a_{1-b}, z_{1-b}) \leftarrow \mathsf{Sim}(x, c_{1-b})$. It sends (a_0, a_1) to the verifier, who returns a challenge c. The prover computes $c_b = c - c_{1-b}$, runs $z_b \leftarrow P_2(st, c)$, and sends (c_0, c_1, z_0, z_1) to the verifier, who checks that $V_2(x_0, a_0, c_0, z_0) = V_2(x_1, a_1, c_1, z_1) = 1$ and $c = c_0 + c_1$. The resulting protocol is called an *OR Σ-protocol*. The theorem below modifies this to the setting of *gap Σ-protocols*.

THEOREM 2.1. *Let R_{zk} and R_{sound} be two NP relations such that $R_{\mathsf{zk}} \subseteq R_{\mathsf{sound}} \subseteq \{0,1\}^* \times \{0,1\}^*$, and let $\langle P, V \rangle$ be a gap Σ-protocol for $(R_{\mathsf{zk}}, R_{\mathsf{sound}})$ such that the*

challenge space is a finite additive group. The construction described above is a gap OR Σ-protocol for $(R_{\sf zk}, R_{\sf sound})$.

Noninteractive zero-knowledge. We also recall the notion of *noninteractive zero-knowledge (NIZK)* [24]. For our purposes, it is more convenient to use the notion of *(same-string) NIZK arguments* from [126]. This definition and all our constructions that use it can be extended in the natural way to NIZK proofs, where soundness holds for all unbounded adversaries.[5] Furthermore, similar to our presentation of Σ-protocols, we choose to present the notion of *gap* NIZKs, where zero-knowledge holds for all statements in some NP relation $R_{\sf zk}$ but soundness only holds w.r.t. $R_{\sf sound} \supseteq R_{\sf zk}$. In other words, zero-knowledge is guaranteed for an honest prover holding a statement in $R_{\sf zk}$, but an honest verifier is only convinced that the statement is in a larger set $R_{\sf sound} \supseteq R_{\sf zk}$.

DEFINITION 2.2 (gap NIZK). *Let $R_{\sf zk}$ and $R_{\sf sound}$ be two NP relations such that $R_{\sf zk} \subseteq R_{\sf sound} \subseteq \{0,1\}^* \times \{0,1\}^*$, and let $L_{\sf zk}$ and $L_{\sf sound}$ be their corresponding NP languages. A* gap NIZK argument system *for $(R_{\sf zk}, R_{\sf sound})$ consists of three algorithms* (Setup, Prove, Verify) *with syntax:*

- $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa)$*: Outputs a CRS* crs *and a trapdoor key* tk *to the CRS.*
- $\pi \leftarrow \mathsf{Prove}_{\sf crs}(x, w)$*: Given a statement and witness pair (x, w), outputs an argument π.*
- $0/1 \leftarrow \mathsf{Verify}_{\sf crs}(x, \pi)$*: Given a statement x and an argument π, verifies whether or not the argument π is correct.*

For the sake of clarity, we write Prove *and* Verify *without the* crs *in the subscript when the* crs *can be inferred from context. We require that the following three properties hold:*

Completeness: *For any $(x, w) \in R_{\sf zk}$,*

$$\Pr\left[\ \mathsf{Verify}(x, \pi) = 1 \ \middle| \ \begin{array}{c} (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa) \\ \pi \leftarrow \mathsf{Prove}(x, w) \end{array} \right] = 1.$$

Soundness: *For any* PPT *adversary $\widetilde{P}$,*

$$\Pr\left[\ \begin{array}{c} \mathsf{Verify}(x^*, \pi^*) = 1 \\ x^* \notin L_{\sf sound} \end{array} \ \middle| \ \begin{array}{c} (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa) \\ (x^*, \pi^*) \leftarrow \widetilde{P}(\mathsf{crs}) \end{array} \right] = \mathsf{negl}(\kappa).$$

Unbounded zero-knowledge: *There exists a* PPT *simulator* Sim *that "simulates" valid proofs without knowing a witness but with the aid of the trapdoor key. We start by defining two oracles.*

The Prover Oracle: *A query to the prover oracle $\mathcal{P}(\cdot)$ consists of a pair (x, w). The oracle checks if $(x, w) \in R_{\sf zk}$. If so, it outputs a valid argument* Prove(x, w)*; otherwise it outputs $\perp$.*

The Simulation Oracle: *A query to the simulation oracle $\mathcal{SIM}_{\sf tk}(\cdot)$ consists of a pair (x, w). The oracle checks if $(x, w) \in R$. If so, it ignores w and outputs a simulated argument* Sim(tk, x)*; otherwise it outputs $\perp$.*

Formally, we require that for any PPT *adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game is negligible (in κ):*

- *The challenger samples $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\kappa)$ and gives* crs *to $\mathcal{A}$. The challenger also samples a bit $b \leftarrow \{0, 1\}$.*

---

[5]Apart from modifying the soundness condition, in the setting of proofs key generation samples a CRS but not a trapdoor, and the zero-knowledge simulator first samples a *simulated CRS* that is computationally indistinguishable from the real CRS and a trapdoor to this CRS.

- If $b = 0$, the adversary $\mathcal{A}$ is given access to the prover oracle $\mathcal{P}(\cdot)$. If $b = 1$, $\mathcal{A}$ is given access to the simulation oracle $\mathcal{SIM}_{\mathsf{tk}}(\cdot)$. In either case, the adversary can adaptively access its oracle.
- The adversary $\mathcal{A}$ outputs a bit $\tilde{b}$.

The advantage of $\mathcal{A}$ is defined to be $|\Pr[\tilde{b} = b] - \frac{1}{2}|$.

Fiat and Shamir [60] showed how to convert a Σ-protocol $\langle P, V \rangle$ for an NP relation R into a NIZK argument for R secure in the random oracle model [13]. Informally, the CRS contains a description of a hash function H, which is modeled as a random oracle. To compute a noninteractive argument, the prover runs $(a, st) \leftarrow P_1(x, w)$ and obtains the verifier's challenge by applying the hash function to a and x: $c := H(a, x)$. It then computes $z \leftarrow P_2(st, c)$ and sends the argument $\pi = (a, c, z)$. The verifier runs $V_2(x, a, c, z)$ to verify the argument. The theorem below modifies this to the setting of *gap* Σ-protocols and *gap* NIZKs.

THEOREM 2.2 (see [60]). *Let R_{zk} and R_{sound} be two NP relations such that $R_{\mathsf{zk}} \subseteq R_{\mathsf{sound}} \subseteq \{0,1\}^* \times \{0,1\}^*$, and let $\langle P, V \rangle$ be a gap Σ-protocol for $(R_{\mathsf{zk}}, R_{\mathsf{sound}})$. Applying the Fiat–Shamir transform to $\langle P, V \rangle$ yields a gap NIZK argument system where soundness holds w.r.t. R_{sound} and completeness and zero-knowledge hold w.r.t. R_{zk}, secure in the random oracle model.*

Though secure in the random oracle model, we remark that in some cases standard-model security of the resulting NIZK appears to be harder to achieve [49, 22]. In particular, if the language L is quasi-polynomially hard and the protocol has messages of size $\mathsf{polylog}(\kappa)$ and is $\kappa^{\log \kappa}$-HVZK, then the resulting NIZK cannot be proven sound via a black-box reduction to a (superpolynomially hard) falsifiable assumption [112].

2.3. Succinct noninteractive arguments: SNARGs and SNARKs. We review the definitions of succinct noninteractive arguments (SNARGs) and succinct noninteractive arguments of knowledge (SNARKs); we use the formalization of Gentry and Wichs [80] and Bitansky et al. [19]. As in the work of Bitansky et al., we allow the proof size to be polynomial in the size of the statement but require it to be polylogarithmic in the size of the witness. We also require fast proof verification.

DEFINITION 2.3 (SNARG). *Let R be an NP relation on pairs (x, w) with corresponding language $L = \{x \mid \exists w \ s.t. \ (x, w) \in R\}$. A SNARG system for L consists of three algorithms* (Setup, Prove, Verify) *with syntax:*

- $(\mathsf{vrs}, \mathsf{priv}) \leftarrow \mathsf{Setup}(1^\kappa)$: *Outputs a verification reference string* vrs *and a private verification state* priv.
- $\varphi \leftarrow \mathsf{Prove}(\mathsf{vrs}, x, w)$: *Outputs an argument* φ *showing that* $R(x, w) = 1$.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{priv}, x, \varphi)$: *Verifies whether or not the argument* φ *is correct.*

We require that the following properties hold:

Completeness: *For any* $(x, w) \in R$,

$$\Pr\left[\ \mathsf{Verify}(\mathsf{priv}, x, \varphi) = 1 \ \middle| \ \begin{array}{l} (\mathsf{vrs}, \mathsf{priv}) \leftarrow \mathsf{Setup}(1^\kappa) \\ \varphi \leftarrow \mathsf{Prove}(\mathsf{vrs}, x, w) \end{array} \right] = 1.$$

In addition, $\mathsf{Prove}(\mathsf{vrs}, x, w)$ *runs in time* $\mathsf{poly}(\kappa, |x|, |w|)$.

Adaptive soundness: *For any* PPT *adversary* $\widetilde{P}$,

$$\Pr\left[\ \begin{array}{l} \mathsf{Verify}(\mathsf{priv}, x^*, \varphi^*) = 1 \ \wedge \\ x^* \notin L \end{array} \ \middle| \ \begin{array}{l} (\mathsf{vrs}, \mathsf{priv}) \leftarrow \mathsf{Setup}(1^\kappa) \\ (x^*, \varphi^*) \leftarrow \widetilde{P}(\mathsf{vrs}) \end{array} \right] = \mathsf{negl}(\kappa).$$

Succinctness: *The length of the proof and the time required for its verification are polylogarithmic in the size of the witness, i.e.,* $\mathsf{poly}(\kappa) \left(\mathsf{poly}(|x|) + \mathsf{polylog}(|w|) \right)$.

DEFINITION 2.4 (SNARK). *A SNARG* $\Phi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *is additionally a proof of knowledge or a SNARK if it satisfies the following stronger definition of soundness:*

Adaptive extractability: *There exists an extractor* Ext *that "extracts" a valid witness from any valid proof* φ. *Formally, for any* PPT *adversary* $\widetilde{P}$, *there exists a* PPT *algorithm* Ext *such that*

$$\Pr \left[\begin{array}{c} \mathsf{Verify}(\mathsf{priv}, x^*, \varphi^*) = 1 \ \wedge \\ R(x^*, w') = 0 \end{array} \ \middle| \ \begin{array}{c} (\mathsf{vrs}, \mathsf{priv}) \leftarrow \mathsf{Setup}(1^\kappa) \\ (x^*, \varphi^*) \leftarrow \widetilde{P}(\mathsf{vrs}) \\ w' \leftarrow \mathsf{Ext}(x^*, \varphi^*) \end{array} \right] = \mathsf{negl}(\kappa).$$

Public vs. private verifiability. In the case where $\mathsf{priv} = \mathsf{vrs}$, we say that the SNARG or SNARK is *publicly verifiable.* In this case, anyone can verify all proofs. Otherwise, we say that it is a *designated-verifier* SNARG/SNARK, in which case soundness/extractability is only guaranteed as long as priv remains secret to the prover. In this case, only the party holding priv can verify the proof.

2.3.1. Delegation of computation from SNARGs. In delegation of computation we are concerned with a client C, who wishes to delegate the computation of a prespecified polynomial-time algorithm M on an input x, to a worker W. The client additionally wishes to verify the correctness of the output y returned by W (i.e., verify that $y = M(x)$) in time that is significantly smaller than the time required to compute $M(x)$ from scratch.

SNARGs can be used in this setting as follows: Define the NP language: $L_M = \{ (x, y) \text{ such that } M(x) = y \}$. A straightforward witness to the statement $(x, y) \in L_M$ consists of the steps taken by M in a computation of $M(x)$ resulting in the output y. The size of this witness is proportional to the size of the computation. Using a SNARG guarantees that the size of the proof is polylogaritmic in the size of the witness and therefore polylogarithmic in the size of the computation.

2.3.2. Constructions. Gentry and Wichs [80] proved that standard-model security of SNARGs with adaptive soundness and proof size sublinear in the witness and statement sizes cannot be based on any falsifiable assumption [112]. The constructions we show below either assume a random oracle [13] or a nonfalsifiable assumption.

CS proofs. Kilian [99, 100] showed how to perform succinct *interactive* verification for any NP language. His solution describes a 4-round protocol, where the prover first constructs a PCP for the correctness of the computation and then uses Merkle hashes to compress it to a sufficiently small proof. Micali's CS proofs [108] apply the Fiat–Shamir transform [60] to Kilian's protocol, obtaining a noninteractive solution. CS proofs are publicly verifiable SNARGs (and SNARKs under Valiant's analysis [131]); indeed, the only "setup" required is a description of a hash function H to use as the random oracle. This can be ensured by letting the vrs be a random key for a (say) collision-resistant hash function.

Due to its use of the Fiat–Shamir transform, Micali's solution is only secure in the *random oracle model* [13]. Unfortunately, several results have shown the implausibility of instantiating the random oracle in the Fiat–Shamir transform with any explicit hash function [94, 9, 33, 58, 86]. In particular, Dachman-Soled and co-authors [49, 22] show that the security of CS proofs (even with nonadaptive soundness) cannot be based on any falsifiable assumption. On the other hand, it has been shown that the security of the Fiat–Shamir paradigm can be based on specific nonfalsifiable assumptions regarding the existence of robust randomness condensers for seed-dependent sources [12, 56].

Constructions will small CRS. Bitansky et al. [19, 20] and Goldwasser, Lin, and Rubinstein [89] revisit the construction of CS proofs and, based on the works of Di Crescenzo and Lipmaa [48] and Valiant [131], show how to construct SNARGs and SNARKs based on a different nonfalsifiable assumption relating to the existence of extractable collision-resistant hash functions. In these works, the verifier's entire computation (both in computing its reference string vrs and in verifying the proof) depends only polylogarithmically in the size of the witness (i.e., the delegated computation).

Allowing a large CRS. Another series of works constructs SNARGs and SNARKs where the verifier's reference string vrs is allowed to depend on the circuit being delegated. In particular, Groth's construction [90] has a CRS of size quadratic in the circuit size. Lipmaa [103] reduces this size to be quasi-linear, and the works of Gennaro et al. [69] and Parno et al. [115] further reduce it to linear in the circuit size. Bitansky et al. [21] give a framework for construction of such SNARKs. Further improvements and variants are given by [104, 59, 17, 105, 102, 136, 54, 134, 8, 91]. All of these constructions are based on certain nonfalsifiable assumptions.

2.4. Secure multiparty computation. Let f be an N-input function with single output. A *multiparty protocol* Π for f is a protocol between N interactive Turing machines $P_1, \ldots, P_N$, called *parties*, such that for all $\vec{x} = (x_1, \ldots, x_N)$, the output of Π in an execution where P_i is given x_i as input, is $y \stackrel{\mathsf{def}}{=} f(\vec{x})$.

2.4.1. Security in the ideal/real paradigm. Informally, a multiparty protocol Π is *secure* if after running Π, no colluding set of corrupt parties can learn anything about an honest player's input or change the output of an honest party. We formalize this in the ideal/real paradigm (see, e.g., [82]).

Ideal and real worlds. We define an *ideal world* in which the computation of f is performed through a trusted functionality $\mathcal{F}$ that receives inputs x_i from each party P_i, computes $y \stackrel{\mathsf{def}}{=} f(x_1, \ldots, x_N)$, and gives y to all parties $P_1, \ldots, P_N$. It is clear that in the ideal world, the only information that any party learns is its own input and the output y. We also define a *real world* in which parties $P_1, \ldots, P_N$ run the protocol Π.

The network. We assume that the real-world execution of the protocol is performed over a *secure* and *synchronous* network; that is, we assume that parties can reliably send messages to other parties without these being read or altered in transmission and that all point-to-point communications happen at the same time. We also assume that a secure *broadcast* channel is available to all parties.

The adversary. In either world, we consider a single adversary that is allowed to corrupt any subset of $t < N$ parties. An adversary is modeled as an interactive Turing machine that receives all messages directed to the corrupted parties and controls the messages sent by them. In this work, we consider only *static* adversaries, that is, adversaries that select the subset of corrupted parties nonadaptively, before any computation is performed. On the other hand, we assume that in each round of the protocol, the adversary chooses the messages for the corrupted parties adaptively, based on the entire transcript of the protocol, up to that round.

We remark that our results can be extended to achieve security against *rushing* real-world adversaries who, on any given round, choose the messages for the corrupted parties adaptively, based on the entire transcript of the protocol *and the messages of the honest parties on that round*. Note that rushing adversaries correspond to a semisynchronous model of communication.

Output distributions. We use $\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\vec{x})$ to denote the joint output of an ideal-world adversary $\mathcal{S}$ and parties $P_1, \ldots, P_N$ in an ideal execution with functionality $\mathcal{F}$ and inputs $\vec{x} = (x_1, \ldots, x_N)$. Similarly, we use $\text{REAL}_{\Pi,\mathcal{A}}(\vec{x})$ to denote the joint output of a real-world adversary $\mathcal{A}$ and parties $P_1, \ldots, P_N$ in an execution of protocol Π with inputs $\vec{x} = (x_1, \ldots, x_N)$.

We say that a protocol Π *securely realizes $\mathcal{F}$ against the class of adversaries* Adv if for every real-world adversary $\mathcal{A} \in \mathsf{Adv}$, there exists an ideal-world adversary $\mathcal{S}$ with black-box access to $\mathcal{A}$ such that for all input vectors $\vec{x}$,

$$\text{IDEAL}_{\mathcal{F},\mathcal{S}}(\vec{x}) \overset{c}{\approx} \text{REAL}_{\Pi,\mathcal{A}}(\vec{x}).$$

2.4.2. Types of adversaries. As stated above, in this work we only consider classes of adversaries Adv containing static adversaries that corrupt any subset of $t < N$ parties. We now describe three different types of adversaries: malicious, semihonest, and semimalicious. The first two are used extensively in the literature, while the latter was introduced recently by Asharov et al. [7, 6]. Of these, malicious adversaries are the strongest, and it is our end goal to achieve security against them in all our protocols.

It is customary to prove security against semihonest adversaries as a stepping stone to proving security against malicious adversaries. However, in this work we follow a different path and first prove security against semimalicious adversaries. We then show how to modify the protocol at hand to achieve security against malicious adversaries. For completeness, we describe all three types of adversaries below and describe how security against one type is related to security against another.

Semihonest adversaries. A semihonest adversary, also known as an honest-but-curious adversary, is one that follows the protocol as described (samples randomness from the correct distribution and computes the specified message at each round) but given its view of the protocol will try to learn information about honest players' inputs.

Malicious adversaries. A malicious adversary is not restricted in how it samples random elements or how it computes its messages at each round. It can sample random elements from any arbitrary distribution and compute the messages of corrupted parties in any arbitrary way, adaptively, according to the partial view it has seen up to that point.

Semimalicious adversaries. Recall that an adversary is modeled as an interactive Turing machine (ITM). A semimalicious adversary is an ITM with an additional *witness tape*. At each round ℓ and for every corrupted party P_j, the adversary must write on the special witness tape, *some* witness pair $(x_j^{(\ell)}, r_j^{(\ell)})$ of input and randomness that explains the message $m_j^{(\ell)}$ sent by P_j on that round. More formally, the messages of a corrupted party P_j must match those of the specified honest protocol when at each round ℓ party P_j is run with input and randomness $(x_j^{(\ell)}, r_j^{(\ell)})$.

A semimalicious adversary can sample random elements from any arbitrary distribution, but it must follow the correct behavior of the honest protocol with inputs and randomness that it *knows*. It is therefore weaker than a malicious adversary, who might not know witnesses for the messages it sends at every round, but stronger than a semihonest adversary, whose witnesses at every round are distributed honestly.

From semimalicious to malicious security. Asharov et al. [7, 6] show how to generically transform a protocol that is secure against semimalicious adversaries into one that is secure against malicious adversaries. The idea behind the compiler is to have each party prove in zero-knowledge that every message it sends follows the

honest protocol and is consistent with all previous messages. In particular, this forces all parties to know witnesses that explain their behavior at every round. The same compiler works in our security model with one subtlety: instead of using standard zero-knowledge proofs, the protocol must use *zero-knowledge proofs of knowledge*. This is to ensure that the simulator can extract the witness $w_j^{(\ell)}$ from the proof sent on round ℓ by the malicious adversary on behalf of the corrupted party P_j. We refer the reader to the work of Asharov et al. [7, 6] for more details.

Finally, we note that unlike the standard GMW compiler from semihonest security to malicious security [84], the parties are not required to perform any coin-flipping. This, in particular, reduces the round complexity of the resulting protocol.

2.5. Fully homomorphic encryption. We review the definitions of fully and leveled homomorphic encryption.

DEFINITION 2.5 (*$\mathcal{C}$-homomorphic encryption* [71]). *For a class of circuits $\mathcal{C}$, a $\mathcal{C}$-homomorphic encryption scheme is a tuple of algorithms $\mathcal{E} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ with the following syntax:*

- $\mathsf{params} \leftarrow \mathsf{Setup}(1^\kappa)$: *For security parameter κ, outputs public parameters params. All other algorithms, $\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}$, implicitly take params as input, even when not explicitly stated.*
- $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Keygen}(1^\kappa)$: *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 pk and a message m, outputs a ciphertext c.*
- $m := \mathsf{Dec}(\mathsf{sk}, c)$: *Given a secret key sk and a ciphertext c, outputs a message m.*
- $c := \mathsf{Eval}(\mathsf{ek}, C, c_1, \ldots, c_\ell)$: *Given an evaluation key ek, a (description of a) circuit C and ℓ ciphertexts $c_1, \ldots, c_\ell$, outputs a ciphertext c.*

We require that for all $c \in \mathcal{C}$, all $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$ in the support of $\mathsf{Keygen}(1^\kappa)$ and all plaintexts $(m_1, \ldots, m_\ell)$ and ciphertexts $(c_1, \ldots, c_\ell)$ such that c_i is in the support of $\mathsf{Enc}(\mathsf{pk}, m_i)$, if $c := \mathsf{Eval}(\mathsf{ek}, C, c_1, \ldots, c_\ell)$, then $\mathsf{Dec}(\mathsf{sk}, c) = C(m_1, \ldots, m_\ell)$.

DEFINITION 2.6 (*fully homomorphic encryption* [71]). *An encryption scheme $\mathcal{E}$ is fully homomorphic if it satisfies the following properties:*
Correctness: *$\mathcal{E}$ is $\mathcal{C}$-homomorphic for the class $\mathcal{C}$ of all circuits.*
Compactness: *The computational complexity of $\mathcal{E}$'s algorithms is polynomial in the security parameter κ, and in the case of the evaluation algorithm, the size of the circuit.*

We now state the definition of leveled homomorphic encryption from [27], which is a relaxation of the original definition of fully homomorphic encryption (Definition 2.6). The main difference is that Definition 2.6 requires all algorithms (decryption in particular) to be independent of the circuit(s) that the scheme can evaluate. Leveled homomorphic encryption relaxes this definition to let all algorithms (including decryption) depend on the circuit depth D.

DEFINITION 2.7 (*leveled homomorphic encryption* [27]). *Let $\mathcal{C}^{(D)}$ be the class of all circuits of depth at most D (that use some specified complete set of gates). We say that a family of homomorphic encryption schemes $\{\mathcal{E}^{(D)} : D \in \mathbb{Z}^+\}$ is leveled fully homomorphic if, for all $D \in \mathbb{Z}^+$, it satisfies the following properties:*
Correctness: *$\mathcal{E}^{(D)}$ is $\mathcal{C}^{(D)}$-homomorphic.*
Compactness: *The computational complexity of $\mathcal{E}^{(D)}$'s algorithms is polynomial in the security parameter κ and D, and in the case of the evaluation algorithm, the*

size of the circuit. We emphasize that this polynomial must be the same for all D.

2.5.1. Bootstrapping. We remind the reader of the definition of a bootstrappable encryption scheme and present Gentry's bootstrapping theorem [71, 70] that states that a bootstrappable scheme can be converted into a fully homomorphic one.

DEFINITION 2.8 (bootstrappable scheme). *Let $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a $\mathcal{C}$-homomorphic encryption scheme, and let f_{add} and f_{mult} be the augmented decryption functions of the scheme defined as*

$$f_{\mathsf{add}}^{c_1,c_2}(\mathsf{sk}) = \mathsf{Dec}(\mathsf{sk}, c_1) \quad XOR \quad \mathsf{Dec}(\mathsf{sk}, c_2),$$
$$f_{\mathsf{mult}}^{c_1,c_2}(\mathsf{sk}) = \mathsf{Dec}(\mathsf{sk}, c_1) \quad AND \quad \mathsf{Dec}(\mathsf{sk}, c_2).$$

$\mathcal{E}$ *is* bootstrappable *if* $\{f_{\mathsf{add}}^{c_1,c_2}, f_{\mathsf{mult}}^{c_1,c_2}\}_{c_1,c_2} \subseteq \mathcal{C}$, *namely, if it can homomorphically evaluate* f_{add} *and* f_{mult}.

