Interaction-Preserving Compilers for Secure Computation

Nico Döttling[1], Vipul Goyal[2,3⋆], Giulio Malavolta[4], and Justin Raizes[2⋆]

[1] CISPA Helmholtz Center for Information Security
nico.doettling@gmail.com
[2] Carnegie Mellon University
vipul@cmu.edu, jraizes@andrew.cmu.edu
[3] NTT Research
[4] Max Planck Institute for Security and Privacy
giulio.malavolta@hotmail.it

Abstract. In this work we consider the following question: What is the cost of security for multi-party protocols? Specifically, given an *insecure* protocol where parties exchange (in the worst case) Γ bits in N rounds, is it possible to design a *secure* protocol with communication complexity close to Γ and N rounds? We systematically study this problem in a variety of settings and we propose solutions based on the intractability of different cryptographic problems.

For the case of two parties we design an interaction-preserving compiler where the number of bits exchanged in the secure protocol approaches Γ and the number of rounds is exactly N, assuming the hardness of standard problems over lattices. For the more general multi-party case, we obtain the same result assuming either (i) an additional round of interaction or (ii) the existence of extractable witness encryption and succinct non-interactive arguments of knowledge. As a contribution of independent interest, we construct the first multi-key fully homomorphic encryption scheme with message-to-ciphertext ratio (i.e., rate) of $1 - o(1)$, assuming the hardness of the learning with errors (LWE) problem.

We view our work as a support for the claim that, as far as interaction and communication are concerned, one does not need to pay a significant price for security in multi-party protocols.

1 Introduction

Secure multi-party computation (MPC) allows several mutually distrustful parties to compute a joint function on their inputs revealing nothing beyond the output of the computation. This ability to compute on private datasets enables applications of tremendous benefits to the society. General positive results for secure computation were obtained over three decades by Yao [46] in the two-party setting and then by Goldreich et al. [28] in the multi-party setting.

Since the initial feasibility results, a large body of literature has focused on improving the efficiency. In particular, much effort has been poured on optimizing the communication of secure MPC protocol, both in terms of the number of bits exchanged as well as the number of rounds. Breakthrough results on fully-homomorphic encryption allowed us to obtain secure computation protocols whose communication complexity could be below (or even independent of) the circuit size of the functionality [23]. For the important case of two parties, many recent works have studied "Alice optimized" and "Bob optimized" protocols, where the communication is independent of either Alice's input length and Bob's input length, respectively, while simultaneously optimizing the number of rounds (see, e.g., [43, 16]).

Given the current state of affairs, a natural question to ask is whether we can design secure computation protocols where the communication complexity could be lower than the size of *both* inputs (or *all* inputs in the multi-party setting). With the growing importance of being able to operate on big data, this question has become particularly compelling. Unfortunately, the answer to this question is, in general, negative: Known communication complexity lower bounds indicate that, for certain functions, communication proportional

⋆ Research supported by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

to at least one of the inputs may be necessary to compute it (even without any security requirements). On the other hand, there are many interesting examples of function classes which *can* be computed with communication complexity much lower than the size of either input. The rich line of research on communication complexity (c.f. [44]) has unveiled a large (and relevant) class of functions where non trivial communication savings can be made by cleverly designing the protocol. This raises the following question:

Given an N-round (insecure) protocol for some functionality F, can we design a secure protocol for F with (roughly) the same communication complexity, both in terms of bits exchanged and number of rounds?

Such a result would show that one does not need to pay a significant price for security (at least as far as interaction is concerned). Indeed there are a number of natural problems where the communication complexity of the insecure protocol could be significantly lower than the size of either input:

(1) Playing Private Combinatorial Games: Suppose two players wish to play chess over the internet and would like to determine who is the winner while hiding their individual strategies entirely. This in particular means that all their intermediate moves during the game must also be hidden from each other. This can be achieved using secure MPC where the input of each player is their strategy while the output is the identity of the winner. A generic MPC protocol for this task would require communication linear in the size of the strategy of at least one of the players. However if we observe that an insecure protocol for this task only requires the players to communicate their next move (thus resulting in communication independent of the strategy size), there is hope to compile this protocol in a secure one while preserving the communication complexity.

(2) Comparing Artificial Intelligences: Suppose two companies have developed independently algorithms to play some online game (e.g., Starcraft) and would like to determine which approach is superior. Since training such algorithms is typically expensive, they want to keep their strategy (and consequently their moves) secret. Using a general purpose MPC for this task would require one to communicate (at least one of) the entire algorithms, which is typically prohibitively large. On the other hand, online games are routinely played on personal computers with minimal communication overhead by leveraging interaction.

(3) Yao's Millionare Problem: Consider the classic Yao's millionaire problem where the task is to compute the "greater than" function on two inputs x and y (of n bit each). A randomized interactive protocol for this problem requires only $O(\log n)$ bits and $O(\log n)$ rounds of interaction. A conceptually simple protocol requiring $O(\log(n) \log\log(n))$ bits and $O(\log n)$ rounds can be found in [44]. The rough outline of the approach is to perform binary search for the position i such that x and y differ in the i-th bit but are identical in the first $i-1$.

Indeed this list is not exhaustive. Several other interesting examples include private property testing, Kachamar-Wigderson games, and so on.

1.1 Our Results

We systematically study the question of compiling insecure protocols to secure ones with minimal communication overhead in several settings and under a variety of cryptographic assumptions. We consider the strongest level of security where all but one parties are potentially corrupted. We assume that all communications among M parties happen through a broadcast channel. Specifically, given as input an N-round (where $N \geq 2$) *insecure* multi-party protocol $(\mathsf{Next}, \mathsf{Output})$ for a functionality F we obtain the following implication.

Theorem 1 (Informal). *If the circular LWE problem is hard, then there exists*

- *an N rounds (semi-honest) 2PC protocol*
- *an $N+1$ rounds (semi-honest) MPC protocol*

for F with communication complexity

$$\sum_{i=1}^{N} |m_i|(1+o(1)) + NM^2\mathsf{poly}(\lambda) + \mathsf{LFECC}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$$

in the common reference string model. Here:

- $|m_i|$ *is the maximum size of the i'th message in the insecure protocol over all inputs (*$|\mathsf{last}|$ *is $|m_{N-1}| + |m_N|$ for the 2PC protocol and $|m_N|$ for the MPC protocol)*
- $|\mathsf{depth}|$ *is the depth of the post-processing function* Output *(for the 2PC case, this also includes the depth of the computation of the last round message).*
- $|\mathsf{out}|$ *is the size of the output.*
- $\mathsf{LFECC}(s_{\mathsf{in}}, d, s_{\mathsf{out}}, \lambda)$ *is the communication complexity of a Laconic Function Evaluation scheme*[5] *for a circuit with input size s_{in}, depth d, and output size s_{out}, with security parameter λ.*

The communication complexity is roughly the worst case communication overhead of the insecure protocol plus an overhead which scales with the security parameter λ. Our results are particularly interesting in the case where the worst case communication complexity is large and the last message and output are small compared to the entire protocol. In this case, the overhead is overwhelmed by the insecure protocol's communication and the overall communication approaches the worst case communication complexity of the insecure protocol.

For the two-party case, this represents a near complete resolution of the problem, i.e., we show an N to N rounds compiler with communication complexity close to that of the insecure protocol.[6] Thus, the natural next question is whether the extra round of interaction is inherent in the multi-party settings. We show that this is in fact not the case and, under strong cryptographic assumptions (at the cost of a slightly higher communication), we obtain a round-preserving compiler.

Theorem 2 (Informal). *If the circular LWE problem is hard and assuming the existence of extractable witness encryption and succinct non-interactive arguments of knowledge, then there exists an N rounds (semi-honest) MPC protocol for F with communication complexity*

$$\sum_{i=1}^{N} |m_i|(1+o(1)) + NM^2\mathsf{poly}(\lambda) + \mathsf{LFECS}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$$
$$+ \mathsf{WEncCC}(\mathsf{LFECRS}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + \mathsf{poly}(\lambda) + |m_N|, |m_{N-1}| + M\mathsf{poly}(\lambda), |m_N| + M\mathsf{poly}(\lambda), \lambda)$$

where

- $\mathsf{WEncCC}(m, x, w, \lambda)$ *is the communication complexity of witness encryption of a message of size m under a statement of size x with witness size w using security parameter λ.*
- $\mathsf{LFECS}(s_{\mathsf{in}}, d, s_{\mathsf{out}}, \lambda)$ *is the circuit size of the ciphertext computation for a Laconic Function Evaluation scheme for a circuit with input size s_{in}, depth d, and output size s_{out}, with security parameter λ.*
- $\mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$ *the size of the common reference string for a Laconic Function Evaluation scheme for a circuit with input size s_{in}, depth d, and output size s_{out}, with security parameter λ*

We remark that extractable witness encryption [29] and succinct non-interactive arguments of knowledge [36] have been shown to be a very powerful machinery but we have evidence [20, 26] that they are

[5] See technical overview. Any improvements to the communication complexity of Laconic Function Evaluation schemes will directly improve our results. The scheme given by [43] has communication complexity $\tilde{O}(s_{\mathsf{in}} + s_{\mathsf{out}})\mathsf{poly}(d, \lambda)$.

[6] The dependency on the size of the output seems somewhat inherent to our approach, since it has been shown [33] that overcoming this barrier require some form of program obfuscation. Achieving this under LWE is a fascinating open question.

hard to instantiate from "standard" cryptographic assumptions. Nevertheless, candidate constructions exist [38, 21] and we can expect that our understanding will improve in the near future. A more pessimistic interpretation of our result is that it highlights a barrier against ruling out the existence of N to N rounds interaction-preserving compilers for MPC.

Finally, we discuss how to remove the need for a trusted setup (assuming that the rounds of the insecure protocol are at least $N \geq 3$) and how to lift all of our protocols to the malicious settings, without increasing the number of rounds.

Theorem 3 (Informal). *If the circular LWE problem is hard and assuming the existence of an MK-FHE scheme without setup, simulation-extractable succinct non-interactive arguments of knowledge, 2 round semi-malicious MPC, and 4 round (inefficient) delayed-input malicious MPC, then there exists a* $\max(4, N+1)$ *round malicious MPC protocol for computing* F *with communication complexity*

$$\sum_{i=1}^{N} |m_i|(1+o(1)) + (NM^2 + \mathsf{LFECC}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + \mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + |m_2|)\mathsf{poly}(\lambda)$$

Furthermore if extractable witness encryption exists, then there exists an $\max(4, N)$ *round malicious MPC protocol for* F *with communication complexity*

$$\sum_{i=1}^{N} |m_i|(1+o(1)) + (NM^2 + \mathsf{LFECS}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + \mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$$
$$+\mathsf{WEncCC}\left(\mathsf{LFECRS}\left(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda\right) + \mathsf{poly}(\lambda) + |m_N|, |m_{N-1}| + M\mathsf{poly}(\lambda), |m_N| + M\mathsf{poly}(\lambda), \lambda\right) + |m_2|)\mathsf{poly}(\lambda)$$

Rate-1 MK-FHE. One of our main technical tools that allows us to achieve these results is the first construction of a multi-key fully-homomorphic encryption (MK-FHE) [37] scheme with message-to-ciphertext ratio (i.e., the rate) that approaches 1 as the size of the message and the security parameter grow. The construction is proven secure against the standard LWE problem, plus an additional circularity assumption to apply the bootstrapping theorem [23].

Theorem 4 (Informal). *If the LWE problem is hard, then there exists a leveled MK-FHE scheme with ciphertext rate approaching* 1*, for a growing message size and security parameter. By making an additional circularity assumption, there exists a ciphertext rate-*1 *MK-FHE scheme.*

1.2 Related Works

To the best of our knowledge, the question of designing general compilers to go from insecure protocols to secure protocols while preserving the communication complexity was first studied by Naor and Nissim [41]. Naor and Nissim showed, given a protocol with communication complexity Γ, how to obtain a secure protocol with communication proportional to $\mathsf{poly}(\Gamma, \lambda)$. The number of rounds in the resulting protocol is $O(\Gamma)$ (even if the original protocol, say, required only a constant number of rounds). Ignoring the factor related to the security parameter, the communication complexity of the resulting protocol is at least Γ^2. Furthermore, the computational blowup of the secure protocol could be even exponential in Γ. Their approach is restricted to the two party settings.

A number of works have focused on optimizing the communication complexity [35, 15, 14] and the round complexity [18, 40, 22, 6] of MPC for generic functions. For all of these solutions, the communication of the secure protocol grows proportionally to the inputs of *all* parties (and sometimes even with the circuit representation of the functionality). A recent work [33] investigates the dependency of the communication complexity and the output size and gives positive results for the semi-honest settings and negative results for the (semi-)malicious settings. We also mention a related work of Boyle et al. [9] that compiles an insecure protocol to a secure one while preserving the communication as a function of M, where M is the number of parties. The overall communication (as a function of the input size) can increase significantly. Furthermore,

their approach adds an additional $2M$ rounds to the protocol in order to construct incremental FHE keys and decrypt the result. A closely related work [1] gives a method for compressing secure multiparty computation protocols into two round protocols (in the case of semi-honest adversaries) or three round protocols (in the case of malicious adversaries). The key difference between their approach and ours is that their approach compiles protocols which are already secure, whereas our approach compiles insecure protocols. As a result, their compiler inherits the overheads of existing secure computation protocols, which as mentioned previously, grows proportionally to the inputs of *all* parties. In contrast, our approach of compiling insecure protocols allows directly lifting work from the communication complexity literature into the secure computation setting. Additionally, their compiler incurs a $\mathsf{poly}(\lambda)$ blowup in the communication complexity of the compiled secure protocol and requires an honest majority.

In a different line of work, Fernando et al. [17] study the communication complexity of secure computation for massively parallel circuits. In contrast to our work, they do not focus on optimizing the communication rate (their compiler uses existing low-rate MK-FHE schemes) nor the round complexity (their compiler adds a constant number of rounds).

2 Technical Overview

In the following we give an informal overview of the techniques that we develop in order to achieve our results. Before delving into the details of our constructions we recall the useful notion of multi-key fully-homomorphic encryption (MK-FHE) developed in the context of round-optimized MPC protocols [37, 4, 40]. An MK-FHE scheme allow M distinct parties to locally sample a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and encrypt a message m_i under their public key pk_i. Then there exists a public evaluation algorithm that allows one to compute any function f over ciphertexts $(c_1, \ldots, c_M)$ encrypted under *independently sampled* keys. The resulting ciphertext $\tilde{c}$ encodes the function output $f(m_1, \ldots, m_M)$ and requires the knowledge of all secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_M)$ in order to be decrypted (this is inherent since it would otherwise violate semantic security). We say that an MK-FHE scheme has *threshold decryption* if the decryption of an evaluated ciphertext $\tilde{c}$ is split in the following two subroutines:

(1) A *local phase* where each party processes $\tilde{c}$ with its own secret key sk_i and produces a decryption share s_i which is then broadcast to all other participants.
(2) A *public phase* where the decryption shares $(s_1, \ldots, s_M)$ are publicly combined to reconstruct the message $f(m_1, \ldots, m_M)$.

Clearly the shares $(s_1, \ldots, s_M)$ should not reveal anything beyond allowing one to reconstruct the designated message.

A Dummy MPC Protocol. Equipped with an MK-FHE scheme, there is a natural way to compile and N-round *insecure* MPC protocol into a secure one. This dummy protocol is outlined in the following.

- **Pre-Processing:** Prior to the beginning of the execution of the protocol, each party P_i, on input x_i, samples an MK-FHE key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and computes an encryption of its input $c_i = \mathsf{Enc}(\mathsf{pk}_i, x_i)$.
- **Round 1 ... N:** Let Next_i be the next-message function of the i-th party P_i for the n-th round (for $n \in \{1, \ldots, N\}$) of the insecure protocol. P_i evaluates Next_i homomorphically over the ciphertexts exchanged so far and the input ciphertext c_i. Note that the resulting ciphertext potentially contains information which was originally encrypted under all public keys $(pk_1, \ldots, \mathsf{pk}_M)$. The resulting (multi-key) ciphertext is then broadcast to all parties.
- **Round N+1:** Let Output_i be the post-processing function of the i-th party P_i that takes as input the protocol transcript and the input x_i and returns the output of the protocol. P_i evaluates Output_i homomorphically over the ciphertexts exchanged so far and the input ciphertext c_i. At this point P_i holds an encrypted version $\tilde{c}_i$ of its output, which has to be decrypted with the help of all participants. $\tilde{c}_i$ is broadcast to all parties.
- **Round N+2:** Each party P_i receives a set of multi-key ciphertexts $(\tilde{c}_1, \ldots, \tilde{c}_M)$ and computes the partial decryption shares using its own secret key sk_i. Then P_i sends the share of the j-th ciphertext $s_{j,i}$ to P_j.

– **Post-Processing:** Each party P_i receives a set of shares $(s_{i,1}, \ldots, s_{i,M})$ which are locally reconstructed to recover the output of the protocol.

It is not hard to see that the resulting protocol is correct[7] and secure (in a semi-honest sense) as long as the MK-FHE scheme is semantically secure and admits an efficient threshold decryption procedure. Let Γ be the worst-case communication complexity of the insecure protocol, then the communication complexity of the resulting MPC is $\Gamma \cdot \mathsf{poly}(\lambda)$. We stress that the fact that the communication of the secure protocol grows with the *worst-case* complexity of the insecure one is always the case, at least for strict notions of security: The early termination of the protocol could leak some additional information about the parties' inputs, which is imperative to avoid.

Challenges. There are multiple reasons why this approach is unsatisfactory and falls short in answering our original question for an interaction-preserving compiler, which we elaborate below.

(1) The complexity of the secure MPC protocol is far from optimal as the compilation introduces a polynomial overhead in the security parameter λ. Note that this is not inherent for secure protocols: As an example, for the encryption functionality we can achieve almost optimal message-to-ciphertext size ratio (i.e., the rate of the encryption) by hybrid encryption. That is, an encryption for a message m is $\mathsf{Enc}(\mathsf{seed}), \mathsf{PRG}(\mathsf{seed}) \oplus m$, where PRG is a cryptographic pseudorandom generator. Here the size of the ciphertext approaches that of the message, for a growing $|m|$, since the size of the first component is fixed. Unfortunately this simple trick does not apply to the case of homomorphic encryption, where there does not seem to be any obvious way to convert evaluated ciphertexts back to this hybrid form. Thus, improving the rate of the above protocol requires one to answer the following question.

Can we construct a rate-1 multi-key fully-homomorphic encryption scheme?

(2) The most evident issue with the protocol described above is the fact that it requires 2 additional rounds of interaction. Intuitively, this is because at the end of the N-th round the parties only learn an *encrypted* version of the protocol's output. It is tempting to conclude that 2 more rounds are necessary to perform the joint decryption of the output ciphertexts, since generic MPC protocols require at least 2 rounds of interaction. However, recent developments in round-optimal MPC suggest that we can hope to do better. Overcoming this obstacle boils down to the following challenge.

Can we construct a round-preserving MPC protocol without any extra round of communication?

In this work we give positive answers to the questions above. Setting aside for the moment the issue of constructing a rate-1 MK-FHE scheme (which we are going to address at the end of this overview), the next paragraphs are dedicated to solving the challenge of the round complexity for interaction-preserving MPC compilers.

2.1 An N to N + 1 Rounds Compiler from Standard Assumptions

We first discuss how to reduce the round complexity of the protocol from $N + 2$ to $N + 1$, only assuming the hardness of the circular LWE problem. For the special case of two parties, the same protocol does not require *any* extra round of communication, i.e., we obtain an N to N rounds compiler.

N+1 Rounds via Laconic Function Evaluation. Before showing how to construct an N to $N+1$ round compiler, we first recall the notion of laconic function evaluation (LFE), a cryptographic primitive recently introduced by Quach et al. [43]. An LFE scheme allows a receiver to compress a large circuit $\mathcal{C}$ into a short hash h. Then the sender, on input x and the digest h, can compute an encryption $\mathsf{LFEEnc}(h, x)$ such that the receiver can recover $\mathcal{C}(x)$ and nothing more. In the instantiation proposed in [43], the size of the hash is a fixed polynomial $\mathsf{poly}(\lambda)$ and the size of the ciphertext depends only on $|x|$, on the size of the output, and on the depth of $\mathcal{C}$.

[7] As standard in MPC, correctness requires that the output of the secure evaluation of F on input $(x_1, \ldots, x_M)$ equals $F(x_1, \ldots, x_M)$. I.e., the secure protocol leaks exactly as much information as the insecure one.

To see why this machinery is useful to save rounds of communication, consider the special case of two parties, where only a single one receives the output: In the N-th round the receiver computes the LFE hash h of the circuit that hardwires all of the ciphertexts in its view (including the encryption of its own input) and takes as input the last ciphertext sent by the sender and its secret key. The circuit computes the post-processing function Output homomorphically over the ciphertexts and outputs the partial decryption of the resulting ciphertext. Then in the subsequent round the sender computes and sends $\mathsf{LFEEnc}(h, (\mathsf{sk}, c^{(N)})$, where sk is its MK-FHE secret key and $c^{(N)}$ is the ciphertext that the sender would have broadcast in the last round. With this information available, the receiver recovers the partial decryption of the encrypted output and completes the decryption locally.

Loosely speaking, this approach allows us to save one round of communication by outsourcing its computation to the LFE scheme. This intuition trivially extends to the multi-party case, where each party P_i acts as a sender and computes $\mathsf{LFEEnc}(h, (\mathsf{sk}_i, (c_1^{(N)}, \ldots, c_M^{(N)}))$, where $(c_1^{(N)}, \ldots, c_M^{(N)})$ are the ciphertexts broadcast in the last round. This allows the receiver to recover all of the decryption shares and thus the output of the computation. It is important to observe that this mechanism crucially exploits the fact that the MK-FHE threshold decryption consists of a local phase (which is computed under the hood of the LFE) and of a public reconstruction phase (which is executed in plain by the receiver). The price that we pay is that of an additive term proportional on the depth of the post-processing function Output of the insecure protocol. This is due to the specific instantiation proposed by Quach et al. [43] and it seems plausible that future LFE schemes (possibly from different assumptions) might surpass this barrier.

The Case of Two Parties. It turns out that, for the special case of two parties, the above approach can be slightly modified to yield a round-preserving (i.e., N to N) compiler. Without entering in the details of the transformation, the basic idea is to anticipate the computation of the LFE hash h to the round $N-1$ and complete the remainder of the computation under the hood of the LFE. The reason why this works for the two-party case (and it does not seem to extend to the more general multi-party case) is that the hashed circuit has hardcoded the complete view of the receiver and the sender can compute the LFE encryption containing its missing ciphertexts $(c^{(N-1)}, c^{(N)})$ already by the N-th round.

In contrast, in the multi-party case each sender would need to compute $\mathsf{LFEEnc}(h, (\mathsf{sk}_i, (c_1^{(N-1)}, \ldots, c_M^{(N-1)}, c_1^{(N)}, \ldots, c_M^{(N)}))$, which contains all ciphertexts broadcast in the last round. Therefore, the LFE ciphertext can only be computed *after* the N-th round is complete. At this point we seem to have encountered a roadblock: We cannot hope to transmit enough information to recompute the last round for all parties, as it would require sending an encryption of their entire input. Thus it is tempting to draw the conclusion that, for the multi-party case, one additional round of interaction is indeed necessary. In the following we show that this intuition is in fact wrong and that an N to N compiler exists under additional (strong) cryptographic assumptions.

2.2 An N to N Rounds Compiler

We now discuss a compiler which transforms an insecure N-round into a semi-honestly secure N-round protocol while preserving the communication complexity up to a $\mathsf{poly}(\lambda)$ factor. The starting point for this compiler is our previous N to $N+1$ rounds compiler discussed in the last section. Recall that in round $N+1$ the only action performed by each party is encrypting the MK-FHE cihertexts received in the last round in an LFE ciphertext addressed to each other party. In particular, the computation in round $N+1$ is succinct and in independent of the (possibly large) inputs x_i.

Flawed Attempts and Why They Fail. A first attempt could be to implement a similar strategy as in the two-party case and have the parties send the LFE hashes in round $N-1$ and LFE ciphertexts in round N. However, in the multiparty case this LFE ciphertext cannot depend on the ciphertexts sent by other parties which themselves might depend on the respective inputs in complex ways.

Our approach to shave off round $N+1$ is to *delegate* the computation of the LFE ciphertexts to the party receiving it. Say for concreteness we have parties P_1, P_2, P_3 and party P_1 wants to delegate computation of its LFE ciphertext to party P_2. Consider the following naive approach to achieve this: In round N party P_1 sends an obfuscated circuit $\tilde{\mathcal{C}}$ which computes the LFE ciphertext along with the other messages it sends

in round N. Since this computation does not depend on the input of P_1 this circuit is small. Now, once P_2 has received all MK-FHE ciphertexts in round N, it can evaluate $\tilde{\mathcal{C}}$ on these ciphertexts obtaining an LFE encryption. However, this approach introduces obvious problems: Since the LFE ciphertext produced by $\tilde{\mathcal{C}}$ also encrypts the secret key sk_1, the circuit $\tilde{\mathcal{C}}$ must know sk_1. But this means that a collusion consisting of P_2 and P_3 can now run $\tilde{\mathcal{C}}$ on many different ciphertexts. Even if P_2 was committed to a specific ciphertext as in the two party case, a semi-honest adversary with the view of P_2 and P_3 could still *replay* round N of P_3 in his head and modify the ciphertext sent by P_3. Such an adversary could clearly learn more than the output of the insecure protocol.

Preventing Replay Attacks via SNARKs. To overcome this issue, we need to make sure that there exists only a single valid input tuple on which P_2 can evaluate $\tilde{\mathcal{C}}$. To enforce this, we modify the protocol such that in round $N-1$ every party commits to their current state. Since the inputs are large, this needs to be done using a succinct commitment scheme (e.g. a Merkle tree). Party P_1 then hardwires these commitments into the circuit $\tilde{\mathcal{C}}$ and each input must be provided together with a proof that it has been correctly computed relative to the committed state and the ciphertexts sent by the other parties in round $N-1$. Since the witness for such a proof contains the input of the respective party, we have to use succinct arguments [38, 7, 8] to implement these proofs. Moreover, since the statements to be proven are only known in round N, we have to use succinct non-interactive arguments (SNARKs).

One issue with this approach is that an argument (as opposed to a proof) can only fix the correct inputs computationally and therefore we cannot rely on indistinguishability obfuscation [5, 19] to compute the obfuscated circuit $\tilde{\mathcal{C}}$. However, a closer look at our setting reveals that P_2 only needs to evaluate the circuit $\tilde{\mathcal{C}}$ once. Consequently, we can implement a *token-based obfuscation* [30] approach, which can be instantiated from witness-encryption and garbled circuits.

