Interaction-Preserving Compilers for Secure Computation

Nico Döttling ✉
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Vipul Goyal ✉
Carnegie Mellon University, Pittsburgh, PA, USA
NTT Research, Sunnyvale, CA, USA

Giulio Malavolta ✉
Max Planck Institute for Security and Privacy, Bochum, Germany

Justin Raizes ✉
Carnegie Mellon University, Pittsburgh, PA, USA

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.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Multiparty Computation, Communication Complexity, Fully Homomorphic Encryption

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.57

Related Version *Full Version*: `https://eprint.iacr.org/2021/1503`

Funding *Vipul Goyal*: 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.
Justin Raizes: 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.

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

© Nico Döttling, Vipul Goyal, Giulio Malavolta, and Justin Raizes;
licensed under Creative Commons License CC-BY 4.0

the society. General positive results for secure computation were obtained over three decades by Yao [40] in the two-party setting and then by Goldreich et al. [25] 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 [22]. 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., [37, 15]).

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 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. [38]) 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 [38]. 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[1] 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.

[1] See technical overview. Any improvements to the communication complexity of Laconic Function Evaluation schemes will directly improve our results. The scheme given by [37] has communication complexity $\tilde{O}(s_{\mathsf{in}} + s_{\mathsf{out}})\mathsf{poly}(d, \lambda)$.

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.[2] 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*

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

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 [26] and succinct non-interactive arguments of knowledge [32] have been shown to be a very powerful machinery but we have evidence [19, 24] that they are hard to instantiate from "standard" cryptographic assumptions. Nevertheless, candidate constructions exist [34, 20] 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*

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

[2] The dependency on the size of the output seems somewhat inherent to our approach, since it has been shown [29] that overcoming this barrier require some form of program obfuscation. Achieving this under LWE is a fascinating open question.

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

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

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) [33] 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 [22].

▶ **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 [36]. 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 [31, 14, 13] and the round complexity [17, 35, 21, 5] 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 [29] 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. [8] 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. [16] 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 [33, 3, 35]. 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 $(c_1, \dots, c_M)$ encrypted under *independently sampled* keys. The resulting ciphertext $\tilde{c}$ encodes the function output $f(m_1, \dots, m_M)$ and requires the knowledge of all secret keys $(\mathsf{sk}_1, \dots, \mathsf{sk}_M)$ 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 $(s_1, \dots, s_M)$ are publicly combined to reconstruct the message $f(m_1, \dots, m_M)$.

Clearly the shares $(s_1, \dots, s_M)$ 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 $n \in \{1, \dots, N\}$) 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 $(pk_1, \dots, \mathsf{pk}_M)$. 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 $(\tilde{c}_1, \dots, \tilde{c}_M)$ 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 $(s_{i,1}, \dots, s_{i,M})$ which are locally reconstructed to recover the output of the protocol.

It is not hard to see that the resulting protocol is correct[3] 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 subparagraphs 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.

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

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. [37]. 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 [37], 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 $\mathcal{C}$.

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 $\mathsf{LFEEnc}(h, (\mathsf{sk}_i, (c_1^{(N)}, \dots, c_M^{(N)}))$, where $(c_1^{(N)}, \dots, c_M^{(N)})$ 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. [37] 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 $\mathsf{LFEEnc}(h, (\mathsf{sk}_i, (c_1^{(N-1)}, \dots, c_M^{(N-1)}, c_1^{(N)}, \dots, c_M^{(N)}))$, 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 [34, 6, 7] 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 [4, 18] 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* [27] 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 H_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 H_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 $(\mathsf{H}_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, b)$ 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 H_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 $(\mathsf{H}_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, 0)$ and the 1-label under the statement $(\mathsf{H}_j, c_j^{(N-1)}, \ldots, c_M^{(N-1)}, \kappa, 1)$. 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 [20] only provides security for messages encrypted under false statements. However, a closer look at the language $\mathcal{L}$ reveals that for given H_j, $c_j^{(N-1)}, \ldots, c_M^{(N-1)}$ and κ both the statements

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

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* [26]. 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 H_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 [26].

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 [33, 2], 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 p, (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 the full version 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 [37] 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., [21, 5]) 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 [3]. To ensure succinctness, we need to rely on simulation-extractable zero knowledge SNARKs instead (e.g. [28]). 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[4] 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. [11]. 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. These SNARKs are computed and broadcast underneath

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

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 $\max(4, N')$, 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. [9] and of Gentry and Halevi [23] 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 $\vec{c}$ and a secret key $\vec{s}$ 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) $L_{\omega,\vec{c}}$ such that

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

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 $L_{\omega,\vec{c}}$. 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 $\vec{c}$ and the concatenation of all of the corresponding secret keys $(\vec{s}_1, \dots, \vec{s}_M)$, we require the existence of a linear function $L_{\omega,\vec{c}}$ such that

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

where ω and e are defined as before. Fortunately, one can show that existing MK-FHE schemes [12, 35] 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 [39] 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

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

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

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

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 $\vec{c}_2 - \mathbf{S} \cdot \vec{c}_1 \pmod q$ 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 [30].