DEFINITION 2.9 (weak circular security). *A public-key encryption scheme $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ 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 this additional information.*

THEOREM 2.3 (bootstrapping theorem). *Let $\mathcal{E}$ be a bootstrappable scheme that is also weakly circular secure. Then there exists a fully homomorphic encryption scheme $\mathcal{E}'$.*

2.6. Rings. In this section we introduce preliminaries to our concrete constructions, which are all ring-based. Some of the discussion in this section is taken verbatim from the work of Brakerski and Vaikuntanathan [30].

We work over rings $R \overset{\mathsf{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$ and $R_q \overset{\mathsf{def}}{=} R/qR$ for some degree $n = n(\kappa)$ integer polynomial $\phi(x) \in \mathbb{Z}[x]$ and a prime integer $q = q(\kappa) \in \mathbb{Z}$. Note that R_q is isomorphic to $\mathbb{Z}_q[x]/\langle \phi(x) \rangle$, the ring of degree n polynomials modulo $\phi(x)$ with coefficients in $\mathbb{Z}_q$. Addition in these rings is done componentwise in their coefficients (thus, their additive group is isomorphic to $\mathbb{Z}^n$ and $\mathbb{Z}_q^n$, respectively), and multiplication is polynomial multiplication modulo $\phi(x)$ (and also q, in the case of the ring R_q). An element in R (or R_q) can be viewed as a polynomial of degree at most $(n-1)$ over $\mathbb{Z}$ (or $\mathbb{Z}_q$). We represent such an element using the vector of its n coefficients. In the case of R_q each coefficient is in the range $\{-\lfloor \frac{q}{2} \rfloor, \ldots, \lfloor \frac{q}{2} \rfloor\}$. For an element $a(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in R$, we let $\|a\|_\infty = \max |a_i|$ denote its ℓ_∞ norm.

In this work, we set $\phi(x) = x^n + 1$, where n is a power of two, and use distributions over the ring $R \overset{\mathsf{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$. For the purpose of homomorphism, the only important property of these distributions is the magnitude of the coefficients of a polynomial output by the distribution. Hence, we define a B-bounded distribution to be a distribution over R where the ℓ_∞-norm of a sample is bounded by B.

DEFINITION 2.10 (B-bounded polynomial). *A polynomial $e \in R$ is called B-bounded if $\|e\|_\infty \leq B$.*

DEFINITION 2.11 (B-bounded distribution). *A distribution ensemble $\{\chi_\kappa\}_{\kappa \in \mathbb{N}}$, supported over R, is called B-bounded (for $B = B(\kappa)$) if for all e in the support of χ_κ, we have $\|e\|_\infty < B$. In other words, a B-bounded distribution over R outputs a B-bounded polynomial.*

The following lemma 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 2.4. *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$, and let $R = \mathbb{Z}[x]/\langle \phi(x)\rangle$. For any $s, t \in R$,*

$$\|s \cdot t\| \le \sqrt{n} \cdot \|s\| \cdot \|t\| \quad and \quad \|s \cdot t\|_\infty \le n \cdot \|s\|_\infty \cdot \|t\|_\infty.$$

Lemma 2.4 yields the following corollary.

COROLLARY 2.5. *Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$, and let $R = \mathbb{Z}[x]/\langle \phi(x)\rangle$. Let χ be a B-bounded distribution over the ring R and let $s_1, \ldots, s_k \leftarrow \chi$. Then $s \stackrel{\mathsf{def}}{=} \prod_{i=1}^{k} s_i$ is $(n^{k-1}B^k)$-bounded.*

2.6.1. Discrete Gaussians. For any real $r > 0$ the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter r is defined as

$$\forall \mathbf{x} \in \mathbb{R}^n \ : \quad \rho_{r,\mathbf{c}}(\mathbf{x}) \stackrel{\mathsf{def}}{=} e^{-\pi\|\mathbf{x}-\mathbf{c}\|^2/r^2}.$$

DEFINITION 2.12. *For any $n \in \mathbb{N}$ and for any $\mathbf{c} \in \mathbb{R}^n$ and real $r > 0$, the* discrete Gaussian distribution *over $\mathbb{Z}^n$ with standard deviation r and centered at $\mathbf{c}$ is defined as*

$$\forall \mathbf{x} \in \mathbb{Z}^n \ : \quad D_{\mathbb{Z}^n,r,\mathbf{c}} \stackrel{\mathsf{def}}{=} \frac{\rho_{r,\mathbf{c}}(\mathbf{x})}{\rho_{r,\mathbf{c}}(\mathbb{Z}^n)},$$

where $\rho_{r,\mathbf{c}}(\mathbb{Z}^n) \stackrel{\mathsf{def}}{=} \sum_{\mathbf{x}\in\mathbb{Z}^n} \rho_{r,\mathbf{c}}(\mathbf{x})$ is a normalization factor.

We present some elementary facts about the Gaussian distribution. The first fact shows that the discrete Gaussian distribution over $\mathbb{Z}^n$ with standard deviation 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.

LEMMA 2.6 (see [109]). *For any real number $r > \omega(\sqrt{\log n})$, we have*

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

Using Lemma 2.6 together with the fact that for all $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\| \ge \|\mathbf{x}\|_\infty$ yields the following bound.

LEMMA 2.7. *Let $n = \omega(\log \kappa)$. For any real number $r > \omega(\sqrt{\log n})$, we have*

$$\Pr_{\mathbf{x} \leftarrow D_{\mathbb{Z}^n,r}} \left[\|\mathbf{x}\|_\infty > r\sqrt{n}\right] \le 2^{-n+1} = \mathsf{negl}(\kappa).$$

Define the *truncated discrete Gaussian distribution* with standard deviation r and centered at $\mathbf{c}$, denoted by $\overline{D}_{\mathbb{Z}^n,r,\mathbf{c}}$, to be one that samples a polynomial according to the discrete Gaussian $D_{\mathbb{Z}^n,r,\mathbf{c}}$ and repeats the sampling if the polynomial is not $(r\sqrt{n})$-bounded. As long as $n = \omega(\log(\kappa))$, Lemma 2.7 implies that this distribution is statistically close to the discrete Gaussian : $\overline{D}_{\mathbb{Z}^n,r,\mathbf{c}} \approx_s D_{\mathbb{Z}^n,r,\mathbf{c}}$.

The second fact says that the statistical distance between a discrete Gaussian with standard deviation r and centered at 0 and one centered at $\mathbf{c} \in \mathbb{Z}^n$ is at most $\|\mathbf{c}\|/r$. In particular, if r is superpolynomially larger than $\|\mathbf{c}\|$, then the two distributions are statistically close.

LEMMA 2.8 (see [87, Lemma 3]). *Let $n \in \mathbb{N}$. For any real number $r > \omega(\sqrt{\log n})$ and any $\mathbf{c} \in \mathbb{Z}^n$, the statistical distance between the distributions $D_{\mathbb{Z}^n,r}$ and $D_{\mathbb{Z}^n,r,\mathbf{c}}$ is at most $\|\mathbf{c}\|/r$.*

COROLLARY 2.9. *Let $\mathbf{c} \in \mathbb{Z}^n$. For any real number $r \geq 2^{\omega(\log \kappa)} \|\mathbf{c}\|$, the distributions $D_{\mathbb{Z}^n,r}$ and $D_{\mathbb{Z}^n,r,\mathbf{c}}$ are statistically close.*

2.6.2. The ring-LWE and polynomial LWE assumptions. We now discuss the ring learning with errors (RLWE) assumption introduced by Lyubaskevsky, Peikert, and Regev [107] and a close variant called Polynomial Learning with Errors (PLWE) introduced by Brakerski and Vaikuntanathan [30].

The PLWE assumption is analogous to the standard LWE assumption, first defined by Regev [120, 121] (generalizing the learning parity with noise assumption of Blum et al. [23]). The $\mathsf{PLWE}_{\phi,q,\chi}$ 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 [30], 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 work and is equivalent to the original up to obtaining one additional sample [4, 107].

We note that PLWE is different from, but closely related to, the RLWE assumption defined in [107]. In particular, the two differ in the fact that in PLWE, the secret s is chosen from R whereas in RLWE, it is chosen from the "dual ring" of R (which is a fractional ideal). There are also differences in the choice of the error distribution. However, in the case of cyclotomic rings $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$, where n is a power of two, the only rings we will consider in this paper, these two assumptions are essentially the same modulo a scaling in the noise distribution. For a detailed discussion of the relation between the assumptions, we refer the reader to [107, 30, 57]. Henceforth in this paper, we will use the terms RLWE and PLWE interchangeably.

DEFINITION 2.13 (the PLWE assumption—Hermite normal form [107]). *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 an odd prime integer, let the ring $R \stackrel{\mathsf{def}}{=} \mathbb{Z}[x]/\langle\phi(x)\rangle$ and $R_q \stackrel{\mathsf{def}}{=} R/qR$, and let χ denote a distribution over the ring R.*

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

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

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 now present a couple of facts about the PLWE assumption. The first says that the assumption also holds if the error is multiplied by 2 in every sample. This follows immediately from the fact that q is an odd prime and therefore relatively prime with 2.

FACT 2.10. *The $\mathsf{PLWE}_{\phi,q,\chi}$ assumption implies that for any $\ell = \mathsf{poly}(\kappa)$,*

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

where a_i, u_i are uniformly random in R_q and s, e_i are drawn from the error distribution χ.

The second fact says that the assumption also holds if the distinguisher is additionally given samples with the same parameter a_j and different secret key s_i. This follows from a hybrid argument that slowly changes the samples, one secret at a time, from PLWE to uniform.

FACT 2.11. *The* $\mathsf{PLWE}_{\phi,q,\chi}$ *assumption implies that for any* $\ell = \mathsf{poly}(\kappa), \ell' = \mathsf{poly}(\kappa)$,

$$\{(a_j, a_j \cdot s_i + e_{i,j})\}_{i\in[\ell],j\in[\ell']} \overset{\mathrm{c}}{\approx} \{(a_j, u_{i,j})\}_{i\in[\ell],j\in[\ell']},$$

where $a_j, u_{i,j}$ *are uniformly random in* R_q *and* $s_i, e_{i,j}$ *are drawn from the error distribution* χ.

2.6.3. 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 the following hold:

- We set $\phi(x) \overset{\mathsf{def}}{=} x^n + 1$, where n is a power of two.
- The error distribution χ is the truncated discrete Gaussian distribution $\overline{D}_{\mathbb{Z}^n, r}$ with standard deviation $r > 0$. A sample from this distribution is a $(r\sqrt{n})$-bounded polynomial $e \in R$. As described above, this is done by sampling a vector $(p_0, \ldots, p_{n-1})$ from the discrete Gaussian distribution over $\mathbb{Z}^n$ and outputting a polynomial $p(x) = \sum_{0 \le i < n} p_i x^i$ whose coefficients are elements of this vector.

2.6.4. The worst-case to average-case connection. We now state a worst-case to average-case reduction from the shortest vector problem (SVP) on ideal lattices to the PLWE problem for our setting of parameters. For details and the formal definition of the (approximate) R-SVP problem, we refer the readers to the original work [107].

THEOREM 2.12 (see [107]). *Let* $\phi(x) = x^n + 1$, *where* n *is a power of two. Let* $r \ge \omega(\sqrt{\log n})$ *be a real number, and let* $q \equiv 1 \pmod{2n}$ *be a prime integer. Let* $R \overset{\mathsf{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$. *Then there is a randomized reduction from the approximate shortest vector problem on* R-*ideal lattices, namely, the* $2^{\omega(\log n)} \cdot (q/r)$-*approximate* R-SVP, *to* $\mathsf{PLWE}_{\phi,q,\chi}$, *where* $\chi = \overline{D}_{\mathbb{Z}^n, r}$ *is the truncated discrete Gaussian distribution.*

Solving approximate R-SVP to within a subexponential factor is believed to be hard. Thus, if $q/r = 2^{o(n)}$, then the $\mathsf{PLWE}_{\phi,q,\chi}$ assumption is believed to be hard.

2.7. NTRU encryption. We describe the NTRU encryption scheme of Hoffstein, Pipher, and Silverman [96], with the modifications proposed by Stehlé and Steinfeld [130]. For security parameter κ, the scheme is parameterized by a prime number $q = q(\kappa)$, a degree $n = n(\kappa)$ polynomial $\phi(x) \in \mathbb{Z}[x]$, and an error distribution $\chi = \chi(\kappa)$ over the ring $R \overset{\mathsf{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$. The parameters n, ϕ, q, χ are public and we assume that given κ, there are polynomial-time algorithms that output ϕ and q and sample from the error distribution χ. The message space is $\mathcal{M} = \{0, 1\}$, and all operations are carried out in the ring R (i.e., modulo $\phi(x)$).

- Keygen(1^κ): Sample polynomials $f', g \leftarrow \chi$ and set $f \overset{\mathsf{def}}{=} 2f' + 1$ so that $f \equiv 1 \pmod{2}$. If f is not invertible in R_q, resample f'; otherwise, let f^{-1} be the inverse of f in R_q. Set

$$\mathsf{pk} \overset{\mathsf{def}}{=} h := \left[2gf^{-1}\right]_q \in R_q \quad , \quad \mathsf{sk} \overset{\mathsf{def}}{=} f \in R.$$

- Enc(pk, m): To encrypt a bit $m \in \{0, 1\}$ with public key pk $= h$, sample polynomials $s, e \leftarrow \chi$ and output the ciphertext

$$c \stackrel{\mathrm{def}}{=} [hs + 2e + m]_q \in R_q.$$

- Dec(sk, c): To decrypt a ciphertext $c \in R_q$ with secret key sk $= f$, let $\mu \stackrel{\mathrm{def}}{=} [fc]_q$ and output $m \stackrel{\mathrm{def}}{=} \mu \pmod 2$.

It is easily seen that this scheme is correct as long as there is no reduction modulo q. To decrypt a ciphertext c, we compute

$$[fc]_q = [fhs + 2fe + fm]_q = [2gs + 2fe + fm]_q.$$

If there is no reduction modulo q, then

$$[fc]_q \pmod 2 = 2gs + 2fe + fm \pmod 2 = fm \pmod 2 = m.$$

Furthermore, our choice of parameter $\phi(x) = x^n + 1$ ensures there is no reduction modulo q. Notice that since the coefficients of g, s, e are all bounded by B and the coefficients of f are bounded by $2B + 1$. By Corollary 2.5, we know that the coefficients of $[fc]_q$ are bounded by $2nB^2(2nB + 1)(2B + 1)$. As long as we set q to be large enough so that $q/2$ is larger than this quantity, a fresh ciphertext generated by Enc is guaranteed to decrypt correctly. From here on, we refer to $\mu = [fc]_q \in R_q$ as the "*error* in ciphertext c."

2.7.1. Security. The security of the (modified) NTRU encryption scheme can be based on two assumptions—the PLWE assumption described in section 2.6 as well as an assumption that we call the (decisional) small polynomial ratio (DSPR) assumption.

DEFINITION 2.14 (decisional small polynomial ratio assumption). *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 $R \stackrel{\mathrm{def}}{=} \mathbb{Z}[x]/\langle\phi(x)\rangle$. The DSPR $\mathrm{DSPR}_{\phi,q,\chi}$ says that it is hard to distinguish the following two distributions:*
- *a polynomial $h \stackrel{\mathrm{def}}{=} [2gf^{-1}]_q$, where f' and g are sampled from the distribution χ (conditioned on $f \stackrel{\mathrm{def}}{=} 2f + 1$ and g being invertible over R_q) and f^{-1} is the inverse of f in R_q,*
- *a polynomial u sampled uniformly at random over R_q.*

The security proof uses a hybrid argument, in two steps.
1. The hardness of $\mathrm{DSPR}_{\phi,q,\chi}$ allows us to change the public key $h = [2gf^{-1}]_q$ to a uniformly sampled h.
2. Once this is done, we can use $\mathrm{PLWE}_{\phi,q,\chi}$ to change the challenge ciphertext $c^* = [hs + 2e + m]_q$ to $c^* = [u + m]_q$, 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.

Stehlé and Steinfeld [130] showed that the $\mathrm{DSPR}_{\phi,q,\chi}$ assumption is unconditionally true even for unbounded adversaries (namely, the two distributions above are statistically close) if n is a power of two, $\phi(x) = x^n + 1$, and χ is the discrete Gaussian

$D_{\mathbb{Z}^n, r}$ for $r > \sqrt{q} \cdot \mathsf{poly}(n)$. Thus, with this setting of parameters, semantic security of the modified NTRU scheme can be based on the $\mathsf{PLWE}_{\phi, q, \chi}$ assumption alone.

On the DSPR assumption. Subsequent to our work, Albrecht, Bai, and Ducas [3] and Cheon, Jeong, and Lee [37] independently presented two attacks against the NTRU scheme and in particular, the DSPR assumption, with an extreme choice of parameters. More precisely, both works show (different) polynomial-time algorithms that solve $\mathsf{DSPR}_{\phi, q, \chi}$ for a class of rings ϕ including $\mathbb{Z}[x] / \langle x^n + 1 \rangle$, when $\chi = D_{\mathbb{Z}^n, r}$ for $r = \mathsf{poly}(n)$ and $q = 2^{\tilde{O}(\sqrt{n})}$. The DSPR assumption remains unbroken for smaller q, namely, $q = 2^{n^\epsilon}$ for some small constant $\epsilon < 1/2$, to the best of our knowledge.

3. Multikey FHE. As mentioned earlier, the main building block in our construction of on-the-fly MPC is multikey FHE: fully homomorphic encryption that allows homomorphic evaluation on ciphertexts encrypted under different and independent keys. In this chapter, we formally define multikey FHE and show a construction for any number of keys based on the NTRU encryption scheme [96, 130] described in section 2.7. We also show that any FHE scheme is inherently multikey for a constant number of keys (in the security parameter) and that the Brakerski–Vaikuntanathan scheme [30, 27] is somewhat homomorphic for a logarithmic number of keys.

3.1. Definition. To formally define multikey fully homomorphic encryption, we introduce a parameter N, which is the number of distinct keys that the scheme can handle; all algorithms will depend polynomially on N. This is similar to the definition of leveled homomorphic encryption from [27] (see Definition 2.7), but 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 ("nonleveled") 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.

Finally, we note that to guarantee semantic security, decryption requires all corresponding secret keys.

DEFINITION 3.1 (multikey $\mathcal{C}$-homomorphic encryption). *Let $\mathcal{C}$ be a class of circuits. A family $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ of algorithms is* multikey $\mathcal{C}$-homomorphic *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 pk and message m, outputs a ciphertext c.*
- $m := \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$*: Given N secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ and a ciphertext c, outputs a message m.*
- $c := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_\ell, \mathsf{pk}_\ell, \mathsf{ek}_\ell))$*: Given a (description of) a boolean circuit C along with ℓ 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 ℓ key tuples $\{(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i)\}_{i \in [\ell]}$ each of which is in $\{(\mathsf{pk}'_j, \mathsf{sk}'_j, \mathsf{ek}'_j)\}_{j \in [N]}$, and all plaintexts $(m_1, \ldots, m_\ell)$ and ciphertexts $(c_1, \ldots, c_\ell)$ such that c_i is in the support of $\mathsf{Enc}(\mathsf{pk}_i, m_i)$, Eval satisfies the following properties:

Correctness: *Let* $c := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_\ell, \mathsf{pk}_\ell, \mathsf{ek}_\ell))$. *Then*
$$\mathsf{Dec}(\mathsf{sk}'_1, \ldots, \mathsf{sk}'_N, c) = C(m_1, \ldots, m_\ell).^6$$

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

DEFINITION 3.2 (multikey fully homomorphic encryption). *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 $\mathcal{C}$-homomor-phic 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 Definitions 3.1 and 3.2 yields the standard definitions of $\mathcal{C}$-homomorphic and fully homomorphic encryption schemes (Definitions 2.5 and 2.6).

3.2. Multikey FHE for a small number of keys. As a prelude to our main result in section 3.3, we show that multikey homomorphic encryption for a small number of keys can be easily achieved. In particular, we show that any (standard) FHE can be converted into a multikey FHE for a constant number of keys, $N = O(1)$. Furthermore, we show that the Brakerski–Vaikuntanathan (ring-based) FHE [30] is multikey homomorphic for a logarithmic number of keys, $N = O(\log \kappa)$. Unfortunately, once we introduce multiple keys we are unable to use either relinearization or squashing and can therefore only obtain a somewhat homomorphic encryption scheme.

3.2.1. $O(1)$-multikey FHE from any FHE. We show that any FHE scheme is inherently multikey for a constant number of keys, $N = O(1)$.[7] Let $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme with message space $\{0, 1\}$ and ciphertext space $\{0, 1\}^\lambda$, where $\lambda = p(\kappa)$ for some polynomial $p(\cdot)$. For $x \in \{0, 1\}^*$, define $x[i]$ to be the ith bit of x and define $\widetilde{\mathsf{Enc}}$ to be the bitwise encryption of x:

$$\widetilde{\mathsf{Enc}}(\mathsf{pk}, x) \stackrel{\text{def}}{=} (\mathsf{Enc}(\mathsf{pk}, x[1]), \ldots, \mathsf{Enc}(\mathsf{pk}, x[|x|])).$$

Furthermore, for any $k \in \mathbb{N}$, recursively define "onion" encryption and decryption:

$$\mathsf{Enc}^*(\mathsf{pk}, x) \stackrel{\text{def}}{=} \mathsf{Enc}(\mathsf{pk}, x),$$

$$\mathsf{Enc}^* (\mathsf{pk}_1, \ldots, \mathsf{pk}_k, x) \stackrel{\text{def}}{=} \mathsf{Enc}^* (\mathsf{pk}_1, \ldots, \mathsf{pk}_{k-1}, \mathsf{Enc} (\mathsf{pk}_k, x)),$$
$$= \mathsf{Enc} (\mathsf{pk}_1, \mathsf{Enc} (\mathsf{pk}_2, \ldots, \mathsf{Enc} ((\mathsf{pk}_k, x)))),$$

$$\mathsf{Dec}^*(\mathsf{sk}, x) \stackrel{\text{def}}{=} \mathsf{Dec}(\mathsf{sk}, x),$$

$$\mathsf{Dec}^* (\mathsf{sk}_1, \ldots, \mathsf{sk}_k, x) \stackrel{\text{def}}{=} \mathsf{Dec}^* (\mathsf{sk}_2, \ldots, \mathsf{pk}_k, \mathsf{Dec} (\mathsf{sk}_1, x)),$$
$$= \mathsf{Dec} (\mathsf{sk}_k, \mathsf{Dec} (\mathsf{sk}_{k-1}, \ldots, \mathsf{Dec} (\mathsf{sk}_1, x))).$$

We highlight two properties of onion encryption and decryption:

---

[6]Note that correctness still holds even if the circuit C completely ignores all ciphertexts encrypted under a public key 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).

[7]The idea for this construction was originally suggested to us by an anonymous STOC 2012 reviewer. We include it in this dissertation and formally prove its correctness for the sake of completeness.

1. First, note that Enc^* and Dec^* satisfy correctness: if $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ for all $i \in [k]$, and then for all $m \in \{0, 1\}$:

$$\mathsf{Dec}^* (\mathsf{sk}_1, \ldots, \mathsf{sk}_k, \mathsf{Enc}^* (\mathsf{pk}_1, \ldots, \mathsf{pk}_k, m)) = m.$$

2. Second, note that the bit-size of the ciphertext $\mathsf{Enc}^* (\mathsf{pk}_1, \ldots, \mathsf{pk}_k, m)$ is λ^k. Recall that the ciphertext space of Enc is $\{0, 1\}^\lambda$ and $\lambda = p(\kappa)$ for some polynomial $p(\cdot)$.

Construction overview. We now give an overview of the construction. Given N ciphertexts $c_i \leftarrow \mathsf{Enc} (\mathsf{pk}_i, m_i)$ encrypting plaintext m_i under key pk_i, for all $i \in [N]$, it is possible to homomorphically compute onion ciphertexts:

$$z_i \approx \mathsf{Enc}^* (\mathsf{pk}_1, \ldots, \mathsf{pk}_N, m_i).$$

This is done by homomorphically evaluating the function $\mathsf{Enc}^*(\mathsf{pk}_{i+1}, \ldots, \mathsf{pk}_N, \cdot)$ on ciphertext c_i. This outputs an onion encryption $\widetilde{z}_i \approx \mathsf{Enc}^*(\mathsf{pk}_i, \ldots, \mathsf{pk}_N, m_i)$. The ciphertext z_i can be obtained by onion encrypting $\widetilde{z}_i$ with the remaining keys: $z_i = \mathsf{Enc}^*(\mathsf{pk}_1, \ldots, \mathsf{pk}_{i-1}, \widetilde{z}_i)$

Once the ciphertexts $z_1, \ldots, z_N$ have been obtained, we can recursively perform homomorphic evaluations corresponding to the keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ (in that order) to obtain a ciphertext:

$$c \approx \mathsf{Enc}^* (\mathsf{pk}_1, \ldots \mathsf{pk}_N, C (m_1, \ldots, m_N)).$$

By correctness of onion encryption, decrypting c can be easily achieved using onion decryption:

$$\mathsf{Dec}^* (\mathsf{sk}_1, \ldots, \mathsf{sk}_k, c) = C (m_1, \ldots, m_N).$$

However, recall that the size of each ciphertext z_i is $\lambda^N = p(\kappa)^N$ for some polynomial $p(\cdot)$. This means that the multikey homomorphic evaluation is efficient only if $N = O(1)$. Thus, this generic construction of multikey FHE from (standard) FHE allows only a constant number of keys.