Our Solution. The final M-party protocol proceeds as follows: In round $N-1$ each party P_i computes a commitment ζ_i of an MK-FHE encryption $c_i^{(0)}$ of their (possibly large) input x_i as well as the MKE-FHE ciphertexts $(c_1^{(1)}, \ldots, c_M^{(N-2)})$. This commitment ζ_i is broadcast along with the round $N-1$ message $c_i^{(N-1)}$ of the underlying protocol. In round N, each party P_i computes a SNARK π_i which establishes that the N-th round message $c_i^{(N)}$ of the underlying protocol was computed consistently with the committed value. For each possible receiver P_j, the party P_i also computes a garbling of a circuit $\mathcal{C}$ which takes as input the N-th round messages $(c_1^{(N)}, \ldots, c_M^{(N)})$ as well as the LFE hashes $(h_1, \ldots, h_M)$ and computes and outputs the round $N+1$ message to P_j, i.e., an LFE encryption $\mathsf{LFEEnc}(h_j, (\mathsf{sk}_i, (c_1^{(N)}, \ldots, c_M^{(N)}))$.

To transmit the labels for this garbled circuit, we rely on witness encryption. For the sake of example, we consider a single k-bits input $c_1^{(N)}$ (the general case is handled analogously). For all $\kappa \in [k]$, our goal is to provide the label corresponding to the κ-th bit of $c_1^{(N)}$ to P_j, but keep the label corresponding to the complementary bit hidden. Define a language $\mathcal{L}$ such that a statement $(\zeta_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, b)$ is in $\mathcal{L}$, if there exist a $\tilde{c}_j^{(N)}$ and a SNARK proof π_j such that

(1) The κ-th bit of $\tilde{c}_j^{(N)}$ is b and
(2) π_j is a verifying SNARK proof which asserts that $\tilde{c}_j^{(N)}$ is the correct next message of P_j given the state committed to in ζ_j and $(c_j^{(N-1)}, \ldots, c_M^{(N-1)})$.

Assume that we are given a witness encryption for $\mathcal{L}$. For each wire-index κ, P_i encrypts the 0-label under the statement $(\zeta_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, 0)$ and the 1-label under the statement $(\zeta_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, 1)$. Finally, to complete round N, P_i sends $c_i^{(N)}$, the SNARK proof π_i, the garbled circuit $\tilde{\mathcal{C}}$, and the witness-encrypted labels to P_j.

To evaluate the garbled circuit $\tilde{\mathcal{C}}$, P_j first obtains the *label-encodings* of the inputs $(c_1^{(N)}, \ldots, c_1^{(N)})$ by decrypting the witness encryptions using the witnesses $\mathbf{w} = (c_1^{(N)}, \pi_1, \ldots, c_M^{(N)}, \pi_M)$. At this point P_j can evaluate the garbled circuit and obtain an LFE encryption $\mathsf{LFEEnc}(h_j, (\mathsf{sk}_i, (c_1^{(N)}, \ldots, c_M^{(N)}))$, where $(c_1^{(N)}, \ldots, c_M^{(N)})$. P_j can then proceed as in the $N+1$-round protocol.

On Extractability Assumptions. To prove semi-honest security, we need to argue that no collusion of parties is able to obtain garbled circuit-labels that do not corresponding to the legitimate inputs. Once this is established, security of the protocol follows routinely from the simulation-security of the garbling scheme. The standard notion of witness encryption [21] only provides security for messages encrypted under false statements. However, a closer look at the language $\mathcal{L}$ reveals that for given ζ_j, $c_j^{(N-1)}, \ldots, c_M^{(N-1)}$ and κ both the statements

$$\mathbf{x}_0 = \left(\zeta_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, 0\right) \text{ and } \mathbf{x}_1 = \left(\zeta_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, 1\right)$$

might be true. The reason for this is that the SNARK, which is used in the definition of $\mathcal{L}$, is computationally sound. Consequently, we need to rely on the stronger notion of *extractable witness encryption* [29]. This notion requires that any (adversarial) machine which can decrypt a witness-encryption ciphertext v (with non-negligible probability) *must know* a corresponding witness for the statement x under which v was encrypted. This is formalized via a non-black-box knowledge extractor. Thus, in the security proof we can argue that such a machine could produce SNARK proofs for the wellformedness of ciphertexts $\tilde{c}_j^{(N)} \neq c_j^{(N)}$, which, by the proof-of-knowledge property of the SNARK, would mean that it can consistently open the commitment ζ_j in different ways, which contradicts its binding property.

The combination of SNARKs and extractable witness encryption in our construction is inspired by the construction of attribute-based encryption for Turing-machines/RAM programs in [29].

2.3 Plain Model

The protocols presented so far require the sampling of a common reference string by a trusted party, which is then made available to all participants. This additional assumption is clearly undesirable and thus a natural question is whether compilers with similar efficiency exist also in the plain model. In the following we sketch how to lift the previously described protocols in the plain model *without adding any additional round of interaction*, at the cost of slightly increasing the communication complexity of the resulting MPC. In a nutshell, our idea is to begin the computation of the protocol under the hood of a MK-FHE without setup (but not necessarily with rate-1), such as the one described in [37, 3], and piggyback the round needed to generate the public parameters in the first round of interaction. Once this operation is completed, we can move the encrypted protocol execution under the rate-1 MK-FHE scheme via standard key-switching techniques.

More specifically, assume for the moment that one round of interaction is sufficient to generate the public parameters of the protocol. Then our augmented protocol proceeds as follows: The parties encrypt their input under the MK-FHE without setup and compute the first message of the protocol homomorphically. Then they broadcast the ciphertexts, together with the corresponding public keys and the information necessary to compute the public parameters of the system. In the second round, the parties proceed as before except that they additionally sample a key pair for the rate-1 MK-FHE (recall that at this point the public parameters are established) together with an encryption of the secret key of the MK-FHE without setup. Once these ciphertexts are sent around, the parties can evaluate the decryption circuit of the MK-FHE without setup *homomorphically*, thus switching to encryptions under the hood of the rate-1 MK-FHE. The remainder of the protocol proceeds as before.

The security of this protocol can be shown with the same strategy as before, except that we have to add an additional intermediate hybrid where we substitute the encryption of the secret key of the MK-FHE without setup with an encryption of a fixed string. Clearly the number of rounds needed for this protocol is identical, given that $N \geq 3$. However, the communication complexity now grows by an additional additive factor $\mathsf{poly}(\lambda, |\mathsf{msg}|)$ where msg is the communication complexity of the *second round* of the insecure protocol, due to the fact that we do not know of any rate-1 MK-FHE without setup (from standard assumptions). The reason why we do not have an additional factor proportional to the size of the first round is that we can assume without loss of generality that *non-evaluated* MK-FHE ciphertexts have rate-1: Simply modify the encryption algorithm to compute the hybrid encryption

$$(\mathsf{Enc}(\mathsf{seed}), \mathsf{PRG}(\mathsf{seed}) \oplus m)$$

where PRG is a pseudorandom generator and $\mathsf{seed} \leftarrow_{\$} \{0,1\}^{\lambda}$. Thus what is left to be shown is that the public parameters of the scheme can be computed in one round. Recall that the public parameters of the schemes consist of (i) the public parameters of the MK-FHE scheme pp, (ii) the key of the collision resistant hash K, (iii) the common reference string of the LFE crs, and the (iv) common reference string of the SNARK $\mathsf{crs}_{\mathsf{SNARK}}$. For the semi-honest case, we can simply let an arbitrary party sample the public parameters and broadcast them in the first round.

2.4 Malicious Security

A natural question that arises from the above protocols is whether we can obtain similar results in the malicious settings. First observe that all of our protocols (or minor modification thereof) satisfy the notion of semi-malicious security: In the semi-malicious security experiment, the distinguisher is still given honestly computed view but it is allowed to choose the random coins for the corrupted parties. Since the MK-FHE scheme(s) that we deploy satisfy perfect correctness, the evaluated ciphertexts are always well-formed as long as the keys and the fresh encryptions are in the support of the algorithms KeyGen and Enc, respectively. Furthermore, we show in Appendix 7.4 that our rate-1 MK-FHE scheme is semantically secure for all (possibly adversarial) choices of the public parameters corresponding to the corrupted parties. Finally recall that the LFE scheme described in [43] has been shown to satisfy semi-malicious security. The caveat here is that the analysis assumes that the crs is sampled by a trusted party, whereas in our protocol Π_{N+1} the crs is chosen by some arbitrary (and potentially corrupted) party in the first round. This issue can be easily resolved by using a generic 2-round semi-malicious MPC protocol (e.g., [22, 6]) to generate the common reference string crs.

It is well known that any semi-malicious protocol can be compiled into a simulation-secure one by using universally composable non-interactive zero-knowledge proofs [4]. To ensure succinctness, we need to rely on simulation-extractable zero knowledge SNARKs instead (e.g. [32]). However, in the malicious setting, generating the common reference string crs for both the SNARKs and the LFE seems to require first running a 4 round[8] maliciously-secure MPC . Since both Π_{N+1} and Π_N require crs in the N'th round, this would only result in compilers for $N \geq 5$.

Early CRS Usage To avoid the unsatisfactory warm-up period, observe that the crs can be generated and used as early as the third round, *as long as it is not used publicly*. Specifically, we can run the 2 round crs generation protocol underneath a MK-FHE scheme without setup, as well as anything which requires it (e.g. the LFE and SNARKs). Upon receiving a proof that the crs was generated semi-maliciously, the entire interaction involving the crs can be safely decrypted to finish the protocol. The extra round for decrypting after the proof is complete can be avoided by using a generic *inefficient* 4 round MPC implementing the multiparty Conditional Disclosure of Secrets (MCDS) functionality suggested by Choudhuri et. al. [12]. Upon input of a witness that the crs was generated semi-maliciously, it reveals the "outer" plain model MK-FHE secret keys. This provides the guarantee that if any party cheated during the crs generation, no party can see the portions of the protocol involving the crs. If the outer low-rate MK-FHE secret keys are revealed, all parties can safely decrypt the portions of the transcript involving the crs without further interaction.

Our Solution The protocol specifications are identical to the plain model semi-honest protocol (which takes N' rounds) except for the following modifications:

(1) In round $N' - 3$, the parties also sample fresh (low-rate) MK-FHE keys and broadcast the public keys. They use these "outer" keys to run an encrypted generic 2-round semi-malicious MPC protocol (using fresh randomness) in rounds $N' - 3$ and $N' - 2$ to generate the crs for the LFE and SNARK.
(2) In the last round of Π_N (last two rounds of Π_{N+1}), instead of sending the LFE hash (and ciphertext) in the clear, it is computed and broadcast underneath the outer keys. This gives it access to the encrypted crs.
(3) Rounds $N' - 1$ and N' are additionally augmented with encrypted zero knowledge SNARKs proving that the transcript so far has been honestly computed with respect to the same input and randomness.

[8] Or 3 rounds with non-black-box simulation.

These SNARKs are computed and broadcast underneath the outer keys, which gives them access to the encrypted crs. If at any point a party P_i receives a faulty SNARK, which is checkable underneath the outer MK-FHE, it implicitly aborts by sending encryptions of a fixed string instead of further messages (e.g. the LFE ciphertext) under the MK-FHE.

(4) During the last four rounds, the parties run a generic 4 round malicious-secure MPC protocol implementing the MCDS functionality. Upon input of the outer secret keys and the randomness used in the crs generation, the MCDS reveals the outer MK-FHE secret keys.

The modifications introduced above do not change the communication complexity from that of the semi-honest plain model protocol, since the crs is bounded by a fixed $\mathsf{poly}(\lambda)$, the generic MPCs only compute circuits involving the crs and plain model MK-FHE keys, the SNARKs are succinct, and the LFE already had size $\mathsf{poly}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$. The round complexity becomes $\mathsf{max}(4, N')$, where N' is the round complexity of the underlying plain model semi-honest secure protocol. The compiler can be applied to any insecure protocol with $N \geq 3$.

At a high level, the simulator extracts the adversary's outer keys and randomness used in generating the crs using the MCDS simulator. During this time, it replaces the encrypted second message of the crs generation protocol with an encryption of 0. It rewinds to round $N' - 2$ and re-simulates MCDS while forcing the crs using the extracted randomness (which was fixed in round $N' - 3$) and the 2 round semi-malicious simulator. It decrypts the SNARKs in round $N' - 1$ using the extracted outer keys then invokes the knowledge extractor to retrieve the adversary's $\Pi_{N'}$ input. Armed with $\Pi_{N'}$ input, it finishes the simulation of the plain model secure protocol underneath the outer MK-FHE keys. To avoid issues from playing protocols in parallel, we rely on the semantic security of the plain model MK-FHE to hide components until we are ready to simulate them, the 2 round nature of the crs generation MPC, the ability to locally compute intermediate messages of $\Pi_{N'}$ given an encryption of the input, and the non-interactive nature of the components used in $\Pi_{N'}$ to force the output.

2.5 Rate-1 Multi-Key Fully-Homomorphic Encryption

The missing piece to complete the picture is the description of a rate-1 MK-FHE. In the following we present a MK-FHE scheme with message-to-ciphertext ratio of $1 - o(1)$ which is proven secure against the standard learning with errors (LWE) problem.

Compressible MK-FHE. Our starting point is the recent works of Brakerski et al. [10] and of Gentry and Halevi [24] where they propose a general construction paradigm for rate-1 FHE. We recall the main ideas in the following. One of their main leverages is that many (low rate) FHE schemes from the literature have a very structured decryption algorithm. More concretely, the decryption circuit for a ciphertext $\mathbf{c}$ and a secret key $\mathbf{s}$ can be rewritten as a linear operation followed by a rounding. That is, for an LWE modulus q there exists a *linear* function (mod q) $L_{\omega,\mathbf{c}}$ such that

$$L_{\omega,\mathbf{c}}(\mathbf{s}) = \omega \cdot m + e$$

where q is the LWE modulus, ω is an arbitrary constant and e is a noise term such that $|e| < B$ for some fixed bound B. By choosing ω to be large enough, the message can be decoded with probability 1. This property is referred to as linear decrypt-and-multiply. The idea to increase the rate of the FHE scheme is to key-switch low rate FHE ciphertext into high rate ciphertexts. High rate encryption schemes exhibit only linear homomorphic properties, which are however sufficient due to the linearity of $L_{\omega,\mathbf{c}}$. Compression is achieved by carefully packing multiple bits into high rate ciphertexts and by setting the constant ω appropriately.

Our observation is that a similar transformation works also for the case of MK-FHE with linear decrypt-and-multiply. Specifically, given a multi-key ciphertext $\mathbf{c}$ and the concatenation of all of the corresponding secret keys $(\mathbf{s}_1, \ldots, \mathbf{s}_M)$, we require the existence of a linear function $L_{\omega,\mathbf{c}}$ such that

$$L_{\omega,\mathbf{c}}(\mathbf{s}_1, \ldots, \mathbf{s}_M) = \omega \cdot m + e$$

where ω and e are defined as before. Fortunately, one can show that existing MK-FHE schemes [13, 40] have precisely this structure. Clearly if we want the final scheme to support multi-key operations, then we also require the high rate scheme to be multi-key homomorphic. Thus, the only missing ingredient is a multi-key linearly-homomorphic encryption (MK-LHE) with high rate. The main technical contribution of this work is the construction of a such a scheme, assuming the hardness of the LWE problem.

Rate-1 MK-LHE from LWE. Our main observation is that Regev encryption [45] can be extended to (i) pack arbitrarily many messages into a single ciphertexts and (ii) support multi-key evaluations of linear functions. In the following we recall the packed version of the scheme and we show a multi-key evaluation algorithm for linear functions over $\mathbb{Z}_q^\eta$, where q is the LWE modulus and η is proportional to the rate of the scheme. The key generation algorithm samples a matrix $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{n \times m}$ uniformly at random. Then it chooses a secret key $\mathbf{S} \leftarrow_\$ \mathbb{Z}_q^{\eta \times n}$ uniformly at random and samples $\mathbf{E} \leftarrow_\$ \chi^{\eta \times m}$, where χ is a discrete Gaussian. The public key of the scheme consists of the matrices

$$\mathbf{A} \text{ and } \mathbf{B} = \mathbf{S} \cdot \mathbf{A} + \mathbf{E}.$$

To encrypt a column vector of messages $(m_1, \ldots, m_\eta)$, one samples a random vector $\mathbf{r} \leftarrow_\$ \{0,1\}^m$ and sets the ciphertext to

$$\mathbf{c}_1 = \mathbf{A} \cdot \mathbf{r} \text{ and } \mathbf{c}_2 = \mathbf{B} \cdot \mathbf{r} + \omega \cdot (m_1, \ldots, m_\eta)$$

where ω is a constant (which is typically set to $q/2$). Given the secret key $\mathbf{S}$, one can recover the encrypted plaintext by computing $\mathbf{c}_2 - \mathbf{S} \cdot \mathbf{c}_1 \pmod q$ and rounding to the nearest multiple of ω. The scheme can be shown to be semantically secure with a canonical reduction to the LWE problem and an invocation of the leftover hash lemma [34].

It is well known that the scheme is (bounded) additively homomorphic for ciphertexts encrypted under the same key. However, by slightly modifying the evaluation and the decryption procedure one can show that the same holds for ciphertexts encrypted under *independently sampled* keys. We exemplify this for the case of homomorphic addition of two ciphertexts: On input $(\mathbf{c}_1, \mathbf{c}_2)$ and $(\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2)$, the multi-key evaluation algorithm computes $(\mathbf{c}_1, \tilde{\mathbf{c}}_1, \mathbf{c}_2 + \tilde{\mathbf{c}}_2)$. Given both secret keys $\mathbf{S}$ and $\tilde{\mathbf{S}}$ one can recover the sum of the vectors by computing

$$\begin{aligned}
&\mathbf{c}_2 + \tilde{\mathbf{c}}_2 - \mathbf{S} \cdot \mathbf{c}_1 - \tilde{\mathbf{S}} \cdot \tilde{\mathbf{c}}_1 \\
&= \mathbf{B} \cdot \mathbf{r} + \omega \cdot (m_1, \ldots, m_\eta) + \tilde{\mathbf{B}} \cdot \tilde{\mathbf{r}} + \omega \cdot (\tilde{m}_1, \ldots, \tilde{m}_\eta) - \mathbf{S} \cdot \mathbf{A} \cdot \mathbf{r} - \tilde{\mathbf{S}} \cdot \tilde{\mathbf{A}} \cdot \tilde{\mathbf{r}} \\
&= \mathbf{E} \cdot \mathbf{r} + \tilde{\mathbf{E}} \cdot \tilde{\mathbf{r}} + \omega \cdot (m + \tilde{m}_1, \ldots, m + \tilde{m}_\eta) \\
&= \mathbf{z} + \omega \cdot (m + \tilde{m}_1, \ldots, m + \tilde{m}_\eta)
\end{aligned}$$

which can be efficiently decoded as long as $\|\mathbf{z}\|_\infty$ is small enough. One limitation of the scheme is that messages cannot be multiplied by large constants since it would also be absorbed by the noise term and would violate correctness. This shortcoming can be easily bypassed by encrypting multiple copies of the each message multiplied by increasing powers of 2. By summing the terms corresponding to the bit representation of the constant, we obtain the same result while keeping the noise growth contained.

It is not immediately clear that the resulting multi-key ciphertexts have a high rate, since the evaluated ciphertexts now contain as many $\mathbb{Z}_q^n$ elements as the number of public keys. Fortunately, we can increase the parameter η (i.e., the dimensions of the encrypted messages) to be large enough to amortize for the additional overhead. Achieving actual rate-1 requires a little more work, but it can be done almost generically using the ciphertext compression algorithm developed by Brakerski et al. [10].

Threshold Decryption. The missing ingredient to use such a scheme in our MPC protocol is to argue that it satisfies threshold decryption. While the resulting MK-FHE falls short in achieving this, such issue can be easily resolved by another application of key-switching, i.e., homomorphically evaluate the compressed decryption algorithm under an MK-FHE with threshold decryption (but low rate). More details can be found in Appendix 7.4.

Removing the Common Reference String. While we deliberately glossed over the instantiations of our building blocks in our overview, one important aspect of our scheme is that it requires a common

reference string. This is because the schemes from [13, 40] require a trusted setup, which is inherited by our construction.[9] We will sketch a way to bypass the need for a trusted setup in our MPC protocols, without adding any extra round of communication but at the cost of a slight increase in the communication complexity of the protocol. To this end, we recall that (low-rate) MK-FHE schemes in the plain model exist [37] from NTRU-type assumptions. Given that N is large enough, we can afford to compute a few rounds of the protocol in under the hood of a plain model MK-FHE and then key-switch into a rate-1 MK-FHE with threshold decryption. These initial rounds can now be used by the participants to jointly compute the common reference string for our MK-FHE, without any extra interaction.

More precisely, we begin the execution of the dummy protocol as described above using a plain-model MK-FHE scheme (with low rate). In parallel with the first rounds of the protocol, the parties also engage in a generic MPC to compute the reference string crs of an MK-FHE with threshold decryption. For an appropriate instantiation of MK-FHE with threshold decryption (e.g., the one that we present in Section 7) the size of the crs is bounded by a fixed polynomial in the security parameter and therefore this trick adds only an additive overhead $\mathsf{poly}(\lambda)$ to the protocol communication. In N is large enough, then no additional rounds are needed. In fact, we will show in Appendix 7.4 the common reference string of our MK-FHE scheme can be computed in a *single* broadcast round. Once the crs is established, the parties sample a MK-FHE key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ and publish $\mathsf{Enc}(\mathsf{pk}_i, \overline{\mathsf{sk}}_i)$, where $\overline{\mathsf{sk}}_i$ is the secret key of the plain model MK-FHE scheme. At this point all parties can non-interactively convert all of the previously exchanged ciphertexts in multi-key ciphertext under $(\mathsf{pk}_1, \ldots, \mathsf{pk}_M)$. The rest of the execution is unchanged.

3 Preliminaries

We denote by $\lambda \in \mathbb{N}$ the security parameter. We we say that a function $\mathsf{negl}(\cdot)$ is negligible if it vanishes faster than any polynomial. Given a set S, we denote by $s \leftarrow\!\!\$\, S$ the uniform sampling from S. We say that an algorithm is PPT if it can be implemented by a probabilistic Turing machine running in time $\mathsf{poly}(\lambda)$. We abbreviate the set $\{1, \ldots, n\}$ as $[n]$. Matrices are denoted by $\mathbf{M}$ and vectors are denoted by $\mathbf{v}$. We denote the infinity norm of a vector by $\|\mathbf{v}\|_\infty$.

3.1 Learning with Errors

The (decisional) learning with errors (LWE) problem was introduced by Regev [45]. The LWE problem is parametrized by a modulus q, positive integers n, m and an error distribution χ. An adversary is either given $(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e})$ or $(\mathbf{A}, \mathbf{u})$ and has to decide which is the case. Here, $\mathbf{A}$ is chosen uniformly from $\mathbb{Z}_q^{n \times m}$, $\mathbf{s}$ is chosen uniformly from $\mathbb{Z}_q^n$, $\mathbf{u}$ is chosen uniformly from $\mathbb{Z}_q^m$ and $\mathbf{e}$ is chosen from χ^m. The matrix version of this problem asks to distinguish $(\mathbf{A}, \mathbf{S} \cdot \mathbf{A} + \mathbf{E})$ from $(\mathbf{A}, \mathbf{U})$, where the dimensions are accordingly. It follows from a simple hybrid argument that the matrix version is as hard as the standard version.

As shown in [45, 42], for *any* sufficiently large modulus q the LWE problem where χ is a discrete Gaussian distribution with parameter $\sigma = \alpha q \geq 2\sqrt{n}$ (i.e. the distribution over $\mathbb{Z}$ where the probability of x is proportional to $e^{-\pi(|x|/\sigma)^2}$), is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\alpha)$ in *worst case* dimension n lattices. We refer to $\alpha = \sigma/q$ as the *modulus-to-noise* ratio, and by the above this quantity controls the hardness of the LWE instantiation. Hereby, LWE with polynomial α is (presumably) harder than LWE with super-polynomial or sub-exponential α. We can truncate the discrete gaussian distribution χ to $\sigma \cdot \omega(\sqrt{\log(\lambda)})$ while only introducing a negligible error. Consequently, we omit the actual distribution χ but only use the fact that it can be bounded by a (small) value B.

3.2 Multi-Party Computation Protocols

In the following we recall the definition of multi-party computation (MPC) protocols. Our definitions are taken almost in verbatim from [27].

[9] We are not aware of any other approach to build rate-1 MK-FHE in the plain model.

Definition 1 (Functionality). *Let M be a positive integer. An M-party functionality f with input is a deterministic function from $\{0,1\}^{k\cdot M}$ to $\{0,1\}^{\ell\cdot M}$, for all $(k,\ell)\in\mathbb{N}$.*

We denote by $f_i(x)$ the i-th block of the output of f on input some $x\in\{0,1\}^{k\cdot M}$. We assume that all communications among parties happen through a broadcast channel.

Definition 2 (MPC Protocol). *Let M be a positive integer, let $N=N(\lambda)$ be a polynomial in the security parameter, and let f be an M-party functionality. An N-round protocol Π for f is a tuple of deterministic polynomial-time algorithms* $(\mathsf{Next},\mathsf{Output})$ *defined as follows.*

$\mathsf{Next}_i\left(1^\lambda,x_i,r_i,\mathbb{T}^{(n)}\right)$: *On input the security parameter 1^λ, an input $x_i\in\{0,1\}^k$, a random tape $r_i\in\{0,1\}^R$, and the transcript of all messages $\mathbb{T}^{(n)}$ up to the n-th round, return the $n+1$ round message $y_i^{(n+1)}$ for the party P_i.*

$\mathsf{Output}_i\left(1^\lambda,x_i,r_i,\mathbb{T}^{(N)}\right)$: *On input the security parameter 1^λ, an input $x_i\in\{0,1\}^k$, a random tape $r_i\in\{0,1\}^R$, and the transcript of all messages $\mathbb{T}^{(N)}$, return the output z_i for the party P_i.*

The MPC protocol must satisfy correctness in the following sense.

