Non-Interactive Multiparty Computation Without Correlated Randomness

Shai Halevi[1], Yuval Ishai[2], Abhishek Jain[3], Ilan Komargodski[4], Amit Sahai[5], and Eylon Yogev[6]

[1] IBM Research shaih@alum.mit.edu
[2] Technion and UCLA yuvali@cs.technion.ac.il
[3] Johns Hopkins University abhishek@cs.jhu.edu
[4] Cornell Tech komargodski@cornell.edu
[5] UCLA sahai@cs.ucla.edu
[6] Weizmann Institute eylon.yogev@weizmann.ac.il

Abstract. We study the problem of non-interactive multiparty computation (NI-MPC) where a group of completely asynchronous parties can evaluate a function over their joint inputs by sending a *single* message to an evaluator who computes the output. Previously, the only general solutions to this problem that resisted collusions between the evaluator and a set of parties were based on multi-input functional encryption and required the use of complex correlated randomness setup.

In this work, we present a new solution for NI-MPC against arbitrary collusions using a public-key infrastructure (PKI) setup supplemented with a common random string. A PKI is, in fact, the minimal setup that one can hope for in this model in order to achieve a meaningful "best possible" notion of security, namely, that an adversary that corrupts the evaluator and an arbitrary set of parties only learns the residual function obtained by restricting the function to the inputs of the uncorrupted parties. Our solution is based on indistinguishability obfuscation and DDH both with sub-exponential security. We extend this main result to the case of *general interaction patterns*, providing the above best possible security that is achievable for the given interaction.

Our main result gives rise to a novel notion of *(public-key) multiparty obfuscation*, where n parties can *independently* obfuscate program modules M_i such that the obfuscated modules, when put together, exhibit the functionality of the program obtained by "combining" the underlying modules M_i. This notion may be of independent interest.

1 Introduction

The recent breakthrough on general-purpose program obfuscation [16] has spurred a large body of research that studies different flavors of obfuscation and their applications. Much of this research is driven by the goal of *minimizing interaction*. In this work we revisit the question of minimizing interaction in the broad context of secure multiparty computation (MPC). This question is independently motivated by the goal of obtaining useful multi-party variants of obfuscation and related primitives such as functional encryption [29,5,28].

Non-Interactive MPC. Consider the following *non-interactive* model of computation: there are n parties who wish to securely evaluate a function f on their joint inputs. For instance, the parties may wish to compute a majority of their $0/1$ inputs for the purpose of voting. The parties are completely asynchronous. At any point, a party can "wake up" and submit a *single* message to a central server, also referred to as the *evaluator*. Upon receiving all the messages, the evaluator computes the output.

The above model represents an example of MPC with limited interaction, and contrasts with traditional MPC models [31,18] where the parties freely interact with each other. Indeed, there are many scenarios where more limited forms of interaction are desirable, e.g., when the parties cannot be simultaneously available due to physical constraints or due to efficiency considerations.

The non-interactive model of computation was first proposed by Feige, Kilian and Naor (FKN) [13]. In their model, they allow the messages of the parties to depend on secret randomness that is unknown to the evaluator. As a result, a major disadvantage of their model is that it does not provide any security in case the evaluator colludes with one or more parties.[7]

The study of *collusion-resistant* MPC protocols with restricted interaction was initiated by Halevi, Lindell, and Pinkas (HLP) [24]. They considered a variation of the non-interactive model in which the parties *sequentially* interact with the evaluator, where this interaction can depend on a public-key infrastructure (PKI). Subsequent to their work, Beimel et al. [3] and Halevi et al. [23] considered a model (similar to FKN) where each of the parties can *independently* interact with the evaluator by sending it a single message. (It was also implicitly considered in the context of multi-input functional encryption by Goldwasser et al. [19].) Crucially, in this model, unlike the FKN model, security does not completely break down in the presence of collusion attacks. We refer to MPC in this model as *non-interactive MPC*, or NI-MPC for short.