Formal description. We now give a formal description of the generic multikey construction and prove its correctness. Let $\mathcal{E} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE scheme with message space $\{0, 1\}$ and ciphertext space $\{0, 1\}^\lambda$, where $\lambda = p(\kappa)$ for some polynomial $p(\cdot)$. Let Enc^* and Dec^* be the onion encryption and decryption algorithms described above.

- $\mathsf{GMK.Keygen}(1^\kappa)$: Run $\mathsf{Keygen}(1^\kappa)$.
- $\mathsf{GMK.Enc}(\mathsf{pk}, m)$: Run $\mathsf{Enc}(\mathsf{pk}, m)$.
- $\mathsf{GMK.Dec} (\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$: Output $\mathsf{Dec}^* (\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$.
- $\mathsf{GMK.Eval} (C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_N, \mathsf{pk}_N, \mathsf{ek}_N))$: For $i \in [N]$, define

$$G_i(x) \stackrel{\text{def}}{=} \mathsf{Enc}^* (\mathsf{pk}_{i+1}, \ldots, \mathsf{pk}_N, x \; ; \; r)$$

for some fixed and valid randomness r[8] and recursively define

$$C^{(k)} (x_1, \ldots, x_N) \stackrel{\text{def}}{=} \begin{cases} C (x_1, \ldots, x_N) & \text{for } k = N. \\ \mathsf{Eval} (\mathsf{ek}_{k+1}, C^{(k+1)}, x_1, \ldots, x_N) & \text{for } k < N. \end{cases}$$

---

[8] We need to include the randomness in the definition because we want $G_i(x)$ to be a deterministic circuit with x as its sole input.

For $i \in [N]$, compute

$$\widetilde{z}_i \overset{\text{def}}{=} \text{Eval}\left(\text{ek}_i, G_i, c_i\right) \quad , \quad z_i \overset{\text{def}}{=} \text{Enc}^*\left(\text{pk}_1, \ldots, \text{pk}_{i-1}, \widetilde{z}_i\right)$$

and output the ciphertext $c \overset{\text{def}}{=} \text{Eval}(\text{ek}_1, C^{(1)}, z_1, \ldots, z_N)$.

THEOREM 3.1. *The encryption scheme* $\mathcal{E}_{\text{GMK}} = (\text{GMK.Keygen}, \text{GMK.Enc}, \text{GMK.Dec}, \text{GMK.Eval})$ *is multikey fully homomorphic for* $N = O(1)$ *keys.*

Proof. To prove correctness of evaluation, we wish to prove that if $(\text{pk}_i, \text{sk}_i, \text{ek}_i)$ is in the support of $\text{GMK.Keygen}(1^\kappa) = \text{Keygen}(1^\kappa)$ and $c_i \leftarrow \text{GMK.Enc}(\text{pk}_i, m_i) = \text{Enc}(\text{pk}_i, m_i)$, then

$$\text{GMK.Dec}\left(\text{sk}_1, \ldots, \text{sk}_N, c\right) = \text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_N, c\right) = C\left(m_1, \ldots, m_N\right).$$

We first show that each z_i is a valid onion encryption of m_i. By correctness of evaluation with evaluation key ek_i, we know that

$$\text{Dec}\left(\text{sk}_i, \widetilde{z}_i\right) = G_i\left(m_i\right) = \text{Enc}^*\left(\text{pk}_{i+1}, \ldots, \text{pk}_N, m_i \ ; \ r\right),$$

and by correctness of encryption, we conclude that

$$\text{Dec}^*\left(\text{sk}_i, \ldots, \text{sk}_N, \widetilde{z}_i\right) = m_i \quad \text{and} \quad \text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_N, z_i\right) = m_i.$$

We now make the following claim, which constitutes the bulk of the proof.

CLAIM 3.1.1. *For every* $k \in [N]$,

$$\text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_k, c\right) = C_k\left(z_1^{(k)}, \ldots, z_N^{(k)}\right),$$

where $z_i^{(k)} \overset{\text{def}}{=} \text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_k, z_i\right)$.

In particular, for $k = N$, this claim implies

$$\text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_N, c\right) = C^{(N)}\left(z_1^{(N)}, \ldots, z_N^{(N)}\right) = C\left(m_1, \ldots, m_N\right),$$

where the second equality follows from the fact that $C_N = C$ by definition and the fact that $z_i^{(N)} = \text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_N, z_i\right) = m_i$, which we proved earlier.

It thus suffices to prove Claim 3.1.1 to conclude the proof of the theorem.

Proof. We prove Claim 3.1.1 by induction. The base case, $k = 1$, follows directly from correctness of evaluation and correctness of decryption:

$$\text{Dec}^*\left(\text{sk}_1, c\right) = C^{(1)}\left(\text{Dec}\left(\text{sk}_1, z_1\right), \ldots, \text{Dec}\left(\text{sk}_1, z_N\right)\right) = C^{(1)}\left(z_1^{(1)}, \ldots, z_N^{(1)}\right).$$

Now suppose that the claim holds for $k - 1$; that is, suppose

$$\text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_{k-1}, c\right) = C^{(k-1)}\left(z_1^{(k-1)}, \ldots, z_N^{(k-1)}\right).$$

Decrypting both sides by sk_k yields

$$\begin{aligned}
\text{Dec}^*\left(\text{sk}_1, \ldots, \text{sk}_k, c\right) &= \text{Dec}\left(\text{sk}_k, C^{(k-1)}\left(z_1^{(k-1)}, \ldots, z_N^{(k-1)}\right)\right) \\
&= \text{Dec}\left(\text{sk}_k, \text{Eval}\left(\text{ek}_k, C^{(k)}, z_1^{(k-1)}, \ldots, z_N^{(k-1)}\right)\right) \\
&= C^{(k)}\left(\text{Dec}\left(\text{sk}_k, z_1^{(k-1)}\right), \ldots, \text{Dec}\left(\text{sk}_k, z_N^{(k-1)}\right)\right) \\
&= C^{(k)}\left(z_1^{(k)}, \ldots, z_1^{(k)}\right), \qquad\qquad\qquad \square
\end{aligned}$$

where the second-to-last equality follows from correctness of evaluation and correctness of decryption. This concludes the inductive step and the proof. □

3.2.2. $O(\log \kappa)$-multikey FHE from ring-LWE. We now show that the Brakerski–Vaikuntanathan FHE [30] based on the RLWE assumption is multikey somewhat homomorphic for $N = O(\log \kappa)$ keys.

Decryption in Regev-style encryption consists of computing the inner product $\langle \mathbf{c}, \mathbf{s} \rangle \pmod 2$, where $\mathbf{c}, \mathbf{s} \in R_q^2$ are the ciphertext and secret key, respectively. Brakerski and Vaikuntanathan [30] generalize this to allow the ciphertext and secret key to grow in dimension. For $\mathbf{c}, \mathbf{s} \in R_q^d$, they define $\mathsf{Dec}(\mathbf{s}, \mathbf{c}) = \langle \mathbf{c}, \mathbf{s} \rangle \pmod 2$. Homomorphic operations are then defined as follows:

- Given two *same-length* ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$, output the ciphertext $\mathbf{c}_{\mathsf{add}} \stackrel{\mathsf{def}}{=} \mathbf{c}_1 + \mathbf{c}_2$ as an encryption of the *sum* of the underlying messages.
  The ciphertext $\mathbf{c}_{\mathsf{add}}$ is decryptable with the same secret key $\mathbf{s}$ since

$$\langle \mathbf{c}_1 + \mathbf{c}_2, \mathbf{s} \rangle = \langle \mathbf{c}_1, \mathbf{s} \rangle + \langle \mathbf{c}_2, \mathbf{s} \rangle.$$

- Given two ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$ *of potentially different length*, output the ciphertext $\mathbf{c}_{\mathsf{mult}} \stackrel{\mathsf{def}}{=} \mathbf{c}_1 \otimes \mathbf{c}_2$ as the *product* of the underlying messages.
  The ciphertext $\mathbf{c}_{\mathsf{mult}}$ is now decryptable with the secret key $\mathbf{s} \otimes \mathbf{s}$ since

$$\langle \mathbf{c}_1 \otimes \mathbf{c}_2, \mathbf{s} \otimes \mathbf{s} \rangle = \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle.$$

We can extend this to the multikey setting. Multiplication is trivial, but some changes are necessary in the case of addition.

- Given two *same-length* ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$ decryptable with secret keys $\mathbf{s}_1, \mathbf{s}_2$, respectively, output the ciphertext $\mathbf{c}_{\mathsf{add}} \stackrel{\mathsf{def}}{=} (\mathbf{c}_1, \mathbf{c}_2)$ as an encryption of the *sum* of the underlying messages.
  The ciphertext $\mathbf{c}_{\mathsf{add}}$ is decryptable with the same secret key $(\mathbf{s}_1, \mathbf{s}_2)$ since

$$\langle (\mathbf{c}_1, \mathbf{c}_2), (\mathbf{s}_1, \mathbf{s}_2) \rangle = \langle \mathbf{c}_1, \mathbf{s}_1 \rangle + \langle \mathbf{c}_2, \mathbf{s}_2 \rangle.$$

- Given two ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$ decryptable with secret keys $\mathbf{s}_1, \mathbf{s}_2$, respectively, and *of potentially different length*, output the ciphertext $\mathbf{c}_{\mathsf{mult}} \stackrel{\mathsf{def}}{=} \mathbf{c}_1 \otimes \mathbf{c}_2$ as the *product* of the underlying messages.
  The ciphertext $\mathbf{c}_{\mathsf{mult}}$ is now decryptable with the secret key $\mathbf{s}_1 \otimes \mathbf{s}_2$ since

$$\langle \mathbf{c}_1 \otimes \mathbf{c}_2, \mathbf{s}_1 \otimes \mathbf{s}_2 \rangle = \langle \mathbf{c}_1, \mathbf{s}_1 \rangle \cdot \langle \mathbf{c}_2, \mathbf{s}_2 \rangle.$$

Observe that each homomorphic operation (at most) doubles the size of the ciphertext. Starting with fresh ciphertexts of length 2, after $(N-1)$ operations (which can combine ciphertexts encrypted under at most N distinct keys), the size of both the ciphertext and the joint decryption key is 2^N. This is only feasible if $N = O(\log \kappa)$.

As shown in the work of Brakerski and Vaikuntanathan [30], the scheme can evaluate circuits of depth $D < \varepsilon \log n - \log \log n + \Theta(1)$, where $q = 2^{n^\varepsilon}$ for constant $\varepsilon \in (0, 1)$. Unfortunately, we do not know how to apply relinearization or squashing in the multikey setting and are therefore not able to convert the resulting multikey scheme into a leveled or fully homomorphic one.

3.3. Multikey somewhat homomorphic encryption for any number of keys. We now turn to our main result in this section: we construct a multikey

somewhat homomorphic encryption scheme based on the (modified) NTRU encryption scheme [96, 130] described in section 2.7. Unlike the schemes in section 3.2, the scheme we describe in this section will be multikey for $N \approx n^\varepsilon$ keys with constant $\varepsilon \in (0, 1)$. In section 3.4, we show how to convert the scheme into a multikey fully homomorphic scheme for $N \approx n^\varepsilon$ keys. By setting $n \approx N^{1/\varepsilon}$, we can construct a multikey FHE for any number of keys N, as long as N is known a priori.

We begin by informally describing the multikey homomorphic properties of NTRU encryption and some of the problems encountered when trying to convert the scheme from section 2.7 into a somewhat homomorphic one. We then show a formal description of our somewhat homomorphic scheme, formally prove its homomorphic properties, and discuss its security. In section 3.4, we show how to convert this scheme into a fully homomorphic scheme.

3.3.1. Multikey homomorphism. Recall from section 2.7 that an NTRU key pair consists of ring elements (h, f) such that $h = [2gf^{-1}]_q$, where g, f are "small" ring elements sampled from a B-bounded distribution χ, and f^{-1} is the inverse of f in R_q. Further recall that an NTRU ciphertext has the form $c = [hs + 2e + m]_q$ for small elements s, e sampled from χ, and decryption computes $[fc]_q \pmod{2}$.

Let (h_1, f_1) and (h_2, f_2) be two different NTRU public/secret key pairs, and let $c_1 \overset{\text{def}}{=} [h_1 s_1 + 2e_1 + m_1]_q$ and $c_2 \overset{\text{def}}{=} [h_2 s_2 + 2e_2 + m_2]_q$ 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 the underlying plaintexts, m_1 and m_2. In particular, we show that the ciphertexts $c_{mult} \overset{\text{def}}{=} c_1 \cdot c_2$ and $c_{add} \overset{\text{def}}{=} c_1 + c_2$ can be decrypted to the product and the sum of m_1 and m_2, respectively, using the secret key $f_{12} \overset{\text{def}}{=} f_1 \cdot f_2$.

To see this, note that

$$[f_1 f_2 (c_1 + c_2)]_q = [2f_1 f_2 e_1 + 2f_1 f_2 e_2 + 2f_2 g_1 s_1 + 2f_1 g_2 s_2 + f_1 f_2 (m_1 + m_2)]_q,$$

$$\begin{aligned}[f_1 f_2 (c_1 \cdot c_2)]_q = [4g_1 g_2 s_1 s_2 &+ 2g_1 s_1 f_2 (2e_2 + m_2) + 2g_2 s_2 f_1 (2e_1 + m_1), \\ &+ 2f_1 f_2 (e_1 m_2 + e_2 m_1 + 2e_1 e_2) + f_1 f_2 (m_1 m_2)]_q, \\ &= m_1 \cdot m_2 \pmod{2}.\end{aligned}$$

Since $f_1 \equiv f_2 \equiv 1 \pmod{2}$, we can conclude that as long as there is no reduction modulo q,

$$[f_1 f_2 (c_1 + c_2)]_q \pmod{2} = m_1 + m_2 \pmod{2},$$

$$[f_1 f_2 (c_1 \cdot c_2)]_q \pmod{2} = m_1 \cdot m_2 \pmod{2}.$$

In other words, the "joint secret key" $f_{12} \overset{\text{def}}{=} f_1 f_2$ can be used to decrypt $c_{\text{add}} = [c_1 + c_2]_q$ and $c_{\text{mult}} = [c_1 \cdot c_2]_q$. 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 $\varepsilon \log(n)$ when $q = 2^{n^\varepsilon}$ and $B = \mathsf{poly}(n)$.

Problems in multikey decryption. An astute reader will have observed 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 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 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

$$[f_1 f_2 \cdot c]_q = 2e + f_1 f_2 m \quad , \quad [f_2 f_3 \cdot c']_q = 2e' + f_2 f_3 m'$$

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 $f_1 f_2 f_3$:

$$[f_1 f_2 f_3 \cdot (c + c')]_q = [f_3(f_1 f_2 \cdot c) + f_1(f_2 f_3 \cdot c')]_q = 2(f_3 e + f_1 e') + f_1 f_2 f_3(m + m').$$

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 the union $K \cup K'$. We note that the absolute magnitude of the coefficients of this product grows exponentially with the number of keys in $K \cup K'$, i.e., the total number of keys involved in the homomorphic computation.

Analogously, in the context of homomorphic multiplication, we need $f_1 f_2^2 f_3$ to decrypt $c \cdot c'$:

$$\left[f_1 f_2^2 f_3 \cdot (c \cdot c') \right]_q = [(f_1 f_2 \cdot c) \cdot (f_2 f_3 \cdot c')]_q = 2E_{\mathsf{mult}} + f_1 f_2^2 f_3(m \cdot m'),$$

where $E_{\mathsf{mult}} \stackrel{\mathsf{def}}{=} 2ee' + e f_2 f_3 m' + e' f_1 f_2 m$. This hints at the fact that the magnitude of the coefficients 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 keys involved, as in the case of addition). Indeed, after M multiplications, the joint secret key needed to decrypt the evaluated ciphertext will be the product of M polynomials, and the magnitude of the coefficients of this product will be exponential in M.

Our solution. To solve the above problems, we use *relinearization* (also known as *key-switching*), a technique first introduced by Brakerski and Vaikuntanathan [29]. Informally, we 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 the new ciphertext depends only on the number of keys N and not on the degree of the circuit C that was evaluated.

Though we are able to eliminate the dependence (of the magnitude of the coefficients of the joint secret key) on the degree of the circuit, we remark that we do not succeed in eliminating the exponential dependence on N, the number of keys—indeed, this is the reason why our solution will eventually assume an a priori upper bound on N.

3.3.2. Formal description. We present a formal description of our multikey somewhat homomorphic encryption scheme based on the (modified) NTRU encryption

scheme [96, 130] described in section 2.7. The scheme uses two subroutines that expand a polynomial in R_q into $\lceil \log q \rceil$ polynomials. For any $x \in R_q$, the following hold:

- $\mathsf{Bit}(x)$ decomposes x into its bit representation. Namely,

$$\mathsf{Bit}(x) = (x_0, x_1, \ldots, x_{\lfloor \log q \rfloor}) \in R_2^{\lceil \log q \rceil} \text{ such that } x = \sum_{j=0}^{\lfloor \log q \rfloor} 2^j x_j.$$

- $\mathsf{Pow}(x) = \left[(x, 2 \cdot x, \ldots, 2^{\lfloor \log q \rfloor} \cdot x) \right]_q \in R_q^{\lceil \log q \rceil}$.

Observe that for any $x, z \in R_q$,

$$[\langle \mathsf{Bit}(x), \mathsf{Pow}(z) \rangle]_q = \left[\sum_{j=0}^{\lfloor \log q \rfloor} x_j \cdot 2^j z \right]_q = \left[z \cdot \sum_{j=0}^{\lfloor \log q \rfloor} 2^j x_j \right]_q = x \cdot z.$$

The encryption scheme is defined as follows:

- $\mathsf{SH.Keygen}(1^\kappa)$: 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'; otherwise let f^{-1} be the inverse of f in R_q. Set

$$\mathsf{pk} \stackrel{\mathsf{def}}{=} h := \left[2gf^{-1}\right]_q \in R_q \quad , \quad \mathsf{sk} \stackrel{\mathsf{def}}{=} f \in R.$$

Sample $\widetilde{\mathbf{s}}, \widetilde{\mathbf{e}} \leftarrow \chi^{\lceil \log q \rceil}$ and compute $\mathsf{ek} \stackrel{\mathsf{def}}{=} \left[h\widetilde{\mathbf{s}} + 2\widetilde{\mathbf{e}} + \mathsf{Pow}(f)\right]_q \in R_q^{\lceil \log q \rceil}$. Output the key tuple $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$.

- $\mathsf{SH.Enc}(\mathsf{pk}, m)$: Sample $s, e \leftarrow \chi$. Output the ciphertext $c := hs+2e+m \in R_q$.
- $\mathsf{SH.Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$: Parse $\mathsf{sk}_i = f_i$ for $i \in [N]$. Compute $\mu = [f_1 \cdots f_N \cdot c]_q \in R_q$ and output $m := \mu \pmod 2$.
- $\mathsf{SH.Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_\ell, \mathsf{pk}_\ell, \mathsf{ek}_\ell))$: We show how to evaluate an ℓ-variate boolean circuit $C : \{0,1\}^\ell \to \{0,1\}$ of depth D. To this end, we show how to homomorphically add and multiply two elements in $\{0,1\}$. Given two ciphertexts c, c', we assume that we also have a set of distinct public keys associated with each ciphertext.[9] We will denote these lists by K, K', respectively. The public-key set of a fresh encryption is simply the set $\{\mathsf{pk}\}$ containing the public key under which it was encrypted. For convenience, in our analysis we sometimes assume that the set contains the indices of the public keys instead of the keys themselves.
  - Given two ciphertexts c and c' with corresponding public-key sets K and K', output the ciphertext

$$c_{\mathsf{add}} = [c + c']_q \in R_q$$

  as an encryption of the *sum* of the underlying messages. Output the set $K_{\mathsf{add}} = K \cup K'$ as its corresponding public-key set.
  - Given two ciphertexts c and c' with corresponding public-key sets K and K', compute ciphertext $c_0 = [c \cdot c']_q \in R_q$.
    * If $K \cap K' = \emptyset$, let $c_{\mathsf{mult}} = c_0$.
    * Otherwise, let $K \cap K' = \{\mathsf{pk}_{i_1}, \ldots, \mathsf{pk}_{i_t}\}$. For $j \in [t]$, compute $c_j = [\langle \mathsf{Bit}(c_{j-1}), \mathsf{ek}_{i_j} \rangle]_q$, and let $c_{\mathsf{mult}} = c_t$ at the end of the iteration.

---

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

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

For a set $S \subseteq [N]$, let $f_S \overset{\mathsf{def}}{=} \prod_{i \in S} f_i$. Note that the ciphertext c_0 can be decrypted to $m \cdot m'$ with the "joint" secret key $f_K f_{K'}$, which contains the term $f_{i_1}^2 \ldots f_{i_t}^2$. The goal of relinearization is to convert it into a ciphertext that decrypts to the same message under the secret key

$$f_K f_{K'} \left(\prod_{j \in K \cap K'} f_j \right)^{-1} = f_{K \cup K'},$$

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

We first show that the scheme works correctly as advertised.

LEMMA 3.2. *If $q = 2^{n^{\varepsilon}}$ for $\varepsilon \in (0, 1)$ and χ is a B-bounded distribution for $B = \mathsf{poly}(n)$, then the encryption scheme $\mathcal{E}_{\mathsf{SH}} = (\mathsf{SH.Keygen}, \mathsf{SH.Enc}, \mathsf{SH.Dec}, \mathsf{SH.Eval})$ described above is multikey homomorphic for $N = O(n^{\delta})$ keys and circuits of depth $D < (\varepsilon - \delta) \log n - \log \log n - O(1)$.*

Proof. We define the (multikey) error of a ciphertext c with corresponding public-key set K to be $\mu \overset{\mathsf{def}}{=} [f_K \cdot c]_q$. We start by showing that the magnitude of the error coefficients does not grow too much after a homomorphic evaluation.

CLAIM 3.2.1. *Let c, c' be ciphertexts encrypting m and m', respectively, and suppose that the magnitude of their error coefficients is bounded by $E < q/2$. Then c_{add} and c_{mult} correctly decrypt to $m + m'$ and $m \cdot m'$, respectively, and their error coefficients are bounded by $(nB)^{2N} E^2$.*

Proof. Let c, c' be encryptions of m, m', respectively, with corresponding public-key sets K, K'. We know that for some $e, e' \in R$ we have

$$[f_K \cdot c]_q = 2e + m \quad , \quad [f_{K'} \cdot c']_q = 2e' + m'$$

and $\|2e + m\|_{\infty}, \|2e' + m\|_{\infty} < E$. Then

$$[f_{K_{\mathsf{add}}} \cdot c_{\mathsf{add}}]_q = [f_{K \cup K'} \cdot (c + c')]_q = \left[f_{K \setminus K'}(f_K \cdot c) + f_{K' \setminus K}(f_{K'} \cdot c') \right]_q$$
$$= f_{K \setminus K'}(2e + m) + f_{K' \setminus K}(2e' + m').$$

We can thus bound the magnitude of the coefficients of $[f_{K_{\mathsf{add}}} \cdot c_{\mathsf{add}}]_q$ by $2(nB)^N E < (nB)^{2N} E^2$, as desired. Furthermore, it easy to see that $[f_{K_{\mathsf{add}}} \cdot c_{\mathsf{add}}]_q \pmod{2} = m + m'$.