Definition 3 (Correctness). *For all integers $\lambda\in\mathbb{N}$, all inputs $(x_1,\ldots,x_M)\in\{0,1\}^{k\cdot M}$, and all random tapes $(r_1,\ldots,r_M)\in\{0,1\}^{R\cdot M}$ it holds that*

$$\left\{\mathsf{Output}_i\left(1^\lambda,x_i,r_i,\mathbb{T}^{(N)}\right)\right\}_{i\in[M]}=f(x_1,\ldots,x_M)$$

where $\mathbb{T}^{(N)}=\left\{\mathsf{Next}_i\left(1^\lambda,x_i,r_i,\mathbb{T}^{(n)}\right)\right\}_{i\in[M],n\in[N]}$.

Let $I\subseteq[M]$ be a subset of parties. We denote the view of parties $\{P_i\}_{i\in I}$ by

$$\mathsf{View}_I\left(1^\lambda,(x_1,\ldots,x_M),(r_1,\ldots,r_M)\right)=\left(\{x_i,r_i\}_{i\in I},\mathbb{T}^{(N)}\right).$$

Similarly, we define their output by

$$\mathsf{Output}_I\left(1^\lambda,(x_1,\ldots,x_M),(r_1,\ldots,r_M)\right)=\left\{\mathsf{Output}_i\left(1^\lambda,x_i,r_i,\mathbb{T}^{(N)}\right)\right\}_{i\in I}.$$

We are now in the position to define the notion of semi-honest security for MPC protocols. Semi-honest MPC is known to be realizable in 2-rounds [22, 6] from the minimal assumption of the existence of a 2-round oblivious transfer.

Definition 4 (Semi-Honest Security). *Let M be a positive integer, let f be an M-party functionality, and let Π be an MPC protocol for f. Then Π is secure against semi-honest adversaries if there exists a PPT algorithm* Sim *such that for the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{View}_I\left(1^\lambda,(x_1,\ldots,x_M),(r_1,\ldots,r_M)\right),\mathsf{Output}_I\left(1^\lambda,(x_1,\ldots,x_M),(r_1,\ldots,r_M)\right)\right\}_{\lambda\in\mathbb{N},I\subseteq[M],(x_1,\ldots,x_M)\in\{0,1\}^{k\cdot M}}$$
$$\overset{c}{\approx}\left\{\mathsf{Sim}(1^\lambda,I,\{x_i,f_i(x_1,\ldots,x_M)\}_{i\in I}),\{f_i(x_1,\ldots,x_M)\}_{i\in I}\right\}_{\lambda\in\mathbb{N},I\subseteq[M],(x_1,\ldots,x_M)\in\{0,1\}^{k\cdot M}}$$

where the probabilities are taken the random choice of $(r_1,\ldots,r_M)\leftarrow_{\$}\{0,1\}^{R\cdot M}$ and the random coins of the simulator Sim.

We say that a protocol is *semi-maliciously* secure if the distinguisher is allowed to (adaptively) choose the random coins of the corrupted parties but otherwise does not deviate from the specifications of the protocol. We refer the reader to [4] for a formal definition. For the settings of fully corrupted parties we consider the standard notion of simulation security for MPC [27].

3.3 Multi-Key Fully-Homomorphic Encryption

A multi-key homomorphic encryption supports the evaluation of functions over ciphertexts computed under different (possibly independently sampled) keys. The result of the computation can then be decrypted using all of the corresponding secret keys. We include a setup procedure in the syntax although it may not be necessary for certain instantiations.

Definition 5 (Multi-Key Homomorphic Encryption). *A multi-key homomorphic encryption scheme consists of the following efficient algorithms.*

$\mathsf{Setup}(1^\lambda)$: *On input the security parameter* 1^λ*, the setup algorithm returns the public parameters* pp.
$\mathsf{KeyGen}(\mathsf{pp})$: *On input the public parameters* pp*, the key generation algorithm returns a key pair* $(\mathsf{sk}, \mathsf{pk})$.
$\mathsf{Enc}(\mathsf{pk}, m)$: *On input a public key* pk *and a message* m*, the encryption algorithm returns a ciphertext* c.
$\mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), f, (c_1, \ldots, c_M))$: *On input a vector of public keys* $(\mathsf{pk}_1, \ldots, \mathsf{pk}_M)$*, an* M*-argument function* f*, and a vector of ciphertexts* $(c_1, \ldots, c_M)$*, the evaluation algorithm returns an evaluated ciphertext* c.
$\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_M), c)$: *On input a vector of secret keys* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_M)$ *and a ciphertext* c*, the decryption algorithm returns a message* m.

We say that the scheme is *fully* homomorphic (MK-FHE) if it is homomorphic for P/poly. The definition of correctness is given in the following. Note that we require the scheme to satisfy the strong notion of multi-hop correctness as defined in [25].

Definition 6 (Multi-Hop Correctness). *A multi-key homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is multi-hop correct if for all* $\lambda \in \mathbb{N}$*, all polynomials* $M = M(\lambda)$*, all* M*-argument functions* f *in the supported family, all inputs* $(m_1, \ldots, m_M)$*, all* pp *in the support of* $\mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *in the support of* $\mathsf{KeyGen}(\mathsf{pp})$*, and all ciphertexts* $(c_1, \ldots, c_M)$ *in the support of the encryptions of* $(m_1, \ldots, m_M)$*, it holds that*

$$\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_M), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), f, (c_1, \ldots, c_M))) = f(m_1, \ldots, m_M)$$

where c *is in the support of the encryptions of* m *if*

$$c = \mathsf{Enc}(\mathsf{pk}, m) \text{ or } c = \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \tilde{f}, (\tilde{c}_1, \ldots, \tilde{c}_M))$$

where $(\tilde{c}_1, \ldots, \tilde{c}_M)$ *are in the support of the encryptions of* $(\tilde{m}_1, \ldots, \tilde{m}_M)$ *and* $\tilde{f}(\tilde{m}_1, \ldots, \tilde{m}_M) = m$.

We say that a scheme is *compact* if the size of the evaluated ciphertext does not depend on the circuit size of the evaluated function f. The notion of security is standard for public-key encryption [31].

Definition 7 (Semantic Security). *A multi-key homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is semantically secure if for all* $\lambda \in \mathbb{N}$ *and for all pairs of messages* (m_0, m_1) *the following distributions are computationally indistinguishable:*

$$(\mathsf{pp}, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_0)) \overset{c}{\approx} (\mathsf{pp}, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_1))$$

where $\mathsf{pp} \leftarrow\!\!\$\, \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\, \mathsf{KeyGen}(\mathsf{pp})$.

Finally we define the rate of an encryption scheme as the asymptotic message-to-ciphertext size ratio.

Definition 8 (Rate). *We say that a homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *has rate* $\rho = \rho(\lambda)$*, if for all* $\lambda \in \mathbb{N}$*, all polynomials* $M = M(\lambda)$*, all* pp *in the support of* $\mathsf{Setup}(1^\lambda)$*, all* pk *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, all supported functions* f *with sufficiently large output size, all messages* $(m_1, \ldots, m_M)$ *in the message space, and all* $(c_1, \ldots, c_M)$ *in the support of the encryptions of* $(m_1, \ldots, m_M)$*, it holds that*

$$\frac{|f(m_1, \ldots, m_M)|}{|\mathsf{Eval}(\mathsf{pk}, f, (c_1, \ldots, c_M))|} \geq \rho.$$

We also say that a scheme has rate 1, if it holds that

$$\liminf_{\lambda \to \infty} \rho(\lambda) = 1.$$

Threshold Decryption We define a special syntax for the decryption of MK-FHE schemes that was first introduced in [40]. At a high level, this protocol consists of (1) a local phase where each party computes a partial decryption of a ciphertext given their secret key and (2) a public phase where all the partial decryption shares are recombined to reconstruct the plaintext.

Definition 9 (Threshold Decryption). *We say that a homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *supports* threshold decryption *if there exist two algorithms* PDec *and* Rec *with the following syntax.*

$\mathsf{PDec}(\mathsf{sk}, c, i)$: *The partial decryption algorithm takes as input a secret key* sk*, a ciphertext* c*, and an index* i*, and returns a decryption share* s*.*

$\mathsf{Rec}(s_1, \ldots, s_M)$: *The reconstruction algorithm takes as input a set of decryption shares* $(s_1, \ldots, s_M)$ *and returns a message* m*.*

The definition of correctness follows (with minor adjustments) from Definition 6. Next we define the notion of simulatability for decryption shares. Note that the definition of simulatability is given for the case of all but one corrupted parties, as in [40].

Definition 10 (Simulatability). *A multi-key homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{PDec}, \mathsf{Rec})$ *satisfies simulatability if there exists a PPT simulator* Sim *such that for all* $\lambda \in \mathbb{N}$*, all polynomials* $M = M(\lambda)$*, all indices* $i \in [M]$*, all* pp *in the support of* $\mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *in the support of* $\mathsf{KeyGen}(\mathsf{pp})$*, all messages* m *in the message domain, and all ciphertexts* c *in the support of the encryption of* m*, it holds that the following distributions are statistically close*

$$\mathsf{PDec}(\mathsf{sk}_i, c, i) \overset{s}{\approx} \mathsf{Sim}\left(m, c, i, \{\mathsf{sk}_j\}_{j \in [M] \setminus \{i\}}\right)$$

where the probability is taken over the random coins of the PDec *algorithm and of the simulator* Sim*.*

Ciphertext Compression for Multi-Key FHE It is useful to augment the syntax of a MK-FHE with a compression algorithm, which operates on standard ciphertexts and transforms them into compressed ciphertexts. To amend this, we also introduce an additional decryption algorithm that allows one to recover the plaintext from compressed ciphertexts, given the secret keys. We recall the definition from [10].

Definition 11 (Compressible MK-FHE). *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be an MK-FHE scheme and let* $\eta = \eta(\lambda)$ *be a polynomial. We say that the MK-FHE scheme supports* η-ciphertext compression *if there exist two algorithms* $\mathsf{Compress}$ *and* $\mathsf{CompDec}$ *with the following syntax.*

$\mathsf{Compress}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), (c_1, \ldots, c_\eta))$: *Takes as input a set of public keys* $(\mathsf{pk}_1, \ldots, \mathsf{pk}_M)$ *and* η *ciphertexts* $(c_1, \ldots, c_\eta)$ *and outputs a compressed ciphertext* c^**.*

$\mathsf{CompDec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_M), c^*)$: *Takes as input a set of secret keys* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_M)$ *and a compressed ciphertext* c^* *and outputs* η *messages* $(m_1, \ldots, m_\eta)$*.*

In terms of correctness we require that for all $(\mathsf{sk}_i, \mathsf{pk}_i)$ in the support of $\mathsf{KeyGen}(1^\lambda)$ and for all ciphertexts $(c_1, \ldots, c_\eta)$ in the support of the encryptions of $(m_1, \ldots, m_\eta)$ it holds that

$$\mathsf{CompDec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_M), \mathsf{Compress}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), (c_1, \ldots, c_\eta))) = (m_1, \ldots, m_\eta).$$

The definition of rate for compressed ciphertexts is analogous to that for standard MK-FHE scheme.

3.4 Laconic Function Evaluation

We recall the definition of laconic function evaluation (LFE), a primitive recently introduced by Quach, Wichs, and Wee [43]. The scheme is defined with respect to a class of circuits parametrized by the input and the circuit size, which we denote by Φ.

Definition 12 (Laconic Function Evaluation). *A laconic function evaluation scheme* $(\mathsf{LFEGen}, \mathsf{LFEHash}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is defined as the following tuple of algorithms.*

$\mathsf{LFEGen}(1^\lambda)$ *On input the security parameter* 1^λ*, the generation algorithm returns a common reference string* crs.

$\mathsf{LFEHash}(\mathsf{crs}, \mathcal{C})$: *On input the common reference string* crs *and a circuit* $\mathcal{C}$*, the compression algorithm returns a digest* h *and a decoding information* d.

$\mathsf{LFEEnc}(\mathsf{crs}, h, x)$: *On input the common reference string* crs*, a digest* h*, and a message* x*, the encryption algorithm returns a ciphertext* c.

$\mathsf{LFEDec}(\mathsf{crs}, c, d)$: *On input the common reference string* crs*, a ciphertext* c*, and a decoding information* d*, the decoding algorithm returns a message* y.

The definition of correctness is given in the following.

Definition 13 (Correctness). *A laconic function evaluation* $(\mathsf{LFEGen}, \mathsf{LFEHash}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is correct if for all* $\lambda \in \mathbb{N}$*, for all circuits* $\mathcal{C} \in \Phi$*, and all messages* x *it holds that*

$$\mathcal{C}(x) = \mathsf{LFEDec}(\mathsf{crs}, c, r)$$

where $\mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda)$*,* $(h, d) \leftarrow \mathsf{LFEHash}(\mathsf{crs}, \mathcal{C})$*, and* $c \leftarrow \mathsf{LFEEnc}(\mathsf{crs}, h, x)$

It is required that the encryption of a message x with respect to a compressed circuit $\mathcal{C}$ reveals nothing beyond $\mathcal{C}(x)$.

Definition 14 (Simulation Security). *A laconic function evaluation* $(\mathsf{LFEGen}, \mathsf{LFEHash}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is simulation secure if for all* $\lambda \in \mathbb{N}$ *there exists a PPT simulator* LFESim *such that for all admissible PPT attackers* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left| \Pr\left[1 = \mathcal{A}_2(c, \mathsf{state}) \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda) \\ (x, \mathcal{C}, s, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{crs}) \\ (h, d) \leftarrow \mathsf{LFEHash}(\mathsf{crs}, \mathcal{C}; s) \\ c \leftarrow \mathsf{LFEEnc}(\mathsf{crs}, h, x) \end{array} \right] - \Pr\left[1 = \mathcal{A}_2(c, \mathsf{state}) \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda) \\ (x, \mathcal{C}, s, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{crs}) \\ (h, d) \leftarrow \mathsf{LFEHash}(\mathsf{crs}, \mathcal{C}; s) \\ c \leftarrow \mathsf{LFESim}(\mathsf{crs}, h, \mathcal{C}, \mathcal{C}(x)) \end{array} \right] \right| = \mathsf{negl}(\lambda)$$

where the attacker is admissible if $\mathcal{C} \in \Phi$ *and the probability is taken over the random coins of* LFEGen*,* $\mathcal{A}_1$*,* $\mathsf{LFEHash}$*,* LFEEnc *and* LFESim.

Function hiding states that the hash hides the circuit to all computationally bounded attackers.

Definition 15 (Function-Hiding). *A laconic function evaluation* $(\mathsf{LFEGen}, \mathsf{LFEHash}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *is function-hiding if for all* $\lambda \in \mathbb{N}$ *there exists a PPT simulator* $\mathsf{LFESimFH}$ *such that for all admissible PPT attackers* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(h, \mathsf{state}) \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda) \\ (\mathcal{C}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{crs}) \\ (h, d) \leftarrow \mathsf{LFEHash}(\mathsf{crs}, \mathcal{C}) \end{array} \right] - \Pr\left[1 \leftarrow \mathcal{A}_2(h, \mathsf{state}) \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda) \\ (\mathcal{C}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{crs}) \\ h \leftarrow \mathsf{LFESimFH}(\mathsf{crs}, \Phi) \end{array} \right] \right| = \mathsf{negl}(\lambda)$$

where the attacker is admissible if $\mathcal{C} \in \Phi$ *the probability is taken over the random coins of* LFEGen*,* $\mathcal{A}_1$*, and* $\mathsf{LFESimFH}$.

3.5 Extractable Witness Encryption

We will now define extractable witness encryption [29, 20].

Definition 16 (Extractable Witness Encryption). *A witness encryption scheme for an NP-language* $\mathcal{L}$ *(with a corresponding witness relation* $\mathcal{R}$*) consists of a pair of PPT-algorithms* $(\mathsf{WEEnc}, \mathsf{WEDec})$ *with the following syntax.*

$\mathsf{WEEnc}(1^\lambda, \mathbf{x}, m)$ *: Takes as input a security parameter* 1^λ*, a statement* $\mathbf{x}$ *and a message* m *and outputs a ciphertext* v*. We will typically omit the input* 1^λ*.*

$\mathsf{WEDec}(v, \mathbf{x}, \mathbf{w})$ *: Takes as input a ciphertext* v*, a statement* $\mathbf{x}$ *and a witness* $\mathbf{w}$ *and outputs a message* m

We require the following properties:

- ***Correctness:*** *It holds for any message* m *and all* $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$ *that*

$$\mathsf{WEDec}(\mathsf{WEEnc}(\mathbf{x}, m), \mathbf{x}, \mathbf{w}) = m.$$

- ***Extractable Security:*** *For any PPT-sampler* $(\mathbf{x}, \mathbf{aux}, m_0, m_1) \leftarrow \mathsf{Samp}(1^\lambda)$ *and any PPT-distinguisher and any non-negligible* ϵ *there exists a PPT-extractor* Ext *and a non-negligible* ϵ' *such that if*

$$|\Pr[\mathcal{D}(\mathsf{WEEnc}(\mathbf{x}, m_0), \mathbf{x}, \mathbf{aux}) = 1] - \Pr[\mathcal{D}(\mathsf{WEEnc}(\mathbf{x}, m_1), \mathbf{x}, \mathbf{aux}) = 1]| > \epsilon,$$

then

$$\Pr[\mathsf{Ext}(\mathbf{x}, \mathbf{aux} = \mathbf{w} \text{ } s.t. \text{ } (\mathbf{x}, \mathbf{w}) \in \mathcal{R})] > \epsilon',$$

where $(\mathbf{x}, \mathbf{aux}, m_0, m_1) \leftarrow \mathsf{Samp}(1^\lambda)$*.*

We use a definition of witness encryption where multi-bit messages are encrypted, whereas [29, 20] use a definition in which the plaintext is just a single bit. A multi-bit scheme scheme can be constructed from a single-bit scheme via bitwise encryption, and security follows routinely via a hybrid argument.

3.6 Succinct Non-Interactive Arguments of Knowledge

We define succinct non-interactive arguments of knowledge (SNARKs) [36, 38].

Definition 17 (Succinct Non-Interactive Arguments of Knowledge). *A succinct non-interactive argument of knowledge* $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ *for an NP-language* $\mathcal{L}$ *with relation* $\mathcal{R}$ *is defined as the following tuple of algorithms.*

$\mathsf{SNARKGen}(1^\lambda)$: *On input the security parameter* 1^λ *the setup algorithm returns a common reference string* crs*.*

$\mathsf{SNARKProve}(\mathsf{crs}, x, w)$: *On input the common reference string* crs*, a statement* x *and a witness* w*, the proving algorithm returns a proof* π*.*

$\mathsf{SNARKVerify}(\mathsf{crs}, x, \pi)$: *On input the common reference string* crs*, a statement* x*, and a proof* π*, the verifier algorithm returns a bit* $\{0, 1\}$*.*

When necessary, we may specify the random coins used in $\mathsf{SNARKProve}$ to be r by the notation $\mathsf{SNARKProve}(\mathsf{crs}, x, w, r)$.

We say that an argument of knowledge is *succinct* if the proof size $|\pi|$ is independent of the size of the witness $|w|$ or the statement $|x|$. Correctness is defined in a standard way.

Definition 18 (Correctness). *A succinct non-interactive argument of knowledge* $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ *is correct if for all* $\lambda \in \mathbb{N}$ *and for all pairs* $(x, w) \in \mathcal{R}$ *it holds that*

$$1 = \mathsf{SNARKVerify}(\mathsf{crs}, x, \pi)$$

where $\mathsf{crs} \leftarrow \mathsf{SNARKGen}(1^\lambda)$ *and* $\pi \leftarrow \mathsf{SNARKProve}(\mathsf{crs}, x, w)$*.*

We say that a SNARK is an argument of knowledge if computing valid proofs require the knowledge of the witness. In the following we define the adaptive notion where the adversary is allowed to choose the statement adaptively after seeing the common reference string.

Definition 19 (Argument of Knowledge). *A succinct non-interactive argument of knowledge* $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ *is an argument of knowledge if there exists a PPT machine* Ext *and a polynomial* poly *such that for all* $\lambda \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[1 = \mathcal{R}(x,w) \wedge 1 = \mathsf{SNARKVerify}(\mathsf{crs}, x, \pi) \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{SNARKGen}(1^\lambda) \\ (\pi, x) \leftarrow \mathcal{A}(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}(\mathsf{crs}, \pi, x) \end{array}\right] = 1/\mathsf{poly}(\lambda)$$

and the probability is taken over the random coins of $\mathsf{SNARKGen}$ *and* $\mathcal{A}$*.*

We define the classical notion of simulation-based zero-knowledge.

Definition 20 (Zero-Knowledge). *A succinct non-interactive argument of knowledge* $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ *is zero-knowledge if there exists a PPT simulator* $(\mathsf{SNARKGenSim}, \mathsf{SNARKSim})$ *such that for all* $\lambda \in \mathbb{N}$*, all* $(x,w) \in \mathcal{R}$*, and all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}, \pi) \middle| \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{SNARKGen}(1^\lambda) \\ \pi \leftarrow \mathsf{SNARKProve}(\mathsf{crs}, x, w) \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}, \pi) \middle| \begin{array}{l} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SNARKGenSim}(1^\lambda) \\ \pi \leftarrow \mathsf{SNARKSim}(\mathsf{crs}, \mathsf{td}, x) \end{array}\right]\right| = \mathsf{negl}(\lambda)$$

where the probabilities are taken over the random coins of $\mathsf{SNARKGen}$*,* $\mathsf{SNARKProve}$*,* $\mathsf{SNARKGenSim}$*,* $\mathsf{SNARKSim}$*, and* $\mathcal{A}$*.*

Simulation extractability requires that the argument of knowledge property holds even when the adversary may receive simulated proofs beforehand for arbitrary statements. We follow the definition for simulation-extractable NIZK given by [32].

Definition 21 (Simulation-Extractability). *A succinct non-interactive argument of knowledge* $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ *for an NP relation* R *is simulation extractable if there exists a PPT simulator* $(\mathsf{SNARKGenSim}, \mathsf{SNARKSim})$ *and a PPT machine* Ext *such that for all* $\lambda \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$ *the probability of the adversary winning the following game is* $\mathsf{negl}(\lambda)$*:*

$\underline{\mathsf{MAIN}G(\lambda)}$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SNARKGenSim}(\lambda)$
$Q = \emptyset$
$(x, \pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}_{\mathsf{crs},\mathsf{td},x_i}}(\mathsf{crs})$
$w \leftarrow \mathsf{Ext}(\mathsf{crs}, \pi, x)$
Return $(x,\pi) \notin Q \wedge (x,w) \notin R \wedge \mathsf{SNARKVerify}(\mathsf{crs}, x, \pi) = 1$

$\underline{\mathsf{SimProve}_{\mathsf{crs},\mathsf{td},x_i}}$
$\pi_i \leftarrow \mathsf{SNARKSim}(\mathsf{crs}, \mathsf{td}, x_i)$
$Q = Q \cup \{(x_i, \pi_i)\}$
Return π_i

3.7 Collision Resistant Hash

We recall the standard definition of collision resistant hash functions.

Definition 22 (Hash Function). *A hash function is a tuple of PPT algorithms* $(\mathsf{HashGen}, \mathsf{Hash})$ *defined as follows:*

$\mathsf{HashGen}(1^\lambda)$: *The key generation algorithm takes as input the security parameter* 1^λ *returns a key* K*.*
$\mathsf{Hash}(\mathsf{K}, m)$: *The hashing algorithm takes as input a key* K *and a message* m *and returns a digest* h*.*

Collision resistance implies that it is hard to find two messages hashing to the same digest.

Definition 23 (Collision Resistance). *A hash function* $(\mathsf{HashGen}, \mathsf{Hash})$ *is collision resistant if for all* $\lambda \in \mathbb{N}$ *and for all PPT attacker* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\mathsf{Hash}(\mathsf{K}, m) = \mathsf{Hash}(\mathsf{K}, \tilde{m}) \wedge m \neq \tilde{m} \middle| \begin{array}{l} \mathsf{K} \leftarrow \mathsf{HashGen}(1^\lambda) \\ (m, \tilde{m}) \leftarrow \mathcal{A}(\mathsf{K}) \end{array}\right] = \mathsf{negl}(\lambda)\,.$$

3.8 Garbled Circuits

We recall the definition of Yao's garbled circuits [46]. A garbling scheme allows one to obfuscate a circuit and provide enough information to learn a single output and nothing more. Garbled circuits can be constructed assuming the existence of one-way functions.

Definition 24 (Garbled Circuits). *A garbling scheme is a tuple of PPT algorithms* $(\mathsf{Garble}, \mathsf{GCEval})$ *defined as follows:*

$\mathsf{Garble}(1^\lambda, \mathcal{C})$: *The garbling algorithm takes as input the security parameter* 1^λ *and the description of a circuit* $\mathcal{C}$ *and returns a garbled circuit* $\tilde{\mathcal{C}}$ *and a set of labels* $\{\mathsf{lbs}_i^{(0)}, \mathsf{lbs}_i^{(1)}\}_{i=1}^n$ *for each input wire of the circuit.*

$\mathsf{GCEval}(\tilde{\mathcal{C}}, \{\mathsf{lbs}_i^{(x_i)}\}_{i=1}^n)$: *The evaluation algorithm takes as input a garbled circuit* $\tilde{\mathcal{C}}$ *and one label per input wire* $\mathsf{lbs}_i^{(x_i)}$ *and returns an output* y*.*

The definition of correctness is given in the following.

Definition 25 (Correctness). *A garbling scheme* $(\mathsf{Garble}, \mathsf{GCEval})$ *is correct if for all* $\lambda \in \mathbb{N}$*, for all polynomial size circuits* $\mathcal{C}$ *with input length* n*, and for all inputs* $x \in \{0,1\}^n$ *it holds that*

$$\mathcal{C}(x) = \mathsf{GCEval}(\tilde{\mathcal{C}}, \{\mathsf{lbs}_i^{(x_i)}\}_{i=1}^n)$$

where $(\tilde{\mathcal{C}}, \{\mathsf{lbs}_i^{(0)}, \mathsf{lbs}_i^{(1)}\}_{i=1}^n) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C})$.

Security requires that the evaluation of a circuit reveals nothing beyond its output.