Best-Possible Security for NI-MPC. An NI-MPC protocol should provide the "best-possible security" for any given set of corrupted parties. The latter can be formalized via the following notion of *residual function* [24]: when the evaluator colludes with a set T of corrupted parties, it can inevitably learn the value of the original function f on the honest parties' inputs coupled with *every possible choice* of inputs from T. Thus, the adversary effectively has oracle access to the residual function obtained by restricting f to the honest parties' inputs and allowing a free choice of the corrupted parties' inputs. The security requirement of NI-MPC informally asserts that the adversary learns no more than the residual function. This gives a meaningful security guarantee in many natural cases. For instance, for the above mentioned example of voting or, more

[7] Indeed, it is not hard to see that in any protocol for a non-trivial function in this model, all inputs can be recovered from the n messages and the common randomness. This makes such protocols inherently vulnerable to collusions.

generally, in the case of symmetric functions, the residual function still hides the sensitive information about the inputs of the uncorrupted parties. See [3] for a more detailed discussion.

The strongest formalization of the above security requirement is via efficient simulation. However, this cannot be achieved in general. Indeed, NI-MPC for general functions can be easily seen to imply "virtual black-box" obfuscation for general functions [19,3], which was ruled out in [2]. Instead, we consider a natural relaxation that replaces efficient simulation by unbounded simulation, or equivalently *indistinguishability* of different sets of inputs that induce the same residual function. Roughly, the indistinguishability-based security definition is described as follows – for simplicity, we state it for the three-party case where the last party is corrupted and is colluding with the evaluator: for any input pairs (x, y) and (x', y') for the honest parties such that $f(x, y, \cdot) \equiv f(x', y', \cdot)$, the adversary's view is indistinguishable when the inputs of the honest parties to the computation are (x, y) or (x', y'). We refer the reader to the technical sections for the general definition.

Necessity of Setup and iO. We next turn our attention to the necessity of setup for NI-MPC. Unlike the FKN model, collusion-resistant NI-MPC cannot be realized with a single common source of randomness that is shared by all clients. Instead, NI-MPC requires a setup that allows authentication of the messages of parties, while remaining robust to collusion. This is because otherwise, an adversarial evaluator can spoof an honest party in order to learn private information about the inputs of the other honest parties: consider NI-MPC for three parties where the last party and the evaluator are corrupted. The security definition for this case was described above. However, suppose there exists a "splitting input" y^* for f s.t. $f(x, y^*, z) \neq f(x', y^*, z)$, where z is some third party input. In the absence of authentication, the adversary can prepare a message $\widehat{y}^*$ on its own using y^* and then evaluate it together with the first and third party messages to distinguish whether the first honest party's input to the computation is x or x'.

Further, as already noted in [3,19,23], NI-MPC for general functions implies indistinguishability obfuscation (iO) [2,15]: recall the security definition of three-party NI-MPC as described above, where the last party and the evaluator are corrupted. Substituting x and x' with two circuits C_0 and C_1, y and y' with null inputs and f with a universal circuit such that $f(C, \perp, z) = C(z)$, we can recover the security definition of iO. We stress that this implication holds regardless of the choice of setup used in the NI-MPC protocol.

Our Question: NI-MPC with minimal setup? Known n-party NI-MPC protocols [19,3,23] require intricate n-wise correlated randomness setups where the *private* randomness r_i input to each party i is computed as $G(r) = (r_1, \ldots, r_n)$ for some function G. In particular, such a correlated randomness setup can only be realized by using a trusted dealer with private channels to each protocol participant, or alternatively a fully interactive MPC protocol that is executed during an offline phase. The main question we ask is whether the use of such *general* correlated randomness is necessary for NI-MPC.

Indeed, secure protocols in the HLP model or even the FKN model can be realized using a standard PKI setup.[8] A PKI setup can be viewed as a special form of "pairwise" correlation that can be realized with no direct interaction between the clients: it suffices for every client to post its own public key and read all of the other clients' public keys. Given that some form of setup is necessary to prevent the aforementioned spoofing attacks in NI-MPC, using a PKI setup is the best one could hope for.[9]

1.1 Our Results

I. NI-MPC without Correlated Randomness Setup. We construct the first collusion-resistant NI-MPC protocols without general correlated randomness. Instead, we use a PKI and a common *random* string (CRS).[10] The security of our construction is proven against malicious adversaries assuming iO and DDH, both with sub-exponential security.

Theorem 1 (Informal). *Assuming indistinguishability obfuscation and DDH, both with sub-exponential security, there exists a collusion-resistant non-interactive multi-party computation protocol that achieves security in the malicious model given a CRS and a PKI.*

As explained earlier, iO is a necessary assumption for NI-MPC for general functions.