The multiplication case is more complex. Let $K \cap K' = \{i_1, \ldots, i_t\}$, as before. Define $F_0 \overset{\mathsf{def}}{=} f_K f_{K'}$, and for $j \in [t]$, define $F_j = F_{j-1} \cdot f_{i_j}^{-1}$. Then $F_r = f_{K \cup K'}$ is a simple product of the secret keys f_i without any quadratic terms. We know that

$$[F_0 \cdot c_0]_q = [(f_K \cdot c)(f_{K'} \cdot c_K)]_q = (2e + m)(2e' + m')$$

so that $\|[F_0 \cdot c_0]_q\|_{\infty} < nE^2$ and $[F_0 \cdot c_0]_q \pmod{2} = m \cdot m'$. Furthermore, for all $j \in [t]$,

$$[F_j \cdot c_j]_q = \left[F_j \cdot \left\langle \mathsf{Bit}\left(c_{j-1}\right), h_{i_j} \widetilde{\mathbf{s}} + 2\widetilde{\mathbf{e}} + \mathsf{Pow}\left(f_{i_j}\right) \right\rangle \right]_q$$
$$= \left[F_j \cdot \left\langle \mathsf{Bit}\left(c_{j-1}\right), h_{i_j} \widetilde{\mathbf{s}} \right\rangle + F_j \cdot \left\langle \mathsf{Bit}\left(c_{j-1}\right), 2\widetilde{\mathbf{e}} \right\rangle + F_j c_{j-1} f_{i_j} \right]_q$$
$$= F_j f_{i_j}^{-1} \cdot \left\langle \mathsf{Bit}\left(c_{j-1}\right), 2g_{i_j} \widetilde{\mathbf{s}} \right\rangle + F_j \cdot \left\langle \mathsf{Bit}\left(c_{j-1}\right), 2\widetilde{\mathbf{e}} \right\rangle + F_{j-1} c_{j-1}.$$

Using the fact that each F_j is the product of at most $(2N - j)$ keys, we have that

$$
\begin{aligned}
\left\| [F_j \cdot c_j]_q \right\|_\infty &< 2 \lceil \log q \rceil n^2 B^2 \cdot (nB)^{2N-j-1} + 2 \lceil \log q \rceil nB \cdot (nB)^{2N-j} \\
&\quad + \left\| [F_{j-1} \cdot c_{j-1}]_q \right\|_\infty \\
&= 4 \lceil \log q \rceil (nB)^{2N-j+1} + \left\| [F_{j-1} \cdot c_{j-1}]_q \right\|_\infty .
\end{aligned}
$$

This yields the following bound on the error of c_{mult}:

$$
\begin{aligned}
\left\| [F_{K \cup K'} \cdot c_{\mathsf{mult}}]_q \right\|_\infty = \left\| [F_t \cdot c_t]_q \right\|_\infty &\le nE^2 + \sum_{j=1}^{t} 4 \lceil \log q \rceil (nB)^{2N-j+1} \\
&= nE^2 + 4 \lceil \log q \rceil (nB)^{2N+1} \sum_{j=1}^{t} (nB)^{-j} \\
&\le nE^2 + 8 \lceil \log q \rceil (nB)^{2N+1} \\
&\le (nB)^{2N} E^2 ,
\end{aligned}
$$

where the last inequality holds by the fact that $q = 2^{n^\varepsilon}$.

Furthermore, notice that $[F_j \cdot c_j]_q \equiv F_{j-1} c_{j-1} \pmod 2$. Since $[F_0 \cdot c_0]_q \pmod 2 = m \cdot m'$, we can conclude that $[F_{K \cup K'} \cdot c_{\mathsf{mult}}]_q \pmod 2 = [F_t \cdot c_t]_q \pmod 2 = m \cdot m'$. □

Once we have bounded the magnitude of the error coefficients after a homomorphic operation, we can bound the overall error incurred after evaluating a circuit of depth D. Starting with error $E_0 \le 3(nB)^2$, after D levels of homomorphic operations, the error magnitude can grow to at most

$$
\left((nB)^{2N} E_0 \right)^{2^D} \le \left((3nB)^{2^D \cdot (2N+2)} \right) .
$$

This results in correct decryption as long as $D < \log \log q - \log \log n - \log N - O(1)$, where we use the fact that $B = \mathsf{poly}(n)$. In particular, for $N = O(n^\delta)$ keys and $q = 2^{n^\varepsilon}$, we get $D < (\varepsilon - \delta) \log n - \log \log n - O(1)$. □

3.3.3. Security. Recall from section 2.7 that the security of the (modified) NTRU encryption scheme can be based on two assumptions—the RLWE assumption and the DSPR assumption. Recall further that Stehlé and Steinfeld [130] showed that the $\mathsf{DSPR}_{\phi,q,\chi}$ assumption is unconditionally true if n is a power of 2, $\phi(x) = x^n + 1$ is the nth cyclotomic polynomial, and χ is the discrete Gaussian $D_{\mathbb{Z}^n, r}$ for $r > \sqrt{q} \cdot \mathsf{poly}(n)$. This allowed them to prove semantic security for the modified NTRU scheme under the $\mathsf{RLWE}_{\phi,q,\chi}$ assumption alone.

Unfortunately, their result holds only if $r > \sqrt{q} \cdot \mathsf{poly}(n)$, which is too large to permit even a single homomorphic multiplication.[10] 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. Thus, it is necessary to assume that the decisional small polynomial ratio problem is hard for our choice of parameters.

---

[10]Nevertheless, Bos et al. [25] show a different way to achieve (single-key) homomorphism for parameters that result from [130], using different methods. Their parameters still do not support homomorphic evaluation with even two different keys. See the paragraph after Theorem 1.2 and [25] for more detailed discussion.

Additionally, note that the evaluation key ek contains elements of the form $[hs_\tau + 2e_\tau + 2^\tau f]_q$, which can be thought of as "pseudoencryptions" of (multiples of) the secret key f under the corresponding public key h.[11] The security of the scheme must then additionally rely on a "circular security" assumption that 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).

Thus, we can base the security of the scheme on the DSPR assumption, the RLWE assumption, and the circular security assumption described above.

LEMMA 3.3. *Let n be a power of 2, let $\phi(x) = x^n + 1$, and let $q = 2^{n^\varepsilon}$ for $\varepsilon \in (0,1)$ and $\chi = D_{\mathbb{Z}^n,r}$ with $r = \mathsf{poly}(n)$. Then the somewhat homomorphic encryption scheme $\mathcal{E}_{\mathsf{SH}} = \{\mathsf{SH.Keygen}, \mathsf{SH.Enc}, \mathsf{SH.Dec}, \mathsf{SH.Eval}$ described above is secure under the $\mathsf{DSPR}_{\phi,q,\chi}$ and $\mathsf{RLWE}_{\phi,q,\chi}$ assumptions and the assumption that the scheme remains semantically secure even given the evaluation key* ek.

3.4. From somewhat to fully homomorphic encryption. We use a generalization of Gentry's bootstrapping theorem [71, 70] (see section 2.5) to convert the multikey somewhat homomorphic scheme from section 3.3 into a multikey fully homomorphic one. We modify Gentry's bootstrapping theorem and the corresponding definitions from their original presentation to generalize them to the multikey setting.

DEFINITION 3.3 (multikey bootstrappable schemes). *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*

$$f_{\mathsf{add}}^{c_1,c_2}(\mathsf{sk}_1,\ldots,\mathsf{sk}_N) = \mathsf{Dec}(\mathsf{sk}_1,\ldots,\mathsf{sk}_N,c_1) \quad XOR \quad \mathsf{Dec}(\mathsf{sk}_1,\ldots,\mathsf{sk}_N,c_2),$$
$$f_{\mathsf{mult}}^{c_1,c_2}(\mathsf{sk}_1,\ldots,\mathsf{sk}_N) = \mathsf{Dec}(\mathsf{sk}_1,\ldots,\mathsf{sk}_N,c_1) \quad AND \quad \mathsf{Dec}(\mathsf{sk}_1,\ldots,\mathsf{sk}_N,c_2).$$

Then $\mathcal{E}$ is bootstrappable *if $\{f_{\mathsf{add}}^{c_1,c_2}, f_{\mathsf{mult}}^{c_1,c_2}\}_{c_1,c_2} \subseteq \mathcal{C}$. Namely, the scheme can homomorphically evaluate f_{add} and f_{mult}.*

We now state a generalization of Gentry's bootstrapping theorem to the multikey setting. Taking $N = 1$ yields the theorem and the definitions from [71, 70] and section 2.5.

THEOREM 3.4 (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 of encryption schemes $\mathcal{E}'$.*

Unfortunately, the somewhat homomorphic scheme described in section 3.3 is not bootstrappable. Recall that the scheme can only evaluate circuits of depth less than $\varepsilon \log(n)$, where $\varepsilon < 1$. 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, a technique introduced by [29] and refined by [27], to convert our somewhat homomorphic scheme into a bootstrappable scheme.

3.4.1. Modulus reduction. Modulus reduction [29, 27] 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

---

[11]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 $m \in R_2$.

section 3.3.1, 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.

Let (h, f) be a key pair, and let c be a ciphertext under public key h. Recall that for decryption to be successful, we need the error $[fc]_q$ to be equal to $fc \in R$. However, each homomorphic operation increases this error. Modulus reduction allows us to keep the error magnitude small by simply scaling the ciphertext after each operation. The key idea is to exploit the difference in how the error affects security and homomorphism:

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

This suggests a method of reducing the magnitude of the error 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 $c' \in R_p$ (for $p < q$) 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 3.5 (see [27]). *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*

$$[fc']_p = [fc]_q \pmod 2 \qquad and \qquad \|[fc']_p\|_\infty < (p/q) \cdot \|[fc]_q\|_\infty + \|f\|_1,$$

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

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

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

so that

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

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

$$e_i - kp \in \left[-\frac{p}{2}, \frac{p}{2} \right] \qquad \text{and} \qquad [e_i]_p = e_i - kp.$$

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

$$e_i - kp \equiv d_i - kq \pmod 2,$$
$$[e_i]_p \equiv [d_i]_q \pmod 2,$$
$$[fc']_p \equiv [fc]_q \pmod 2. \qquad \Box$$

The beauty of Lemma 3.5 is that if we know the depth D of the circuit we want to evaluate, 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. This yields a leveled homomorphic encryption scheme. A bootstrappable scheme can then be obtained by setting $D = D_{\mathsf{dec}}$, the depth of the augmented decryption circuit.

3.4.2. Obtaining a leveled homomorphic scheme. We change the somewhat homomorphic scheme from section 3.3 to use modulus reduction during the evaluation. The main changes to the scheme are as follows:

- The scheme is now additionally parametrized by an integer D, which is the maximum circuit depth that it can homomorphically evaluate, and a ladder of decreasing moduli $q_0, \ldots q_D$.
- We cannot use a single key f for all levels (at the expense of assuming the circular security), as in section 3.3. This is because the public key h depends on the modulus q. (Recall that $h = 2gf^{-1}$, where f^{-1} is the inverse of f in R_q.) With the new ladder of moduli, this would require that the following two conditions be met simultaneously: first, that f^{-1} is the inverse of f in R_{q_D} (to guarantee correctness of decryption) and second, that $h = 2gf^{-1}$ is (indistinguishable from) uniformly random in R_{q_0} (to guarantee semantic security). This would require making a much stronger (and perhaps false) hardness assumption.
  Instead, key generation computes a different key pair $(h^{(d)}, f^{(d)})$ for each level $d \in \{0, \ldots, D\}$. Encryption uses $\mathsf{pk} \stackrel{\mathsf{def}}{=} h^{(0)}$ as the public key, and decryption uses $\mathsf{sk}^{(d)} \stackrel{\mathsf{def}}{=} f^{(d)}$ to decrypt a "level-d" ciphertext, i.e., a ciphertext that is the output of a depth-d circuit evaluation. Without loss of generality (w.l.o.g.) we assume any ciphertext to be decrypted is a level-D ciphertext and set the secret key to be $\mathsf{sk} = f^{(D)}$.
  Homomorphic operations will ensure that if c, c' are level-$(d-1)$ ciphertexts in $R_{q_{d-1}}$ decryptable with $f^{(d-1)}$, then c_{add} and c_{mult} are level-d ciphertexts in R_{q_d} decryptable with $f^{(d)}$.
- Relinearization will now serve two purposes: it will ensure that only linear terms of keys are needed to decrypt the resulting ciphertext, but it will also switch the level-$(d-1)$ key to the corresponding level-(d) key. (Indeed, relinearization is also known as *key-switching* in the literature). Moreover, note that we must perform the key-switching step not only for quadratic terms but also for linear terms. Thus, we now perform relinearization/key-switching after *every* homomorphic operation, both addition and multiplication, and furthermore, we relinearize/key-switch every key in $K \cup K'$, instead of only those in $K \cap K'$.
- To perform the relinearization/key-switching step described above, the evaluation key consists of pseudoencryptions of $f^{(d-1)}$ and $(f^{(d-1)})^2$ under the public key $h^{(d)}$ for all $d \in [D]$.
  Note in particular that we now need pseudoencryptions of the quadratic terms of the key. In the scheme from section 3.3, relinearization only required pseudoencryptions of (multiples of) f because the term $\langle \mathsf{Bit}\,(c)\,, \mathsf{Pow}\,(f) \rangle$ only performed "partial decryption" of the ciphertext c; it computes fc but f^2 is

required to decrypt c. Decryption of c was completed at decryption time when the ciphertext was multiplied by f once more, obtaining $f^2 c$.

In our new setting, because decryption is performed using a *different* key, re-linearization needs to "completely decrypt" c with the original key. For a key in $K \cap K'$, this means computing $[\langle \mathsf{Bit}(c), \mathsf{Pow}((f^{(d-1)})^2) \rangle]_q = [(f^{(d-1)})^2 c]_q$. Since $\mathsf{Pow}((f^{(d-1)})^2)$ is encrypted under $h^{(d)}$, the new ciphertext can be decrypted using $f^{(d)}$.

Pseudoencryptions of the linear terms of the keys are also required in order to relinearize/key-switch keys in $K \triangle K'$, the symmetric difference of K, K'.

3.4.3. Formal description. We now give a formal description of the leveled homomorphic encryption scheme resulting from applying the changes described above to the somewhat homomorphic scheme $\mathcal{E}_{\mathsf{SH}}$ described in section 3.3.

- LH.Keygen(1^κ): For every $i \in \{0, \dots, D\}$, 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 $u^{(i)}$; otherwise, let $(f^{(i)})^{-1}$ be the inverse of $f^{(i)}$ in R_q. Let $h^{(i)} \overset{\mathsf{def}}{=} [2g^{(i)}(f^{(i)})^{-1}]_{q_i} \in R_{q_i}$, and set

$$\mathsf{pk} \overset{\mathsf{def}}{=} h^{(0)} \in R_{q_0} \quad , \quad \mathsf{sk} \overset{\mathsf{def}}{=} f^{(D)} \in R_{q_D}.$$

For all $i \in [D]$, sample $\widetilde{\mathbf{s}}_\gamma^{(i)}, \widetilde{\mathbf{e}}_\gamma^{(i)}, \widetilde{\mathbf{s}}_\zeta^{(i)}, \widetilde{\mathbf{e}}_\zeta^{(i)} \leftarrow \chi^{\lceil \log q \rceil}$ and compute

$$\boldsymbol{\gamma}^{(i)} := \left[h^{(i)} \widetilde{\mathbf{s}}_\gamma^{(i)} + 2\widetilde{\mathbf{e}}_\gamma^{(i)} + \mathsf{Pow}\left(f^{(i-1)} \right) \right]_{q_i} \in R_{q_i}^{\lceil \log q_i \rceil},$$

$$\boldsymbol{\zeta}^{(i)} := \left[h^{(i)} \widetilde{\mathbf{s}}_\zeta^{(i)} + 2\widetilde{\mathbf{e}}_\zeta^{(i)} + \mathsf{Pow}\left(\left(f^{(i-1)} \right)^2 \right) \right]_{q_i} \in R_{q_i}^{\lceil \log q_i \rceil}.$$

Set $\mathsf{ek} \overset{\mathsf{def}}{=} \{\boldsymbol{\gamma}^{(i)}, \boldsymbol{\zeta}^{(i)}\}_{i \in [D]}$, and output the key tuple $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$.

- LH.Enc(pk, m): Sample $s, e \leftarrow \chi$. Output the ciphertext $c := [hs + 2e + m]_{q_0} \in R_{q_0}$.

- LH.Dec($\mathsf{sk}_1, \dots, \mathsf{sk}_N, c$): Assume without loss of generality that $c \in R_{q_D}$. Parse $\mathsf{sk}_i = f_i$ for $i \in [N]$. Let $\mu := [f_1 \cdots f_N \cdot c]_{q_D} \in R_{q_D}$. Output $m' := \mu \pmod 2$.

- LH.Eval($C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \dots, (c_\ell, \mathsf{pk}_\ell, \mathsf{ek}_\ell)$): We show how to evaluate an ℓ-variate boolean circuit $C : \{0,1\}^\ell \to \{0,1\}$ of depth D. To this end, we show how to homomorphically add and multiply two elements in $\{0,1\}$. As before, given two ciphertexts c, c', we assume that we also have a set of distinct public keys associated with each ciphertext and denote these lists by K, K', respectively. The public-key set of a fresh encryption is simply the set $\{\mathsf{pk}\}$ containing the public key under which it was encrypted. For convenience, in our analysis we sometimes assume that the set contains the indices of the public keys instead of the keys themselves.

  - Given two ciphertexts $c, c \in R_{q_d}$ with corresponding public-key sets K, K', compute $c_0 = [c + c']_{q_d} \in R_{q_d}$; let $K \cup K' = \{\mathsf{pk}_{i_1}, \dots, \mathsf{pk}_{i_t}\}$. For $j = 1, \dots, r$, parse $\mathsf{ek}_{i_j} = \{\boldsymbol{\gamma}_{i_j}^{(\delta)}, \boldsymbol{\zeta}_{i_j}^{(\delta)}\}_{\delta \in [D]}$ and compute

  $$c_j = \left[\left\langle \mathsf{Bit}(c_{j-1}), \boldsymbol{\gamma}_{i_j}^{(d)} \right\rangle \right]_q \in R_{q_d}.$$

  Finally, reduce the modulus: let c_{add} be the integer vector closest to $(q_{d+1}/q_d) \cdot c_t$ such that $c_{\mathsf{add}} \equiv c_t \pmod 2$. Output $c_{\mathsf{add}} \in R_{q_{d+1}}$ as

an encryption of the *sum* of the underlying messages. Output the set $K_{\mathsf{add}} \stackrel{\text{def}}{=} K \cup K'$ as its corresponding public-key set.

– Given two ciphertexts $c, c \in R_{q_d}$ with corresponding public-key sets K, K', compute $c_0 = [c \cdot c']_{q_d} \in R_{q_d}$ and let $K \cup K' = \{\mathsf{pk}_{i_1}, \ldots, \mathsf{pk}_{i_t}\}$. For $j = 1, \ldots, r$, parse $\mathsf{ek}_{i_j} = \{\boldsymbol{\gamma}_{i_j}^{(\delta)}, \boldsymbol{\zeta}_{i_j}^{(\delta)}\}_{\delta \in [D]}$ and compute c_j as follows:
  * If $\mathsf{pk}_{i_j} \in K \cap K'$, let

$$c_j = \left[\left\langle \mathsf{Bit}\left(c_{j-1}\right), \boldsymbol{\zeta}_{i_j}^{(d)} \right\rangle \right]_q \in R_{q_d}.$$

  * Otherwise, let

$$c_j = \left[\left\langle \mathsf{Bit}\left(c_{j-1}\right), \boldsymbol{\gamma}_{i_j}^{(d)} \right\rangle \right]_q \in R_{q_d}.$$

Finally, reduce the modulus: let c_{mult} be the integer vector closest to $(q_{d+1}/q_d) \cdot c_t$ such that $c_{\mathsf{mult}} \equiv c_t \pmod 2$. Output $c_{\mathsf{mult}} \in R_{q_{d+1}}$ as an encryption of the *product* of the underlying messages. Output the set $K_{\mathsf{mult}} \stackrel{\text{def}}{=} K \cup K'$ as its corresponding public-key set.

Leveled homomorphism. We can now show the following lemma, characterizing the circuits and number of keys that the scheme can handle in evaluation.

LEMMA 3.6. *Let χ be a B-bounded distribution for $B = \mathsf{poly}(n)$, let $q_0 = 2^{n^\varepsilon}$ for $\varepsilon \in (0, 1)$ and for $d \in [D]$, and let $q_{d-1}/q_d = 8n(nB)^{2N+2}$. Then the encryption scheme $\mathcal{E}_{\mathsf{LH}} = (\mathsf{LH.Keygen}, \mathsf{LH.Enc}, \mathsf{LH.Dec}, \mathsf{LH.Eval})$ described above is multikey homomorphic for N keys and circuits of depth D as long as $ND = O\left(n^\varepsilon / \log n\right)$.*

Proof. We claim that for all $d \in \{0, \ldots, D\}$, the error of a level-d ciphertext is bounded by $E \stackrel{\text{def}}{=} (1/2n) \cdot (q_{d-1}/q_d) = 4(nB)^{2N+2}$, and we prove it by induction. The base case follows immediately since the error of a freshly encrypted ciphertext is bounded by $3(nB)^2 < 4(nB)^{2N+2}$.

We now turn to the inductive step. Let c, c' be level-$(d-1)$ ciphertexts with corresponding public key sets K, K'. The inductive hypothesis tells us the error in c and c' is bounded by E. Using the same analysis as in the proof of Lemma 3.2, we can show that relinearizing all keys in $K \cup K'$ generates an additive error less than $8\lceil \log q_d \rceil (nB)^{2N+1} < (nB)^{2N+2}$, where we used the fact that $q_d < q_0 = 2^{n^\varepsilon}$ for $\varepsilon < 1$. Recall that c_t is the ciphertext obtained in a homomorphic operation after relinearization has been completed but before modulus reduction is performed. Then the following hold:

• In a homomorphic addition, the error of c_t is bounded by $2(nB)^N E + (nB)^{2N+2}$. By Lemma 3.5, the error of c_{add} is bounded by

$$
\begin{aligned}
\frac{q_d}{q_{d-1}} \cdot \left(2(nB)^N E + (nB)^{2N+2}\right) + \|f\|_1 &\leq \frac{2(nB)^N E + (nB)^{2N+2}}{2nE} + nB \\
&\leq \frac{2(nB)^N E}{2nE} + (nB)^{2N+2} + nB \\
&\leq \frac{(nB)^N}{n} + (nB)^{2N+2} + nB \\
&\leq 4(nB)^{2N+2} = E.
\end{aligned}
$$

• In a homomorphic multiplication, the error of c_t is bounded by $nE^2 + (nB)^{2N+2}$.

By Lemma 3.5, the error of c_{mult} is bounded by

$$
\begin{aligned}
\frac{q_d}{q_{d-1}} \cdot \left(nE^2 + (nB)^{2N+2}\right) + \|f\|_1 &\le \frac{nE^2 + (nB)^{2N+2}}{2nE} + nB \\
&\le \frac{nE^2}{2nE} + 2(nB)^{2N+2} \\
&\le \frac{E}{2} + \frac{E}{2} = E.
\end{aligned}
$$

This concludes the inductive step and the proof that ciphertexts of all levels have error at most E.

To correctly decrypt a level-D ciphertext, we must have that

$$
(nB)^{2N+2} = E < \frac{q_D}{2} < \frac{q_0}{2(8n(nB)^{2N+2})^D} = \frac{2^{n^\varepsilon}}{2(8n(nB)^{2N+2})^D},
$$

which yields the theorem statement: $ND = O(n^\varepsilon / \log n)$. □

Security. As in section 3.3, the security of the scheme $\mathcal{E}_{\mathsf{LH}}$ can be based in the $\mathsf{DSPR}_{\phi,q,\chi}$ and $\mathsf{RLWE}_{\phi,q,\chi}$ assumptions. However, unlike in section 3.3, we do not need to assume circular security of the encryption scheme. This is due to the fact that the evaluation key consists of pseudoencryptions of (multiples of) $f^{(d-1)}$ and $(f^{(d-1)})^2$ under a *different* public key $h^{(d)}$ for all $d \in [D]$. Semantic security even given the evaluation key can then be established by a hybrid argument that converts all pseudoencryptions in the evaluation key, one-by-one, to uniform elements in R_q.

LEMMA 3.7. *Let n be a power of 2, let $\phi(x) = x^n + 1$, and let $q = 2^{n^\varepsilon}$ for $\varepsilon \in (0,1)$ and $\chi = D_{\mathbb{Z}^n,r}$ with $r = \mathsf{poly}(n)$. Then the multikey leveled homomorphic encryption scheme $\mathcal{E}_{\mathsf{LH}} = (\mathsf{LH.Keygen}, \mathsf{LH.Enc}, \mathsf{LH.Dec}, \mathsf{LH.Eval})$ described above is secure under the $\mathsf{DSPR}_{\phi,q,\chi}$ and $\mathsf{RLWE}_{\phi,q,\chi}$ assumptions.*

3.4.4. Multikey fully homomorphic encryption. To convert the leveled homomorphic encryption scheme described in section 3.4.2 into a fully homomorphic scheme, we use the multikey bootstrapping theorem (Theorem 3.4). First, we show an upper bound on the depth of the decryption circuit and show that the scheme is bootstrappable.

LEMMA 3.8. *The N-key decryption circuit of the leveled homomorphic encryption scheme described above can be implemented as a polynomial-size arithmetic circuit over $GF(2)$ of depth $O(\log N \cdot (\log \log q_D + \log n))$.*

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_D} can be done using a polynomial-size boolean circuit of depth $O(\log \log q_D + \log n)$ (see, e.g., [29, Lemma 4.5] for a proof). Using a binary tree of polynomial multiplications, the decryption operation above can then be done in depth $O(\log N \cdot (\log \log q_D + \log n))$, as claimed. □

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

THEOREM 3.9. *Let χ is a B-bounded distribution for $B = \mathsf{poly}(n)$, let $q_0 = 2^{n^{\varepsilon}}$ for $\varepsilon \in (0,1)$ and for $d \in [D]$, and let $q_{d-1}/q_d = 8n(nB)^{2N+2}$. Then, there exists a multikey fully homomorphic encryption scheme for $N = O(n^{\varepsilon}/\log^3 n)$ keys, secure under the $\mathsf{DSPR}_{\phi,q,\chi}$ and $\mathsf{RLWE}_{\phi,q,\chi}$ assumptions, and the assumption that the leveled homomorphic encryption scheme $\mathcal{E}_{\mathsf{LH}} = (\mathsf{LH.Keygen}, \mathsf{LH.Enc}, \mathsf{LH.Dec}, \mathsf{LH.Eval})$ described above is weakly circular secure.*

Proof. To apply the multikey bootstrapping theorem (Theorem 3.4), we require that the depth of the decryption circuit is smaller than the depth of the circuits that the scheme can evaluate. That is, we require that

$$\log N \cdot (\log \log q_D + \log n) < C \cdot \frac{\log q_0}{N \cdot \log n}$$

for some universal constant $C > 0$. For $N \leq \sqrt{C/2} \cdot (n^{\varepsilon/2}/\log n)$, we have,

$$N \cdot \log n \cdot \log N \cdot (\log \log q_D + \log n) \leq N^2 \cdot \log n \cdot \cdot (\log \log q_0 + \log n)$$
$$\leq \frac{C}{2} \cdot \frac{n^{\varepsilon}}{\log^2 n} \cdot (1 + \varepsilon) \cdot \log^2 n$$
$$\leq C \cdot n^{\varepsilon} = C \cdot \log q_0,$$

as required. $\qquad\square$

Remark 1. Theorem 3.9 implies that for any $N \in \mathbb{N}$, there exists a multikey fully homomorphic encryption scheme for N keys. This is achieved by choosing ε' such that $n^{\varepsilon'} \leq \sqrt{C/2} \cdot (n^{\varepsilon/2}/\log n)$ and setting $n \geq N^{1/\varepsilon'}$.