Definition 26 (Security). *A garbling scheme* $(\mathsf{Garble}, \mathsf{GCEval})$ *is secure if for all* $\lambda \in \mathbb{N}$ *there exists a PPT algorithm* GCSim *such that for all PPT attacker* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(\tilde{\mathcal{C}}, \{\mathsf{lbs}_i^{(x_i)}\}_{i=1}^n, \mathsf{state}) \middle| \begin{array}{l} (\mathcal{C}, x, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\tilde{\mathcal{C}}, \{\mathsf{lbs}_i^{(0)}, \mathsf{lbs}_i^{(1)}\}_{i=1}^n) \leftarrow \mathsf{Garble}(1^\lambda, \mathcal{C}) \end{array} \right] - \Pr\left[1 \leftarrow \mathcal{A}_2(\tilde{\mathcal{C}}, \{\mathsf{lbs}_i\}_{i=1}^n, \mathsf{state}) \middle| \begin{array}{l} (\mathcal{C}, x, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda) \\ (\tilde{\mathcal{C}}, \{\mathsf{lbs}_i\}_{i=1}^n) \leftarrow \mathsf{GCSim}(1^\lambda, |\mathcal{C}|, \mathcal{C}(x)) \end{array} \right] \right| = \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of $\mathcal{A}_1$*,* Garble*, and* GCSim*.*

4 Interaction-Preserving Compilers for Multi-Party Computation

We present our results in an incremental way, starting from a compiler which introduces one extra round of interaction (with respect to the optimal insecure protocol). Then we show how to upgrade to a round-preserving compiler by introducing progressively more powerful cryptographic machinery.

4.1 From N to N + 1 Rounds from Standard Assumptions

We now provide a transformation which compiles an insecure N-round protocol into a semi-honestly secure $N+1$ round protocol. The protocol assumes the existence of

(1) a multi-hop MK-FHE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with threshold decryption, and
(2) a laconic function evaluation scheme $(\mathsf{LFEGen}, \mathsf{LFEHash}, \mathsf{LFEEnc}, \mathsf{LFEDec})$.

The description of Π_{N+1} for a functionality f is specified below. Without loss of generality, we assume that f is a deterministic function.[10]

[10] Randomized functions can always be simulated by artificially adding some random bits to the inputs of the parties and using them as the random coins of the function.

Let $f : \{0,1\}^{k \cdot M} \to \{0,1\}^{\ell \cdot M}$ be and M-party functionality and let $(\mathsf{Next}, \mathsf{Output})$ be an N-round (insecure) protocol for f.

Input: Each party P_i is given as input a string $x_i \in \{0,1\}^k$.

Setup: Execute $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda)$ and set the public parameters to $(\mathsf{pp}, \mathsf{crs})$, which are made available to all parties.

Round 1: Each party P_i samples a random tape $r_i \leftarrow\!\!\$\, \{0,1\}^R$, for some polynomial $R = R(\lambda)$, then it samples a random key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and computes

$$c_i^{(0)} \leftarrow \mathsf{Enc}\left(\mathsf{pk}_i, x_i\right)$$

where the random coins are uniformly sampled. Then P_i computes the ciphertext

$$c_i^{(1)} \leftarrow \mathsf{Eval}\left(\overline{\mathsf{pk}}_i, \mathsf{Next}_i\left(1^\lambda, \cdot, r_i, \emptyset\right), c_1^{(0)}\right)$$

and broadcasts $\left(\mathsf{pk}_i, c_i^{(1)}\right)$.

Round $\mathbf{n} \in \{2, \ldots, N-1\}$: Let $(c_1^{(1)}, \ldots, c_M^{(n-1)})$ be the set of ciphertext exchanged up to the previous round. Each party P_i computes and broadcasts

$$c_i^{(n)} \leftarrow \mathsf{Eval}\left((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \mathsf{Next}_i\left(1^\lambda, \cdot, r_i, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(n-1)}\right)\right).$$

Round N: Let $(c_1^{(1)}, \ldots, c_M^{(N-1)})$ be the set of ciphertext exchanged up to the previous round. Each party P_i computes

$$c_i^{(N)} \leftarrow \mathsf{Eval}\left((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \mathsf{Next}_i\left(1^\lambda, \cdot, r_i, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)}\right)\right).$$

Let $\Omega_i\left[1^\lambda, r_i, c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)}\right]$ be the circuit that, on input $(\mathsf{sk}, j, r, c_1^{(N)}, \ldots, c_M^{(N)})$, computes

$$c^* \leftarrow \mathsf{Eval}\left((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \mathsf{Output}_i\left(1^\lambda, \cdot, r_i, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N)}\right)\right)$$

and returns $\mathsf{PDec}\left(\mathsf{sk}, c^*, j; r\right)$. I.e., Ω_i is obtained by hardwiring 1^λ, r_i, and $(c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)})$ and leaving sk, j, and $(c_1^{(N)}, \ldots, c_M^{(N)})$ as variable inputs. The circuit then homomorphically evaluates the output function over the variables and returns the partial decryption of the resulting ciphertext. Finally P_i computes

$$(h_i, d_i) \leftarrow \mathsf{LFEHash}\left(\mathsf{crs}, \Omega_i\left[1^\lambda, r_i, c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)}\right]\right)$$

and broadcasts $(c_i^{(N)}, h_i)$.

Round $\mathbf{N+1}$: Let $(c_1^{(N)}, h_1), \ldots, (c_M^{(N)}, h_M)$ be the messages received in the last round. For all $j \in [M]$, the i-th party P_i computes

$$\tilde{c}_{i,j} \leftarrow \mathsf{LFEEnc}\left(\mathsf{crs}, h_j, \left(\mathsf{sk}_i, i, \tilde{r}_{i,j}, c_1^{(N)}, \ldots, c_M^{(N)}\right)\right).$$

(where the random coins for LFEEnc and $\tilde{r}_{i,j}$ are uniformly sampled) and sends $\tilde{c}_{i,j}$ to P_j.

Output: The let $(\tilde{c}_{1,i}, \ldots, \tilde{c}_{M,i})$ be the ciphertexts received by the i-th party P_i in the last round. For all $j \in [M]$, P_i computes

$$s_{j,i} \leftarrow \mathsf{LFEDec}(\mathsf{crs}, \tilde{c}_{j,i}, d_i)$$

and outputs

$$z_i \leftarrow \mathsf{Rec}(s_{1,i}, \ldots, s_{M,i}).$$

Communication. We now discuss the communication complexity of Π_{N+1}, assuming that $N \geq 2$. When instantiating the MK-FHE with distributed setup with the rate-1 scheme (as described in Section 7) and setting the size of the MK-FHE message space to be $M\mathsf{poly}(\lambda)$ regardless of the insecure protocol message size[11], we obtain a communication complexity that approaches that of the insecure protocol for the rounds 1 to N. More precisely, the communication complexity is identical to that of the insecure protocol except that all messages are encrypted, requiring a minimum communication of $M\mathsf{poly}(\lambda)$ per round for a single ciphertext message, and that each party needs to broadcast their corresponding public key of size $M\mathsf{poly}(\lambda)$. As the security parameter λ grows to infinity, the size of the ciphertexts approaches the sum of the worst case communication complexities per round of the insecure protocol $\sum_{i=1}^{N} |m_i|$.

Additionally, an LFE hash and ciphertext are sent. The LFE circuit class contains circuits of depth $|\mathsf{depth}|$ which take inputs of size $|\mathsf{last}|$ and have output size $|\mathsf{out}|$, where last is the communication complexity of the N-th round of the insecure protocol, depth is the maximum depth of (the circuit representation of) the output function Output, and $|\mathsf{out}|$ is the size of the output.

The total communication complexity of Π_{N+1} is therefore bounded by

$$\sum_{i=1}^{N} |m_i|(1 + o(1)) + NM^2\mathsf{poly}(\lambda) + \mathsf{LFECC}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$$

Instantiating the LFE scheme with the construction of Quach et al. [43], the LFE ciphertext has size $\tilde{O}(|\mathsf{last}| + |\mathsf{out}|)\mathsf{poly}(|\mathsf{depth}|, \lambda)$.

Correctness. Assuming the perfect correctness of the MK-FHE and LFE schemes, the output of the protocol equals $f(x_1, \ldots, x_M)$ with probability 1.

Security. We analyze the (semi-honest) security of our scheme assuming that the adversary corrupts *exactly* $M - 1$ parties, as opposed to *at most* $M - 1$ parties. The following Lemma shows that this is without loss of generality.

Lemma 1 ([40]). *Let Π be a semi-honest MPC protocol secure against the corruption of exactly $M - 1$ parties, then there exists a semi-honest MPC protocol Π' secure against the corruption of any subset of parties.*

We briefly recall the transformation in the following and we refer the reader to [40] for more details. The modified protocol Π' is identical to Π, except that instead of computing the function f it computes the extended function f' defined as follows: Return $f(x'_1 \ldots x'_M)$ where $x'_i = x_i$ (the input of the i-th party) if $\mathsf{mode} = 1$ else $x'_i = x_i \oplus \mathsf{PRF}(K, i)$ if $\mathsf{mode} = 2$, where K is a key sampled by one of the parties.

As for the communication complexity of the resulting protocol, we only need to argue how the communication complexity of the insecure protocol (to compute f' instead of f) changes. The PRF and the addition modulo 2 can be computed locally by each party, so they do not affect the communication complexity of the protocol. The only extra cost is that of an additional round for the designated party to communicate the PRF key K (where $K \in \{0,1\}^\lambda$).

We then proceed by establishing the (semi-honest) security of our scheme.

Theorem 5 (Security). *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a semantically-secure MK-FHE scheme with threshold decryption and let* $(\mathsf{LFEGen}, \mathsf{LFEHash}, \mathsf{LFEEnc}, \mathsf{LFEDec})$ *be a simulatable and function-hiding LFE scheme. Then Π_{N+1} is a semi-honest secure MPC protocol.*

Proof. Let P_i be the honest party. We modify the protocol through a series of hybrids, which are defined below.

Hybrid 0: This is the real experiment.

[11] Messages not fitting into a single ciphertext can be encrypted as multiple ciphertexts due to the fully homomorphic nature of the MK-FHE.

Hybrid 1 ... M − 1: The Hybrid j (for all $j \in [M-1]$) is identical to the previous one except that the ciphertext $\tilde{c}_{i,j}$ is computed as

$$\tilde{c}_{i,j} \leftarrow \mathsf{LFESim}(\mathsf{crs}, h_j, \Omega_j, s_{i,j})$$

where $s_{i,j}$ is computed as

$$s_{i,j} \leftarrow \Omega_j \left[1^\lambda, r_j, c_j^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)}\right] \left(\mathsf{sk}_i, i, \tilde{r}_{i,j}, c_1^{(N)}, \ldots, c_M^{(N)}\right).$$

Note that the modified ciphertext is functionally equivalent to the original and therefore Hybrid j is computationally indistinguishable from the previous one from an invocation of the simulatability of the LFE.

Hybrid M ... 2M − 2: The Hybrid $M - 1 + j$ (for all $j \in [M-1]$) is identical to the previous one except that the share $s_{i,j}$ is computed as follows

$$s_{i,j} \leftarrow \mathsf{Sim}\left(z_j, c^*, i, \{\mathsf{sk}_j\}_{j \in [M] \setminus \{i\}}\right)$$

where z_j is the output of P_j and c^* is defined as in Ω_j, i.e.,

$$c^* \leftarrow \mathsf{Eval}\left(\left(\mathsf{pk}_1, \ldots, \mathsf{pk}_M\right), \mathsf{Output}_i\left(1^\lambda, \cdot, r_i, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N)}\right)\right).$$

By the simulatability of the MK-FHE with threshold decryption, the distribution induced by this hybrid is statistically close to that of the previouse one.

Hybrid 2M − 1: In this hybrid we modify the hash h_i computed by the honest party to

$$h_i \leftarrow \mathsf{LFESimFH}\left(\mathsf{crs}, \Phi\right)$$

By the function-hiding of the LFE, this hybrid is computationally indistinguishable from the previous one.

Hybrid 2M: In this hybrid the ciphertext $\overline{c}_i^{(0)}$ sent in the first round by the honest party is substituted by encryptions of a fixed string (e.g., a concatenation of 0s) of the appropriate length. This modification is indistinguishable by the semantic security of the MK-FHE scheme.

Simulator: The simulator computes the views of the corrupted parties as dictated by the protocol specifications and the views of the honest party as in the previous hybrid. Note that the view induced by the simulator is identical to the previous hybrid and therefore is computationally indistinguishable from that of the original protocol.

Note that the same argument can be shown to prove the security of the construction even if we include all of the ciphertexts encrypting the parties' inputs in the view of the adversary. This observation is going to be useful to greatly simplify the analysis of our subsequent protocol.

The Two-Party Case. In the following we show that, for the special case of $M = 2$, a slight modification of Π_{N+1} yields a round-preserving (i.e., N to N) compiler with the similar communication complexity and from the same assumptions. We consider the case where a single party (say P_1) receives an output but the same procedure can be repeated in parallel to extend the protocol to the more general case where both parties receive an output. The first modification that we introduce is that the LFE hash is computed by P_1 in the $N - 1$ and the compressed circuit $\tilde{\Omega}$ is defined as follows.

$\tilde{\Omega}\left[1^\lambda, r_1, \mathsf{sk}_1, c_i^{(0)}, c_1^{(1)}, \ldots, c_1^{(N-1)}, c_2^{(1)}, \ldots, c_2^{(N-2)}\right]\left(\mathsf{sk}_2, 2, r, c_2^{(N-1)}, c_2^{(N)}\right) =$

- Compute $c_1^{(N)} \leftarrow \mathsf{Eval}\left(\left(\mathsf{pk}_1, \mathsf{pk}_2\right), \mathsf{Next}_1\left(1^\lambda, \cdot, r_1, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_2^{(N-1)}\right)\right)$
- Compute $c^* \leftarrow \mathsf{Eval}\left(\left(\mathsf{pk}_1, \mathsf{pk}_2\right), \mathsf{Output}_1\left(1^\lambda, \cdot, r_1, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_2^{(N)}\right)\right)$
- Return $\mathsf{PDec}(\mathsf{sk}_1, c^*, 1; 0) \| \mathsf{PDec}\left(\mathsf{sk}_2, c^*, 2; r\right)$.

On a high level, the circuit $\tilde{\Omega}$ hardwires all ciphertexts seen by P_1 up to the $(N-1)$-th round (included) and sent by P_2 up to the $(N-2)$-th round (included). On input the secret key sk_2, a randomness r, and the last two messages of P_2 $(c_2^{(N-1)}, c_2^{(N)})$, the circuit recomputes the last message $c_1^{(N)}$ and uses all of the ciphertexts to complete the homomorphic computation of the output c^*. Finally the circuit outputs the partial decryptions of c^* using sk_1 and sk_2 and some randomness r (for sk_1 we can fix the randomness to a known string, e.g., a concatenation of 0s). The message sent by P_2 in the N-th round consists of the LFE ciphertext $\tilde{c}$ computed as

$$\tilde{c} \leftarrow \mathsf{LFEEnc}\left(\mathsf{crs}, h, \left(\mathsf{sk}_2, 2, \tilde{r}_2, c_2^{(N-1)}, c_2^{(N)}\right)\right)$$

where h is the hash of the circuit Ω as defined above and $\tilde{r}_2$ is taken from the random tape r_2 of P_2. After the N-th round, P_i can recover the output by evaluating $\mathsf{LFEDec}(\mathsf{crs}, \Omega, \tilde{c})$ and running the threshold reconstruction algorithm. This allows P_1 to complete the protocol execution locally, without any further interaction with P_2. Note that all information needed to compute $\tilde{c}$ is available to P_2 by the end of the $(N-1)$-th round. This gives us an N-to-N communication-efficient compiler for the case of 2 parties. The communication complexity of the protocol is increased by a term $\mathsf{poly}(|\mathsf{depth}|, |\mathsf{enc}|, |\mathsf{out}|, \lambda)$, where $|\mathsf{depth}|$ is the depth of the circuit representation of $\tilde{\Omega}$ and $|\mathsf{enc}|$ is the size of the encodings. When instantiating with the LFE protocol of [43], the term $|\mathsf{depth}|$ is proportional to the depth of computation of the last message and the post-processing of the insecure protocol, and the term $|\mathsf{enc}|$ is proportional to the size of the last two messages of the insecure protocol.

While this gives us a round-preserving solution of the two-party case, the same ideas do not seem to extend to the more general case of M parties. In the what follows we show how to overcome this barrier by leveraging stronger cryptographic machinery.

4.2 From N to N Rounds

We will now show how to modify the protocol of the last section to obtain a secure N-round protocol. This transformation will require heavier tools, namely SNARKs and extractable witness encryption. Let $(\mathsf{HashGen}, \mathsf{Hash})$ be a collision-resistant hash function and let $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ be a succinct non-interactive argument of knowledge system for the following language $\mathcal{L}$: A tuple

$$(\mathsf{K}, \zeta_i, (c_i^{(N)}, h_i), (c_1^{(N-1)}, \ldots, c_M^{(N-1)}))$$

is in the language $\mathcal{L}$, if there exist a state

$$\mathsf{state}_i = (r_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_M, c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-2)})$$

such that

$$\begin{aligned}
\zeta_i &= \mathsf{Hash}(\mathsf{K}, \mathsf{state}_i) \\
c_i^{(N)} &= \mathsf{Eval}\left((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \mathsf{Next}_i\left(1^\lambda, \cdot, r_i, \cdot\right), \left(c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)}\right)\right) \\
(h_i, d_i) &= \mathsf{LFEHash}\left(\mathsf{crs}, \Omega_i\left[1^\lambda, c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-1)}\right]\right).
\end{aligned}$$

Assume that the bit-length of $\mathbf{t} = ((c_j^{(N)}, h_j))_{j\in[M]}$ is k bits and that for a $\kappa \in [k]$ the function $\mathsf{bit}_\kappa(\mathbf{t})$ computes the κ-th bit of $\mathbf{t}$. Let $(\mathsf{WEEnc}, \mathsf{WEDec})$ be an extractable witness encryption scheme for the following language $\mathcal{L}'$. A statement $\mathbf{x} = (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, c_1^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, b)$ is in the language $\mathcal{L}'$, if there exist $\tilde{\mathbf{t}} = ((\tilde{c}_j^{(N)}, \tilde{h}_j))_{j\in[M]}$ and $\tilde{\pi}_1, \ldots, \tilde{\pi}_M$ such that $\mathsf{bit}_\kappa(\tilde{\mathbf{t}}) = b$ and for all $i \in [M]$ it holds that

$$\mathsf{SNARKVerify}(\mathsf{crs}_{\mathsf{SNARK}}, (\mathsf{K}, \zeta_j, (\tilde{c}_j^{(N)}, \tilde{h}_j), (c_1^{(N-1)}, \ldots, c_M^{(N-1)})), \tilde{\pi}_j) = 1.$$

Let

$$\mathbf{x} \leftarrow (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, c_1^{(N-1)}, \ldots, c_M^{(N-1)})$$
$$\mathbf{w} \leftarrow (\mathbf{t}, \pi_1, \ldots, \pi_M).$$

For better readability, we will define the follow *batch version* of $(\mathsf{WEEnc}, \mathsf{WEDec})$.

$\mathsf{BatchWEEnc}(\mathbf{x}, \mathsf{lbs})$**:** Takes as input a statement $\mathbf{x}$ and labels $\mathsf{lbs} \in (\{0,1\}^\lambda)^{k \times 2}$ and computes and outputs $(\mathsf{WEEnc}((\mathbf{x}, \kappa, b), \mathsf{lbs}_{\kappa,b}))_{\kappa \in [k], b \in \{0,1\}}$.

$\mathsf{BatchWEDec}(v, \mathbf{x}, \mathbf{w})$**:** Takes as input a ciphertext $v = (v_{\kappa,b})_{\kappa \in [k], b \in \{0,1\}}$, a statement $\mathbf{x}$ and a witness $\mathbf{w}$ and outputs $(\mathsf{WEDec}(v_{\kappa, \mathsf{bit}_\kappa(\mathbf{t})}, (\mathbf{x}, \kappa, \mathsf{bit}_\kappa(\mathbf{t})), \mathbf{w}))_{\kappa \in [k]}$.

Let $(\mathsf{Garble}, \mathsf{GCEval})$ be a projective garbling scheme. We can now describe a modification which collapses rounds N and $N+1$ of protocol Π_{N+1} into a single round. Up until round $N-2$ the protocol is identical to the previous protocol, with the modification that $\mathsf{SNARKGen}$ and $\mathsf{HashGen}$ are computed in the setup and the public parameters are augmented with a common reference string $\mathsf{crs}_{\mathsf{SNARK}}$ and a hashing key K. Consequently, for Π_N we describe only the modified rounds $N-1$, N and the output function.

Round N − 1: Party P_i proceeds as follows.

- Compute $c_i^{(N-1)}$ as in protocol Π_{N+1}
- Set $\mathsf{state}_i = (r_i, \mathsf{pk}_1, \ldots, \mathsf{pk}_M, c_i^{(0)}, c_1^{(1)}, \ldots, c_M^{(N-2)})$
- Compute $\zeta_i \leftarrow \mathsf{Hash}(\mathsf{K}, \mathsf{state}_i)$.
- Broadcast $(c_i^{(N-1)}, \zeta_i)$

Round N: Upon receiving the $((c_j^{(N-1)}, \zeta_j))_{j \in [M]}$ party P_i proceeds as follows.

- Compute $(c_i^{(N)}, h_i)$ as in protocol Π_{N+1}
- Compute

$$\pi_i \leftarrow \mathsf{SNARKProve}(\mathsf{crs}_{\mathsf{SNARK}}, (\mathsf{K}, \zeta_i, c_i^{(N)}, (c_1^{(N-1)}, \ldots, c_M^{(N-1)})), \mathsf{state}_i)$$

- Define a circuit $\mathsf{C}[\mathsf{sk}_i, i, j, \tilde{r}, \rho'](\mathbf{t})$, which takes as hardwired inputs sk_i, i, j and random coins ρ', an explicit input $\mathbf{t} = ((c_j^{(N)}, h_j))_{j \in [M]}$, and computes and outputs $\mathsf{LFEEnc}\left(\mathsf{crs}, h_j, \left(\mathsf{sk}_i, i, \tilde{r}_{i,j}, c_1^{(N)}, \ldots, c_M^{(N)}\right); \rho'\right)$.
- Set $\mathbf{x} \leftarrow (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, c_1^{(N-1)}, \ldots, c_M^{(N-1)})$
- For every $j \in [M]$ choose random coins $\rho'_j \leftarrow_\$ \{0,1\}^\lambda$ and compute $(\hat{\mathsf{C}}_{i,j}, \mathsf{lbs}_{i,j}) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{C}[\mathsf{sk}_i, i, j, \tilde{r}_{i,j}, \rho'_j])$. Compute $v_{i,j} \leftarrow \mathsf{BatchWEEnc}(\mathbf{x}, \mathsf{lbs}_{i,j})$.
- Send $(c_i^{(N)}, h_i, \pi_i, \hat{\mathsf{C}}_{i,j}, v_{i,j})$ to P_j

Output: Upon receiving the $((c_j^{(N)}, h_j, \pi_j, \hat{\mathsf{C}}_{j,i}, v_{j,i}))_{j \in [M]}$ party P_i proceeds as follows.

- For all $j \in [M]$:
  - Set $\mathbf{x} \leftarrow (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, c_1^{(N-1)}, \ldots, c_M^{(N-1)})$
  - Set $\mathbf{t} = ((c_j^{(N)}, h_j))_{j \in [M]}$
  - Set $\mathbf{w} \leftarrow (\mathbf{t}, \pi_1, \ldots, \pi_M)$
  - Compute $\overline{\mathsf{lbs}}_j \leftarrow \mathsf{BatchWEDec}(v_{j,i}, \mathbf{x}, \mathbf{w})$.
  - Compute $\tilde{c}_{i,j} \leftarrow \mathsf{GCEval}(\hat{\mathsf{C}}_{j,i}, \overline{\mathsf{lbs}}_j)$
  - Compute $s_{i,j} \leftarrow \mathsf{LFEDec}(\mathsf{crs}, \Omega_i, \tilde{c}_{i,j})$.
- Compute and output $z_i \leftarrow \mathsf{Rec}(s_{1,i}, \ldots, s_{M,i})$.

Communication. To analyze the communication complexity, note that compared to protocol Π_{N+1}, the only additional values which need to be sent are the hash ζ_i in round $N-1$ as well as the garbled circuit

$(\hat{\mathsf{C}}_{i,j}, v_{i,j})$ and the witness encryption $v_{i,j}$ in round N. The hash is of size $\mathsf{poly}(\lambda)$. The garbled circuit encodes the LFE encryption procedure, which has size $\mathsf{LFECS}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$. Note that this garbled circuit replaces the sending of the actual LFE encryption from protocol Π_{N+1}. The witness encryption size scales with the statement and witness. The statement consists of the encrypted m_{N-1}, M hashes, $\mathsf{crs}_{\mathsf{SNARK}}$, the hash key K, and a single bit. A witness consists of the encrypted m_N, M LFE hashes, and M SNARKs. Recall that the LFE hashes and SNARKs have size $\mathsf{poly}(\lambda)$. Additionally, the witness encrypted message are the labels to the garbled circuit, which computes the LFE encryption. The input to the LFE encryption includes the LFE crs, the LFE hash, an MK-FHE secret key, some randomness, and message m_N.

Overall, this gives a communication complexity of

$$\sum_{i=1}^{N} |m_i|(1+o(1)) + NM^2\mathsf{poly}(\lambda) + \mathsf{LFECS}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$$
$$+ \mathsf{WEncCC}(\mathsf{LFECRS}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + \mathsf{poly}(\lambda) + |m_N|, |m_{N-1}| + M\mathsf{poly}(\lambda), |m_N| + M\mathsf{poly}(\lambda), \lambda)$$

Instantiating the LFE scheme with the construction of Quach et al. [43], the LFE encryption algorithm takes time $\tilde{O}(|\mathsf{last}| + |\mathsf{out}|)\mathsf{poly}(|\mathsf{depth}|, \lambda)$, so the circuit size can be bounded by the square.