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 $(\vec{c}_1, \vec{c}_2)$ and $(\tilde{\vec{c}}_1, \tilde{\vec{c}}_2)$, the multi-key evaluation algorithm computes $\left(\vec{c}_1, \tilde{\vec{c}}_1, \vec{c}_2 + \tilde{\vec{c}}_2\right)$. Given both secret keys $\mathbf{S}$ and $\tilde{\mathbf{S}}$ one can recover the sum of the vectors by computing

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

which can be efficiently decoded as long as $\|\vec{z}\|_\infty$ 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. [9].

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). See the full version for more details.

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 [12, 35] require a trusted setup, which is inherited by our construction.[5] 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 [33] 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 (see the full version) 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 the full version that the common reference string of our MK-FHE scheme can be computed in a *single* broadcast round using techniques from [10]. 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.

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

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. doi:10.1007/978-3-030-64840-4_7.

2 Prabhanjan Ananth, Abhishek Jain, ZhengZhong Jin, and Giulio Malavolta. Multikey fhe in the plain model. Cryptology ePrint Archive, Report 2020/180, 2020. URL: https://eprint.iacr.org/2020/180.

3 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, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, 2012. doi:10.1007/978-3-642-29011-4_29.

4 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, 2001. doi:10.1007/3-540-44647-8_1.

5 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, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, 2018. doi:10.1007/978-3-319-78375-8_17.

6 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*, pages 326–349. ACM, 2012. doi:10.1145/2090236.2090263.

7 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 ACM STOC*, pages 111–120. ACM Press, 2013. doi:10.1145/2488608.2488623.

8 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*, volume 107 of *LIPIcs*, pages 24:1–24:16. Schloss Dagstuhl, 2018. doi:10.4230/LIPIcs.ICALP.2018.24.

9 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.

10 Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 645–677. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-70500-2_22.

11 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. doi:10.1007/978-3-030-64378-2_11.

12 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, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, 2015. doi:10.1007/978-3-662-48000-7_31.

13 Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor,

EUROCRYPT 2010, volume 6110 of *LNCS*, pages 445–465. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-13190-5_23.

14 Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 241–261. Springer, Heidelberg, 2008. doi:10.1007/978-3-540-85174-5_14.

15 Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In *FOCS*. IEEE, 2019.

16 Rex Fernando, Ilan Komargodski, Yanyi Liu, and Elaine Shi. Secure massively parallel computation for dishonest majority. Cryptology ePrint Archive, Report 2020/1157, 2020. URL: https://eprint.iacr.org/2020/1157.

17 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, 2014. doi:10.1007/978-3-642-54242-8_4.

18 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, 2013. doi:10.1109/FOCS.2013.13.

19 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, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 518–535. Springer, Heidelberg, 2014. doi:10.1007/978-3-662-44371-2_29.

20 Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, 2013. doi:10.1145/2488608.2488667.

21 Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, 2018. doi:10.1007/978-3-319-78375-8_16.

22 Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, 2009. doi:10.1145/1536414.1536440.

23 Craig Gentry and Shai Halevi. Compressible FHE with Applications to PIR. In *TCC*, Lecture Notes in Computer Science. Springer, 2019.

24 Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, 2011. doi:10.1145/1993636.1993651.

25 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 ACM STOC*, pages 218–229. ACM Press, 1987. doi:10.1145/28395.28420.

26 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, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40084-1_30.

27 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 ACM STOC*, pages 555–564. ACM Press, 2013. doi:10.1145/2488608.2488678.

28 Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-63715-0_20.

29 Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, 2015. doi:10.1145/2688073.2688105.

30 Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, 1989. doi:10.1145/73007.73009.

31 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442. ACM Press, 2008. doi:10.1145/1374376.1374438.

32 Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, 1992. doi:10.1145/129712.129782.

33 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 ACM STOC*, pages 1219–1234. ACM Press, 2012. doi:10.1145/2213977.2214086.

34 Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, 1994. doi:10.1109/SFCS.1994.365746.

35 Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, 2016. doi:10.1007/978-3-662-49896-5_26.

36 Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *33rd ACM STOC*, pages 590–599. ACM Press, 2001. doi:10.1145/380752.380855.

37 Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, 2018. doi:10.1109/FOCS.2018.00086.

38 Anup Rao and Amir Yehudayoff. *Communication Complexity and Applications*. Cambridge University Press, 2020.

39 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, 2005. doi:10.1145/1060590.1060603.

40 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, 1986. doi:10.1109/SFCS.1986.25.