We emphasize the fact that bootstrapping (and therefore assuming weak circular security) can be avoided at the cost of obtaining a leveled homomorphic encryption scheme.

4. On-the-fly MPC from multikey FHE. We now show how to construct on-the-fly MPC from multikey FHE. We first construct a basic protocol that is secure against semimalicious adversaries and then describe how to modify the protocol to obtain security against malicious adversaries. As mentioned earlier, the main building block of our construction is multikey fully homomorphic encryption, defined and constructed in section 3.

4.1. 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. The following construction is an on-the-fly MPC protocol secure against semimalicious adversaries. The protocol is defined as follows:

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 $\Pi_{\mathrm{DEC}}{}^{\mathrm{SM}}$ to compute the function $g_c(\mathsf{sk}_1, \ldots, \mathsf{sk}_N) \stackrel{\mathrm{def}}{=} \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$.

We remark that an upper bound on the number of computing parties must be known in advance. This is a direct consequence of the "leveled" nature of our multikey FHE construction with respect to the number of keys.

4.1.1. Security against semimalicious adversaries.

THEOREM 4.1. *Let* $\{\mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})\}_{N>0}$ *be a multikey fully homomorphic encryption scheme, and let* ${\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$ *be an N-party MPC protocol for computing the decryption function* $g_c(\mathsf{sk}_1, \ldots, \mathsf{sk}_N) \stackrel{\mathrm{def}}{=} \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c)$. *If $\mathcal{E}$ is semantically secure and* ${\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$ *is secure against semihonest adversaries corrupting $t < N$ parties, then the above construction is an on-the-fly MPC protocol secure against (static) semimalicious adversaries corrupting t parties and possibly the server S.*

Proof. 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 MPC protocol ${\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$ for decryption.

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: We show security for the case when the server is corrupted; the case when the server is honest is analogous. Let $\mathcal{A}^{\mathrm{SM}}$ be a real-world semimalicious adversary corrupting t clients and the server. 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. Thus, we assume $t < N$, let $T \subsetneq [N]$ be the set of corrupted clients and let $\overline{T} = [N] \backslash T$.

We construct a simulator $\mathcal{S}^{\mathrm{SM}}$ as follows. The simulator receives the inputs of the corrupted parties $\{x_i\}_{i \in T}$ and runs $\mathcal{A}^{\mathrm{SM}}$ on these inputs $\{x_i\}_{i \in T}$. It simulates the messages for all honest parties in the protocol execution with $\mathcal{A}^{\mathrm{SM}}$ by sampling all key tuples correctly but encrypting 0 instead of the honest input x_i (which it doesn't know). In Step 3, it runs the simulator $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ for the protocol ${\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$.

Step 1: For noncomputing parties $i \in \{N+1, \ldots, U\}$ and for honest parties $i \in \overline{T}$, $\mathcal{S}^{\mathrm{SM}}$ 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, the simulator sends $(c_i, \mathsf{pk}_i, \mathsf{ek}_i)$ to $\mathcal{A}^{\mathrm{SM}}$ on behalf of P_i.

At the end of this round, it reads from $\mathcal{A}^{\mathrm{SM}}$'s witness tape the secret keys $\{\mathsf{sk}_i\}_{i \in T}$ and the inputs $\{\widetilde{x}_i\}_{i \in T}$. The simulator sends these inputs to the trusted functionality $\mathcal{F}$ and receives the output $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where $\widetilde{x}_i = x_i$ for honest inputs $i \in \overline{T}$.

Step 2: The simulator receives c from $\mathcal{A}^{\mathrm{SM}}$ as the server's broadcast message.

Step 3: The simulator $\mathcal{S}^{\mathrm{SM}}$ runs the simulator $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ for the decryption protocol (interacting with $\mathcal{A}^{\mathrm{SM}}$). When $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ queries the ideal decryption functionality with secret keys $\{\widetilde{\mathsf{sk}}_i\}_{i \in T}$, $\mathcal{S}^{\mathrm{SM}}$ returns $\widetilde{y}$.

Output: The simulator receives the output of the corrupted parties from $\mathcal{A}^{\mathrm{SM}}$ and returns these as its output.

We prove that $\mathrm{IDEAL}_{\mathcal{F}, \mathcal{S}^{\mathrm{SM}}}(\vec{x}) \stackrel{c}{\approx} \mathrm{REAL}_{\Pi^{\mathrm{SM}}, \mathcal{A}^{\mathrm{SM}}}(\vec{x})$ via a series of hybrids.

Hybrid 0: This is the real-world execution of the protocol.

Hybrid 1: We change how Step 3 is performed. Instead of executing the protocol ${\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$, where honest parties use their individual secret keys, we run the simulator $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ (interacting with $\mathcal{A}^{\mathrm{SM}}$). When $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ queries

the ideal decryption functionality with secret keys $\{\widetilde{\mathsf{sk}}_i\}_{i \in T}$, we return

$$\widetilde{y} = g_c(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N) = \mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c),$$

where $\widetilde{\mathsf{sk}}_i = \mathsf{sk}_i$ for honest secret keys $i \in \overline{T}$. The output of the corrupted parties is defined to be the output of $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$, and the output of the honest parties is defined to be $\widetilde{y}$.

We claim that Hybrid 0 is computationally indistinguishable from Hybrid 1 by the *security of* $\Pi_{\mathrm{DEC}}{}^{\mathrm{SM}}$. Indeed, the security of the decryption protocol $\Pi_{\mathrm{DEC}}{}^{\mathrm{SM}}$ guarantees that as long as we correctly emulate the ideal decryption functionality, the joint output of all parties is computationally indistinguishable in a real-world execution of the protocol with adversary $\mathcal{A}^{\mathrm{SM}}$ (Hybrid 0) and in an ideal-world execution of the protocol with adversary $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$ (Hybrid 1). We correctly emulate the ideal decryption functionality, by definition.

Hybrid 2: We now change how we compute $\widetilde{y}$, the value returned to the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$ when it queries the decryption ideal functionality. Instead of computing $\widetilde{y} = g_c(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N) = \mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c)$, we instead compute

$$\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N),$$

where $\widetilde{x}_i = x_i$ for honest inputs $i \in \overline{T}$, and where for corrupt parties $i \in T$, we recover $\widetilde{x}_i$ by reading $\mathcal{A}^{\mathrm{SM}}$'s witness tape at the end of Step 1. We claim that Hybrid 1 and Hybrid 2 are identically distributed. The adversary $\mathcal{A}^{\mathrm{SM}}$ follows the protocol as specified, so in particular, it performs the homomorphic evaluation correctly. By *correctness of multikey evaluation* we know that c decrypts to $f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$ when decrypted using the secret keys it computed in Step 1, $\{\mathsf{sk}_i\}_{i \in [N]}$; that is, $\mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c) = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$.

Furthermore, because the adversary $\mathcal{A}^{\mathrm{SM}}$ follows the protocol as specified, we know that the secret keys it uses in Step 3 are the same as the ones it computed in Step 1, i.e., $\mathsf{sk}_i = \widetilde{\mathsf{sk}}_i$ for all $i \in T$. We conclude that $\mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c) = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$.

Hybrids 3.k *for* $k = 1, \ldots, N - t$: Let $\overline{T} = \{i_1, \ldots, i_{N-t}\}$. In Hybrid 3.k we change c_{i_k} so that instead of encrypting x_{i_k} it now encrypts 0. More formally, in Hybrid 3.k we have

$$\left\{ c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0) \right\}_{j \leq k} \quad , \quad \left\{ c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j}) \right\}_{j > k}.$$

For ease of notation we let Hybrid 2 be Hybrid 3.0. We claim that the view of $\mathcal{A}^{\mathrm{SM}}$ in Hybrid 3.k is indistinguishable from its view in Hybrid 3.(k−1) by the *semantic security of* $\mathcal{E}$ under public key pk_{i_k}. Indeed, now that we run the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{SM}}$ in Step 3 instead of the real decryption protocol, the secret key sk_{i_k} is only used to encrypt c_{i_k}. So suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 3.k and 3.(k − 1). We construct an adversary $\mathcal{B}$ that breaks the semantic security of $\mathcal{E}$ under public key pk_{i_k}. The reduction $\mathcal{B}$ works as follows:

1. The reduction chooses arbitrary $\{x_i\}$.
2. It receives $(\mathsf{pk}, \mathsf{ek})$ from the semantic security challenger and sets $\mathsf{pk}_{i_k} = \mathsf{pk}$ and $\mathsf{ek}_{i_k} = \mathsf{ek}$; gives $m_0 = 0$ and $m_1 = x_{i_k}$ to the

challenger and receives $c = \mathsf{Enc}(\mathsf{pk}, m_b)$; and sets $c_{i_k} = c$. For all $i \in \overline{T}, i \neq i_k$, it computes $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ honestly. For $j < k$, it computes $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)$, and for $j > k$, it computes $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})$.

3. The reduction runs $\mathcal{A}^{\mathrm{SM}}$: for all $i \in \overline{T}$ gives $(\mathsf{pk}_i, \mathsf{ek}_i, c_i)$ to $\mathcal{A}^{\mathrm{SM}}$ on behalf of P_i and receives c from $\mathcal{A}^{\mathrm{SM}}$.

4. It reads from $\mathcal{A}^{\mathrm{SM}}$'s witness tape the inputs $\{\widetilde{x}_i\}_{i \in T}$ and runs the simulator $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ (interacting with $\mathcal{A}^{\mathrm{SM}}$). When $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$ queries the ideal decryption functionality, it returns $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where $\widetilde{x}_i = x_i$ for inputs $i \in \overline{T}$.

5. The reduction then gives $\mathcal{D}$ $\widetilde{y}$ as the output of all honest parties, as well as the output of $\mathcal{S}^{\mathrm{SM}}_{\Pi_{\mathrm{DEC}}}$.

6. Finally, $\mathcal{B}$ outputs the bit output by $\mathcal{D}$.

When $b = 0$, $\mathcal{B}$ perfectly emulates Hybrid 3.k, whereas if $b = 1$, $\mathcal{B}$ perfectly emulates Hybrid 3.$(k - 1)$. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 3.k and 3.$(k - 1)$, 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 joint output in Hybrid 0 is computationally indistinguishable from the joint output in Hybrid 3.$(N - t)$. But notice that the joint output in Hybrid 3.$(N - t)$ is precisely $\mathrm{IDEAL}_{\mathcal{F}, \mathcal{S}^{\mathrm{SM}}}(\vec{x})$, and the joint output in Hybrid 0 is defined to be $\mathrm{REAL}_{\Pi^{\mathrm{SM}}, \mathcal{A}^{\mathrm{SM}}}(\vec{x})$. We conclude that $\mathrm{IDEAL}_{\mathcal{F}, \mathcal{S}^{\mathrm{SM}}}(\vec{x}) \overset{c}{\approx} \mathrm{REAL}_{\Pi^{\mathrm{SM}}, \mathcal{A}^{\mathrm{SM}}}(\vec{x})$, as desired. $\quad\square$

4.2. Achieving security against malicious adversaries. The protocol described in section 4.1, though secure against semimalicious adversaries, is not secure against fully malicious adversaries. We transform the protocol into one that is secure against malicious corruptions in three steps:

1. First, we replace the decryption protocol in Step 3 with one that is secure against malicious corruptions. More importantly, we change the function it computes to ensure that the secret key used in this protocol is consistent with the public and evaluation keys that the parties computed in Step 1.

2. Second, we add zero-knowledge proofs at each step in the protocol, following the AJW compiler [7, 6] (which is based on the GMW compiler [84]).

3. Finally, in order to maintain the performance guarantees of the scheme, in Step 2 we replace the server's proof with a *succinct argument* (not necessarily ZK). This allows the server to prove that it correctly performed the homomorphic evaluation and the clients to verify the validity of the proof in time that is significantly less than the size of the circuit.

The new decryption protocol. Our first step in handling malicious attacks is to replace the decryption protocol $\Pi_{\mathrm{DEC}}{}^{\mathrm{SM}}$ with one that is secure against malicious adversaries; we will denote it by $\Pi_{\mathrm{DEC}}{}^{\mathrm{MAL}}$. The function being computed by this protocol also needs to change in order to guarantee that the secret key used by each party is consistent with its public and evaluation keys:

$$g_{c, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}((\mathsf{sk}_1, r_1) \ldots, (\mathsf{sk}_N, r_N))$$

$$\overset{\mathrm{def}}{=} \begin{cases} \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c) & \text{if } (\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) = \mathsf{Keygen}(1^\kappa; r_i) \ \ \forall i \in [N], \\ \bot & \text{otherwise.} \end{cases}$$

Intuitively, if the protocol outputs something other than $\bot$, then in particular every corrupt party P_i "knows" a secret key $\widetilde{\mathsf{sk}}_i$ that is consistent with its public and

evaluation keys $(\mathsf{pk}_i, \mathsf{ek}_i)$. By correctness of decryption, this binds P_i to the input $\widetilde{x}_i = \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$, which by semantic security of the FHE must be independent of the honest party's inputs.

We remark that the proceedings version of this work [106] does not change the decryption function but instead adds to Step 1 a zero-knowledge proof π_i^{GEN} for the relation

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

While this guarantees that the public and evaluation keys are well-formed, it does not guarantee that the secret key used in the decryption protocol in Step 3 is consistent with the public and evaluation keys $(\mathsf{pk}_i, \mathsf{ek}_i)$ created and used in Step 1. This allows a corrupt party to use a different secret key sk_i^* in Step 3 and potentially change the outcome of the decryption. We are therefore unable to prove security of that construction. However, the zero-knowledge proofs π_i^{GEN} can be required as an *optimization* to guarantee that an honest server does not accept, store, or compute on ciphertexts that are encrypted under malformed keys (even though the outcome of any joint computation on such a ciphertext would not be decryptable using protocol $\Pi_{\mathrm{DEC}}^{\mathrm{MAL}}$).

Finally, we highlight the fact that if the protocol $\Pi_{\mathrm{DEC}}^{\mathrm{MAL}}$ can implemented using the cloud-assisted protocol of Asharov et al. [7, 6]. Jumping ahead, this yields a 5-round on-the-fly MPC protocol in the CRS-model, secure against malicious corruptions of any $t < N$ parties and possibly the server.

Adding zero-knowledge proofs. The second step in our transformation is to apply the AJW compiler [7, 6] (based on the GMW compiler [84]) to the rest of the protocol (Steps 1 and 2) in order to ensure that parties do not deviate from the protocol specifications. This entails having each party and the server compute a zero-knowledge proof at every round, proving that their message in that round is well-formed and consistent with the protocol transcript.

Because the well-formedness of the public and evaluation keys $(\mathsf{pk}_i, \mathsf{ek}_i)$ is checked in the decryption protocol $\Pi_{\mathrm{DEC}}^{\mathrm{MAL}}$, the parties do not need to compute a separate zero-knowledge proof for this statement (unless required for the optimization described above). Therefore, each party only needs to prove that their ciphertext c_i is well-formed by providing NIZK proof for the NP relation:

$$R^{\mathrm{ENC}} = \{ \ (\ (\mathsf{pk}_i, c_i) \ , \ (x_i, s_i) \ ) \ | \ c_i = \mathsf{Enc}(\mathsf{pk}_i, x_i \ ; \ s_i) \ \} \, .$$

Note that the proof π_i^{ENC} must indeed be noninteractive, for reasons that will become apparent shortly. Informally, this proof will either be broadcast by the server in Step 2 for all parties to verify, or it will be used as a witness in the proof of another NP relation. An interactive zero-knowledge proof would not be convincing in either of these cases, since a valid transcript of the interaction may be simulated without knowing a witness and without the use of any trapdoors.

Maintaining performance guarantees. Verifying a standard zero-knowledge proof for the server's computation in Step 2 requires time proportional to the size of the circuit. However, this computation is deterministic and public; indeed, anyone can have verified the validity of the server's broadcast message by performing the homomorphic evaluation themselves (in time proportional to the size of the circuit). Thus, we replace the server's proof with a *succinct argument* (not necessarily ZK), which allows the server to prove that it correctly performed the homomorphic evaluation,

so that the clients can verify the validity of the proof in time that is significantly less than the size of the circuit. 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, allowing communication complexity in the online phase to be linear in the total input size of the participating parties. In this case, the server will broadcast all ciphertexts and proofs $\{c_i, \pi_i^{\text{ENC}}\}_{i \in [N]}$, the evaluated ciphertext c, and a succinct argument φ showing that it performed the homomorphic evaluation correctly. 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 $\mathsf{poly}(|C|, \kappa)$ accepts. For any uniform circuit C (i.e., locally computable by some Turing machine in time polynomial in their input and κ), the following offer $\mathsf{poly}(\kappa, \log(|C|))$ communication and verification efficiency.[12]

1. Use the argument system of Kilian [99, 100], yielding *interactive* 4-round verification. It relies on expensive PCPs but has been implemented in practice [15].
2. Use the succinct noninteractive arguments (SNARGs and SNARKs) of Micali [108], Bitansky et al. [19], or Goldwasser, Lin, and Rubinstein [89] (see section 2.3). These are noninteractive[13] but are secure only in the random oracle model [13] (in the case of CS proofs) or hold in the standard model but require a nonfalsifiable assumption [112, 80, 19]. Some variants rely on PCPs, PIR, or FHE.
3. Use SNARKs built via the bootstrapping technique of Bitansky et al. [20] applied to a suitable preprocessing SNARK (e.g., [69, 115, 21]). Such boostrapped SNARKs have been implemented in practice [16].[14]

Note that we cannot directly use verifiable computation protocols in the preprocessing model (e.g., [68, 40, 5]) or SNARGs/SNARKs where the CRS depends on the circuit to be computed or where its size is at least as big as the computation, e.g., [90, 103, 69, 115, 104, 21]. These have two problems. First, they require the clients to participate in a preprocessing phase where their computation is proportional to the size of the circuit, violating the performance requirements of on-the-fly MPC. Second, with this preprocessing step the model loses its dynamic nature, where users can compute many different functions on their inputs and can choose these functions dynamically, on-the-fly. Indeed, using these solutions would limit the parties to only compute functions for which they have already performed the corresponding preprocessing work or computed the corresponding CRS. (The second problem can be remedied by using a SNARK with universal preprocessing, as implemented by Ben-Sasson et al. [17].)

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

---

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

[13]In our protocol, each party can run Gen in Step 1 and send the vrs to the server in that step. Or in the case of CS proofs, where only a description of a hash function is required, this can be added to the CRS of the protocol.

[14]The boostrapping is phrased in terms of Turing machines (or RAM machines). Since C is assumed to be a uniform circuit, we can consider the Turing machine that computes the gates of C and evaluates them, whose running time is polynomial in $|C|$ and κ.

4. Use the argument system of Goldwasser, Kalai, and Rothblum [88] for a 1-round solution.[15] It relies on PIR.

Verification for large inputs: We can make the communication and verification complexities depend merely polylogarithmically on the size of the relevant inputs $x_1, \ldots, x_N$. This requires a succint argument system that is a *proof of knowledge*. This is satisfied by Micali's construction of CS proofs under Valiant's analysis [108, 131] and by SNARKs [19, 20]. The complexity of these arguments depends polynomially on the size of the statement being proven but merely polylogarithmically on the size of the witness for the statement. 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 $\mathcal{H} = \{H_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^\kappa\}$.

In the offline stage, every party P_i samples a hash key hk_i and computes the digest $d_i = H_{\mathsf{hk}_i}(c_i)$. Party P_i then sends $(c_i, \pi_i^{\mathrm{ENC}}, \mathsf{hk}_i, d_i)$ to the cloud. Each party P_i remembers its own (hk_i, d_i) pair but can forget the potentially long $x_i, c_i, \pi_i^{\mathrm{ENC}}$. 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 $\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{ENC}}, \ldots, \widetilde{c}_N, \widetilde{\pi}_N^{\mathrm{ENC}}$ such that $d_i = H_{\mathsf{hk}_i}(\widetilde{c}_i)$ 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 $\widetilde{\pi}_i^{\mathrm{ENC}}$ is a valid proof."

The construction is secure, since whenever the server convinces the clients, it actually "knows" such $\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{ENC}}, \ldots, \widetilde{c}_N, \widetilde{\pi}_N^{\mathrm{ENC}}$ 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 (by the soundness of $\widetilde{\pi}_i^{\mathrm{ENC}}$) and its plaintext can be efficiently extracted using the secret key used by P_i in the decryption protocol in Step 3.

4.2.1. Formal protocol. We now write a formal description of our construction of on-the-fly MPC, secure against malicious adversaries, and providing correct verification for large inputs. Our construction requires the following building blocks:

- A semantically secure multikey fully homomorphic family of encryption schemes $\mathcal{E} = \big\{ \mathcal{E}^{(N)} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}) \big\}_{N>0}$.
- A family of collision-resistant hash functions $\mathcal{H} = \big\{ H_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^\kappa \big\}_{\mathsf{hk}}$.
- A NIZK argument system $\Pi^{\mathrm{ENC}} = (\mathsf{Setup}^{\mathrm{ENC}}, \mathsf{Prove}^{\mathrm{ENC}}, \mathsf{Verify}^{\mathrm{ENC}}, \mathsf{Sim}^{\mathrm{ENC}})$ for the NP relation $R^{\mathrm{ENC}} = \{ ((\mathsf{pk}, c) , (x, s)) \mid c = \mathsf{Enc}(\mathsf{pk}, x \; ; \; s) \}$.
- An adaptively extractable SNARK system $\Phi = (\mathsf{Setup}^\Phi, \mathsf{Prove}^\Phi, \mathsf{Verify}^\Phi, \mathsf{Ext}^\Phi)$ for all of NP.
- An N-party MPC protocol, secure against malicious adversaries corrupting $t < N$ parties, for computing the family of decryption functions

$$g_{c, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}((\mathsf{sk}_1, r_1) \ldots, (\mathsf{sk}_N, r_N))$$

$$\stackrel{\mathrm{def}}{=} \begin{cases} \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c) & \text{if } (\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) = \mathsf{Keygen}(1^\kappa; r_i) \;\; \forall i \in [N], \\ \bot & \text{otherwise.} \end{cases}$$

The protocol is defined as follows:

Input: All parties and the server receive as input the common reference string $\mathsf{crs}^{\mathrm{ENC}}$ for the NIZK proof system Π^{ENC}. If CS proofs are used as the SNARK system,

---

[15]The protocol has two rounds, but (as in the case of SNARGs and SNARKs) the first round is a challenge that is independent of the language and the statement and can therefore be precomputed by the clients in Step 1 of our protocol. Each challenge can only be used for one proof, so the client must refresh the challenge after each computation.

the (description) of the random-oracle hash function is also given to all parties and the server.

Step 1: For $i \in [U]$, party P_i samples a key tuple $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i)$, encrypts its input x_i, and computes a NIZK showing that the ciphertext is well-formed:

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

$$\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Prove}^{\mathrm{ENC}}(\; (\mathsf{pk}_i, c_i) \; , \; (x_i, s_i) \;).$$

It also samples a hash key hk_i and computes the digest of the ciphertext: $d_i = H_{\mathsf{hk}_i}(c_i)$. It additionally creates a verification reference string and private verification key: $(\mathsf{vrs}_i, \mathsf{priv}_i) \leftarrow \mathsf{Setup}^\Phi(1^\kappa)$.

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

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 its secret key and key-generation randomness (sk_i, r_i), the hash key and digest (hk_i, d_i), and its private verification key priv_i.

A function F, represented as a uniform 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 $c := \mathsf{Eval}(C, (c_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (c_N, \mathsf{pk}_N, \mathsf{ek}_N))$ and creates succinct NIZK arguments $\{\varphi_i\}_{i \in [N]}$ for the NP language

$$
\begin{aligned}
L = \{ \quad & \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]} \quad | \\
& \exists \; (\widetilde{c}_1, \widetilde{\pi}_1^{\mathrm{ENC}}) \; , \ldots, (\widetilde{c}_N, \widetilde{\pi}_N^{\mathrm{ENC}}) \;\; \text{such that} \\
& d_i = H_{\mathsf{hk}_i}(\widetilde{c}_i) \quad\;\; \text{and} \\
& \mathsf{Verify}^{\mathrm{ENC}}(\; (\mathsf{pk}_i, \widetilde{c}_i) \; , \widetilde{\pi}_i^{\mathrm{ENC}}) = 1 \quad\;\; \text{and} \\
& c = \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N)) \; \},
\end{aligned}
$$

where C is given in its succinct uniform representation.