Correctness. We will first show correctness of the protocol. To establish correctness, first note that it holds for all $i \in [M]$ that $(\mathsf{K}, \zeta_i, c_i^{(N)}, (c_1^{(N-1)}, \ldots, c_M^{(N-1)})) \in \mathcal{L}$ and that state_i is a valid witness for this statement. Consequently, by the completeness of $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ it holds for all $i \in [M]$ that

$$\mathsf{SNARKVerify}(\mathsf{crs}_{\mathsf{SNARK}}, (\mathsf{K}, \zeta_i, (c_i^{(N)}, h_i), (c_1^{(N-1)}, \ldots, c_M^{(N-1)})), \pi_i) = 1.$$

Consequently, it also holds for all $\kappa \in [k]$ that $\mathbf{x}' = (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, c_1^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, \mathsf{bit}_\kappa(\mathbf{t})) \in \mathcal{L}'$ and $\mathbf{w} = (\mathbf{t}, \pi_1, \ldots, \pi_M)$ is a witness. Setting $\mathbf{x} = (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, c_1^{(N-1)}, \ldots, c_M^{(N-1)})$, as $v_{i,j} = \mathsf{BatchWEEnc}(\mathbf{x}, \mathsf{lbs}_{i,j})$, it holds by the correctness of $(\mathsf{BatchWEEnc}, \mathsf{BatchWEDec})$ that

$$\overline{\mathsf{lbs}}_j = \mathsf{BatchWEDec}(v_{j,i}, \mathbf{x}, \mathbf{w}) = ((\mathsf{lbs}_{i,j})_{\kappa, \mathsf{bit}_\kappa(\mathbf{t})})_\kappa.$$

Thus, since $(\hat{\mathsf{C}}_{i,j}, \mathsf{lbs}_{i,j}) \leftarrow \mathsf{Garble}(1^\lambda, \mathsf{C}[\mathsf{sk}_i, i, j, \tilde{r}_{i,j}, \rho'_j])$ it holds by the correctness of the garbling scheme $(\mathsf{Garble}, \mathsf{GCEval})$ we get that

$$\tilde{c}_{i,j} = \mathsf{GCEval}(\hat{\mathsf{C}}_{j,i}, \overline{\mathsf{lbs}}_j) = \mathsf{LFEEnc}\left(\mathsf{crs}, h_j, \left(\mathsf{sk}_i, i, \tilde{r}_{i,j}, c_1^{(N)}, \ldots, c_M^{(N)}\right); \rho'\right).$$

We conclude that $\tilde{c}_{i,j}$ has the same value as in protocol Π_{N+1}, and we can conclude from the correctness of protocol Π_{N+1} that Π_N is also correct.

Security. We show semi-honest security of our protocol in the following.

Theorem 6. *Assume that Π_{N+1} is a semi-honest MPC protocol, that* $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ *is a non-adaptively sound succinct non-interactive argument system of knowledge,* $(\mathsf{HashGen}, \mathsf{Hash})$ *is a collision-resistant hash function, and* $(\mathsf{BatchWEEnc}, \mathsf{BatchWEDec})$ *is an extractable witness encryption scheme. Then Π_N is a semi-honest MPC protocol.*

Proof. Assume without loss of generality that P_M is the honest party. We modify the protocol through a series of hybrids, which are defined below. Fix inputs $x_1, \ldots, x_M$.

Hybrid 0 : This is the real experiment.

Hybrids $1 \ldots M-1$: Hybrid j for $j \in [M-1]$ is identical to the previous hybrid, except that party P_M computes $v_{M,j}$ by $v_{M,j} \leftarrow \mathsf{BatchWEEnc}((\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M), \mathsf{lbs}^*_{M,j})$, where $\mathsf{lbs}^*_{M,j}$ is such that $\mathsf{lbs}^*_{M,j,\kappa,b} = \mathsf{lbs}^*_{M,j,\kappa,\mathsf{bit}_\kappa(\mathbf{t})}$ for all $\kappa \in [k]$ and all $b \in \{0,1\}$.

Hybrids $M \ldots 2M-2$: Hybrid $M-1+j$ for $j \in [M-1]$ is identical to the previous hybrid, except that party P_M computes $(\hat{\mathsf{C}}_{M,j}, \mathsf{lbs}^*_{M,j})$ by

$$(\hat{\mathsf{C}}_{M,j}, \mathsf{lbs}^*_{M,j}) \leftarrow \mathsf{GCSim}(\mathsf{LFEEnc}\left(\mathsf{crs}, h_j, \left(\mathsf{sk}_M, M, \tilde{r}_{M,j}, c_1^{(N)}, \ldots, c_M^{(N)}\right); \rho'_{M,j}\right)).$$

First note that in hybrid $2M-2$ the garbled circuits $\hat{\mathsf{C}}_{M,j}$ for $j \in [M-2]$ are simulated. Consequently, we can simulate the output of hybrid $2M-2$ given a transcript of protocol Π_{N+1} which contains the views of all parties except P_M but which does include $c_M^{(0)}$ and r_M (which as remarked above can be considered public values). Thus, given a transcript computed by the simulator for Π_{N+1} we can simulate transcripts for Π_N. To see this, note that the only additional values included in transcripts of Π_N are the $(\zeta_j)_j$, $(\pi_j)_{j\in[M]}$ and the $(\hat{\mathsf{C}}_{M,j}, v_{M,j})$. For every party except P_M these values can immediately be computed from their corresponding views. For party P_M, we can compute ζ_M and π_M using $c_M^{(0)}$, r_M, the public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_M)$ and the ciphertexts $(c_1^{(1)}, \ldots, c_M^{(N)})$ which are all included in a view of protocol Π_{N+1}. Finally, we can simulate the $\hat{\mathsf{C}}_{M,j}$ and $v_{M,j}$ by computing $(\hat{\mathsf{C}}_{M,j}, \mathsf{lbs}^*_{i,j}) \leftarrow \mathsf{GCSim}(\tilde{c}_{M,j})$ as the LFE-ciphertexts $\tilde{c}_{M,j}$ are included in the view of protocol Π_{N+1}. By the security of Π_{N+1} we can conclude that the advantage of any PPT-distinguisher $\mathcal{D}$ distinguishing between hybrid $2M-1$ and simulated transcripts is negligible.

Indistinguishability between hybrids $M \ldots 2M-2$ follows routinely from the simulation-security of the garbling scheme $(\mathsf{Garble}, \mathsf{GCEval}, \mathsf{GCSim})$. The main technical challenge is establishing indistinguishability between hybrids $1 \ldots M-1$. We will use the security of $(\mathsf{BatchWEEnc}, \mathsf{BatchWEDec})$, the proof-of-knowledge property of $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ and the collision resistance of $(\mathsf{HashGen}, \mathsf{Hash})$ to establish this.

Thus, assume towards contradiction there was a PPT distinguisher $\mathcal{D}$ which distinguishes with non-negligible advantage ϵ between two hybrids j and $j+1$ for some $j \in [M-1]$. We will first go through a sequence of sub-hybrids hybrid' $1 \ldots k$ over all $\kappa \in [k]$ where we change the way in which $v_{M,j}$ is computed. $v_{M,j}$ is computed by

$$v_{M,j} \leftarrow \mathsf{BatchWEEnc}(\mathbf{x}, \mathsf{lbs}_{M,j}),$$

where $\mathbf{x} = (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M)$. Unrolling the definition of $\mathsf{BatchWEEnc}$, we get that $(v_{M,j})_{\kappa,b}$ is computed by

$$(v_{M,j})_{\kappa,b} \leftarrow \mathsf{WEEnc}((\mathbf{x}, \kappa, b), \mathsf{lbs}_{\kappa,b}).$$

Now, in hybrid' ι for $\iota \in [k]$, for $\kappa \leq \iota$ we compute $(v_{M,j})_{\kappa,1-\mathsf{bit}_\kappa(\mathbf{t})}$ by

$$(v_{M,j})_{\kappa,1-\mathsf{bit}_\kappa(\mathbf{t})} \leftarrow \mathsf{WEEnc}((\mathbf{x}, \kappa, b), \mathsf{lbs}_{\kappa,\mathsf{bit}_\kappa(\mathbf{t})}),$$

whereas for $\kappa > \iota$ we compute

$$(v_{M,j})_{\kappa,1-\mathsf{bit}_\kappa(\mathbf{t})} \leftarrow \mathsf{WEEnc}((\mathbf{x}, \kappa, b), \mathsf{lbs}_{\kappa,1-\mathsf{bit}_\kappa(\mathbf{t})}).$$

Note that in hybrid' k we compute $v_{M,j}$ as in hybrid j. Defining hybrid' 0 to be hybrid $j-1$, we gt by a routine hybrid argument that there exist $\iota \in [k]$ such that $\mathcal{D}$ distinguishes between hybrid' $\iota-1$ and hybrid' ι with advantage at least $\epsilon' = \epsilon/k$.

We will now construct a sampler Samp and a distinguisher $\mathcal{D}'$ where Samp produces a statement $\mathbf{x}'$, and auxiliary output $\mathbf{aux}$ and two messages m_0 and m_1 such that

$$|\Pr[\mathcal{D}'(\mathsf{WEEnc}(\mathbf{x}', m_0), \mathbf{aux}) = 1] - \Pr[\mathcal{D}'(\mathsf{WEEnc}(\mathbf{x}', m_1), \mathbf{aux}) = 1]| = \epsilon'.$$

The sampler Samp simulates hybrid' ι but removes $(v_{M,j})_{\iota,1-\mathsf{bit}_\iota(\mathbf{t})}$ from the resulting view. Call this view $\mathbf{aux}$. Samp further sets $\mathbf{x}' = (\mathbf{x}, \iota, 1-\mathsf{bit}_\iota(\mathbf{t}))$, $m_0 = \mathsf{lbs}_{\kappa,1-\mathsf{bit}_\kappa(\mathbf{t})}$ and $m_1 = \mathsf{lbs}_{\kappa,\mathsf{bit}_\kappa(\mathbf{t})}$. The distinguisher $\mathcal{D}'$ takes a ciphertext v and a view $\mathbf{aux}$, and sets $(v_{M,j})_{\iota,1-\mathsf{bit}_\iota(\mathbf{t})} = v$ in this view obtaining a complete view $\mathbf{aux}'$. It now runs $\mathcal{D}$ on $\mathbf{aux}'$ and outputs whatever $\mathcal{D}$ outputs. Clearly, if $v = \mathsf{WEEnc}(\mathbf{x}', m_0) = \mathsf{WEEnc}(\mathbf{x}', \mathsf{lbs}_{\kappa,1-\mathsf{bit}_\kappa(\mathbf{t})})$ then $\mathcal{D}'$ provides $\mathcal{D}$ is a well-formed view of hybrid' $\iota-1$. On the other hand, if $v = \mathsf{WEEnc}(\mathbf{x}', m_1) = \mathsf{WEEnc}(\mathbf{x}', \mathsf{lbs}_{\kappa,\mathsf{bit}_\kappa(\mathbf{t})})$ then $\mathcal{D}'$ runs $\mathcal{D}$ on a well-formed view of hybrid' ι. Consequently it holds that

$$|\Pr[\mathcal{D}'(\mathsf{WEEnc}(\mathbf{x}', m_0), \mathbf{aux}) = 1] - \Pr[\mathcal{D}'(\mathsf{WEEnc}(\mathbf{x}', m_1), \mathbf{aux}) = 1]| = \epsilon',$$

where $(\mathbf{x}', \mathbf{aux}, m_0, m_1) \leftarrow \mathsf{Samp}(1^\lambda)$.

Now, by the security of $(\mathsf{WEEnc}, \mathsf{WEDec})$ there exists a PPT machine Ext and a non-negligible ϵ'' such that $\mathsf{Ext}(\mathbf{x}', \mathbf{aux})$ outputs a witness $\mathbf{w}'$ for $\mathbf{x}'$ with probability ϵ''. The statement $\mathbf{x}'$ is of the form $\mathbf{x}' = (\mathbf{x}, \iota, 1-\mathsf{bit}_\iota(\mathbf{t})) = (\mathsf{crs}_{\mathsf{SNARK}}, \mathsf{K}, \zeta_1, \ldots, \zeta_M, \iota, 1-\mathsf{bit}_\iota(\mathbf{t}))$, and the witness $\mathbf{w}'$ is of the form $\mathbf{w}' = (\tilde{\mathbf{t}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_M)$ where $\tilde{\mathbf{t}} = ((\tilde{c}_j^{(N)}, \tilde{h}_j))_{j\in[M]}$ such that it holds that

$$\mathsf{bit}_\iota(\tilde{\mathbf{t}}) = 1 - \mathsf{bit}_\iota(\mathbf{t}) \tag{1}$$

$$\text{Forall } j \in [M]: \quad \mathsf{SNARKVerify}(\mathsf{crs}_{\mathsf{SNARK}}, (\mathsf{K}, \zeta_j, (\tilde{c}_j^{(N)}, \tilde{h}_j), (c_1^{(N-1)}, \ldots, c_M^{(N-1)})), \pi_j) = 1. \tag{2}$$

Due to equation 1 it holds that $\mathbf{t}$ and $\tilde{\mathbf{t}}$ differ in their ι-th bit position. Assume that bit position ι corresponds to an index $j^* \in [M]$ such that $(\tilde{c}_{j^*}^{(N)}, \tilde{h}_{j^*}) \neq (c_{j^*}^{(N)}, h_{j^*})$ (Recall that $\mathbf{t} = ((c_j^{(N)}, h_j))_{j\in[M]}$ is specified in the view $\mathbf{aux}$). Consequently by 2 it also holds that

$$\mathsf{SNARKVerify}(\mathsf{crs}_{\mathsf{SNARK}}, (\mathsf{K}, \zeta_{j^*}, (\tilde{c}_{j^*}^{(N)}, \tilde{h}_{j^*}), (c_1^{(N-1)}, \ldots, c_M^{(N-1)})), \pi_{j^*}) = 1.$$

Using Ext we can immediately construct a malicious prover $\mathcal{P}^*$ which takes as input the view aux and a common-reference string $\mathsf{crs}_{\mathsf{SNARK}}$ and outputs the statement $\mathbf{x}'' = (\mathsf{K}, \zeta_{j^*}, (\tilde{c}_{j^*}^{(N)}, \tilde{h}_{j^*}), (c_1^{(N-1)}, \ldots, c_M^{(N-1)}))$ and a proof π_{j^*} such that π_{j^*} verifies for $\mathbf{x}''$ with probability ϵ' and K, ζ_{j^*} have the same values as specified in $\mathbf{aux}$.

Now, by the proof-of-knowledge property of $(\mathsf{SNARKGen}, \mathsf{SNARKProve}, \mathsf{SNARKVerify})$ there exists a PPT-machine Ext' which on input aux and $\mathsf{crs}_{\mathsf{SNARK}}$ outputs a witness

$$\tilde{\mathsf{state}}_{j^*} = (\tilde{r}_{j^*}, \tilde{\mathsf{pk}}_1, \ldots, \tilde{\mathsf{pk}}_M, \tilde{c}_{j^*}^{(0)}, \tilde{c}_1^{(1)}, \ldots, \tilde{c}_M^{(N-2)})$$

such that

$$\zeta_{j^*} = \mathsf{Hash}(\mathsf{K}, \tilde{\mathsf{state}}_{j^*}) \tag{3}$$

$$\tilde{c}_{j^*}^{(N)} = \mathsf{Eval}\left((\tilde{\mathsf{pk}}_1, \ldots, \tilde{\mathsf{pk}}_M), \mathsf{Next}_i\left(1^\lambda, \cdot, \tilde{r}_{j^*}, \cdot\right), \left(\tilde{c}_{j^*}^{(0)}, \tilde{c}_1^{(1)}, \ldots, \tilde{c}_M^{(N-2)}, c_1^{(N-1)}, \ldots, c_M^{(N-1)}\right)\right) \tag{4}$$

$$\left(\tilde{h}_{j^*}, \tilde{d}_{j^*}\right) = \mathsf{LFEHash}\left(\mathsf{crs}, \Omega_i\left[1^\lambda, \tilde{c}_{j^*}^{(0)}, \tilde{c}_1^{(1)}, \ldots, \tilde{c}_M^{(N-2)}, c_1^{(N-1)}, \ldots, c_M^{(N-1)}\right]\right). \tag{5}$$

Now recall that $(\tilde{c}_{j^*}^{(N)}, \tilde{h}_{j^*}) \neq (c_{j^*}^{(N)}, h_{j^*})$, but since by (4) and (5) both $\tilde{c}_{j^*}^{(N)}$ and $\tilde{h}_{j^*}$ are deterministically computed from $\tilde{\mathsf{state}}_{j^*}$ and $(c_1^{(N-1)}, \ldots, c_M^{(N-1)})$, this means that $\tilde{\mathsf{state}}_{j^*} \neq \mathsf{state}_{j^*}$. But since ζ_{j^*} is computed by $\zeta_{j^*} \leftarrow \mathsf{Hash}(\mathsf{K}, \mathsf{state}_{j^*})$ and by (3) we have that $\zeta_{j^*} = \mathsf{Hash}(\mathsf{K}, \tilde{\mathsf{state}}_{j^*})$, this means that $(\mathsf{state}_{j^*}, \tilde{\mathsf{state}}_{j^*})$ forms a hash collision for key K. Consequently, we can use Ext'' to construct an adversary $\mathcal{A}$ which given a key K finds a hash collision for $(\mathsf{HashGen}, \mathsf{Hash})$ with probability ϵ'', contradicting the collision resistance of $(\mathsf{HashGen}, \mathsf{Hash})$. We can conclude that hybrid $j-1$ and j are indistinguishable, concluding the proof.

5 Semi-Honest Security in the Plain Model

After round 3, the protocol in the plain model is identical to the previous protocol (either Π_N or Π_{N+1}) in the common reference string model, so we describe only the modified rounds 1, 2, and 3.

Round 1: Each party P_i samples a random tape $r_i \leftarrow\!\!\$\, \{0,1\}^R$, for some polynomial $R = R(\lambda)$, and computes the first message

$$m_i^{(1)} \leftarrow \mathsf{Next}_i\left(1^\lambda, x_i, r_i, \emptyset\right)$$

Then P_i samples a random key pair $(\overline{\mathsf{sk}}_i, \overline{\mathsf{pk}}_i) \leftarrow \overline{\mathsf{KeyGen}}(1^\lambda)$ for an MK-FHE scheme in the plain model as well as a PRG key $s_i \leftarrow\!\!\$\, \{0,1\}^\lambda$ and computes the hybrid encryption

$$\hat{c}_i^{(1)} \leftarrow \left(\overline{\mathsf{Enc}}(\overline{pk}_i, s_i), \mathsf{PRG}(s_i) \oplus m_i^{(1)}\right)$$

where the coins for $\overline{\mathsf{Enc}}(\overline{pk}_i, s_i)$ are uniformly sampled. It samples its block of the public parameters $\mathsf{pp}_i \leftarrow \mathsf{Setup}(1^\lambda)$ for our rate-1 MK-FHE scheme, as described in appendix 7.4, and broadcasts $(\hat{c}_i^{(1)}, \mathsf{pp}_i)$. If $i = 1$, it additionally samples and broadcasts $\mathsf{crs} \leftarrow \mathsf{LFEGen}(1^\lambda)$ and, if compiling Π_N, $\mathsf{crs}_{\mathsf{SNARK}} \leftarrow \mathsf{SNARKGen}(1^\lambda)$.

Round 2: Each party P_i parses $\hat{c}_i^{(1)} = (\hat{c}_{i,1}^{(1)}, \hat{c}_{i,2}^{(1)})$ and computes

$$\overline{c}_j^{(1)} \leftarrow \mathsf{Eval}\left(\overline{pk}_j, \mathsf{PRGAndXOR}\left(1^\lambda, \cdot, \hat{c}_{i,2}^{(1)}\right), \hat{c}_{i,1}^{(1)}\right)$$

for each $j = 1, \ldots, M$. It computes its encrypted input

$$\overline{c}_i^{(0)} \leftarrow \overline{\mathsf{Enc}}(\overline{pk}_i, x_i)$$

then uses this to compute

$$\overline{c}_i^{(2)} = \mathsf{Eval}\left((\overline{pk}_1, \ldots, \overline{pk}_M), \mathsf{Next}\left(1^\lambda, \cdot, r_i, \cdot\right), \left(\overline{c}_i^{(0)}, \overline{c}_1^{(1)}, \ldots, \overline{c}_M^{(1)}\right)\right)$$

P_i then samples $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{crs})$ for the rate-1 MK-FHE scheme, and computes

$$c_i^{\overline{\mathsf{sk}}} \leftarrow \mathsf{Enc}\left(pk_i, \overline{sk}_i\right)$$

They broadcast $(\overline{c}_i^{(2)}, pk, c_i^{\overline{\mathsf{sk}}})$.

Round 3: Each party P_i computes

$$c_j^{(1)} \leftarrow \mathsf{Eval}\left(pk_j, \overline{\mathsf{Dec}}(\cdot,\cdot), \left(\overline{c}_j^{(1)}, c_j^{\overline{\mathsf{sk}}}\right)\right)$$
$$c_j^{(2)} \leftarrow \mathsf{Eval}\left((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \overline{\mathsf{Dec}}(\cdot,\cdot), \left(\overline{c}_j^{(2)}, \left(c_1^{\overline{\mathsf{sk}}}, \ldots, c_M^{\overline{\mathsf{sk}}}\right)\right)\right)$$

for each $j = 1, \ldots, M$. They compute

$$c_i^{(0)} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, x_i)$$

then proceed as in the previous protocol.

Communication. The only changes are to send an additional public key, to compute the first message under a hybrid encryption, to compute the second message under an MK-FHE without setup, and to send two additional encrypted keys. The new keys are all of size $\mathsf{poly}(\lambda)$, and the hybrid encryption has size $\mathsf{poly}(\lambda) + |m_1|$, so the only major difference is in the second message which is encrypted by a potentially inefficient MK-FHE. This at most increases the second message size by a multiplicative $\mathsf{poly}(\lambda)$.

Assuming $|m_2| > 0$, the overall communication complexity is $|\Pi_{N'}| + |m_2|\mathsf{poly}(\lambda)$, where $|\Pi_{N'}|$ is the communication complexity of the compiled $\Pi_{N'}$.

Correctness. Correctness follows easily from the correctness of the MK-FHE without setup and the correctness of the compiled $\Pi_{N'}$ when crs is sampled correctly.

Security. We prove the semi-honest security of our protocol in the plain model below.

Theorem 7. *Assume that $\Pi_{N'}$ is a semi-honest MPC protocol, that $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{Eval}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ is a semantically secure MK-FHE scheme with threshold decryption in the plain model, and the existence of PRGs. Then $\Pi_{\mathsf{pl},N'}$ is a semi-honest MPC protocol in the plain model.*

Proof. Let P_i be the honest party. We modify the protocol through a series of hybrids, which are defined below.

H0: This is the real experiment.

H1: In this hybrid, the CRS model simulator is invoked starting from round 3. In the case of Π_{N+1}, this change is exactly the same as using hybrid $2M-1$ in the proof of Theorem 5, and so is indistinguishable from the real transcript. For Π_N, recall that its simulator uses the simulator from Π_{N+1}, and otherwise only modifies round N. Since hybrid $2M-1$ in the proof of Theorem 5 is indistinguishable from the full simulator for Π_{N+1}, the view produced by this change is indistinguishable.

H2: Hybrid 2 is identical to hybrid 1, except in round 2 P_i computes $c^{(sk)_i}$ as an encryption of 0s instead of as an encryption of its plain model MK-FHE secret key $\overline{sk}_i$. Indistinguishability follows from the semantic security of the rate-1 MK-FHE scheme.

H3: Hybrid 3 is identical to hybrid 2, except in round 1 P_i computes the hybrid ciphertext $\hat{c}_i^{(1)}$ as

$$\hat{c}_i^{(1)} = \left(\overline{\mathsf{Enc}}(\overline{pk}_i, 0^\lambda), \mathsf{PRG}(s_i) \oplus m_i^{(1)}\right)$$

Indistinguishability follows from the semantic security of the plain model MK-FHE.

H4: Hybrid 4 is identical to hybrid 3, except in round 1 P_i computes its insecure message $m_i^{(1)}$ as a function of a fixed string (e.g. 0s) instead of as a function of its input x_i. Note that this causes the messages of $\Pi_{N'}$ after round 2 to be as in hybrid $2M$ from the proof of Theorem 5. Indistinguishability follows from the security of the PRG.

H5: Hybrid 5 is identical to hybrid 4, except in round 2 P_i computes $\bar{c}_i^{(0)}$ as an encryption of a fixed string instead of its input x. Indistinguishability follows from the semantic security of the plain model MK-FHE.

6 Malicious Security in the Plain Model

Our malicious-secure protocol compiles insecure protocols of length N to malicious-secure protocols of length N', where $N' = N$ or $N' = N+1$, depending on whether Π_N or Π_{N+1} is used internally. It additionally requires simulation-extractable SNARKs, a plain model MK-FHE scheme, a generic 2 round semi-malicious MPC protocol, and a generic 4 round malicious secure MPC protocol[12].

Before describing the construction, we first specify the exact behavior for each of these tools. At a high level, 2 round semi-malicious MPC protocol generates a crs for use in $\Pi_{\mathsf{pl},N'}$ and the zkSNARKS. The zkSNARKs will ensure adversarial semi-maliciousness in $\Pi_{\mathsf{pl},N'}$. The plain model MK-FHE scheme will hide any components using the generated crs until it is confirmed to be generated semi-maliciously (which the 2 round protocol has security against). The 4 round malicious secure MPC protocol will implement the MCDS functionality to confirm the crs was generated semi-maliciously, then if so, reveal any components which make use of it.

6.1 SNARK Languages

We will use zkSNARKs in both rounds $N'-1$ and N'. The zkSNARK in round $N'-1$ will prove that P_i's messages in the $\Pi_{\mathsf{pl},N'}$ transcript so far have been generated honestly with respect to some input and randomness. The zkSNARK in round N' will then prove that P_i's next message in the transcript has been generated honestly with respect to the same input and randomness used for the previous zkSNARK.

The NP languages $\mathcal{L}_{\mathsf{SNARK},N'-1}$ is defined as follows: a statement

$$\left(\mathbb{T}_{\Pi_{N'}}^{(N'-2)}, m_i^{(N'-1)}\right)$$

[12] The generic 4 round malicious secure MPC protocol may have inefficient communication, and so does not subsume our result.

consisting of a player name i, the $\Pi_{N'}$ transcript $\mathbb{T}_{\Pi_{N'}}^{(N'-2)}$ until round $n-1$, and P_i's next message $m_i^{(N'-1)}$ is in $\mathcal{L}_{\mathsf{SNARK},N'}$ if there exists a witness

$$(x, r)$$

such that

$$m_i^{(k)} = \Pi_{N',i}\left(1^\lambda, x_i, r_i, \mathbb{T}_{\Pi_{N'}}^{(k-1)}\right)$$

for $k = 1, \ldots, N'-1$, where $m_i^{(k)}$ is P_i's k'th message in $\mathbb{T}_{\Pi_{N'}}^{(N'-2)}$ for $k < n$ and $\mathbb{T}_{\Pi_{N'}}^{(k-1)}$ denotes the portion of the $\mathbb{T}_{\Pi_{N'}}^{(N'-2)}$ through round $k-1$.