To compute φ_i, the server uses the verification reference string vrs_i. If CS proofs are used as the SNARK system, the server need only compute a single proof φ that can be verified by all.

The server broadcasts $(c, \varphi_1, \ldots, \varphi_N)$ to all parties $P_1, \ldots, P_N$, together with the tuple $\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]}$.

Step 3: Party P_i runs $\mathsf{Verify}^\Phi(\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]}, \varphi_i)$ to verify the argument φ_i. If verification is successful for all partiers, they run an MPC protocol $\Pi_{\mathrm{DEC}}{}^{\mathrm{MAL}}$ to compute the function

$$
\begin{aligned}
& g_{c, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}((\mathsf{sk}_1, r_1) \ldots, (\mathsf{sk}_N, r_N)) \\
& \overset{\text{def}}{=} \begin{cases} \mathsf{Dec}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, c) & \text{if } (\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) = \mathsf{Keygen}(1^\kappa; r_i) \;\; \forall i \in [N], \\ \bot & \text{otherwise.} \end{cases}
\end{aligned}
$$

4.2.2. Proof of security.

THEOREM 4.2. *Let* $\mathcal{E}, \Pi_{\mathrm{DEC}}{}^{\mathrm{MAL}}, \mathcal{H}, \Pi^{\mathrm{ENC}}, \Phi$ *be as described in section* 4.2.1. *Then the above construction is an on-the-fly MPC protocol secure against malicious adversaries corrupting $t < N$ parties and possibly the server S.*

Proof. 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 $\Pi_{\mathrm{DEC}}^{\mathrm{MAL}}$.

Performance: The zero-knowledge proofs π_i^{ENC} are independent of C (which is chosen only at a later step). The size of c is independent of $|C|$, by the compactness of the homomorphic evaluation. Moreover, by the succinctness of the SNARK and uniformity of C, the proof φ has size polylogarithmic in $|C|$, and its verification depends only polylogarithmically on the size of the ciphertexts c_i (and therefore polylogarithmically on the size of the inputs x_i as well). Thus, the communication complexity of the protocol is polylogarithmic in $|C|$, and the computation time of each party P_i is at most polylogarithmic in $|C|$ and the total size of the inputs, and polynomial in y and its input x_i.

Security: We show security for the case when the server is corrupted; the case when the server is honest is analogous. Let $\mathcal{A}^{\mathrm{MAL}}$ be a real-world semimalicious adversary corrupting t clients and the server. 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. Thus, we assume $t < N$, let $T \subsetneq [N]$ be the set of corrupted clients, and let $\overline{T} = [N] \backslash T$.

We construct a simulator $\mathcal{S}^{\mathrm{MAL}}$ as follows. The simulator receives the inputs of the corrupted parties $\{x_i\}_{i \in T}$ and runs $\mathcal{A}^{\mathrm{MAL}}$ on these inputs $\{x_i\}_{i \in T}$. It simulates the messages for all honest parties in the protocol execution with $\mathcal{A}^{\mathrm{MAL}}$. In Step 1, it samples all key tuples correctly but encrypts 0 instead of the honest input x_i (which it doesn't know) and computes simulated proofs π_i^{ENC}. In Step 2, it fixes an honest party h and extracts the witness $\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in [N]}$ of the argument φ_h. For all corrupted parties $i \in T$, the simulator extracts the corrupted input $\widetilde{x}_i$ from the proof $\widetilde{\pi}_i^{\mathrm{ENC}}$, submits these to the ideal functionality $\mathcal{F}$, and obtains an output $\widetilde{y}$. In Step 3, it runs the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ for the protocol $\Pi_{\mathrm{DEC}}^{\mathrm{MAL}}$, returning $\widetilde{y}$ when it calls the ideal decryption functionality. More formally the following occurs:

Step 1: The simulator creates the CRS for the NIZK Π^{ENC}, together with a trapdoor key and an extraction key:

$$\left(\mathsf{crs}^{\mathrm{ENC}}, \mathsf{tk}^{\mathrm{ENC}}, \mathsf{extk}^{\mathrm{ENC}}\right) \leftarrow \mathsf{Setup}^{\mathrm{ENC}}(1^\kappa).$$

For noncomputing parties $i \in \{N+1, \ldots, U\}$ and for honest parties $i \in \overline{T}$, the simulator computes $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and samples hk_i honestly. The simulator also runs the verification setup honestly: $(\mathsf{vrs}_i, \mathsf{priv}_i) \leftarrow \mathsf{Setup}^\Phi(1^\kappa)$.

The simulator computes an encryption of 0 and simulated zero-knowledge proofs:

$$c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0) \quad , \quad \pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}(\ \mathsf{tk}^{\mathrm{ENC}}\ , \ (\mathsf{pk}_i, c_i)\ ).$$

It computes the digest $d_i = H_{\mathsf{hk}_i}(c_i)$ honestly. For each party P_i, $\mathcal{S}^{\mathrm{MAL}}$ sends $(\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\mathrm{ENC}}, \mathsf{hk}_i, d_i, \mathsf{vrs}_i)$ to $\mathcal{A}^{\mathrm{MAL}}$ on behalf of P_i.

Step 2: The simulator receives $(c, \varphi_1, \ldots, \varphi_N)$ from $\mathcal{A}^{\mathrm{MAL}}$, together with the tuples $\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]}$. The simulator verifies φ_i for all honest parties $i \in \overline{T}$ and for a fixed honest party $h \in \overline{T}$ uses the SNARG extractor to extract witness $\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in [N]}$ from φ_h:

$$\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in [N]} \leftarrow \mathsf{Ext}^\Phi\left(\ \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]}\ , \ \varphi_h\ \right).$$

It outputs $\perp$ if for any $i \in [N]$, verification fails for φ_i or $\widetilde{\pi}_i^{\text{ENC}}$ or if $d_i \neq H_{\mathsf{hk}_i}(\widetilde{c}_i)$. It also outputs $\perp$ if $c \neq \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))$, or if $\widetilde{c}_i \neq c_i$ for some honest $i \in \overline{T}$.

Step 3: The simulator runs the decryption simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ for protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ (interacting with $\mathcal{A}^{\text{MAL}}$). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\widetilde{\mathsf{sk}}_i, \widetilde{r}_i\}_{i \in T}$, the simulator checks that $\mathsf{Keygen}(1^\kappa ; \widetilde{r}_i) = (\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i)$ for all $i \in T$. If the check fails, it outputs $\perp$. Otherwise, it decrypts $\widetilde{c}_i$ with the secret key $\widetilde{\mathsf{sk}}_i$ to obtain the corrupted input $\widetilde{x}_i$ (if $\mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_j) = \perp$, it returns $\perp$):

$$\widetilde{x}_i := \mathsf{Dec}\left(\widetilde{\mathsf{sk}}_i, \widetilde{c}_j\right).$$

Finally, it submits inputs $\{\widetilde{x}_i\}_{i \in T}$ to the ideal functionality $\mathcal{F}$ and obtains output $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where $\widetilde{x}_i = x_i$ for honest parties $i \in \overline{T}$. It returns $\widetilde{y}$ to the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$.

Output: The simulator receives the output of the corrupted parties from $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ and returns these as its output.

We prove that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{MAL}}}(\vec{x}) \overset{c}{\approx} \text{REAL}_{\Pi^{\text{MAL}}, \mathcal{A}^{\text{MAL}}}(\vec{x})$ via a hybrid argument.

Hybrid 0: This is a real-world execution of the protocol.

Hybrid 1: We change how Step 3 is performed. Instead of executing the protocol $\Pi_{\text{DEC}}^{\text{MAL}}$, where honest parties use their individual secret keys, we run the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (interacting with $\mathcal{A}^{\text{MAL}}$). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret keys and randomness $\{\widetilde{\mathsf{sk}}_i, \widetilde{r}_i\}_{i \in T}$, we return

$$\widetilde{y} = g_{c, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}\left(\left(\widetilde{\mathsf{sk}}_1, \widetilde{r}_1\right) \ldots, \left(\widetilde{\mathsf{sk}}_N, \widetilde{r}_N\right)\right),$$

where $\widetilde{\mathsf{sk}}_i = \mathsf{sk}_i$ and $\widetilde{r}_i = r_i$ for honest parties $i \in \overline{T}$. We define the output of the corrupted parties to be the output of $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ and the output of the honest parties to be $\widetilde{y}$.

We claim that Hybrid 0 is computationally indistinguishable from Hybrid 1 by the *security of* $\Pi_{\text{DEC}}^{\text{MAL}}$. Indeed, the security of the decryption protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ guarantees that as long as we correctly emulate the ideal decryption functionality, the joint output of all parties is computationally indistinguishable in a real-world execution of the protocol with adversary $\mathcal{A}^{\text{MAL}}$ (Hybrid 0) and in an ideal-world execution of the protocol with adversary $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (Hybrid 1). We correctly emulate the ideal decryption functionality, by definition.

Hybrid 2: Hybrid 2 is the same as Hybrid 1 except that we use the extractor Ext^Φ to extract a witness $\{(\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}})\}_{i \in [N]}$ from φ_h:

$$\{\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \mathsf{Ext}^\Phi\left(\ \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]}\ ,\ \varphi_h\ \right).$$

We define the output of the protocol to be $\perp$ if for any $i \in [N]$, verification fails for $\widetilde{\pi}_i^{\text{ENC}}$ or $d_i \neq H_{\mathsf{hk}_i}(\widetilde{c}_i)$. We also output $\perp$ if $c \neq \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))$, where c is the ciphertext returned by $\mathcal{A}^{\text{MAL}}$ in Step 2. By the *adaptive extractability* property of Φ, we know that this event happens with negligible probability. Therefore, Hybrid 1 and Hybrid 2 are statistically close.

Note that we require Φ to satisfy *adaptive* extractability because the adversary is free to choose the statement of the proof *after* it sees vrs_h.

Hybrid 3: In Hybrid 3, we additionally let the output of the protocol be $\perp$ if $\widetilde{c}_i \neq c_i$ for any honest $i \in \overline{T}$.

We claim that Hybrids 2 and 3 are statistically close by the *collision-resistance* of $\mathcal{H}$. Indeed, Hybrids 2 and 3 are identical except in the case when all previous checks pass but there exists $j \in \overline{T}$ such that $\widetilde{c}_j \neq c_j$. Let ε be the probability, conditioned on all other checks passing, that there exists such a $j \in \overline{T}$. Suppose, for the sake of contradiction, that ε is nonnegligible. Then we construct an adversary $\mathcal{B}$ that breaks the collision-resistance of $\mathcal{H}$. The reduction $\mathcal{B}$ works as follows:

1. The reduction chooses arbitrary inputs $\{x_i\}$.
2. It creates the NIZK CRS honestly, $(\mathsf{crs}^{\mathrm{ENC}}, \cdot) \leftarrow \mathsf{Setup}^{\mathrm{ENC}}(1^\kappa)$, and runs $\mathcal{A}^{\mathrm{MAL}}$ on inputs $\{x_i\}_{i \in T}$ and $\mathsf{crs}^{\mathrm{ENC}}$ as the CRS.
3. For all noncomputing parties and honest parties, it samples key tuples $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and encrypts the input correctly: $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i \; ; \; s_i)$. It creates honest proofs $\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Prove}^{\mathrm{ENC}}(\; (\mathsf{pk}_i, c_i) \;, \; (x_i, s_i) \;)$. It also runs the verification setup honestly to generate a verification reference string $(\mathsf{vrs}_i, \cdot) \leftarrow \mathsf{Setup}^\Phi(1^\kappa)$.
4. When it receives a hash key hk from the collision-resistance challenger, the reduction guesses an honest index $i^* \leftarrow \overline{T}$ uniformly at random and sets $\mathsf{hk}_{i^*} = \mathsf{hk}$. For all other $i \neq i^*$, it samples hk_i honestly. Finally, for all noncomputing and honest parties, it computes the digest $d_i = H_{\mathsf{hk}_i}(c_i)$.
5. It sends $\{\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\mathrm{ENC}}, \mathsf{hk}_i, d_i\}_{i \in \overline{T}}$ to $\mathcal{A}^{\mathrm{MAL}}$.
6. When it receives a ciphertext c and proofs $\varphi_1, \ldots, \varphi_N$ from $\mathcal{A}^{\mathrm{MAL}}$, along with the set $\{\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\}$, it runs the extractor

$$\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in [N]} \leftarrow \mathsf{Ext}^\Phi\left(\; \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]} \;, \; \varphi_h \;\right).$$

7. Finally, it submits c_{i^*} and $\widetilde{c}_{i^*}$ to the collision-resistance challenger as its collision.

If all previous checks pass, then in both hybrids we have that $H(c_j) = H(\widetilde{c}_j) = d_j$. Therefore the probability that $\mathcal{B}$ submits a valid collision to the collision challenger is $\varepsilon/|\overline{T}|$. If ε is nonnegligible, then $\mathcal{B}$ breaks the collision-resistance property of the hash family $\mathcal{H}$.

Hybrid 4: In Hybrid 4, we additionally let the output of the protocol be $\perp$ if $\mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i) = \perp$ for any corrupt $i \in T$, where $\widetilde{\mathsf{sk}}_i$ is the secret key output by the decryption protocol simulator $\mathcal{S}^{\mathrm{MAL}}_{\Pi_{\mathrm{DEC}}}$ and $\widetilde{c}_i$ is extracted from the succinct argument φ_h, as in Hybrids 2 and 3.

We claim that Hybrid 3 and Hybrid 4 are statistically close by the *soundness* of the NIZK Π^{ENC}. Indeed, Hybrids 3 and 4 are identical except in the case when all previous checks pass but there exists $j \in T$ such that $\mathsf{Dec}(\widetilde{\mathsf{sk}}_j, \widetilde{c}_j) = \perp$. By *correctness of decryption*, this happens if and only if $\not\exists (\widetilde{x}_j, \widetilde{s}_j)$ such that $\mathsf{Enc}(\mathsf{pk}_j, \widetilde{x}_j \; ; \; \widetilde{s}_j) = \widetilde{c}_j$, or in other words, if $(\mathsf{pk}_j, \widetilde{c}_j) \notin L^{\mathrm{ENC}}$. Let ε be the probability, conditioned on all other checks passing, that there exists an index $j \in T$ such that $(\mathsf{pk}_j, \widetilde{c}_j) \notin L^{\mathrm{ENC}}$. Suppose, for the sake of contradiction, that ε is nonnegligible. Then we construct an adversary $\mathcal{B}$ that breaks the soundness of Π^{ENC}. The reduction $\mathcal{B}$ works as follows:

1. The reduction chooses arbitrary inputs $\{x_i\}$.
2. It receives the CRS from the soundness challenger and runs $\mathcal{A}^{\mathrm{MAL}}$ on inputs $\{x_i\}_{i \in T}$ and the CRS.

3. For all noncomputing parties and honest parties, it samples keys $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and encrypts the input correctly: $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i \; ; s_i)$. It creates honest proofs $\pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Prove}^{\mathrm{ENC}}(\; (\mathsf{pk}_i, c_i) \; , \; (x_i, s_i) \;)$. It also runs the verification setup honestly to generate a verification reference string $(\mathsf{vrs}_i, \cdot) \leftarrow \mathsf{Setup}^\Phi(1^\kappa)$.

4. It samples hk_i honestly and computes the digest $d_i = H_{\mathsf{hk}_i}(c_i)$.

5. It sends $\{\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\mathrm{ENC}}, \mathsf{hk}_i, d_i\}_{i \in \overline{T}}$ to $\mathcal{A}^{\mathrm{MAL}}$.

6. When it receives a ciphertext c and proofs $\varphi_1, \ldots, \varphi_N$ from $\mathcal{A}^{\mathrm{MAL}}$, along with the set $\{\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\}$, it runs the extractor

$$\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in [N]} \leftarrow \mathsf{Ext}^\Phi \left(\; \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]} \; , \; \varphi_h \; \right).$$

7. It runs the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ (interacting with $\mathcal{A}^{\mathrm{MAL}}$). When $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\widetilde{\mathsf{sk}}_i, \widetilde{r}_i\}_{i \in T}$, it checks that $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i) = \mathsf{Keygen}(1^\kappa \; ; \; \widetilde{r}_i)$ for all $i \in [N]$. If this check fails, it returns $\bot$. Otherwise, it chooses a corrupt $i^* \leftarrow T$ uniformly at random and submits $\widetilde{\pi}_{i^*}^{\mathrm{ENC}}$ as its proof forgery.

If all previous checks pass, then in both hybrids we have that $\mathsf{Verify}(\; (\mathsf{pk}_i, \widetilde{c}_i) \; , \; \widetilde{\pi}_i^{\mathrm{ENC}} \;) = 1$ for all $i \in [N]$ (see Hybrid 2). Therefore, the probability that $\mathcal{B}$ submits a valid forgery to the soundness challenger is $\varepsilon / |T|$. If ε is nonnegligible, then $\mathcal{B}$ breaks the soundness property of the NIZK Π^{ENC}.

Hybrid 5: We now change how we compute $\widetilde{y}$, the value returned to the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ when it queries the decryption ideal functionality. Instead of computing $\widetilde{y} = g_{c, \mathsf{pk}_1, \mathsf{ek}_1, \ldots, \mathsf{pk}_N, \mathsf{ek}_N}((\widetilde{\mathsf{sk}}_1, \widetilde{r}_1) \ldots, (\widetilde{\mathsf{sk}}_N, \widetilde{r}_N))$, we first check if $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i) = \mathsf{Keygen}(1^\kappa \; ; \; \widetilde{r}_i)$ for all $i \in T$. If this check fails, we return $\bot$; otherwise we decrypt each malicious $\widetilde{c}_i$ and evaluate f on the resulting inputs:

$$\widetilde{y} = \begin{cases} f(\widetilde{x}_1, \ldots, \widetilde{x}_N) & \text{if } (\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i) = \mathsf{Keygen}(1^\kappa \; ; \; \widetilde{r}_i) \; \forall i \in T, \\ \bot & \text{otherwise,} \end{cases}$$

where $\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$ for $i \in T$ and $\widetilde{x}_i = x_i$ for $i \in \overline{T}$.

We claim that Hybrid 5 and Hybrid 4 are statistically close. In the case when $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i) \neq \mathsf{Keygen}(1^\kappa \; ; \; \widetilde{r}_i)$ for some $i \in T$, both hybrids output $\bot$. We focus on the case when this check passes for all parties, so that $\widetilde{\mathsf{sk}}_i$ is guaranteed to be a *valid* secret key for its corresponding public and evaluation keys. In both hybrids, we know that $c = \mathsf{Eval}(C, (\widetilde{c}_1, \mathsf{pk}_1, \mathsf{ek}_1), \ldots, (\widetilde{c}_N, \mathsf{pk}_N, \mathsf{ek}_N))$ (see Hybrid 2). By *soundness* of Π^{ENC}, we know that all $\widetilde{c}_i$'s are fresh encryptions, so by *correctness of multikey evaluation* we know that $\mathsf{Dec}(\widetilde{\mathsf{sk}}_1, \ldots, \widetilde{\mathsf{sk}}_N, c) = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where we define $\widetilde{\mathsf{sk}}_i = \mathsf{sk}_i$ for all honest $i \in \overline{T}$ and $\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$ for all $i \in [N]$. Furthermore, since $\widetilde{c}_i = c_i$ for all honest $i \in \overline{T}$ (see Hybrid 3), we know that $\widetilde{x}_i = x_i$ for all $i \in \overline{T}$ by *correctness of decryption*.

Hybrid 6: In Hybrid 6, we change how we compute the proofs π_i^{ENC}. Instead of computing real proofs, we use the NIZK simulator to create simulated proofs:

$$\left\{ \pi_i^{\mathrm{ENC}} \leftarrow \mathsf{Sim}(\; \mathsf{tk}^{\mathrm{ENC}} \; , \; (\mathsf{pk}_i, c_i) \;) \right\}_{i \in \overline{T}}.$$

We claim that Hybrid 6 is computationally indistinguishable from Hybrid 5 by the *unbounded zero-knowledge property* of the proof system Π^{ENC}. Suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes

between Hybrids 5 and 6. We construct an adversary $\mathcal{B}$ that breaks zero-knowledge of Π^{ENC}. The reduction $\mathcal{B}$ works as follows:

1. The reduction chooses arbitrary inputs $\{x_i\}$.
2. It receives the CRS from the zero-knowledge challenger and runs $\mathcal{A}^{\mathrm{MAL}}$ on inputs $\{x_i\}_{i \in T}$ and the CRS.
3. For all noncomputing parties and honest parties, it samples keys $(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and encrypts the input correctly: $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i ; s_i)$. It creates proofs π_i^{ENC} by calling its oracle with statement (pk_i, c_i) and witness (x_i, s_i). It also runs the verification setup honestly to generate a verification reference string $(\mathsf{vrs}_i, \cdot) \leftarrow \mathsf{Setup}^\Phi(1^\kappa)$.
4. It samples hk_i honestly and computes the digest $d_i = H_{\mathsf{hk}_i}(c_i)$.
5. It sends $\{\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\mathrm{ENC}}, \mathsf{hk}_i, d_i\}_{i \in \overline{T}}$ to $\mathcal{A}^{\mathrm{MAL}}$.
6. When it receives a ciphertext c and proofs $\varphi_1, \ldots, \varphi_N$ from $\mathcal{A}^{\mathrm{MAL}}$, along with the set $\{\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\}$, it runs the extractor

$$\{\widetilde{c}_i, \widetilde{\pi}_i^{\mathrm{ENC}}\}_{i \in [N]} \leftarrow \mathsf{Ext}^\Phi \left(\ \{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]} \ , \ \varphi_h \ \right).$$

7. It runs the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ (interacting with $\mathcal{A}^{\mathrm{MAL}}$). When $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\widetilde{\mathsf{sk}}_i, \widetilde{r}_i\}_{i \in T}$, it checks that $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i) \neq \mathsf{Keygen}(1^\kappa ; \ \widetilde{r}_i)$. If this check fails, it returns $\perp$; otherwise it returns $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where $\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$ for $i \in T$ and $\widetilde{x}_i = x_i$ for $i \in \overline{T}$.
8. At the end of the protocol, it forwards $\mathcal{A}^{\mathrm{MAL}}$'s output to $\mathcal{D}$ as the output of the corrupt parties and gives $\widetilde{y}$ to $\mathcal{D}$ as the output of the honest parties.

When $\mathcal{B}$'s oracle is the prover oracle $\mathcal{P}(\cdot)$, $\mathcal{B}$ perfectly emulates Hybrid 5, whereas if the oracle is the simulation oracle $\mathcal{SIM}_{\mathsf{tk}}(\cdot)$, $\mathcal{B}$ perfectly emulates Hybrid 6. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 5 and 6, then $\mathcal{B}$ breaks the zero-knowledge property of Π^{ENC}.

Hybrids 7.*k for* $k = 1, \ldots, N - t$: Let $\overline{T} = \{i_1, \ldots, i_{N-t}\}$. In Hybrid 7.*k* we change c_{i_k} so that instead of encrypting x_{i_k} it now encrypts 0. More formally, in Hybrid 7.*k* we have

$$\left\{ c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0) \right\}_{j \leq k} \quad , \quad \left\{ c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j}) \right\}_{j > k}.$$

For ease of notation we let Hybrid 6 be Hybrid 7.0. We claim that the view of $\mathcal{A}^{\mathrm{MAL}}$ in Hybrid 7.*k* is indistinguishable from its view in Hybrid 7.(*k*−1) by the *semantic security of* $\mathcal{E}$ under public key pk_{i_k}. Indeed, now that we run the simulator $\mathcal{S}_{\Pi_{\mathrm{DEC}}}^{\mathrm{MAL}}$ in Step 3 instead of the real decryption protocol, the secret key sk_{i_k} is only used to encrypt c_{i_k}. So suppose, for the sake of contradiction, that there exists an algorithm $\mathcal{D}$ that distinguishes between hybrids 7.*k* and 7.(*k*−1). We construct an adversary $\mathcal{B}$ that breaks the semantic security of $\mathcal{E}$ under public key pk_{i_k}. The reduction $\mathcal{B}$ works as follows:

1. The reduction chooses arbitrary $\{x_i\}$.
2. It creates the NIZK CRS honestly, $(\mathsf{crs}^{\mathrm{ENC}}, \mathsf{tk}^{\mathrm{ENC}}) \leftarrow \mathsf{Setup}^{\mathrm{ENC}}(1^\kappa)$, and runs $\mathcal{A}^{\mathrm{MAL}}$ on inputs $\{x_i\}_{i \in T}$ and $\mathsf{crs}^{\mathrm{ENC}}$ as the CRS.
3. It receives $(\mathsf{pk}, \mathsf{ek})$ from the semantic security challenger and sets $\mathsf{pk}_{i_k} = \mathsf{pk}$ and $\mathsf{ek}_{i_k} = \mathsf{ek}$, gives $m_0 = 0$ and $m_1 = x_{i_k}$ to the challenger, and receives $c = \mathsf{Enc}(\mathsf{pk}, m_b)$. It sets $c_{i_k} = c$. For all $i \in \overline{T}, i \neq i_k$, it

computes $(\mathsf{pk}_i, \cdot, \mathsf{ek}_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ honestly. For $j < k$, it computes $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, 0)$, and for $j > k$, and it computes $c_{i_j} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i_j}, x_{i_j})$.