The NP languages $\mathcal{L}_{\mathsf{SNARK},N'}$ is defined as follows: a statement

$$\left(i, \mathbb{T}_{\Pi_{N'}}^{(N'-1)}, m_i^{(N')}, \pi_i\right)$$

consisting of a player name i, the $\Pi_{N'}$ transcript $\mathbb{T}_{\Pi_{N'}}^{(N'-1)}$ through round $N'-1$, P_i's next message $m_i^{(N'-1)}$, and i's zkSNARK proof from round $N'-1$ is in $\mathcal{L}_{\mathsf{SNARK},N'}$ if there exists a witness

$$(x, r, r_\pi)$$

such that

$$m_i^{(N')} = \Pi_{N',i}\left(1^\lambda, x, r, \mathbb{T}_{\Pi_{N'}}^{(N'-1)}\right)$$
$$\pi_i = \mathsf{SNARKProve}\left(\mathsf{crs}, \left(i, \mathbb{T}_{\Pi_{N'}}^{(N'-1)}, m_i^{(N')}, \pi_i\right), (x, w); r_\pi\right)$$

6.2 CRS Generation Functionality

Let CRSGen be a 2 round semi-malicious MPC protocol securely implementing the crs generation functionality. This functionality samples a $\mathsf{crs} \leftarrow (\mathsf{LFEGen}(1^\lambda), \mathsf{SNARKGen}(1^\lambda), \mathsf{HashGen}(1^\lambda))$ and sends it to all parties. To minimize notational clutter, we provide the same concatenated crs to all protocols and let them parse their portion of the crs internally.

6.3 MCDS Functionality

The MCDS functionality for the 4 round generic delayed-input MPC with malicious security takes in a public statement x and a witness $w = (w_1, \ldots, w_M)$. It outputs $(\mathsf{sk}_1, \ldots, \mathsf{sk}_M)$ to all parties if $(x, w) \in \mathcal{R}_{\mathcal{L}_{\mathsf{MCDS}}}$ and $\perp$ otherwise. We denote the 4 round MPC which securely implements this functionality as MCDS. MCDS is required to have the following properties: delayed-input, security against malicious adversaries, its simulator must extract the adversarial inputs, and its simulator must query the ideal functionality with the extracted inputs before the 4th round. The protocol from [12] satisfies these requirements.

The NP language $\mathcal{L}_{\mathsf{MCDS}}$ for the MCDS is defined in terms of M NP languages $\mathcal{L}_{\mathsf{MCDS},i}$: a tuple

$$x = \left(c_j^{(\mathsf{crs},1)}, c_j^{(\mathsf{crs},2)}\right)_{j=1,\ldots,M}$$

is in the language if $x \in \mathcal{L}_{\mathsf{MCDS},i}$ for $i = 1, \ldots, M$. A statement x of this form is in $\mathcal{L}_{\mathsf{MCDS},i}$ if there exists a witness

$$\left(r_i^{(\mathsf{crs})}, \mathsf{sk}_i^{(\mathsf{crs})}\right)$$

such that

$$m_i^{(\mathsf{crs},1)} \leftarrow \overline{\mathsf{Dec}}\left(sk_i^{(\mathsf{crs})}, c_i^{(\mathsf{crs},1)}\right)$$
$$m_i^{(\mathsf{crs},1)} = \mathsf{CRSGen}_i\left(1^\lambda, \perp, r_i^{(\mathsf{crs})}, \emptyset\right)$$
$$c_i^{(\mathsf{crs},2)} = \mathsf{Eval}\left(\left(\mathsf{pk}_1, \ldots, \mathsf{pk}_M\right), \mathsf{CRSGen}_i\left(1^\lambda, \perp, r_i^{(\mathsf{crs})}, \cdot\right), \left(c_1^{(\mathsf{crs},1)}, \ldots, c_M^{(\mathsf{crs},1)}\right)\right)$$

Here $\overline{\mathsf{Dec}}$ is the decryption algorithm in a plain model MK-FHE.

Informally, when using $\mathcal{L}_{\mathsf{MCDS}}$ in the MCDS, each party will provide part of the witness. The witness is accepting if every party submits the randomness used in CRSGen and the private keys used to encrypt CRSGen.

6.4 Additional $\Pi_{\mathsf{pl},N'}$ Properties

We divide the protocol $\Pi_{\mathsf{pl},N'}$ into several protocols, then observe that under this division it satisfies the notion of semi-honest robustness [2]. Informally, even if the adversary behaves maliciously in the first $N'-1$ rounds, it receives no information about the honest party's inputs as long as it behaves semi-honestly in the last round. In our particular case, we will only require semi-honest behavior from the adversary in *part* of the last round. Robustness allows us to delay proving honesty of an execution until the last two rounds, giving time to generate the crs for the zero knowledge SNARKs.

Definition 27 (Robust MPC). *Let F be an M-party functionality. Let $\mathcal{A} = (\mathcal{A}^1, \mathcal{A}^2)$ represent a PPT algorithm controlling all parties except P_i. For a t–round protocol computing F, we let $\mathsf{RealExec}^{\mathcal{A}^1}_{(t-1)}(\mathbf{x}, z)$ denote the view of $\mathcal{A}^1$ during the first $t-1$ rounds in the real execution of the protocol on input $\mathbf{x} = (x_1, \ldots, x_M)$ and auxiliary input z. We require that at the end of the first $t-1$ rounds in the real protocol, $\mathcal{A}^1$ outputs state and $(\mathsf{inp}, \mathsf{rand})$ on a special tape where either $(\mathsf{inp}, \mathsf{rand}) = (\bot, \bot)$ (if $\mathcal{A}^1$ behave maliciously) or $(\mathsf{inp}, \mathsf{rand}) = (\{\hat{x}_j\}_{j\neq i}, \{\hat{r}_j\}_{j\neq i})$ which is consistent with honest behavior for $\mathsf{RealExec}_{(t-1)}$ (first $t-1$ rounds).*

*A protocol $\Pi = (\Pi^1, \Pi^2)$ is said to be a "**robust**" secure multiparty computation protocol for F if for every PPT adversary $\mathcal{A} = (\mathcal{A}^1, \mathcal{A}^2)$ controlling all parties except P_i where $\mathcal{A}^1$ interacts in Π^1 and $\mathcal{A}^2$ interacts semi-honestly in Π^2, there exists a PPT simulator $\mathsf{Sim} = (\mathsf{Sim}^1, \mathsf{Sim}^2)$ such that for every initial input vector $\mathbf{x}$ and every auxiliary input z*

- *If $(\mathsf{inp}, \mathsf{rand}) \neq (\bot, \bot)$, then:*

$$\Big(\mathsf{RealExec}^{\mathcal{A}^1}(\mathbf{x}, z), \mathsf{RealExec}^{\mathcal{A}^1}(\mathbf{x}, \mathsf{state})\Big) \overset{c}{\approx} \Big(\mathsf{RealExec}^{\mathcal{A}^1}(\mathbf{x}, z), \mathsf{Sim}^2(\{\hat{x}_j\}_{j\neq i}, \{\hat{r}_j\}_{j\neq i}, y, \mathsf{state})\Big)$$
$$\overset{c}{\approx} \big(\mathsf{Sim}^1(z), \mathsf{Sim}^2(\{\hat{x}_j\}_{j\neq i}, \{\hat{r}_j\}_{j\neq i}, y, \mathsf{state})\big)$$

  Here $y = F(\hat{x}_1, \ldots, \hat{x}_M)$, where $\hat{x}_i = x_i$, and $\mathsf{RealExec}^{\mathcal{A}^2}(\mathbf{x}, \mathsf{state})$ is the view of adversary $\mathcal{A}^2$.
- *Else,*

$$\mathsf{RealExec}^{\mathcal{A}^1}(\mathbf{x}, z) \overset{c}{\approx} \mathsf{Sim}^1(z)$$

We say Π is **rewinding-robust** if this holds even when the adversary may rewind the honest party (resp. simulator) in rounds 2 through $t-1$ and select new messages a polynomial number of times.

The definition here differs from the original in a few ways. First, we restrict ourselves to the case where the adversary controls all but one party. This will be sufficient for proving security of our malicious construction against an adversary which controls all but one party, which Lemma 1 shows is without loss of generality. Second, we allow Π^1 to extend into the last round, rather than restricting it to round $t-1$ and earlier as in the original definition.

Protocol Division We divide $\Pi_{\mathsf{pl},N'}$ into $\left(\left(\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}, \Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}\right), \Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}\right)$ both by last round and by usage of the crs. The further division of Π^1 by crs usage will be useful in our protocol description later.

$\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}$ consists of all messages in round $N'-1$ requiring a crs. If $N' = N+1$ this consists of the LFE hash h_i. For $N' = N$ this consists of ζ_i.

$\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}$ consists of all messages in round N' requiring a crs. If $N' = N+1$ this consists of the LFE ciphertexts $(\tilde{c}_{i,j})_{j\neq i}$. For $N' = N$ this consists of the LFE hash h_i, the SNARK π_i, the garbled circuit $(C_{i,j})_{j\neq i}$, and the witness encrypted labels $(v_{i,j})_{j\neq i}$.

$\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}$ consists of all other messages, i.e. the ciphertext $\hat{c}^1_i$ in round 1, the ciphertexts and public key $(\bar{c}^{(2)}_i, \mathsf{pk}_i, \overline{c^{\mathsf{sk}}_i})$ in round 2, and the ciphertext $c^{(n)}_i$ for all subsequent rounds n.

Robustness Overview. To see why this division satisfies semi-honest rewinding-robustness, consider the case where $\Pi_{\mathsf{pl},N'-1}$ is executed in the common reference string model, so that crs is provided by a trusted setup instead of by P_1. This will still be useful for us because the crs is only required in the last two rounds[13], giving us time to generate it in the final malicious-secure protocol. First observe that hybrid $2M-2$ in the proof of Theorem 5 gives a simulator for $\Pi^{\mathsf{crs},N+1}_{\mathsf{pl},N+1}$ satisfying the first equation. Plugging this into hybrid $2M-1$ from the proof of Theorem 6 gives a simulator for $\Pi^{\mathsf{crs},N}_{\mathsf{pl},N}$. The changes made to move to $\Pi_{\mathsf{pl},N'}$ in the common reference string model do not change these simulators. There are no rewinding issues because round N' is not permitted to be rewound for rewinding-robustness.

Next, observe that the full simulator for $\Pi_{\mathsf{pl},N'}$ does not require the adversary's input or randomness until the last round, and so if the adversary behaves semi-maliciously, the second equation holds. This fact requires the use of the semi-malicious security of the distributed setup procedure for the rate-1 MK-FHE in Appendix 7.4. To show this is the case even when the middle rounds may be rewound, first recall that this holds even if the MK-FHE ciphertexts including the parties' inputs are included in the view of the adversary. Given P_i's input ciphertexts, the MK-FHE ciphertexts for rounds after 2 can be computed locally. Since round 1 is never rewound, P_i's round 2 message can be replayed during any rewinds. The only remaining components which can be rewound are the collision resistant hash ζ_i (for $\Pi_{\mathsf{pl},N}$) and the LFE hash (for $\Pi_{\mathsf{pl},N+1}$). Similarly to the MK-FHE ciphertexts after round 2, ζ_i can be computed locally given the rate-1 MK-FHE ciphertext of P_i's input. Finally, the LFE hash's function hiding experiment is non-interactive, and so each rewinding can be treated as a separate attempt at the experiment. Using these observations, we can show security of this view against a rewinding semi-malicious adversary in two hybrids. In the first hybrid, $\Pi_{\mathsf{pl},N'}$ is simulated *except* for P_i's LFE hash. Security against rewinding holds from the first set of observations about replaying the round 2 message and locally computing the others. In the second hybrid, the LFE hash is simulated. Security against rewinding holds from the observation about the LFE hash's non-interactivity.

For the third equation, observe that the series of hybrids in the proof of Theorem 7 holds even if $\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}$ is not included in the adversary's view, and against malicious adversaries in the crs model. LFE function hiding (used implicitly in hybrid 1) holds even against malicious adversaries due to its non-interactive nature. We can rely on the semantic security of our rate-1 MK-FHE against a malicious adversary for the distributed setup procedure by relying on the semi-malicious security of the distributed setup procedure and the fact that our rate-1 MK-FHE has efficiently checkable public parameters, allowing parties to detect malicious behavior and abort if necessary. PRG security and the plain model MK-FHE semantic security are non-interactive and so unaffected by malicious adversaries. Security against rewinding follows from similar reasoning as for the second equation.

6.5 Construction

In summary we use the following components:

- A simulation-extractable zero knowledge SNARK $(\mathsf{SNARKGen}_{N'-1}, \mathsf{SNARKProve}_{N'-1}, \mathsf{SNARKVerify}_{N'-1})$ for the language $\mathcal{L}_{\mathsf{SNARK},N'-1}$.
- A simulation-extractable zero knowledge SNARK $(\mathsf{SNARKGen}_{N'}, \mathsf{SNARKProve}_{N'}, \mathsf{SNARKVerify}_{N'})$ for the language $\mathcal{L}_{\mathsf{SNARK},N'}$.
- A plain model MK-FHE scheme $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Eval}}, \overline{\mathsf{Dec}})$.
- A 2 round semi-malicious MPC CRSGen implementing the crs generation functionality.
- A 4 round delayed-input malicious MPC MCDS implementing the MCDS functionality.
- Our N' round interaction preserving protocol $\Pi_{\mathsf{pl},N'} = \left(\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}, \Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}, \Pi^{\mathsf{pl}}_{\mathsf{pl},N'}\right)$.

The only rounds modified are $N'-3$, $N'-2$, $N'-1$, and N', where N' is the number of rounds in the semi-honest secure protocol being compiled (i.e. $N+1$ for Π_{N+1} and N for Π_N). The other rounds are unmodified and their descriptions are omitted here.

The total round complexity is $\max(4, N')$, and insecure protocols with $N \geq 3$ rounds can be compiled.

[13] It is replaced in the rate-1 MK-FHE by a distributed setup procedure during the first round.

Let $\mathbb{T}^{(N'-4)}_{\Pi_{\mathsf{pl},N'}}$ be the transcript of $\Pi_{\mathsf{pl},N'}$ so far.

Round $N'-3$: Party P_i proceeds as follows:

- Compute the message from the underlying protocol $m_i^{(\mathsf{pl},N'-3)} \leftarrow \Pi^{\mathsf{pl}}_{\mathsf{pl},N'}(1^\lambda, x_i, r_i, \mathbb{T}^{(N'-4)}_{\Pi_{\mathsf{pl},N'}})$
- Sample a fresh plain model MK-FHE key $(\overline{\mathsf{sk}}_i^{(\mathsf{crs})}, \overline{\mathsf{pk}}_i^{(\mathsf{crs})}) \leftarrow \overline{\mathsf{KeyGen}}(1^\lambda)$
- Compute the first message of the crs generation $m_i^{(\mathsf{crs},1)} \leftarrow \mathsf{CRSGen}_i(1^\lambda, \perp, r_i^{(\mathsf{crs})}, \emptyset)$ and encrypt it as $c_i^{(\mathsf{crs},1)} \leftarrow \mathsf{Enc}\left(\overline{\mathsf{pk}}_i^{(\mathsf{crs})}, m_i^{(\mathsf{crs},1)}\right)$.
- Using fresh randomness $r_i^{(\mathsf{MCDS})}$, compute the first message of the MCDS

$$m_i^{(\mathsf{mcds},1)} \leftarrow \mathsf{MCDS}_i(1^\lambda, \perp, r_i^{(\mathsf{MCDS})}, \emptyset)$$

- Broadcast $\left(m_i^{(\mathsf{pl},N'-3)}, \overline{\mathsf{pk}}_i^{(\mathsf{crs})}, c_i^{(\mathsf{crs},1)}, m_i^{(\mathsf{mcds},1)}\right)$

Round $N'-2$: Upon receiving the messages from round $N'-3$, party i does the following:

- Compute the message from the underlying protocol $m_i^{(\mathsf{pl},N'-2)} \leftarrow \Pi^{\mathsf{pl}}_{\mathsf{pl},N'}(1^\lambda, x_i, r_i, \mathbb{T}^{(N'-3)}_{\Pi_{\mathsf{pl},N'}})$ where $\mathbb{T}^{(N'-3)}_{\Pi_{\mathsf{pl},N'}} = \mathbb{T}^{(N'-4)}_{\Pi_{\mathsf{pl},N'}} || (m_1^{(\mathsf{pl},N'-3)}, \ldots, m_M^{(\mathsf{pl},N'-3)})$.
- Compute the encrypted second message of the crs generation

$$c_i^{(\mathsf{crs},2)} \leftarrow \mathsf{Eval}\left(\left(\overline{\mathsf{pk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{pk}}_M^{(\mathsf{crs})}\right), \mathsf{CRSGen}_i\left(1^\lambda, \perp, r_i^{(\mathsf{crs})}, \cdot\right), \left(c_1^{(\mathsf{crs},1)}, \ldots, c_M^{(\mathsf{crs},1)}\right)\right)$$

- Let $\mathbb{T}^{(1)}_{\mathsf{MCDS}}$ be the MCDS transcript so far. Compute the second message of the MCDS

$$m_i^{(\mathsf{mcds},2)} \leftarrow \mathsf{MCDS}_i\left(1^\lambda, \perp, r_i^{(\mathsf{MCDS})}, \mathbb{T}^{(1)}_{\mathsf{MCDS}}\right)$$

- Broadcast $\left(m_i^{(\mathsf{pl},N'-2)}, c_i^{(\mathsf{crs},2)}, m_i^{(\mathsf{mcds},2)}\right)$

Round $N'-1$: Upon receiving the messages from round $N'-2$, party i does the following:

- Compute the message from the underlying protocol $m_i^{(\mathsf{pl},N'-1)} \leftarrow \Pi^{\mathsf{pl}}_{\mathsf{pl},N'}(1^\lambda, x_i, r_i, \mathbb{T}^{(N'-2)}_{\Pi_{\mathsf{pl},N'}})$ where $\mathbb{T}^{(N'-2)}_{\Pi_{\mathsf{pl},N'}} = \mathbb{T}^{(N'-3)}_{\Pi_{\mathsf{pl},N'}} || (m_1^{(\mathsf{pl},N'-2)}, \ldots, m_M^{(\mathsf{pl},N'-2)})$.
- Compute the encrypted crs

$$c_i^{(\mathsf{crs})} \leftarrow \mathsf{Eval}\left(\left(\overline{\mathsf{pk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{pk}}_M^{(\mathsf{crs})}\right), \mathsf{CRSGenOutput}\left(1^\lambda, \perp, r_i^{(\mathsf{crs})}, \cdot\right), \left(c_1^{(\mathsf{crs},1)}, \ldots, c_M^{(\mathsf{crs},2)}\right)\right)$$

- Compute the encrypted message from the underlying protocol requiring the encrypted crs

$$c_i^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}} \leftarrow \mathsf{Eval}\left(\left(\overline{\mathsf{pk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{pk}}_M^{(\mathsf{crs})}\right), \Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}(1^\lambda, x_i, r_i, \cdot), c_i^{(\mathbb{T},N'-2)}\right)$$

  where $c_i^{(\mathbb{T},N'-2)} = c_i^{(\mathsf{crs})} || \mathsf{Enc}(\mathsf{pk}_i, ^{(\mathsf{crs})}, \mathbb{T}^{(N'-2)}_{\Pi_{\mathsf{pl},N'}})$.
- Compute its encrypted $\Pi_{\mathsf{pl},N'}$ input and randomness $c_i^{(x,r)} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}_i, (x_i, r_i))$
- Encrypt its next $\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}$ message as $c_i^{\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}} \leftarrow \overline{\mathsf{Enc}}\left(\overline{\mathsf{pk}}_i, m_i^{(\mathsf{pl},N'-1)}\right)$
- Compute the encrypted SNARK proof

$$c_i^{(\pi,N'-1)} \leftarrow \mathsf{Eval}\left(\left(\overline{\mathsf{pk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{pk}}_M^{(\mathsf{crs})}\right), \mathsf{SNARKProve}(\cdot,\cdot,\cdot; r_i^{\mathsf{SNARK}}), \left(c_i^{(\mathsf{crs})}, \left(c_i^{(\mathbb{T},N'-2)}, \left(c_i^{\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}}, c_i^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}\right)\right), c_i^{(x,r)}\right)\right)$$

  using fresh randomness r_i^{SNARK}

– Let $\mathbb{T}^{(2)}_{\mathsf{MCDS}}$ be the MCDS transcript so far. Compute the third message of the MCDS

$$m_i^{(\mathsf{mcds},3)} \leftarrow \mathsf{MCDS}_i\left(1^\lambda, \left(r_i^{(\mathsf{crs})}, \overline{\mathsf{sk}}_i^{(\mathsf{crs})}\right), r_i^{(\mathsf{MCDS})}, \mathbb{T}^{(2)}_{\mathsf{MCDS}}\right)$$

– Broadcast $(m_i^{(\mathsf{pl},N'-1)}, c_i^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}, c_i^{(\pi,N'-1)}, m_i^{(\mathsf{mcds},3)})$.

Round N': Upon receiving the messages from round $N'-1$, party i does the following:

– Compute the message from the underlying protocol

$$m_i^{(\mathsf{pl},N')} \leftarrow \Pi^{\mathsf{pl}}_{\mathsf{pl},N'}(1^\lambda, x_i, r_i, \mathbb{T}^{(N'-1)}_{\Pi_{\mathsf{pl},N'}})$$

where $\mathbb{T}^{(N'-1)}_{\Pi_{\mathsf{pl},N'}} = \mathbb{T}^{(N'-2)}_{\Pi_{\mathsf{pl},N'}} || (m_1^{(\mathsf{pl},N'-1)}, \ldots, m_M^{(\mathsf{pl},N'-1)})$.

– Let $C_i^{\Pi}[1^\lambda]$ be the following circuit. It takes as input a common reference string crs, $M-1$ proofs $(\pi_j)_{j\neq i}$, a transcript $\mathbb{T}^{(N'-1)}$ consisting of $N'-1$ rounds, and some input and randomness (x_i, r_i). If $\mathsf{SNARKVerify}_{N'-1}(\mathsf{crs}, \pi_j, (\mathbb{T}^{(N'-2)}, m_j^{(N'-1)})) = 1$ for all $j \neq i$, where $m_j^{(N'-1)}$ is P_j's message in the transcript $\mathbb{T}^{(N'-1)}$ and $\mathbb{T}^{(N'-2)}$ is the first $N'-2$ rounds of $\mathbb{T}^{(N'-1)}$, output $\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}(1^\lambda, x_i, r_i, \mathbb{T}^{(N'-1)})$. Otherwise output $\perp$.

– Compute the encrypted message from the underlying protocol requiring the encrypted crs

$$c_i^{\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}} \leftarrow \mathsf{Eval}\left(\left(\overline{\mathsf{pk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{pk}}_M^{(\mathsf{crs})}\right), C_i^{\Pi}[1^\lambda](\cdot,\cdot,\cdot,\cdot), \left(c_i^{(\mathsf{crs})}, (c_j^{(\pi,N'-1)})_{j\neq i}, c_i^{(\mathbb{T},N'-1)}, c_i^{(x,r)}\right)\right)$$

where $c_i^{(\mathbb{T},N'-1)} = c_i^{(\mathsf{crs})} || \left(c_1^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}, \ldots, c_M^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}\right) || \mathsf{Enc}(\mathsf{pk}_i, {}^{(\mathsf{crs})}, \mathbb{T}^{(N'-1)}_{\Pi_{\mathsf{pl},N'}})$.

– Encrypt its next $\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}$ message as $c_i^{\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}} \leftarrow \overline{\mathsf{Enc}}\left(\overline{\mathsf{pk}}_i, m_i^{(\mathsf{pl},N')}\right)$

– Let $\mathsf{statement}_i^{N'} = \left(c_i^{(\mathbb{T},N'-2)}, \left(c_i^{\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}}, c_i^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}, c_i^{(\pi,N'-1)}\right)\right)$.
Compute the encrypted SNARK proof

$$c_i^{(\pi,N')} \leftarrow \mathsf{Eval}\left(\left(\overline{\mathsf{pk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{pk}}_M^{(\mathsf{crs})}\right), \mathsf{SNARKProve}(\cdot,\cdot,\cdot), \left(c_i^{(\mathsf{crs})}, \mathsf{statement}_i^{N'}, \left(c_i^{(x,r)}, r_i^{\mathsf{SNARK}}\right)\right)\right)$$

– Let $\mathbb{T}^{(3)}_{\mathsf{MCDS}}$ be the MCDS transcript so far. Compute the fourth message of the MCDS

$$m_i^{(\mathsf{mcds},4)} \leftarrow \mathsf{MCDS}_i\left(1^\lambda, \left(r_i^{(\mathsf{crs})}, \overline{\mathsf{sk}}_i^{(\mathsf{crs})}\right), r_i^{(\mathsf{MCDS})}, \mathbb{T}^{(3)}_{\mathsf{MCDS}}\right)$$

– Broadcast $(m_i^{(\mathsf{pl},N')}, c_i^{\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}}, c_i^{(\pi,N')}, m_i^{(\mathsf{mcds},4)})$.

Output: Upon receiving the messages from round N', party i does the following:

– Let $\mathbb{T}_{\mathsf{MCDS}}$ be the full MCDS transcript. Decode the MCDS output

$$\mathsf{out}_{\mathsf{MCDS}} \leftarrow \mathsf{MCDSOutput}\left(1^\lambda, \left(r_i^{(\mathsf{crs})}, \overline{\mathsf{sk}}_i^{(\mathsf{crs})}\right), r_i^{(\mathsf{MCDS})}, \mathbb{T}_{\mathsf{MCDS}}\right)$$

– If $\mathsf{out}_{\mathsf{MCDS}} = \perp$, output $\perp$. Otherwise, parse $\mathsf{out}_{\mathsf{MCDS}} = \overline{\mathsf{sk}}_1, \ldots, \overline{\mathsf{sk}}_M$.

– Decrypt $\mathsf{crs} \leftarrow \overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}_1^{(\mathsf{crs})}, \ldots, \overline{\mathsf{sk}}_M^{(\mathsf{crs})}\right), c_i^{(\mathsf{crs})}\right)$

– Decrypt the portions of the $\Pi_{\mathsf{pl},N'}$ transcript requiring the crs

$$\mathbb{T}_{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}} \leftarrow \overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}_1^{(\mathsf{crs})},\ldots,\overline{\mathsf{sk}}_M^{(\mathsf{crs})}\right),\left(c_1^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}},\ldots,c_M^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}\right)\right)$$
$$\mathbb{T}_{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}} \leftarrow \overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}_1^{(\mathsf{crs})},\ldots,\overline{\mathsf{sk}}_M^{(\mathsf{crs})}\right),\left(c_1^{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}},\ldots,c_M^{\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}}\right)\right)$$

Denote the full $\Pi_{\mathsf{pl},N'}$ transcript as $\mathbb{T}_{\Pi_{\mathsf{pl},N'}} = \left(\mathbb{T}_{\Pi_{\mathsf{pl},N'}}, \mathbb{T}_{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}, \mathbb{T}_{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}\right)$.

– For each $j \neq i$, decrypt the SNARKs

$$\pi_i^{N'-1} \leftarrow \overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}_1^{(\mathsf{crs})},\ldots,\overline{\mathsf{sk}}_M^{(\mathsf{crs})}\right),c_j^{(\pi,N'-1)}\right)$$
$$\pi_i^{N'} \leftarrow \overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}_1^{(\mathsf{crs})},\ldots,\overline{\mathsf{sk}}_M^{(\mathsf{crs})}\right),c_j^{(\pi,N')}\right)$$

– If $\mathsf{SNARKVerify}_{N'}\left(\mathsf{crs},\left(\mathbb{T}^{(N'-1)}_{\Pi_{\mathsf{pl},N'}},m_j^{(N')},\pi_i^{N'-1}\right),\pi_i^{N'}\right) = 0$ for some j, output $\perp$.

– Otherwise, output the $\Pi_{\mathsf{pl},N'}$ output

$$\mathsf{Output}_i^{\Pi_{\mathsf{pl},N'}}\left(1^\lambda, x_i, r_i, \left(\mathbb{T}_{\Pi^{\mathsf{pl}}_{\mathsf{pl},N'}}, \mathbb{T}_{\Pi^{\mathsf{crs},N'-1}_{\mathsf{pl},N'}}, \mathbb{T}_{\Pi^{\mathsf{crs},N'}_{\mathsf{pl},N'}}\right)\right)$$

Communication. In addition to the $\Pi_{\mathsf{pl},N'}$ transcript, the parties also send an additional MK-FHE key pair, two additional (encrypted) simulation extractable SNARKs, the encrypted (inefficient) 2 round semi-malicious MPC CRSGen, and the (inefficient) 4 round delayed-input malicious MPC MCDS. Furthermore, the messages in $\Pi_{\mathsf{pl},N'}$ requiring crs are encrypted under a (potentially inefficient) MK-FHE without setup. For $\Pi_{\mathsf{pl},N+1}$ these are only the LFE hash and ciphertext, and for $\Pi_{\mathsf{pl},N}$ these are the collision-resistant hash, the LFE hash, an additional SNARK, garbled circuits computing LFE ciphertexts, and the witness encrypted labels for the garbled circuits.

First, we discuss the new messages. The MK-FHE key pair has size $\mathsf{poly}(\lambda)$. The simulation extractable SNARKs are succinct and so have size $\mathsf{poly}(\lambda)$. CRSGen has size $|\mathsf{crs}|\mathsf{poly}(\lambda)$, since it samples a $\mathsf{crs} \leftarrow (\mathsf{LFEGen}(1^\lambda), \mathsf{SNARKGen}(1^\lambda), \mathsf{HashGen}(1^\lambda))$. The SNARK and collision resistant hash functions require only $\mathsf{poly}(\lambda)$-sized common reference strings. The LFE common reference string has size $\mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)$. Finally, MCDS takes in the encrypted CRSGen transcript, the randomness used in the CRSGen protocol, and M plain model MK-FHE secret keys, then uses these to verify that the encrypted CRSGen transcript was generated according to that randomness and encrypted with those keys. Since CRSGen takes communication $\mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)\mathsf{poly}(\lambda)$, this means MCDS takes communication $\mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda)\mathsf{poly}(\lambda)$.

Next we discuss the messages from $\Pi_{\mathsf{pl},N+1}$ which are encrypted using the potentially inefficient MK-FHE without setup. As mentioned in our previous analysis of Π_{N+1} and Π_N, these messages together have size $\tilde{O}(|m_N| + |\mathsf{out}|)\mathsf{poly}(|\mathsf{depth}|, \lambda)$ for Π_{N+1} and $\tilde{O}(|m_N| + |\mathsf{out}|)\mathsf{poly}(|\mathsf{depth}|, \lambda) + \mathsf{poly}(|m_{N-1}|, |m_N|, M, \lambda)$ forΠ_N. Encrypting these under the potentially inefficient MK-FHE increases the communication by at most a multiplicative $\mathsf{poly}(\lambda)$.

Overall the communication complexity of $\Pi_{\mathsf{mal},N+1}$ is

$$\sum_{i=1}^{N} |m_i|(1+o(1)) + (NM^2 + \mathsf{LFECC}(|m_N|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + \mathsf{LFECRS}(|\mathsf{last}|, |\mathsf{depth}|, |\mathsf{out}|, \lambda) + |m_2|)\mathsf{poly}(\lambda)$$

The communication complexity of $\Pi_{\mathsf{mal},N+1}$ is

$$\sum_{i=1}^{N}|m_i|(1+o(1))+(NM^2+\mathsf{LFECS}(|m_N|,|\mathsf{depth}|,|\mathsf{out}|,\lambda)+\mathsf{LFECRS}(|\mathsf{last}|,|\mathsf{depth}|,|\mathsf{out}|,\lambda)$$
$$+\mathsf{WEncCC}\left(\mathsf{LFECRS}\left(|m_N|,|\mathsf{depth}|,|\mathsf{out}|,\lambda\right)+\mathsf{poly}\left(\lambda\right)+|m_N|,|m_{N-1}|+M\mathsf{poly}\left(\lambda\right),|m_N|+M\mathsf{poly}\left(\lambda\right),\lambda\right)+|m_2|\right)\mathsf{poly}\left(\lambda\right)$$

Instantiating the LFE with the scheme of Quach et. al. [43], we have $\mathsf{LFECRS}(|\mathsf{last}|,|\mathsf{depth}|,|\mathsf{out}|,\lambda)=|m_{N-1}|\mathsf{poly}(|\mathsf{depth}|,\lambda)$.

Correctness. To establish correctness, first note that for the MCDS, all parties provide a valid statement and witness. Therefore by the correctness of the MCDS, at the end of the protocol all parties are in possession of $\overline{\mathsf{sk}}_i$ for $i=1,\ldots,M$ and by the correctness of the plain model MK-FHE can successfully decrypt the following: $\mathsf{crs},\mathbb{T}_{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}},\mathbb{T}_{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}},\pi_i^{N'-1},\pi_i^{N'}$. By correctness of CRSGen, crs is sampled from $(\mathsf{LFEGen}(1^\lambda),\mathsf{SNARKGen}(1^\lambda),\mathsf{HashGen}(1^\lambda))$, and so all parties provide a valid witness and statement for the SNARKS. Since this is the case, $\mathsf{SNARKVerify}(\mathsf{crs},\mathsf{statement},\pi_i^{N'-1})=1$ by SNARK correctness. Therefore $\mathbb{T}_{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}}$ contains the transcript of $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}$, rather than $\perp$ (which would be the case if the round $N'-1$ SNARKS had been rejecting). Finally, $\mathsf{SNARKVerify}(\mathsf{crs},\mathsf{statement},\pi_i^{N'})=1$ by SNARK correctness. Therefore all parties output $\mathsf{Output}_i^{\Pi_{\mathsf{pl},N'}}\left(1^\lambda,x_i,r_i,\left(\mathbb{T}_{\Pi_{\mathsf{pl},N'}^{\mathsf{pl}}},\mathbb{T}_{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}},\mathbb{T}_{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}}\right)\right)$, which is correct by the correctness of $\Pi_{\mathsf{pl},N'}$.

Security. We prove the malicious security of our protocol below.

Theorem 8. *Assume that* $\Pi_{\mathsf{pl},N'}=\left(\left(\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1},\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}\right),\Pi_{\mathsf{pl},N'}^{\mathsf{pl}}\right)$ *is a semi-honest rewinding-robust MPC protocol,* $(\mathsf{SNARKGen}_{N'-1},\mathsf{SNARKProve}_{N'-1},\mathsf{SNARKVerify}_{N'-1})$ *and* $(\mathsf{SNARKGen}_{N'},\mathsf{SNARKProve}_{N'},\mathsf{SNARKVerify}_{N'})$ *are simulation-extractable zero knowledge succinct non-interactive argument systems of knowledge,* CRSGen *is a 2 round semi-malicious MPC protocol, and* MCDS *is a 4 round delayed-input malicious MPC protocol. Then* $\Pi_{\mathsf{mal},N'}$ *is a malicious MPC protocol.*

Proof. Let P_i be the honest party. We modify the protocol through a series of hybrids, which are defined below.

H0: This is the real experiment.

H1: In this hybrid, the hybrid simulator invokes $\mathsf{Sim}_{\mathsf{MCDS}}$, acting as the ideal functionality. Note that whenever the ideal functionality is required to output $(\mathsf{sk}_1,\ldots,\mathsf{sk}_M)$, it receives as input valid $\mathsf{sk}_1,\ldots,\mathsf{sk}_M$ due to the definition of the functionality and $\mathcal{L}_{\mathsf{MCDS}}$.
Indistinguishability between hybrids 0 and 1 follows from the simulation security of MCDS. Since the honest output can still be decoded from the transcript in both hybrids, we have indistingishability of the joint distribution of honest output and adversarial view.

H2: In this hybrid, the hybrid simulator creates a series of lookahead threads to extract the adversary's CRSGen randomness $r_j^{(\mathsf{crs})}$ before the final simulation of MCDS. Specifically, it runs $\mathsf{Sim}_{\mathsf{MCDS}}$ until it receives an ideal functionality query from $\mathsf{Sim}_{\mathsf{MCDS}}$ containing the extracted adversarial input in round $N'-1$, rewinds $\mathsf{Sim}_{\mathsf{MCDS}}$ to round $N'-2$, and plays $\mathsf{Sim}_{\mathsf{MCDS}}$ again. If the extracted input $(x,w)\notin\mathcal{R}_{\mathcal{L}_{\mathsf{MCDS}}}$ during the first simulation, it finishes the initial simulation with the message from round N' and does not rewind. Otherwise, to maintain the same probability of an implicit abort, in the subsequent invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$, $\mathsf{Sim}_{\mathsf{MCDS}}$ is repeated until the extracted input from the most recent round $(x,w)\in\mathcal{R}_{\mathcal{L}_{\mathsf{MCDS}}}$. Note that the *only* information kept between the two invocations is the extracted $r_j^{(\mathsf{crs})}$, which is the same in both rewinds, since it is fixed in round $N'-3$.
Indistinguishability between hybrids 1 and 2 follows from the observation that the expected number of rewinds is polynomial.

H3: In this hybrid, the hybrid simulator replaces all ciphertexts encrypted using $\overline{\mathsf{pk}}_i$ after round $N'-3$ with an encryption of 0 during the first invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$. Specifically, these are $c_i^{(\mathsf{crs},2)}$, $c_i^{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}}$, $c_i^{(\pi,N'-1)}$, $c_i^{\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}}$, and $c_i^{(\pi,N')}$.
Indistinguishability between hybrids 2 and 3 reduces to the semantic security of the plain model MK-FHE scheme. Conditioned on the second invocation occurring, the distributions are clearly identical. Therefore if an adversary could distinguish between the two hybrids with noticeable advantage, then with noticeable advantage it could distinguish between the two conditioned on the second invocation not occurring, which happens if and only if MCDS would output $\perp$ in the first invocation. Given the ciphertexts either as an encryption of 0 or according to the protocol specification, the reduction plays hybrid 2 using the received ciphertexts until it extracts (x, w). If $(x, w) \notin \mathcal{R}_{\mathcal{L}_{\mathsf{MCDS}}}$, it finishes H2 and outputs the decision of the H2-H3 distinguisher. Otherwise it guesses randomly.

H4: In this hybrid, the hybrid simulator invokes $\mathsf{Sim}_{\mathsf{CRSGen}}$. $\mathsf{Sim}_{\mathsf{CRSGen}}$ gives the first message of CRSGen, then is provided with the extracted adversarial randomness $r_j^{(\mathsf{crs})}$ to simulate the second message during the second invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$. Note that the simulated first message of CRSGen (round $N'-3$) both does not need the adversary's randomness beforehand and is never rewound. Additionally, the simulated second message of CRSGen (round $N'-2$) can be replayed during any rewinds in the second invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$, since it only depends on the first message of CRSGen, which is never rewound. This change could cause the MCDS functionality to output $\perp$ regardless of what the adversary does; to prevent this, the simulator replies to the ideal functionality query for MCDS as if the input were in $\mathcal{L}_{\mathsf{MCDS}_i}$ anyway.
Indistinguishability from the previous hybrid follows from the semi-malicious simulation security of CRSGen and the observation that the simulated CRSGen transcript is not rewound.

H5: In this hybrid, the hybrid simulator simulates the SNARK in round $N'-1$ using $\mathsf{SNARKGenSim}_{N'-1}$ and $\mathsf{SNARKSim}_{N'-1}$. Note that this changes the crs which $\mathsf{Sim}_{\mathsf{CRSGen}}$ must force. $\mathsf{SNARKGenSim}_{N'-1}$ is invoked from the beginning, and $\mathsf{SNARKSim}_{N'-1}$ is only invoked during the second invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$, since $\mathsf{Sim}_{\mathsf{CRSGen}}$ does not force the crs before then.
Indistinguishability from the previous hybrid follows from the zero knowledge property of the SNARK. Simulation-extractability implies that the adversary's round $N'-1$ SNARKs remain sound.

H6: In this hybrid, the hybrid simulator simulates the SNARK in round N'.
Indistinguishability between hybrids from the previous hybrid follows from the zero knowledge property of the SNARK. Simulation-extractability implies that the adversary's round N' SNARKs remain sound.

H7: In this hybrid, the hybrid simulator extracts the adversary's input in round $N'-1$ by decrypting the SNARKs in round $N'-1$ and invoking its extractor, relying on the simulation-extractability of the SNARK. This hybrid is identically distributed to the previous hybrid, but also includes the adversary's extracted input in its view. This will be important in arguing joint indistinguishability of the honest output and adversarial view in the next hybrid, since the next hybrid will not be able to decode the honest party's output from the transcript. Specifically, in second invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$ if the extracted MCDS input (x, w) is valid with respect to the adversarial input, the hybrid simulator uses the adversarial secret keys $\overline{\mathsf{sk}}_j$ to decrypt the adversary's SNARK in round $N'-1$. It then invokes the SNARK extractor to recover the adversary's input to $\Pi_{N'}$.
Indistinguishability from the previous hybrid (including only the view of the adversary, not the extracted inputs) is trivial, since the transcript is not modified. This is the last hybrid in which the honest output can still be decoded from the transcript.

H8: In this hybrid, the hybrid simulator simulates $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$. Specifically, after extracting the adversary's input and the MCDS input (x, w) in round $N'-1$, if (x, w) is valid with respect to the adversarial input, it simulates $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$ under the plain model MK-FHE.
Indistinguishability from the previous hybrid reduces to the robustness of $\Pi_{N'}$. Observe that conditioned on (x, w) being *invalid* with respect to the adversarial input (which is detectable by the reduction), the hybrids are identical, since $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$ is never revealed and does not need to be simulated.
To show that the joint distribution of the adversarial view and honest output are indistinguishable, observe that the adversarial inputs are indistinguishable across hybrids 7 and 8, so the joint distribution including the honest output is similarly indistinguishable.

H9: In this hybrid, the simulator simulates $\left(\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}, \Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'-1}\right)$.

Indistinguishability from the previous hybrid follows in two steps, both following from the rewinding-robustness of $\Pi_{\mathsf{pl},N'}$. First, we show that the adversary's advantage is negligible conditioned on the second $\mathsf{Sim}_{\mathsf{MCDS}}$ occurring and all decrypted round $N'-1$ SNARKs being accepted. Then, we show that the adversary's advantage is negligible conditioned on the complement of these events, i.e. the second $\mathsf{Sim}_{\mathsf{MCDS}}$ does not occur or one of the decrypted round $N'-1$ SNARKs is rejected.

Assume the first is not the case. Then we can reduce to the second rewinding-robustness property of $\Pi_{\mathsf{pl},N'}$ by running H8 using a $\Pi_{\mathsf{pl},N'}$ interactively provided by a challenger. If the simulator finishes the simulation after one invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$, or if $\pi_j^{N'}$ is accepting for all $j \neq i$, the reduction guesses randomly. Otherwise it outputs what the hybrid distinguisher outputs. Since the first claim is not the case, the hybrid distinguisher has noticeable advantage in the case where the reduction outputs what the hybrid distinguisher outputs. When this occurs, the adversary has behaved semi-maliciously in the first round of CRSGen (otherwise the second invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$ would not have occurred), so the $\mathsf{Sim}_{\mathsf{CRSGen}}$ succeeds in forcing the crs. This means the SNARK simulation extractability property holds and so the adversary's semi-honest round N' input was extracted successfully in round $N'-1$. Therefore the transcript corresponds precisely to either H8 or H9, depending on the challenger's behavior.

If the second claim is not the case, we can similarly reduce to the third rewinding-robustness property of $\Pi_{\mathsf{pl},N'}$. The reduction is the same as the previous one, except half of the challenger (controlling $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$) is run underneath the plain model MK-FHE and the cases where the reduction guesses randomly and where it outputs what the hybrid distinguisher outputs are flipped. Note that whether the reduction guesses randomly or not is decided before it has to provide the messages from $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$, which it cannot get from the challenger. If it outputs what the hybrid distinguisher outputs, then $Ci^{\Pi}[1^\lambda]$ outputs[14] $\perp$ instead of the message for $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$ (in the case of a SNARK not verifying). In the case that the simulator would terminate after a single invocation of $\mathsf{Sim}_{\mathsf{MCDS}}$, it locally computes a faulty $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$ message which is never revealed. In either case, $\Pi_{\mathsf{pl},N'}^{\mathsf{crs},N'}$ is not required for the reduction to proceed, so the transcript given to the hybrid distinguisher exactly corresponds to either H8 or H9, depending on the challenger's behavior.

To show that the joint distribution of the adversarial view and honest output are indistinguishable, observe that the adversarial inputs are indistinguishable across hybrids 8 and 9, so the joint distribution including the honest output is similarly indistinguishable.

7 Rate-1 Multi-Key Fully-Homomorphic Encryption

In the following we present our construction for a rate-1 MK-FHE scheme from the LWE problem.

7.1 Rate-1 Multi-Key Linearly Homomorphic Encryption

In the following we present a multi-key linearly homomorphic encryption (MK-HE) scheme for linear function with rate $1 - 1/\lambda$.

Relaxed Correctness. Before going into the formal details of the scheme, we introduce a more fined grained notion of correctness which separates the decryption procedure with the message decoding. This allows us for a more modular treatment of the scheme in our generic construction. Throughout the following description we assume that the public parameters of the scheme define a message space $\mathbb{Z}_q^\eta$, i.e., the scheme allows us to encrypt η-dimensional vectors and it is linearly homomorphic over $\mathbb{Z}_q^\eta$.

Definition 28 (Relaxed Correctness). *A multi-key homomorphic encryption scheme* (Setup, KeyGen, Enc, Eval, Dec) *is multi-hop correct if there exists a polynomial $B = B(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all polynomials*

[14] This circuit is changed to output either $\perp$ or the challenger's corresponding message. Changing it does not change the transcript since the circuit is only used locally.

$M = M(\lambda)$, all M-argument functions f in the supported family, all inputs $(m_1, \ldots, m_M)$, all pp in the support of $\mathsf{Setup}(1^\lambda)$, all $(\mathsf{sk}_i, \mathsf{pk}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{pp})$, and all ciphertexts $(c_1, \ldots, c_M)$ in the support of the encryptions of $(m_1, \ldots, m_M)$, it holds that

$$\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_M), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), f, (c_1, \ldots, c_M))) = f(m_1, \ldots, m_M) + \mathbf{z}$$

where $\mathbf{z} \in \mathbb{Z}^\eta$ is a noise term with $\|\mathbf{z}\|_\infty \leq \eta \cdot B(\lambda)$. We say that c is in the support of the encryptions of m if

$$c = \mathsf{Enc}(\mathsf{pk}, m) \text{ or } c = \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), \tilde{f}, (\tilde{c}_1, \ldots, \tilde{c}_M))$$

where $(\tilde{c}_1, \ldots, \tilde{c}_M)$ are in the support of the encryptions of $(\tilde{m}_1, \ldots, \tilde{m}_M)$ and $\tilde{f}(\tilde{m}_1, \ldots, \tilde{m}_M) = m$.

Description. In the following we propose an MK-HE for linear functions, which is secure against the LWE assumption. The scheme is linearly homomorphic over the vector space $\mathbb{Z}_q^\eta$. We first show a low rate version and then we discuss a generic methodology to improve the rate of such a scheme. Note that the scheme we describe has no trusted setup but it fixes a modulus q which must be shared by all users. There is no restriction on the structure of q, except on its size.

Let $q = 2q'$ be a k-bit modulus, let (n, m, η) be positive integers and let χ be a B-bounded error distribution defined on $\mathbb{Z}$. Let $\mathbf{G}_i \in \mathbb{Z}_q^{\eta \times k}$ be a matrix which is zero everywhere, but its i-th row is $\mathbf{g}^\top = (1, 2, \ldots, 2^i, \ldots, 2^k)$. For a $y \in \mathbb{Z}_q$ let $\mathbf{g}^{-1}(y) \in \{0,1\}^k$ be the binary expansion of y, i.e., it holds that $\mathbf{g}^\top \cdot \mathbf{g}^{-1}(y) = y$.

Key Generation: On input the security parameter 1^λ, the key generation algorithm samples $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{n \times m}$ uniformly at random. Then it chooses $\mathbf{S} \leftarrow_\$ \mathbb{Z}_q^{\eta \times n}$ uniformly at random and samples $\mathbf{E} \leftarrow_\$ \chi^{\eta \times m}$. Set $\mathbf{B} = \mathbf{S} \cdot \mathbf{A} + \mathbf{E}$. The public key is set to $\mathsf{pk} = (\mathbf{A}, \mathbf{B})$ and the secret key to $\mathsf{sk} = \mathbf{S}$.

Encryption: On input a public key $\mathsf{pk} = (\mathbf{A}, \mathbf{B})$ and a set of messages $(m_1, \ldots, m_\eta)$, the encryption algorithm samples a random matrix $\mathbf{R} \leftarrow_\$ \{0,1\}^{m \times k}$ and sets $\mathbf{C}_1 = \mathbf{A} \cdot \mathbf{R}$ and $\mathbf{C}_2 = \mathbf{B} \cdot \mathbf{R} + \cdot \sum_{i=1}^{\eta} m_i \cdot \mathbf{G}_i$. The algorithm outputs $c = (\mathbf{C}_1, \mathbf{C}_2)$.

Evaluation: On input a linear function $(f_1, \ldots, f_M)$ and a set of ciphertexts $(c_1, \ldots, c_M)$, the evaluation algorithm parses $c_i = (\mathbf{C}_{1,i}, \mathbf{C}_{2,i})$. Then for all $i \in [M]$ it computes $\mathbf{c}_{1,i} = \mathbf{C}_{1,i} \cdot \mathbf{g}^{-1}(f_i)$ and $\mathbf{c}_2 = \sum_{i=1}^{M} \mathbf{C}_{2,i} \cdot \mathbf{g}^{-1}(f_i)$. Finally it outputs $c = (\mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,M}, \mathbf{c}_2)$.

Decryption: On input a set of secret keys $(\mathbf{S}_1, \ldots, \mathbf{S}_M)$ and an evaluated ciphertext $c = (\mathbf{c}_{1,1}, \ldots, \mathbf{c}_{1,M}, \mathbf{c}_2)$, the decryption algorithm computes and outputs $\mathbf{c}_2 - \sum_{i=1}^{M} \mathbf{S}_i \cdot \mathbf{c}_{1,i}$.

We now consider the (relaxed) evaluation correctness of the scheme.

Theorem 9 (Relaxed Correctness). *The scheme as described above satisfies relaxed correctness with $(M \cdot k \cdot m \cdot B)$-noise.*

Proof. Let $\mathbf{f} = (f_1, \ldots, f_M) \in \mathbb{Z}_q^M$ be a linear function and let $(\mathbf{m}_1, \ldots, \mathbf{m}_M) \in \mathbb{Z}_q^{\eta \cdot M}$. For all $i \in [M]$ let $c_i = (\mathbf{C}_{1,i}, \mathbf{C}_{2,i}) = \mathsf{Enc}(\mathsf{pk}_i, m_i)$, i.e., it holds that $\mathbf{C}_{1,i} = \mathbf{A}_i \cdot \mathbf{R}_i$ and $\mathbf{C}_{2,i} = \mathbf{B}_i \cdot \mathbf{R}_i + \sum_{j=1}^{\eta} m_{i,j} \cdot \mathbf{G}_j$. A

routine calculation shows that the output of $\mathsf{Dec}\left((\mathbf{S}_1, \ldots, \mathbf{S}_m), c\right)$ equals

$$\begin{aligned}
&\mathbf{c}_2 - \sum_{i=1}^{M} \mathbf{S}_i \cdot \mathbf{c}_{1,i} \\
&= \sum_{i=1}^{M} \mathbf{C}_{2,i} \cdot \mathbf{g}^{-1}(f_i) - \sum_{i=1}^{M} \mathbf{S}_i \cdot \mathbf{C}_{1,i} \cdot \mathbf{g}^{-1}(f_i) \\
&= \sum_{i=1}^{M} \left(\mathbf{B}_i \cdot \mathbf{R}_i + \sum_{j=1}^{\eta} m_{i,j} \cdot \mathbf{G}_j \right) \cdot \mathbf{g}^{-1}(f_i) - \sum_{i=1}^{M} \mathbf{S}_i \cdot \mathbf{A}_i \cdot \mathbf{R}_i \cdot \mathbf{g}^{-1}(f_i) \\
&= \sum_{i=1}^{M} f_i \cdot \mathbf{m}_i + \mathbf{z}
\end{aligned}$$

where $\mathbf{z} = \sum_{i=1}^{M} \mathbf{E}_i \cdot \mathbf{R}_i \cdot \mathbf{g}^{-1}(f_i)$. We can bound $\|\mathbf{z}\|_\infty$ by

$$\|\mathbf{z}\|_\infty \leq M \cdot k \cdot m \cdot B.$$

Therefore, the scheme satisfies relaxed correctness with $(M \cdot k \cdot m \cdot B)$-noise.