4. For all noncomputing and honest parties, it creates simulated proofs $\pi_i^{\text{ENC}} \leftarrow \mathsf{Sim}(\mathsf{tk}^{\text{ENC}}, (\mathsf{pk}_i, c_i))$ using the trapdoor tk^{ENC}. It also runs the verification setup honestly to generate a verification reference string $(\mathsf{vrs}_i, \cdot) \leftarrow \mathsf{Setup}^\Phi(1^\kappa)$.

5. It samples hk_i honestly and computes the digest $d_i = H_{\mathsf{hk}_i}(c_i)$.

6. It sends $\{\mathsf{pk}_i, \mathsf{ek}_i, c_i, \pi_i^{\text{ENC}}, \mathsf{hk}_i, d_i\}_{i \in \overline{T}}$ to $\mathcal{A}^{\text{MAL}}$.

7. When it receives a ciphertext c and proofs $\varphi_1, \ldots, \varphi_N$ from $\mathcal{A}^{\text{MAL}}$, along with the set $\{\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i\}$, it runs the extractor

$$\{\widetilde{c}_i, \widetilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \mathsf{Ext}^\Phi \left(\{(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right).$$

8. It runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (interacting with $\mathcal{A}^{\text{MAL}}$). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\widetilde{\mathsf{sk}}_i, \widetilde{r}_i\}_{i \in T}$, it checks that $(\mathsf{pk}_i, \widetilde{\mathsf{sk}}_i, \mathsf{ek}_i) \neq \mathsf{Keygen}(1^\kappa ; \widetilde{r}_i)$. If this check fails, it returns $\bot$; otherwise it returns $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_N)$, where $\widetilde{x}_i := \mathsf{Dec}(\widetilde{\mathsf{sk}}_i, \widetilde{c}_i)$ for $i \in T$ and $\widetilde{x}_i = x_i$ for $i \in \overline{T}$.

9. At the end of the protocol, it forwards $\mathcal{A}^{\text{MAL}}$'s output to $\mathcal{D}$ as the output of the corrupt parties and gives $\widetilde{y}$ to $\mathcal{D}$ as the output of the honest parties.

When $b = 0$, $\mathcal{B}$ perfectly emulates Hybrid 7.k, whereas if $b = 1$, $\mathcal{B}$ perfectly emulates Hybrid 7.$(k-1)$. Therefore, if $\mathcal{D}$ can distinguish between Hybrids 7.k and 7.$(k-1)$, 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 joint output in Hybrid 0 is computationally indistinguishable from the joint output in Hybrid 7.$(N-t)$. Notice that the joint output in Hybrid 7.$(N-t)$ is precisely $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{MAL}}}(\vec{x})$, and the joint output in Hybrid 0 is defined to be $\text{REAL}_{\Pi^{\text{SM}}, \mathcal{A}^{\text{MAL}}}(\vec{x})$. We conclude that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{MAL}}}(\vec{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi^{\text{MAL}}, \mathcal{A}^{\text{MAL}}}(\vec{x})$, as desired. $\qquad\square$

4.2.3. Efficient NIZKs to prove plaintext knowledge. The protocol described in section 4.2.1 requires a NIZK argument system for the NP relation $R^{\text{ENC}} = \{ ((\mathsf{pk}, c), (x, s)) \mid c = \mathsf{Enc}(\mathsf{pk}, x ; s) \}$. While it is known how to construct NIZK argument systems for all of NP [92, 93], using this construction requires expensive NP reductions. In this section, we show how to construct an efficient gap Σ-protocol for R^{ENC} when the encryption scheme is the NTRU-based multikey FHE scheme from section 3.4. By Theorem 2.2 this suffices to construct an efficient NIZK argument system for R^{ENC} in the random oracle model. Our construction follows the ideas of Asharov and co-authors [7, 6].

Recall that in the aforementioned FHE scheme, a ciphertext has the form $c = [hs + 2e + m]_q$ for public key h, message $m \in \{0, 1\}$, and ring elements s, e, sampled from B-bounded distribution χ. We construct a gap Σ-protocol for proving that "c encrypts 0 under h." That is, we show a protocol for relation

$$R_0^{\text{ENC}} = \left\{ ((h, c), (s, e)) \mid c = [hs + 2e]_q \ \wedge \ \|s\|_\infty, \|e\|_\infty \leq B \right\}$$

with corresponding language L_0^{ENC}. By Theorem 2.1, we can then construct a gap Σ-protocol for R^{ENC} using an OR protocol to prove that "$c \in L_0^{\text{ENC}}$ or $c - 1 \in L_0^{\text{ENC}}$."

Gap Σ-protocol for encryptions of 0. Our construction of a gap Σ-protocol for R_0^{ENC} uses the same parameters as the encryption scheme: degree n, polynomial $\phi(x) =$

$x^n + 1$, modulus q, and distribution $\chi = \overline{D}_{\mathbb{Z}^n, r}$ over the ring $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$. It is additionally parametrized by a distribution $\widetilde{\chi} = \overline{D}_{\mathbb{Z}^n, \widetilde{r}}$ over R, such that $2^{\omega(\log \kappa)} r \leq \widetilde{r} \leq q/4\sqrt{n} - r$. To simplify notation, we recall from Lemma 2.7 that χ is B-bounded and $\widetilde{\chi}$ is $\widetilde{B}$-bounded for $B = r\sqrt{n}$ and $\widetilde{B} = \widetilde{r}\sqrt{n}$. By our choice of $\widetilde{r}$, this means that $\widetilde{B} + B \leq q/4$.

To formally describe our protocol, we must first define relations R_{zk} and R_{sound}. We set $B_{\mathsf{zk}} = R_0^{\mathrm{ENC}}$ and set B_{sound} to be essentially the same as R_0^{ENC}, differing only in the requirement set for $\|s\|_\infty$ and $\|e\|_\infty$:

$$R_{\mathsf{sound}} = \left\{ \ ((h,c) \ , \ (s,e) \ ) \ \mid \ c = [hs + 2e]_q \ \wedge \ \|s\|_\infty, \|e\|_\infty \leq 4\left(\widetilde{B} + B\right) \ \right\}.$$

Note that since $\widetilde{B} \geq B$, we have $R_{\mathsf{zk}} \subseteq R_{\mathsf{sound}}$. We can now describe our construction:

- $P_1((h,c),(s,e))$: Samples $\widetilde{s}, \widetilde{e} \leftarrow \widetilde{\chi}$ and outputs $a = [h\widetilde{s} + 2\widetilde{e}]_q$ and $st = (\widetilde{s}, s)$.
- $V_1((h,c))$: Outputs a random bit $b \leftarrow \{0,1\}$.
- $P_2(st, b)$: Parses $st = (\widetilde{s}, s)$ and outputs $z = [\widetilde{s} + bs]_q$.
- $V_2((h,c), a, b, z)$: Computes $\varepsilon = [(a + bc) - hz]_q$ and outputs 1 if and only if $\|z\|_\infty \leq \widetilde{B} + B$, $\|\varepsilon\|_\infty \leq 2(B + \widetilde{B})$, and ε is even.

THEOREM 4.3. *Let $R_{\mathsf{zk}}, R_{\mathsf{sound}}$ be the NP relations described above. The construction $\langle P, V \rangle$ with $P = (P_1, P_2)$ and $V = (V_1, V_2)$ is a gap Σ-protocol for $(R_{\mathsf{zk}}, R_{\mathsf{sound}})$.*

Proof. We show that the above construction satisfies the completeness, special soundness, and HVZK properties.

Completeness: Let $((h,c),(s,e)) \in L_{\mathsf{zk}}$, and let (a,b,z) be a transcript for protocol $\langle P, V \rangle$. Then

$$\varepsilon = [(a + bc) - hz]_q = [h\widetilde{s} + 2\widetilde{e} + bhs + 2be - h\widetilde{s} - hbs]_q$$
$$= [2(\widetilde{e} + be)]_q = 2(\widetilde{e} + be),$$

where the last inequality holds by the fact that $\widetilde{B} + B \leq q/4$. It is clear that ε is even and its coefficients are bounded by $2(\widetilde{B} + B)$. Furthermore, $z = \widetilde{s} + bs$, so $\|z\|_\infty \leq \widetilde{B} + B$, as required.

Special soundness: Let (h,c) be a public key and ciphertext pair, and let $(a,0,z_0)$ and $(a,1,z_1)$ be two accepting transcripts. The extractor Ext outputs (s^*, e^*), where $s^* = z_1 - z_0$ and $e^* = [c - hs^*]_q$.

We now argue that $((h,c),(s^*,e^*)) \in R_{\mathsf{sound}}$. By construction, we have that $c = [hs^* + 2e^*]_q$. It remains to show the bound on the size of the coefficients of s^* and e^*. Since $(a,0,z_0)$ and $(a,1,z_1)$ are accepting transcripts, we know that $\|z_0\|_\infty, \|z_1\|_\infty \leq \widetilde{B} + B$, so that $\|s^*\|_\infty \leq 2(\widetilde{B} + B)$.

We now bound e^*. Let $\varepsilon_0 = [a - hz_0]_q$ and $\varepsilon_1 = [(a+c) - hz_1]_q$. Since $(a,0,z_0)$ and $(a,1,z_1)$ are accepting transcripts, we know that $\|\varepsilon_0\|_\infty, \|\varepsilon_1\|_\infty \leq 2(\widetilde{B} + B)$ and both ε_0 and ε_1 are even. Furthermore, $\varepsilon_1 - \varepsilon_0 = [(a+c) - hz_1 - (a - hz_0)]_q = [c - h(z_1 - z_0)]_q = e^*$. This means that e^* is even since both ε_0 and ε_1 are even, and we also have that $\|e^*\|_\infty \leq \|\varepsilon_0\|_\infty + \|\varepsilon_1\|_\infty \leq 4(\widetilde{B} + B)$, as desired.

Honest-verifier zero-knowledge: Let $((h,c),(s,e)) \in L_{\mathsf{zk}}$ and let $b \in \{0,1\}$. The simulator Sim chooses $z', e' \leftarrow \widetilde{\chi}$, sets $a' = hz' + 2e' + bc$, and outputs (a', b, z'). We argue that the output of Sim is statistically close to the transcript (a, b, z) of an execution of the protocol $\langle P, V \rangle$. In a real transcript, we have $a = h\widetilde{s} + 2\widetilde{e}$ and $z = \widetilde{s} + \sigma s$. In the simulated transcript, we have $a' = h(z' + bs) + 2(e' + be)$.

If $b = 0$, then the distributions are *identical* because $\widetilde{s}, \widetilde{e}, z', e'$ are all sampled from the same distribution $\widetilde{\chi}$. On the other hand, if $b = 1$, then the distributions are *statistically close* by Corollary 2.9. $\square$

We remark that the gap between R_{zk} and R_{sound} is superpolynomial due to our use of Corollary 2.9. We leave a better analysis, potentially resulting in a small polynomial gap between R_{zk} and R_{sound}, as an open problem.

Consequences of having a gap. We have shown how to construct efficient NIZK arguments for the relation R^{ENC} for the NTRU-based multikey FHE scheme from section 3.4. However, there is a *gap* in the relations for which soundness and zero-knowledge hold: zero-knowledge holds for an honest prover with a statement in R_{zk}, but an honest verifier is only convinced that the statement is in $R_{\mathsf{sound}} \supseteq R_{\mathsf{zk}}$. We must show that this gap does not affect the correctness of our protocol. It suffices to prove that the scheme is fully homomorphic when the error in fresh ciphertexts is bounded by $B^* \overset{\mathsf{def}}{=} 4(\widetilde{B} + B)$.

Our analysis in section 3.4 does not immediately guarantee this, as it sets $B = \mathsf{poly}(n)$. Since we must have $n = \mathsf{poly}(\kappa)$ for efficiency of the scheme, this means $B = \mathsf{poly}(\kappa)$. However B^* is superpolynomial in κ. Nevertheless, we can easily modify our parameters and analysis to guarantee that the scheme remains fully homomorphic with ciphertext noise that is superpolynomial in κ.

The proof of Lemma 3.6 shows that the leveled homomorphic scheme $\mathcal{E}_{\mathsf{LH}}$ described in section 3.4.2 is multikey homomorphic for N keys and circuits of depth D as long as

$$(nB^*)^{2N+2} < \frac{2^{n^\varepsilon}}{2(8n(nB^*)^{2N+2})^D},$$

which yields the requirement $ND = O(n^\varepsilon/(\log n + \log B^*))$. We follow the proof of Theorem 3.9 and show that there exists a multikey fully homomorphic encryption scheme for $N = O(\sqrt{(n^\varepsilon/\log n(\log n + \log B^*))})$. If we set $\widetilde{B} = 2^{\log^2 \kappa} \cdot B$ for $B = \mathsf{poly}(n)$ and $n \geq \kappa$, this is guaranteed if $N = O(\sqrt{(n^\varepsilon/\log^3 n)})$ since

$$n^\varepsilon/(\log^3 n) = O(n^\varepsilon/(\log n \cdot (\log n + \log^2 \kappa))) = O(n^\varepsilon/(\log n \cdot (\log n + \log B^*))).$$

(In)security in the standard model. We have shown a NIZK argument for relation R^{ENC}. Though secure in the random oracle model, we remark that care must be taken if we want to hope for security in the standard model. More specifically, since our gap Σ-protocol has only constant soundness, we need to use parallel repetition for soundness amplification. For efficiency, we would like to repeat the protocol only $\mathsf{polylog}(\kappa)$ many times as this already achieves negligible soundness. However, Dachman, Soled, and co-authors [49, 22] have shown that if we use such a small number of repetitions, the resulting NIZK cannot be proved sound (in the standard model) via a black-box reduction to a (superpolynomially hard) falsifiable assumption. Also see the remarks after Theorem 2.2.

4.3. Impossibility of a 2-round protocol. We have shown that there exists an on-the-fly MPC protocol with a 5-round online phase. We now ask whether we can achieve the optimal solution of having a completely noninteractive online phase. In this section we answer this question negatively: we show that the existence of such a protocol (secure against semihonest adversaries)[16] implies general circuit obfuscation

---

[16] Considering semihonest adversaries instead of semimalicious or malicious adversaries only makes our result stronger.

as a virtual black-box with single-bit output, which we know to be impossible [10]. Our techniques are inspired by those of van Dijk and Juels [133].

We begin by reviewing the definition of general circuit obfuscation [10].

DEFINITION 4.1 (circuit obfuscation [10]). *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, $|\mathcal{O}(C)| \leq p(|C|)$.*

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

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

Barak et al. [10] show that assuming one-way functions exist, there does not exist any algorithm $\mathcal{O}$ satisfying Definition 4.1, even if we do not require that $\mathcal{O}$ run in polynomial time. Thus, our results imply that assuming one-way functions exist, there does not exist any on-the-fly MPC protocol with a noninteractive online phase.

We now show the connection between on-the-fly MPC and obfuscation. We consider an on-the-fly MPC protocol with a noninteractive online phase and assume that only one function is evaluated and the function is chosen a priori, before the start of the protocol (i.e., it does not depend on the offline stage messages). Let N be the number of inputs of the circuit; without loss of generality, we assume that the computing parties are $P_1, \ldots, P_N$. Note that considering such a restricted protocol only makes our impossibility result stronger. A protocol like this can be modeled by efficient and possibly randomized algorithms: $\mathsf{In}_1, \ldots, \mathsf{In}_U, \mathsf{Compute}, \mathsf{Out}_1, \ldots, \mathsf{Out}_N$, where the following hold:

- $(d_i, c_i) \leftarrow \mathsf{In}_i(x_i)$: 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.
- $(z_1, \ldots, z_N) \leftarrow \mathsf{Compute}(C, c_1, \ldots, c_N)$: On input a circuit C and $c_1, \ldots, c_N$, which are the messages the server received from parties $P_1, \ldots, P_N$, $\mathsf{Compute}$ outputs N elements $z_1, \ldots, z_N$. The server sends back z_i to party P_i.
- $y \leftarrow \mathsf{Out}_i(z_i, d_i)$: 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.

We know from the work of Halevi, Lindell, and Pinkas [95] that in the noninteractive 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. Thus we must relax the definition of security so that when the server is corrupted, the simulator is allowed to submit queries of the form $(S, \vec{x})$, where S is a nonempty subset of the honest parties and $\vec{x}$ is any input vector of size $n - |S|$. The trusted functionality evaluates the function on $\vec{x}$ and the honest inputs in S. Furthermore, our result holds even when the real-world adversary is only allowed to output 1 bit.[17]

THEOREM 4.4. *If there exists an on-the-fly MPC protocol with a noninteractive online phase that computes all efficiently computable functions with two inputs and is secure against semihonest adversaries (with the relaxed definition of security), then there exists a circuit obfuscator $\mathcal{O}$ satisfying Definition 4.1.*

---

[17]Considering a restricted class of adversaries for the on-the-fly MPC protocol only makes our impossibility result stronger.

Proof. We start by defining a family of "meta-circuits" $\{F^{(m)}\}_{m \in \mathbb{N}}$. For a fixed $m \in \mathbb{N}$, $F^{(m)}$ is such that given a circuit C of size m and bit-string x, it evaluates C on x and outputs $C(x)$, i.e., $F^{(m)}(C, x) = C(x)$. van Dijk and Juels [133] show how to construct a family of meta-circuits such that for all $m \in \mathbb{N}$, $|F^{(m)}| = O(m^2)$.

We now show how to construct a circuit obfuscator $\mathcal{O}$ using an on-the-fly MPC protocol $\Pi = (\mathsf{In}_1, \ldots, \mathsf{In}_U, \mathsf{Compute}, \mathsf{Out}_1, \mathsf{Out}_2)$ with the properties described in the theorem statement. Given a circuit C of size m, $\mathcal{O}$ computes $(\cdot, c_1) \leftarrow \mathsf{In}_1(C)$, samples random coins ρ, σ, τ, and outputs a circuit G that on input x:

- Computes $(c_2, d_2) := \mathsf{In}_2(x \; ; \rho)$.
- Computes $(\cdot, z_2) := \mathsf{Compute}(F^{(m)}, c_1, c_2 \; ; \sigma)$.
- Computes and outputs $y := \mathsf{Out}_2(z_2, d_2 \; ; \tau)$. $\qquad\qquad\square$

We now show that this obfuscator satisfies the functionality, polynomial slow-down, and virtual black-box properties from Definition 4.1.

Functionality: The correctness property of the on-the-fly MPC protocol guarantees that for all x, $G(x) = F^{(m)}(C, x) = C(x)$.

Polynomial slowdown: Using van Dijk and Juels's construction [133], we have that $|F^{(m)}| = O(m^2)$. Since all algorithms of the on-the-fly MPC protocol run in polynomial time, we have that there exists a polynomial p such that $|G| = p(|C|)$.

Virtual black-box: To prove the virtual black-box property, we observe that given an attacker $\mathcal{A}$ trying to break the obfuscation, we can construct a real-world semihonest adversary $\mathcal{B}$ attacking the on-the-fly MPC protocol, corrupting the server and party P_2. The honest party receives input C and $\mathcal{B}$ receives a dummy value $\widetilde{x}$ for P_2, which it ignores. Instead it receives c_1 from the honest party, builds G as specified, and runs $\mathcal{A}$ on G. When $\mathcal{A}$ outputs a bit b, $\mathcal{B}$ completes Steps 2 and 3 in the protocol as specified and outputs b. We emphasize that any action taken by $\mathcal{A}$ is valid for a semihonest adversary, so $\mathcal{B}$ is semihonest.

Security of Π implies that there exists simulator $\mathcal{S}$ such that for all inputs $C, \widetilde{x}$, we have $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}}(C, \widetilde{x}) \stackrel{c}{\approx} \mathrm{REAL}_{\Pi,\mathcal{B}}(C, \widetilde{x})$, where in the ideal world, $\mathcal{S}$ is given access to an oracle as described above. In the setting we are considering, the only valid subset that $\mathcal{S}$ can provide in a query to this oracle is $\{1\}$. Thus, $\mathcal{S}$ has oracle access to $F^{(m)}(C, \cdot) = C(\cdot)$. We can build a simulator $\mathcal{S}'$ with oracle access to $C(\cdot)$ that on input $|C|$,[18] chooses an arbitrary $\widetilde{x}$ and runs $\mathcal{S}(\widetilde{x})$ (which runs $\mathcal{B}$, which runs $\mathcal{A}$), anwers $\mathcal{S}$'s queries with its own oracle, and outputs $\mathcal{S}$'s output.

Since $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs and $\mathcal{S}'$ outputs whatever $\mathcal{S}$ outputs, the fact that $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}}(C, \widetilde{x}) \stackrel{c}{\approx} \mathrm{REAL}_{\Pi,\mathcal{B}}(C, \widetilde{x})$ implies that $\mathcal{S}'(|C|) \stackrel{c}{\approx} \mathcal{A}(G)$. The theorem statement follows.

REFERENCES

[1] W. AIELLO, S. N. BHATT, R. OSTROVSKY, AND S. RAJAGOPALAN, *Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP*, in Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP), 2000, Geneva, Switzerland, 2000, pp. 463–474.
[2] M. AJTAI AND C. DWORK, *A public-key cryptosystem with worst-case/average-case equivalence*, in Proceedings of STOC, 1997, pp. 284–293.

---

[18]In most applications it is okay to leak the size of the honest input. Indeed this is implied in most constructions, including our construction from section 4.1.

[3]   M. R. Albrecht, S. Bai, and L. Ducas, *A subfield lattice attack on overstretched NTRU assumptions—Cryptanalysis of some FHE and graded encoding schemes*, in Proceedings of CRYPTO, 2016, pp. 153–178.

[4]   B. Applebaum, D. Cash, C. Peikert, and A. Sahai, *Fast cryptographic primitives and circular-secure encryption based on hard learning problems*, in Proceedings of CRYPTO, S. Halevi, ed., Lecture Notes in Computer Sci. 5677, Springer, New York, 2009, pp. 595–618.

[5]   B. Applebaum, Y. Ishai, and E. Kushilevitz, *From secrecy to soundness: Efficient verification via secure computation*, in Proceedings of ICALP, S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, eds., Lecture Notes in Comput. Sci. 6198, Springer, New York, 2010, pp. 152–163.

[6]   G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, *Multiparty computation with low communication, computation and interaction via threshold FHE*, in Proceedings of EUROCRYPT, 2012, pp. 483–501.

[7]   G. Asharov, A. Jain, and D. Wichs, *Multiparty computation with low communication, computation and interaction via threshold FHE*, in Proceedings of EUROCRYPT, 2012, pp. 483–501.

[8]   M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk, *ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data*, in Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, 2015, IEEE Computer Society, pp. 271–286, https://doi.org/10.1109/SP.2015.24.

[9]   B. Barak, *How to go beyond the black-box simulation barrier*, in Proceedings of FOCS, IEEE Computer Society, 2001, pp. 106–115.

[10] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang, *On the (im)possibility of obfuscating programs*, in CRYPTO, 2001, pp. 1–18.

[11] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang, *On the (im)possibility of obfuscating programs*, J. ACM, 59 (2012), p. 6.

[12] B. Barak, Y. Lindell, and S. P. Vadhan, *Lower bounds for black-box zero knowledge*, J. Comput. System Sci., 72 (2006), pp. 321–391.

[13] M. Bellare and P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, in Proceedings of the ACM Conference on Computer and Communications Security, D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, eds., ACM, 1993, pp. 62–73.

[14] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)*, in Proceedings of STOC, 1988, pp. 1–10.

[15] E. Ben-Sasson, I. Bentov, A. Chiesa, A. Gabizon, D. Genkin, M. Hamilis, E. Pergament, M. Riabzev, M. Silberstein, E. Tromer, and M. Virza, *Computational integrity with a public random string from quasi-linear PCPs*, in Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques Advances in Cryptology, EUROCRYPT 2017, Paris, France, J. Coron and J. B. Nielsen, eds., Lecture Notes in Comput. Sci. 10212, 2017, pp. 551–579, https://doi.org/10.1007/978-3-319-56617-7_19.

[16] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, *Scalable zero knowledge via cycles of elliptic curves*, in Proceedings of the 34th Annual Cryptology Conference on Advances in Cryptology, CRYPTO 2014, Santa Barbara, CA, J. A. Garay and R. Gennaro, eds., Lecture Notes in Comput. Sci. 8617, Springer, New York, 2014, pp. 276–294, https://doi.org/10.1007/978-3-662-44381-1_16.

[17] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, *Succinct non-interactive zero knowledge for a von neumann architecture*, in Proceedings of USENIX, pp. 781–796; also available online from https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson.

[18] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias, *Semi-homomorphic encryption and multiparty computation*, in Proceedings of EUROCRYPT, 2011, pp. 169–188.

[19] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, *From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again*, in Proceedings of ITCS, 2012.

[20] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, *Recursive composition and bootstrapping for SNARKS and proof-carrying data*, in Proceedings of STOC, D. Boneh, T. Roughgarden, and J. Feigenbaum, eds., ACM, 2013, pp. 111–120.

[21] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth, *Succinct non-interactive arguments via linear interactive proofs*, in Proceedings of the 10th Theory of Cryptogra-

phy Conference of Cryptography, TCC 2013, Tokyo, Japan, 2013, A. Sahai, ed., Lecture Notes in Comput. Sci. 7785, Springer, New York, 2013, pp. 315–333, https://doi.org/10.1007/978-3-642-36594-2_18.

[22] N. Bitansky, D. Dachman-Soled, S. Garg, A. Jain, Y. T. Kalai, A. López-Alt, and D. Wichs, *Why "Fiat–Shamir for proofs" lacks a proof*, in Proceedings of TCC, 2013, pp. 182–201.

[23] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton, *Cryptographic primitives based on hard learning problems*, in Proceedings of CRYPTO, D. R. Stinson, ed., Lecture Notes in Comput. Sci. 773, Springer, New York, 1993, pp. 278–291.

[24] M. Blum, P. Feldman, and S. Micali, *Non-interactive zero-knowledge and its applications (extended abstract)*, in Proceedings of STOC, J. Simon, ed., ACM, 1988, pp. 103–112.

[25] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, *Improved security for a ring-based fully homomorphic encryption scheme*, in Proceedings of the IMA International Conference, M. Stam, ed., Lecture Notes in Comput. Sci. 8308, Springer, New York, 2013, pp. 45–64.

[26] Z. Brakerski, *Fully homomorphic encryption without modulus switching from classical GapSVP*, in Proceedings of CRYPTO, 2012, pp. 868–886.

[27] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *Fully homomorphic encryption without bootstrapping*, in Proceedings of ITCS, 2012.

[28] Z. Brakerski and R. Perlman, *Lattice-based fully dynamic multi-key FHE with short ciphertexts*, in Proceedings of CRYPTO, 2016, pp. 190–213.

[29] Z. Brakerski and V. Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, in Proceedings of FOCS, 2011, pp. 97–106.

[30] Z. Brakerski and V. Vaikuntanathan, *Fully homomorphic encryption from ring-LWE and security for key dependent messages*, in Proceedings of CRYPTO, 2011, pp. 505–524.

[31] Z. Brakerski and V. Vaikuntanathan, *Lattice-based FHE as secure as PKE*, in Proceedings of ITCS, M. Naor, ed., ACM, 2014, pp. 1–12.

[32] R. Canetti, U. Feige, O. Goldreich, and M. Naor, *Adaptively secure multi-party computation*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, 1996, pp. 639–648.

[33] R. Canetti, O. Goldreich, and S. Halevi, *The random oracle methodology, revisited*, J. ACM, 51 (2004), pp. 557–594.

[34] R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan, *Obfuscation of probabilistic circuits and applications*, in Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, Y. Dodis and J. B. Nielsen, eds., Warsaw, Poland, Lecture Notes in Comput. Sci. 9015, Springer, New York, 2015, pp. 468–497, http://doi.org/10.1007/978-3-662-46497-7_19.

[35] D. Chaum, C. Crépeau, and I. Damgård, *Multiparty unconditionally secure protocols (extended abstract)*, in Proceedings of STOC, 1988, pp. 11–19.

[36] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, *Batch fully homomorphic encryption over the integers*, in Proceedings of EUROCRYPT, 2013, pp. 315–335.

[37] J. H. Cheon, J. Jeong, and C. Lee, *An Algorithm for NTRU Problems and Cryptanalysis of the GGH Multilinear Map Without an Encoding of Zero*, IACR Cryptology ePrint Archive, 2016, 139, http://eprint.iacr.org/2016/139.

[38] J. H. Cheon and D. Stehlé, *Fully homomophic encryption over the integers revisited*, in Proceedings of EUROCRYPT, E. Oswald and M. Fischlin, eds., Lecture Notes in Comput. Sci. 9056, Springer, New York, 2015, pp. 513–536, https://doi.org/10.1007/978-3-662-46800-5_20.

[39] A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N. P. Smart, *Between a rock and a hard place: Interpolating between MPC and FHE*, in Proceedings of ASIACRYPT, K. Sako and P. Sarkar, eds., Lecture Notes in Comput. Sci. 8270, Springer, New York, 2013, pp. 221–240.

[40] K.-M. Chung, Y. T. Kalai, and S. P. Vadhan, *Improved delegation of computation using fully homomorphic encryption*, in Proceedings of CRYPTO, 2010, pp. 483–501.

[41] M. Clear and C. McGoldrick, *Multi-identity and multi-key leveled FHE from learning with errors*, in Proceedings of the 35th Annual Cryptology Conference on Advances in Cryptology, CRYPTO 2015, Santa Barbara, CA, R. Gennaro and M. Robshaw, eds., Lecture Notes in Comput. Sci. 9216, Springer, New York, 2015, pp. 630–656, http://doi.org/10.1007/978-3-662-48000-7_31.

[42] J.-S. Coron, T. Lepoint, and M. Tibouchi, *Scale-invariant fully homomorphic encryption over the integers*, in Proceedings of the International Workshop on Public Key Cryptog-

raphy, H. Krawczyk, ed., Lecture Notes in Comput. Sci. 8383, Springer, New York, 2014, pp. 311–328.

[43] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, *Fully homomorphic encryption over the integers with shorter public keys*, in Proceedings of CRYPTO, 2011, pp. 487–504.

[44] J.-S. Coron, D. Naccache, and M. Tibouchi, *Public key compression and modulus switching for fully homomorphic encryption over the integers*, in Proceedings of EUROCRYPT, 2012, pp. 446–464.

[45] R. Cramer, ed., *Theory of Cryptography*, in Proceedings of the 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, 2012, Lecture Notes in Comput. Sci. 7194, Springer, New York, 2012.

[46] R. Cramer, I. Damgård, and J. B. Nielsen, *Multiparty computation from threshold homomorphic encryption*, in Proceedings of EUROCRYPT, 2001, pp. 280–299.

[47] R. Cramer, I. Damgård, and B. Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, in Proceedings of CRYPTO, Y. Desmedt, ed., Lecture Notes in Comput. Sci. 839, Springer, New York, 1994, pp. 174–187.

[48] G. Di Crescenzo and H. Lipmaa, *Succinct NP proofs from an extractability assumption*, in Proceedings of CiE, A. Beckmann, C. Dimitracopoulos, and B. Löwe, eds., Lecture Notes in Comput. Sci. 5028, Springer, New York, 2008, pp. 175–185.

[49] D. Dachman-Soled, A. Jain, Y. T. Kalai, and A. López-Alt, *On the (In)security of the Fiat-Shamir Paradigm, Revisited*, IACR Cryptology ePrint Archive, 2012, 706.

[50] I. Damgård, Y. Ishai, and M. Krøigaard, *Perfectly secure multiparty computation and the computational overhead of cryptography*, in Proceedings of EUROCRYPT, 2010, pp. 445–465.

[51] I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. Smith, *Scalable multiparty computation with nearly optimal work and resilience*, in Proceedings of CRYPTO, D. Wagner, ed., Lecture Notes in Comput. Sci. 5157, Springer, New York, 2008, pp. 241–261.

[52] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, *Practical covertly secure MPC for dishonest majority—or: Breaking the SPDZ limits*, in Proceedings of ESORICS, J. Crampton, S. Jajodia, and K. Mayes, eds., Lecture Notes in Comput. Sci. 8134, Springer, New York, 2013, pp. 1–18.

[53] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, *Multiparty computation from somewhat homomorphic encryption*, in Proceedings of CRYPTO, 2012, pp. 643–662.

[54] G. Danezis, C. Fournet, J. Groth, and M. Kohlweiss, *Square span programs with applications to succinct NIZK arguments*, Part I, in Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security Advances in Cryptology, ASIACRYPT 2014, Kaoshiung, Taiwan, 2014, P. Sarkar and T. Iwata, eds., Lecture Notes in Comput. Sci. 8873, Springer, New York, 2014, pp. 532–550, https://doi.org/10.1007/978-3-662-45611-8_28.

[55] Y. Dodis, S. Halevi, R. D. Rothblum, and D. Wichs, *Spooky encryption and its applications*, in Proceedings of the 36th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO 2016, Santa Barbara, CA, M. Robshaw and J. Katz, eds., Lecture Notes in Comput. Sci. 9816, Springer, New York, 2016, pp. 93–122, http://doi.org/10.1007/978-3-662-53015-3_4.

[56] Y. Dodis, T. Ristenpart, and S. P. Vadhan, *Randomness condensers for efficiently samplable, seed-dependent sources*, in Proceedings of TCC, 2012, pp. 618–635.

[57] L. Ducas and A. Durmus, *Ring-LWE in polynomial rings*, in Proceedings of the International Workshop on Public Key Cryptography, 2012, pp. 34–51.

[58] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer, *Magic functions*, J. ACM, 50 (2003), pp. 852–921.

[59] P. Fauzi, H. Lipmaa, and B. Zhang, *Efficient modular NIZK arguments from shift and product*, in Proceedings of the 12th International Conference on Cryptology and Network Security, CANS 2013, Paraty, Brazil, 2013, M. Abdalla, C. Nita-Rotaru, and R. Dahab, eds., Lecture Notes in Comput. Sci. 8257, Springer, New York, 2013, pp. 92–121, https://doi.org/10.1007/978-3-319-02937-5_6.

[60] A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, in Proceedings of CRYPTO, A. M. Odlyzko, ed., Lecture Notes in Comput. Sci. 263, Springer, New York, 1986, pp. 186–194.

[61] M. Fischlin and J. Coron, eds., *Advances in Cryptology—EUROCRYPT Part* II, in Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, 2016, Lecture Notes in Comput. Sci. 9666, Springer, New York, 2016, https://doi.org/10.1007/978-3-662-49896-5.

[62] K. Fu AND J. Jung, EDS., *Proceedings of the* 23*rd USENIX Security Symposium*, San Diego, CA, 2014, USENIX, 2014; also available online from https://www.usenix.org/conference/usenixsecurity14.

[63] T. Elgamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, in Proceedings of CRYPTO, G. R. Blakley and D. Chaum, eds., Lecture Notes in Comput. Sci. 196, Springer, New York, 1984, pp. 10–18.

[64] S. Garg, C. Gentry, AND S. Halevi, *Candidate multilinear maps from ideal lattices*, in EUROCRYPT 2013, pp. 1–17.

[65] S. Garg, C. Gentry, S. Halevi, AND M. Raykova, *Two-round secure MPC from indistinguishability obfuscation*, in Proceedings of TCC, Y. Lindell, ed., Lecture Notes in Comput. Sci. 8349, Springer, New York, 2014, pp. 74–94.

[66] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, AND B. Waters, *Candidate indistinguishability obfuscation and functional encryption for all circuits*, in Proceedings of FOCS, IEEE Computer Society, 2013, pp. 40–49.

[67] S. Garg AND A. Polychroniadou, *Two-round adaptively secure MPC from indistinguishability obfuscation*, in Proceedings of TCC, 2015, pp. 614–637.

[68] R. Gennaro, C. Gentry, AND B. Parno, *Non-interactive verifiable computing: Outsourcing computation to untrusted workers*, in CRYPTO 2010, pp. 465–482.

[69] R. Gennaro, C. Gentry, B. Parno, AND M. Raykova, *Quadratic span programs and succinct NIZKs without PCPs*, in EUROCRYPT 2013, pp. 626–645.

[70] C. Gentry, *A Fully Homomorphic Encryption Scheme*, Ph.D. thesis, Stanford University, 2009; also available online from https://crypto.stanford.edu/craig.

[71] C. Gentry, *Fully homomorphic encryption using ideal lattices*, in Proceedings of STOC, M. Mitzenmacher, ed., ACM, 2009, pp. 169–178.

[72] C. Gentry AND S. Halevi, *Fully homomorphic encryption without squashing using depth*-3 *arithmetic circuits*, in FOCS 2011, pp. 107–109.

[73] C. Gentry AND S. Halevi, *Implementing gentry's fully-homomorphic encryption scheme*, in EUROCRYPT 2011, pp. 129–148.

[74] C. Gentry, S. Halevi, V. Lyubashevsky, C. Peikert, J. Silverman, AND N. Smart, *private communication*, 2011.

[75] C. Gentry, S. Halevi, C. Peikert, AND N. P. Smart, *Ring switching in BGV-style homomorphic encryption*, in Proceedings of SCN, I. Visconti and R. D. Prisco, eds., Lecture Notes in Comput. Sci. 7485, Springer, New York, 2012, pp. 19–37.

[76] C. Gentry, S. Halevi, AND N. P. Smart, *Better bootstrapping in fully homomorphic encryption*, in Proceedings of the International Workshop on Public Key Cryptography, M. Fischlin, J. Buchmann, and M. Manulis, eds., Lecture Notes in Comput. Sci. 7293, Springer, New York, 2012, pp. 1–16.

[77] C. Gentry, S. Halevi, AND N. P. Smart, *Fully homomorphic encryption with polylog overhead*, in Proceedings of EUROCRYPT, 2012, pp. 465–482.

[78] C. Gentry, S. Halevi, AND N. P. Smart, *Homomorphic evaluation of the AES circuit*, in Proceedings of CRYPTO, 2012, pp. 850–867.

[79] C. Gentry, A. Sahai, AND B. Waters, *Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based*, in Proceedings of CRYPTO (1), R. Canetti and J. A. Garay, eds., Lecture Notes in Comput. Sci. 8042, Springer, New York, 2013, pp. 75–92.

[80] C. Gentry AND D. Wichs, *Separating succinct non-interactive arguments from all falsifiable assumptions*, in Proceedings of STOC, L. Fortnow and S. P. Vadhan, eds., ACM, 2011, pp. 99–108.

[81] H. Gilbert, ED., in *Proceedings of the* 29*th Annual International Conference on the Theory and Applications of Cryptographic Techniques Advances in Cryptology—EUROCRYPT* 2010, French Riviera, 2010, Lecture Notes in Comput. Sci. 6110, Springer, New York, 2010.

[82] O. Goldreich, *Foundations of Cryptography: Volume* 2*, Basic Applications*, Cambridge University Press, New York, 2004.

[83] O. Goldreich, S. Goldwasser, AND S. Halevi, *Public-key cryptosystems from lattice reduction problems*, in Proceedings of CRYPTO, B. S. K., Jr., ed., Lecture Notes in Comput. Sci. 1294, Springer, New York, 1997, pp. 112–131.

[84] O. Goldreich, S. Micali, AND A. Wigderson, *How to play any mental game or a completeness theorem for protocols with honest majority*, in Proceedings of STOC, 1987, pp. 218–229.

[85] O. Goldreich AND Y. Oren, *Definitions and properties of zero-knowledge proof systems*, J. Cryptology, 7 (1994), pp. 1–32.

[86] S. Goldwasser and Y. T. Kalai, *On the (in)security of the fiat-shamir paradigm*, in Proceedings of FOCS, IEEE Computer Society, 2003, pp. 102–113.

[87] S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan, *Robustness of the learning with errors assumption*, in Proceedings of Innovations in Computer Science—ICS 2010, Tsinghua University, Beijing, China, 2010, A. C. Yao, ed., Tsinghua University Press, 2010, pp. 230–240; also available online from https://dspace.mit.edu/handle/1721.1/73191.

[88] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, *Delegating computation: Interactive proofs for muggles*, in Proceedings of STOC, C. Dwork, ed., ACM, 2008, pp. 113–122.

[89] S. Goldwasser, H. Lin, and A. Rubinstein, *Delegation of Computation Without Rejection Problem from Designated Verifier CS-proofs*, Cryptology ePrint Archive: Report 2011/456, 2011.

[90] J. Groth, *Short pairing-based non-interactive zero-knowledge arguments*, in Proceedings of ASIACRYPT, M. Abe, ed., Lecture Notes in Comput. Sci. 6477, Springer, New York, 2010, pp. 321–340.

[91] J. Groth, *On the size of pairing-based non-interactive arguments*, in EUROCRYPT, 2016, pp. 305–326, https://doi.org/10.1007/978-3-662-49896-5_11.

[92] J. Groth, R. Ostrovsky, and A. Sahai, *Perfect non-interactive zero knowledge for NP*, in Proceedings of EUROCRYPT, S. Vaudenay, ed., Lecture Notes in Comput. Sci. 4004, Springer, New York, 2006, pp. 339–358.

[93] J. Groth, R. Ostrovsky, and A. Sahai, *New techniques for noninteractive zero-knowledge*, J. ACM, 59 (2012), 11.

[94] S. Hada and T. Tanaka, *On the existence of 3-round zero-knowledge protocols*, in Proceedings of CRYPTO, H. Krawczyk, ed., Lecture Notes in Comput. Sci. 1462, Springer, New York, 1998, pp. 408–423.

[95] S. Halevi, Y. Lindell, and B. Pinkas, *Secure computation on the web: Computing without simultaneous interaction*, in CRYPTO, 2011, pp. 132–150.

[96] J. Hoffstein, J. Pipher, and J. H. Silverman, *NTRU: A ring-based public key cryptosystem*, in Proceedings of ANTS, J. Buhler, ed., Lecture Notes in Comput. Sci. 1423, Springer, New York, 1998, pp. 267–288.

[97] T. Johansson and P. Q. Nguyen, eds., *Advances in Cryptology—EUROCRYPT* 2013, in Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, 2013, Lecture Notes in Comput. Sci. 7881, Springer, New York, 2013.

[98] S. Kamara, P. Mohassel, and M. Raykova, *Outsourcing Multi-Party Computation*, Cryptology ePrint Archive: Report 2011/272, 2011.

[99] J. Kilian, *A note on efficient zero-knowledge proofs and arguments (extended abstract)*, in Proceedings of STOC, ACM, 1992, pp. 723–732.

[100] J. Kilian, *Improved efficient arguments (preliminary version)*, in Proceedings of CRYPTO, D. Coppersmith, ed., Lecture Notes in Comput. Sci. 963, Springer, New York, 1995, pp. 311–324.

[101] J. Kilian, ed., *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology—CRYPTO* 2001, Santa Barbara, CA, 2001, Lecture Notes in Comput. Sci. 23139, Springer, New York, 2001.

[102] A. E. Kosba, D. Papadopoulos, C. Papamanthou, M. F. Sayed, E. Shi, and N. Triandopoulos, *TRUESET: Faster verifiable set computations*, in Proceedings of USENIX 2014, pp. 765–780; also available online from https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/kosba.

[103] H. Lipmaa, *Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments*, in Proceedings of TCC, 2012, pp. 169–189.

[104] H. Lipmaa, *Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes*, in Proceedings of ASIACRYPT, K. Sako and P. Sarkar, eds., Lecture Notes in Comput. Sci. 8269, Springer, New York, 2013, pp. 41–60.

[105] H. Lipmaa, *Efficient NIZK arguments via parallel verification of benes networks*, in Proceedings of the 9th International Conference on Security and Cryptography for Networks, SCN 2014, Amalfi, Italy, 2014, M. Abdalla and R. D. Prisco, eds., Lecture Notes in Comput. Sci. 8642, Springer, New York, 2014, pp. 416–434, https://doi.org/10.1007/978-3-319-10879-7_24.

[106] A. López-Alt, E. Tromer, and V. Vaikuntanathan, *On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption*, in Proceedings of STOC, H. J. Karloff and T. Pitassi, eds., ACM, 2012, pp. 1219–1234.

[107] V. Lyubashevsky, C. Peikert, and O. Regev, *On ideal lattices and learning with errors over rings*, in Proceedings of EUROCRYPT, 2010, pp. 1–23.

[108] S. Micali, *CS proofs (extended abstracts)*, in Proceedings of FOCS, IEEE, 1994, pp. 436–453.

[109] D. Micciancio and O. Regev, *Worst-case to average-case reductions based on Gaussian measures*, SIAM J. Comput., 37 (2007), pp. 267–302.

[110] P. Mukherjee and D. Wichs, *Two round multiparty computation via multi-key FHE*, in Proceedings of EUROCRYPT, 2016, pp. 735–763, http://doi.org/10.1007/978-3-662-49896-5_26.

[111] S. Myers, M. Sergi, and A. Shelat, *Black-box proof of knowledge of plaintext and multiparty computation with low communication overhead*, in Proceedings of TCC, 2013, pp. 397–417.

[112] M. Naor, *On cryptographic assumptions and challenges*, in Proceedings of CRYPTO, D. Boneh, ed., Lecture Notes in Comput. Sci. 2729, Springer, New York, 2003, pp. 96–109.

[113] R. Ostrovsky, ed., *IEEE 52nd Annual Symposium on Foundations of Computer Science*, Proceedings of FOCS 2011, Palm Springs, CA, 2011, IEEE, 2011.

[114] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, in Proceedings of EUROCRYPT, J. Stern, ed., Lecture Notes in Comput. Sci. 1592, Springer, New York, 1999, pp. 223–238.

[115] B. Parno, J. Howell, C. Gentry, and M. Raykova, *Pinocchio: Nearly practical verifiable computation*, in Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society, 2013, pp. 238–252.

[116] K. G. Paterson, ed., *Advances in Cryptology—EUROCRYPT 2011*, in Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 2011, Lecture Notes in Comput. Sci. 6632, Springer, New York, 2011.

[117] C. Peikert and S. Shiehian, *Multi-key FHE from LWE, revisited*, in Proceedings of TCC, 2016, pp. 217–238.

[118] D. Pointcheval and T. Johansson, eds., *Advances in Cryptology—EUROCRYPT 2012*, in Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, 2012, Lecture Notes in Comput. Sci. 7237, Springer, New York, 2012.

[119] T. Rabin, ed., *Proceedings of the 30th Annual Cryptology Conference on Advances in Cryptology—CRYPTO 2010*, Santa Barbara, CA, 2010, Lecture Notes in Comput. Sci. 6223, Springer, New York, 2010.

[120] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, in Proceedings of STOC, H. N. Gabow and R. Fagin, eds., ACM, 2005, pp. 84–93.

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

[122] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, *Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds*, in Proceedings of the ACM Conference on Computer and Communications Security, 2009, pp. 199–212.

[123] R. L. Rivest, L. Adleman, and M. L. Dertouzos, *On data banks and privacy homomorphisms*, Foundations of Secure Computation, Academia Press, Ghent, Belgium, 1978, pp. 169–179.

[124] P. Rogaway, ed., *Proceedings of the 31st Annual Cryptology Conference on Advances in Cryptology—CRYPTO 2011*, Santa Barbara, CA, 2011, Lecture Notes in Comput. Sci. 6841, Springer, New York, 2011.

[125] R. Safavi-Naini and R. Canetti, eds., *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology—CRYPTO 2012*, Santa Barbara, CA, 2012, Lecture Notes in Comput. Sci. 7417, Springer, New York, 2012.

[126] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, *Robust non-interactive zero knowledge*, in CRYPTO, 2001, pp. 566–598.

[127] P. Scholl and N. P. Smart, *Improved key generation for Gentry's fully homomorphic encryption scheme*, in Proceedings of the IMA International Conference, L. Chen, ed., Lecture Notes in Comput. Sci. 7089, Springer, New York, 2011, pp. 10–22.

[128] N. P. Smart and F. Vercauteren, *Fully homomorphic encryption with relatively small key and ciphertext sizes*, in Proceedings of the Internatonal Workship on Public Key Cryptography, P. Q. Nguyen and D. Pointcheval, eds., Lecture Notes in Comput. Sci. 6056, Springer, New York, 2010, pp. 420–443.

[129] N. P. Smart and F. Vercauteren, *Fully homomorphic SIMD operations*, Des. Codes Cryptogr., 71 (2014), pp. 57–81.

[130] D. Stehlé and R. Steinfeld, *Making NTRU as secure as worst-case problems over ideal lattices*, in Proceedings of EURCRYPT, 2011, pp. 27–47.

[131] P. Valiant, *Incrementally verifiable computation or proofs of knowledge imply time/space efficiency*, in Proceedings of TCC, R. Canetti, ed., Lecture Notes in Comput. Sci. 4948, Springer, New York, 2008, pp. 1–18.

[132] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, *Fully homomorphic encryption over the integers*, in Proceedings of EUROCRYPT, 2010, pp. 24–43.

[133] M. van Dijk and A. 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, Berkeley, CA, 2010, USENIX, pp. 1–8; also available online from http://portal.acm.org/citation.cfm?id=1924931.1924934.

[134] R. S. Wahby, S. T. V. Setty, Z. Ren, A. J. Blumberg, and M. Walfish, *Efficient RAM and control flow in verifiable outsourced computation*, in Proceedings of the 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, CA, 2015, The Internet Society, 2015; also available online from http://internetsociety.ong/doc/efficient-ram-and-control-flow-verifiable-outsourced-computation.

[135] A. C.-C. Yao, *Protocols for secure computations (extended abstract)*, in Proceedings of FOCS, 1982, pp. 160–164.

[136] Y. Zhang, C. Papamanthou, and J. Katz, *ALITHEIA: Towards practical verifiable graph processing*, in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, 2014, G. Ahn, M. Yung, and N. Li, eds., ACM, 2014, pp. 856–867, http://doi.acm.org/10.1145/2660267.2660354.