We now argue that the scheme is as hard as the LWE problem.

Theorem 10 (Semantic Security). *Let $m > (n + \eta) \cdot \log(q) + \omega(\log(\lambda))$, then the scheme as described above is semantically secure under the LWE assumption.*

Proof. Observe that the matrix $\mathbf{B}$ in the public key is pseudorandom, by an invocation of the LWE assumption. Then by the leftover-hash lemma [34], the ciphertexts $(\mathbf{C}_1, \mathbf{C}_2)$ are statistically close to uniform.

Instantiating with Rate 1. The scheme as described above does not achieve rate-1, however it satisfies the structural requirement to apply a generic transformation recently introduced by Brakerski et al. [10]. Specifically, let $\mathbf{S}$ be a secret key and let $(\mathbf{c}_1, \mathbf{c}_2)$ be an evaluated MK-HE ciphertext that satisfies relaxed correctness. If the decryption algorithm is of the form

$$\mathsf{Dec}(\mathbf{S}, (\mathbf{c}_1, \mathbf{c}_2)) = \mathbf{c}_2 - \mathbf{S} \cdot \mathbf{c}_1 \mod q$$

for some even modulus q, then there exists a pair of algorithms $(\mathsf{Shrink}, \mathsf{ShrinkDec})$ such that the following conditions are satisfied:

(1) The (public) shrinking algorithm Shrink takes as input a ciphertext $(\mathbf{c}_0, c_1, \ldots, c_\eta)$ encrypting $(m_1, \ldots, m_\eta) \in \{0,1\}^\eta$ and returns $(\mathbf{c}_0, r, w_1, \ldots, w_\eta)$ where $r \in \mathbb{Z}_q$ and $(w_1, \ldots, w_\eta) \in \{0,1\}^\eta$.
(2) The (private) shrinking decryption algorithm $\mathsf{ShrinkDec}$ take as input a shrunk ciphertext and the secret key and returns the plaintext $(m_1, \ldots, m_\eta)$.

In other words, the shrinking algorithm reduces the size of a ciphertext without affecting its correctness. For a detailed description of the algorithm we refer the reader to [10]. When applied to the scheme as described above, the shrunk ciphertexts are of length $M(n+1)\log(q) + \eta$. Loosely bounding $\log(q) \leq \lambda^2$, we obtain

$$\rho = \frac{\eta}{M(n+1)\log(q) + \eta} \geq 1 - \frac{M(n+1)\log(q)}{\eta}$$

which translates to $1 - 1/\lambda$ for $\eta \geq M \cdot (n+1) \cdot \lambda^3$.

7.2 Compressible Multi-Key Fully-Homomorphic Encryption

We construct a MK-FHE augmented with a ciphertext compression algorithm (see Appendix 3.3 for a formal definition). In the following we introduce a compressible MK-FHE assuming the existence of

(1) an MK-FHE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with linear decrypt-and-multiply and plaintext space $\{0,1\}$ (and any rate) and
(2) a rate-1 MK-HE scheme $(\overline{\mathsf{Setup}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Eval}}, \overline{\mathsf{Dec}})$ for linear functions over $\mathbb{Z}_q^\eta$ (i.e., the scheme encrypts η-dimensional vectors)

where the parameters (q, η) will be set later according to our requirements. In abuse of notation we assume that the key-generation algorithm $\mathsf{KeyGen}(1^\lambda; q)$ takes the modulus q as an explicit input. While the former building block can be instantiated from standard MK-FHE constructions (see Appendix A), the latter can be instantiated with the scheme proposed in Section 7.1.

Setup: On input the security parameter 1^λ, the setup algorithm samples

$$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \text{ and } \overline{\mathsf{pp}} \leftarrow \overline{\mathsf{Setup}}(1^\lambda)$$

and sets the public parameters of the scheme to $(\mathsf{pp}, \overline{\mathsf{pp}})$.

Key Generation: On input the public parameters $(\mathsf{pp}, \overline{\mathsf{pp}})$, the key generation algorithm samples

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}; q) \text{ and } (\overline{\mathsf{sk}}, \overline{\mathsf{pk}}) \leftarrow \overline{\mathsf{KeyGen}}(\overline{\mathsf{pp}})$$

where q is the modulus of the plaintext space defined by $\overline{\mathsf{pp}}$. Let $\mathsf{sk} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$, for all $i \in [\eta]$ and for all $j \in [n]$ the algorithm computes

$$\mathsf{ck}_{i,j} \leftarrow \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, s_j \cdot \mathbf{e}_i)$$

where $\mathbf{e}_i$ is the i-th unit vector. Then it sets the public key to $(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ck}_{1,1}, \ldots, \mathsf{ck}_{\eta,n})$ and the secret key to $(\mathsf{sk}, \overline{\mathsf{sk}})$.

Encryption: On input the public key $(\mathsf{pk}, \overline{\mathsf{pk}}, \mathsf{ck}_{1,1}, \ldots, \mathsf{ck}_{\eta,n})$ and a message $m \in \{0,1\}$, the encryption algorithm computes and outputs $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$.

Evaluation: On input a set of public keys $(\mathsf{pk}^{(1)}, \ldots, \mathsf{pk}^{(M)})$, a function f, and a set of ciphertexts $(c^{(1)}, \ldots, c^{(M)})$, the multi-key evaluation algorithm computes and outputs

$$c \leftarrow \mathsf{Eval}\left(\left(\mathsf{pk}^{(1)}, \ldots, \mathsf{pk}^{(M)}\right), f, \left(c^{(1)}, \ldots, c^{(M)}\right)\right).$$

Decryption: On input a set of secret keys $(\mathsf{sk}^{(1)}, \ldots, \mathsf{sk}^{(M)})$ and a ciphertext c, the decryption algorithm computes and outputs

$$m \leftarrow \mathsf{Dec}\left(\left(\mathsf{sk}^{(1)}, \ldots, \mathsf{sk}^{(M)}\right), c\right).$$

Compression: On input a set of public keys $(\overline{\mathsf{pk}}^{(1)}, \mathsf{ck}_{1,1}^{(1)}, \ldots, \mathsf{ck}_{\eta,n}^{(1)}, \ldots, \overline{\mathsf{pk}}^{(M)}, \mathsf{ck}_{1,1}^{(M)}, \ldots, \mathsf{ck}_{\eta,n}^{(M)})$ and a set of ciphertexts $(c_1, \ldots, c_\eta)$. Define f as the linear function

$$f\left(x_{1,1}^{(1)}, \ldots, x_{\eta,n}^{(M)}\right) = \sum_{i=1}^{\eta} \mathsf{Dec\&Mult}\left(\left(x_{i,1}^{(1)}, \ldots, x_{i,n}^{(M)}\right), c_i, q/2\right).$$

The compression function computes and outputs

$$c \leftarrow \overline{\mathsf{Eval}}\left(\left(\overline{\mathsf{pk}}_1, \ldots, \overline{\mathsf{pk}}_M\right), f, \left(\mathsf{ck}_{1,1}^{(1)}, \ldots, \mathsf{ck}_{\eta,n}^{(M)}\right)\right).$$

Compressed Decryption: On input a set of secret keys $(\overline{\mathsf{sk}}^{(1)}, \ldots, \overline{\mathsf{sk}}^{(M)})$ and a compressed ciphertext c, the compressed decryption algorithm computes and outputs

$$m \leftarrow \overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}^{(1)}, \ldots, \overline{\mathsf{sk}}^{(M)}\right), c\right).$$

7.3 Analysis of Compressible MK-FHE

First we argue that the resulting scheme is semantically secure.

Theorem 11 (Semantic Security). *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a secure MK-FHE scheme and let* $(\overline{\mathsf{Setup}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Eval}}, \overline{\mathsf{Dec}})$ *be a secure MK-HE scheme. Then the scheme as described above is semantically secure.*

Proof. Security is shown by a sequence of hybrid experiments.

Hybrid 0: Is defined to be the original scheme.

Hybrid 1 ... $\eta \cdot$ n: For all $i \in [\eta]$ and for all $j \in [n]$ the Hybrid $i + \eta \cdot j$ is defined to be identical to the previous one except that $\mathsf{ck}_{i,j}$ is defined to be the encryption of the column vector 0^η. Indistinguishability follows from an invocation of the semantic security of the MK-HE scheme.

Observe that in the last hybrid no information about sk is contained in the public parameters (beyond what revealed by pk). Furthermore the encryption algorithm returns a fresh MK-FHE ciphertext. Therefore semantic security follows from a canonical reduction to the semantic security of the MK-FHE scheme.

Next we show that our scheme satisfies correctness.

Theorem 12 (Correctness). *Let B be the decryption noise of the MK-FHE scheme an let $\tilde{B}$ be the decryption noise of the MK-HE scheme. Let $q/4 \geq B + M \cdot \eta \cdot n \cdot \tilde{B}$, the the scheme as described above is correct.*

Proof. Fix a pair of public parameters $(\mathsf{pp}, \overline{\mathsf{pp}})$ and a set of M public keys

$$\left(\mathsf{pk}^{(1)}, \overline{\mathsf{pk}}^{(1)}, \mathsf{ck}^{(1)}_{1,1}, \ldots, \mathsf{ck}^{(1)}_{\eta,n}, \ldots, \mathsf{pk}^{(M)}, \overline{\mathsf{pk}}^{(M)}, \mathsf{ck}^{(M)}_{1,1}, \ldots, \mathsf{ck}^{(M)}_{\eta,n}\right)$$

with corresponding secret keys $(\mathsf{sk}^{(1)}, \overline{\mathsf{sk}}^{(1)}, \ldots, \mathsf{sk}^{(M)}, \overline{\mathsf{sk}}^{(M)})$. Furthermore, fix a set of η multi-key ciphertexts $(c_1, \ldots, c_\eta)$ under the set of keys defined above, which are in the support of the encryptions of the messages $(m_1, \ldots, m_\eta)$. Let

$$c = \overline{\mathsf{Eval}}\left(\left(\overline{\mathsf{pk}}_1, \ldots, \overline{\mathsf{pk}}_M\right), f, \left(\mathsf{ck}^{(1)}_{1,1}, \ldots, \mathsf{ck}^{(M)}_{\eta,n}\right)\right)$$

where f is defined as above. Since $\mathsf{ck}^{(k)}_{i,j} = \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}^{(k)}, s^{(k)}_j \cdot \mathbf{e}_i)$, by the multi-key correctness of MK-HE it holds that

$$\overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}^{(1)}, \ldots, \overline{\mathsf{sk}}^{(M)}\right), c\right) = f\left(s^{(1)}_1 \cdot \mathbf{e}_1, \ldots, s^{(M)}_n \cdot \mathbf{e}_\eta\right) + \mathbf{z}$$

where $\|\mathbf{z}\|_\infty \leq M \cdot \eta \cdot n \cdot \tilde{B}$, by the relaxed correctness of the MK-HE scheme. Substituting f it holds that

$$\begin{aligned}
f\left(s^{(1)}_1 \cdot \mathbf{e}_1, \ldots, s^{(M)}_n \cdot \mathbf{e}_\eta\right) &= \sum_{i=1}^{\eta} \mathsf{Dec\&Mult}\left(\left(s^{(1)}_1 \cdot \mathbf{e}_i, \ldots, s^{(M)}_n \cdot \mathbf{e}_i\right), c_i, q/2\right) \\
&= \sum_{i=1}^{\eta} \mathbf{e}_i \cdot \mathsf{Dec\&Mult}\left(\left(s^{(1)}_1, \ldots, s^{(M)}_n\right), c_i, q/2\right) \\
&= \sum_{i=1}^{\eta} \mathbf{e}_i \left(q/2 \cdot m_i + y_i\right) \\
&= q/2 \cdot (m_1, \ldots m_\eta) + \mathbf{y}
\end{aligned}$$

where $\|\mathbf{y}\|_\infty \leq B$ by the decrypt-and-multiply correctness of the MK-FHE scheme. Consequently, we have that

$$\overline{\mathsf{Dec}}\left(\left(\overline{\mathsf{sk}}^{(1)}, \ldots, \overline{\mathsf{sk}}^{(M)}\right), c\right) = q/2 \cdot (m_1, \ldots m_\eta) + \mathbf{y} + \mathbf{z}$$

where

$$\|\mathbf{y} + \mathbf{z}\|_\infty \leq \|\mathbf{z}\|_\infty + \|\mathbf{z}\|_\infty \leq M \cdot \eta \cdot n \cdot \tilde{B} + B \leq q/4$$

which implies that the vector $(m_1, \ldots, m_\eta)$ can be efficiently decoded with probability 1.

Parameters. We discuss our choices of parameters when instantiating the MK-FHE scheme with the scheme from Mukherjee and Wichs [40] and the MK-HE scheme with the construction described in Section 7.1. Let B be the bound on the noise of the MK-FHE scheme and let $\tilde{B} = \mathsf{poly}(\lambda)$ be the decryption noise of MK-HE for some fixed bounded error distribution χ over $\mathbb{Z}$. The analysis (Theorem 12) fixes the following constraint:

$$q/4 \geq B + M \cdot \eta \cdot n \cdot \tilde{B}$$

which is always satisfied for a slightly super-polynomial modulus q. The rate of compressed ciphertexts is identical to the rate of the MK-HE scheme and therefore approaches 1 as η grows. I.e., for η dimensional vectors we have ciphertexts of size $\eta + \mathsf{poly}(\lambda)$ and therefore the rate is $(1 - 1/\lambda)$ for a sufficiently large $\eta = \mathsf{poly}(\lambda)$.

7.4 Additional Properties

Here we discuss on how different instantiations (or minor modifications thereof) of the building blocks of our compiler satisfy desirable properties for an MK-FHE scheme.

Multi-Hop Evaluation. Note that the definition of compressible MK-FHE does not require evaluation correctness for compressed ciphertexts. This is not only a definitional issue and in fact our compressed ciphertext have only partial homomorphic properties. This means that the scheme as described above is only single-hop correct, i.e., the homomorphic evaluations must be completed before compressing the ciphertext. Fortunately there is a generic way to transform such a scheme into a multi-hop (for any *bounded* number of hops h) MK-FHE scheme: Each user can generate a chain of key pairs and publish an encryption of each secret key under the previous public keys, i.e., define the public key of the scheme as

$$\left(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}, \mathsf{Enc}\left(\mathsf{pk}^{(2)}, \mathsf{sk}^{(1)}\right), \ldots, \mathsf{pk}^{(h)}, \mathsf{Enc}\left(\mathsf{pk}^{(h)}, \mathsf{sk}^{(h-1)}\right)\right).$$

Then one can bootstrap compressed ciphertext $c^{(i)}$ under $(\mathsf{pk}_1^{(i)}, \ldots, \mathsf{pk}_M^{(i)})$ into (non-compressed) ciphertext under $(\mathsf{pk}_1^{(i+1)}, \ldots, \mathsf{pk}_M^{(i+1)})$ by evaluating the $\mathsf{CompDec}$ algorithm homomorphically treating $c^{(i)}$ as a constant. Semantic security follow from a standard hybrid argument. The scheme can be upgraded to support an *unbounded* number of hops by setting the users' public keys to

$$(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{sk}))$$

for a uniformly sampled pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, at the cost of introducing an additional circularity assumption.

Threshold Decryption. Another property that is not preserved by our compiler is threshold decryption. That is, even if the original MK-FHE scheme supported threshold decryption, its rate-1 counterpart does not achieve this property (for compressed ciphertexts). To amend this, one can include an additional bootstrapping step that evaluates the compressed decryption algorithm homomorphically and produces a valid (low rate) MK-FHE ciphertext, which is now amenable to threshold decryption. Thus our compiled MK-FHE scheme can be modified to support threshold decryption, as long as the underlying MK-FHE satisfies this property. This however comes at the cost of increasing the rate for the output ciphertext. Since our MPC compilers are anyway subject to a communication overhead polynomial in the output size, this modification does not introduce any further increase in the communication complexity.

For completeness, we show an alternative (more direct) approach to achieve threshold decryption: Instead of sending the decrypted header $(\mathbf{S}_i \cdot \mathbf{c}_{1,i})$, send that plus a smudging noise. If the parameters are set correctly, this smudging noise does not affect correctness, since the decryption has to be robust to noise anyway.

Distributed Setup. One of the major shortcomings of all known MK-FHE schemes with linear decrypt-and-multiply is the presence of a trusted setup to sample the public parameters of the scheme. In the context of MPC, this implies that the protocol inherits the trusted setup assumption even in the *semi-honest* settings. To overcome this nuisance, Brakerski et al. [11] showed how to modify the scheme of [13, 40] to support

a distributed setup: In a distributed setup, each participant locally samples its own block of the public parameters pp_i and broadcasts it to all other users. To accommodate this modification, we refine the syntax of the setup algorithm as follows.

$\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, the (randomized) setup algorithm returns the public parameters block pp_i.

Then the public parameters of the scheme consist of the concatenation of all blocks $\mathsf{pp} = \mathsf{pp}_1 \| \dots \| \mathsf{pp}_M$. Crucially, the modified scheme preserves linear decrypt-and-multiply and therefore can be used as a building block for our compiler. This means that the resulting rate-1 scheme also supports a distributed setup procedure. In terms of security, the scheme is resilient against *semi-malicious* attackers, i.e., the semantic security is preserved even if the attacker is allowed to choose the random coins for the setup generation of all but one users. Note that we consider a rushing adversary that selects the coins after seeing the honestly generated block of parameters.

Definition 29 (Semi-Malicious Security). *A multi-key homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is semi-malicious secure if for all* $\lambda \in \mathbb{N}$*, for all polynomials* $M = M(\lambda)$*, for all* $i \in [M]$*, and for all pairs of messages* (m_0, m_1) *the following distributions are computationally indistinguishable:*

$$(\mathsf{pp}, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_0)) \overset{c}{\approx} (\mathsf{pp}, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m_1))$$

where $(\mathsf{sk}, \mathsf{pk}) \leftarrow_\$ \mathsf{KeyGen}(\mathsf{pp})$*,* $\mathsf{pp} = \mathsf{pp}_1 \| \dots \| \mathsf{pp}_M$*,* $\mathsf{pp}_i \leftarrow \mathsf{Setup}(1^\lambda)$*, and for all* $i \in [M] \setminus \{i\}$ *it holds that* $\mathsf{pp}_j \leftarrow_\$ \mathsf{Setup}(1^\lambda; r_i)$ *where* r_i *is adaptively chosen by the distinguisher on input* pp_i*.*

Our rate 1 MK-FHE scheme allows efficient checkability of the public parameters, since the public parameters are a random matrix. This allows us to rely on its security even when the parameters are generated by a distributed setup with a fully malicious adversary.

References

1. Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Towards efficiency-preserving round compression in MPC - do fewer rounds mean more computation? In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 181–212. Springer, 2020.
2. Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.
3. Prabhanjan Ananth, Abhishek Jain, ZhengZhong Jin, and Giulio Malavolta. Multikey fhe in the plain model. Cryptology ePrint Archive, Report 2020/180, 2020. https://eprint.iacr.org/2020/180.
4. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.
5. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.
6. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

7. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 326–349, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.
8. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 111–120, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.
9. Elette Boyle, Abhishek Jain, Manoj Prabhakaran, and Ching-Hua Yu. The bottleneck complexity of secure multiparty computation. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018: 45th International Colloquium on Automata, Languages and Programming*, volume 107 of *LIPIcs*, pages 24:1–24:16, Prague, Czech Republic, July 9–13, 2018. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
10. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, Lecture Notes in Computer Science. Springer, 2019.
11. Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 645–677, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.
12. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 291–319. Springer, 2020.
13. Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 630–656, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.
14. Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.
15. Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.
16. Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In *FOCS*. IEEE, 2019.
17. Rex Fernando, Ilan Komargodski, Yanyi Liu, and Elaine Shi. Secure massively parallel computation for dishonest majority. Cryptology ePrint Archive, Report 2020/1157, 2020. https://eprint.iacr.org/2020/1157.
18. Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.
19. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.
20. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.
21. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.
22. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

23. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.
24. Craig Gentry and Shai Halevi. Compressible FHE with Applications to PIR. In *TCC*, Lecture Notes in Computer Science. Springer, 2019.
25. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.
26. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press.
27. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
28. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.
29. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
30. Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.
31. Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press.
32. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 581–612, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.
33. Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, January 11–13, 2015. Association for Computing Machinery.
34. Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press.
35. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 433–442, Victoria, BC, Canada, May 17–20, 2008. ACM Press.
36. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 723–732, Victoria, BC, Canada, May 4–6, 1992. ACM Press.
37. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press.
38. Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science*, pages 436–453, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.
39. Daniele Micciancio. From linear functions to fully homomorphic encryption. Technical report, 2019. https://bacrypto.github.io/presentations/2018.11.30-Micciancio-FHE.pdf.
40. Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
41. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *33rd Annual ACM Symposium on Theory of Computing*, pages 590–599, Crete, Greece, July 6–8, 2001. ACM Press.
42. Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

43. Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 859–870, Paris, France, October 7–9, 2018. IEEE Computer Society Press.
44. Anup Rao and Amir Yehudayoff. *Communication Complexity*.
45. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
46. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A Multi-Key Linear Decrypt-and-Multiply

We require the existence of an MK-FHE scheme with some structural properties of the decryption algorithm. Specifically, we require that the multi-key decryption algorithm consists of an application of a linear function (in the concatenation of the secret keys) followed by a rounding operation. Furthermore, we requires that the linear function can be augmented with a constant ω, which can be arbitrarily chosen, and results in the product of the plaintext with ω. We henceforth refer to such a procedure as linear decrypt-and-multiply. This property was explicitly characterized in an oral presentation of Micciancio [39] and was formalized in a recent work of Brakerski et al. [10]. We recall the definition in the following.

Definition 30 (Decrypt-and-Multiply). *We call a multi-key homomorphic encryption scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *a decrypt-and-multiply scheme, if there exists bounds* $B = B(\lambda)$ *and* $Q = Q(\lambda)$ *and an algorithm* $\mathsf{Dec\&Mult}$ *such that the following holds. For all* $\lambda \in \mathbb{N}$*, all polynomials* $M = M(\lambda)$*, all* $q \geq Q$*, all* pp *in the support of* $\mathsf{Setup}(1^\lambda; q)$*, all* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, all* M*-argument functions* f *(in the class supported by the scheme), all inputs* $(m_1, \ldots, m_M)$*, all* c_i *in the support of the encryptions of* $(m_1, \ldots, m_M)$*, and all* $\omega \in \mathbb{Z}_q$ *it holds that*

$$\mathsf{Dec\&Mult}((\mathsf{sk}_1, \ldots, \mathsf{sk}_M), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_M), f, (c_1, \ldots, c_M)), \omega) = \omega \cdot f(m_1, \ldots, m_M) + e \mod q$$

where $\mathsf{Dec\&Mult}$ *is a linear function in the vector* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_M)$ *over* $\mathbb{Z}_q$ *and* $|e| \leq B$ *with all but negligible probability.*

An aspect which is going to be important for our design is the choice of the modulus q. We recall that for essentially all FHE schemes in the literature, the the modulus q does not depend on any secret but depends only on the security parameter. Moreover, LWE-based FHE schemes can be instantiated with any (sufficiently large) modulus q without affecting the worst-case hardness of the underlying LWE problem [42]. This will allow us to set the modulus q on demand in our construction. In abuse of notation, we provide the modulus q as an explicit input to the FHE key generation algorithm.

An MK-FHE Scheme with Linear Decrypt-and-Multiply. It has been shown by Micciancio [39] that any FHE scheme with linear decryption always admits an efficient linear decrypt-and-multiply algorithm. The same argument can be extended to multi-key schemes. Brakerski et al. [10] showed that the MK-FHE scheme of Mukherjee and Wichs [40] (which is a simplified version of the scheme from Clear and McGoldrick [13]) satisfies these structural requirements. The scheme shown in [13, 40] is as secure as the LWE problem (with super-polynomial modulo-to-noise ratio) and we highlight the relevant properties in the following:

(1) The construction is in the common random string model and all parties have access to a uniform matrix $\mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{(n-1)\times m}$.

(2) For any fixed depth parameter d, the scheme supports multi-key evaluation of depth-d circuits using public keys of size $d \cdot \mathsf{poly}(\lambda)$, while secret keys are vectors $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^n$, regardless of the depth parameter. More concretely, there exists an efficient algorithm Eval that is given as input:
   (a) Parameters $(M, d) \in \mathbb{N}$, where M is the number of public keys that perform depth-d computation.
   (b) A depth-d circuit that computes an M-argument Boolean function $f : \{0,1\}^* \to \{0,1\}$.
   (c) A vector of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_M)$ and a fresh (bit-by-bit) encryption of each argument m_i under pk_i, denoted by $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, m_i)$.

   Then Eval outputs a matrix $\mathbf{C} \in \mathbb{Z}_q^{nM \times mM}$ such that

$$\tilde{\mathbf{s}} \cdot \mathbf{C} \cdot \mathbf{g}^{-1}(\boldsymbol{\omega}) = \omega \cdot f(x_1, \ldots, x_M) + e \pmod q$$

   where $\tilde{\mathbf{s}}$ is the row concatenation of $(\mathbf{s}_1, \ldots, \mathbf{s}_M)$, $\boldsymbol{\omega}$ is the vector $(0, \ldots, 0, \omega) \in \mathbb{Z}_q^{nM}$, and $\mathbf{g}^{-1}$ is the bit-decomposition operator. Furthermore, it holds that

$$|e| \leq \beta \cdot (m^4 + m)(mM + 1)^d = \beta \cdot 2^{O(d \cdot \log(\lambda))}$$

   where β is a bound on the absolute value of the noise of fresh ciphertexts.

(3) By further making a circular-security assumption, Eval supports the evaluation of circuits of any depth without increasing the size of the public keys. In this case the bound on the noise is $|e| \leq \beta \cdot 2^{O(d_{\mathsf{Dec}} \cdot \log(\lambda))}$, where d_{Dec} is the depth of the decryption circuit, which is poly-logarithmic in λ.

Observe that $\mathbf{C} \cdot \mathbf{g}^{-1}(\boldsymbol{\omega})$ does not depend on the secret key and therefore defining

$$\mathsf{Dec\&Mult}(\tilde{\mathbf{s}}, \mathbf{C}, \omega) = \tilde{\mathbf{s}} \cdot \mathbf{C} \cdot \mathbf{g}^{-1}(\boldsymbol{\omega})$$

gives a syntactically correct linear decrypt-and-multiply algorithm and $B = |e|$ is the corresponding noise bound. Finally we remark that the MK-FHE scheme does not impose any restriction on the choice of q (except for its size) so we can freely adjust it depending on our requirements.