II. Multi-Party Indistinguishability Obfuscation. Our main result gives rise to a new notion of multiparty obfuscation (and iO) that may be of independent interest. We start by explaining this notion via a motivating example: consider a scenario where there are n parties, each holding a private program module M_i. Suppose there is a public $\mathsf{Combine}$ algorithm that combines $M_1, \ldots, M_n$ to yield a program M. The parties wish to jointly obfuscate M; however, due to proprietary reasons, they cannot share their modules with each other.

One option for the parties is to simply run a standard MPC protocol using their modules as their respective private inputs to jointly compute an obfuscation of M. We ask whether it is possible for the parties to jointly obfuscate M in a non-interactive manner. In particular, we require that each party i can individually obfuscate its module M_i s.t. the obfuscated modules $\overline{M_1}, \ldots, \overline{M_n}$ put together, exhibit the functionality of M while still hiding the individual modules.

[8] Protocols in the FKN model can be realized using standard PKI by combining a non-interactive 2-party key agreement protocol, which can be based on the DDH assumption, with garbled circuits. The idea is to have every party P_i agree with the last party P_n on the input keys of P_i, and have P_n generate and send the garbled circuit based on all keys.

[9] In particular, note that a common random string setup cannot prevent spoofing attacks.

[10] We note that even a construction with a PKI and a common *reference* string was unknown prior to this work.

We refer to this as multiparty obfuscation, and in the context of iO, as multiparty iO. Roughly speaking, the security of multiparty iO says that for any pair of tuples $(M_1^0, \ldots, M_n^0)$ and $(M_1^1, \ldots, M_n^1)$ s.t. $\mathsf{Combine}(M_1^0, \ldots, M_n^0)$ and $\mathsf{Combine}(M_1^1, \ldots, M_n^1)$ are functionally equivalent, the pair of tuples $(\overline{M_1^0}, \ldots, \overline{M_n^0})$ and $(\overline{M_1^1}, \ldots, \overline{M_n^1})$ are computationally indistinguishable. Here $\overline{M_i^b}$ denotes the obfuscation of M_i^b computed by the i'th party.

We note that if we set $n = 1$ and the $\mathsf{Combine}$ function to be the identity function, then we recover the standard notion of iO from the above definition. Further, while the above definition considers the case where all the parties are honest and only the obfuscation evaluator is dishonest, it can be easily modified to allow for dishonest parties. However, in this case, the functional equivalence requirement on the modules must be suitably modified, as in the case of NI-MPC.

Multiparty iO from NI-MPC. We now explain how our NI-MPC can be used to obtain multiparty iO. The setup in multiparty iO is the same as in the NI-MPC, namely, a PKI and a CRS. We consider NI-MPC between $(n+1)$ parties. The inputs of the first n parties are set to be their respective modules $M_1, \ldots, M_n$. The $(n+1)$th party is the evaluator which holds an input x for the "combined" program $M = \mathsf{Combine}(M_1, \ldots, M_n)$. The function computed by the parties contains the $\mathsf{Combine}$ algorithm hardwired in its description. On input $(M_1, \ldots, M_n, x)$, it computes $M = \mathsf{Combine}(M_1, \ldots, M_n)$ and outputs $M(x)$.

Now, suppose that only the $(n+1)$th party and the evaluator are corrupted. Then, this case exactly corresponds to multiparty iO where all the parties (a.k.a., obfuscators) are honest. The case of more general corruptions is defined similarly.

III. General interaction patterns. As observed by Halevi et al. [23], the problem of NI-MPC can be viewed as secure computation on a *star* interaction pattern. Towards a generalization, Halevi et al. also considered the problem of secure multiparty computation with *general* interaction patterns. They presented a result for the same assuming (sub-exponentially secure) indistinguishability obfuscation by using a complex correlated randomness setup.

We improve upon their result by providing a construction in the PKI model assuming (sub-exponentially secure) indistinguishability obfuscation and DDH. In particular, our main protocol for NI-MPC can be extended to handle more general interaction patterns described by directed acyclic graphs, where each node represents a party who expects to receive messages from all of its parents and can then send messages to all of its children, and where the sinks of the graph compute the output.

We note that any interaction pattern gives rise to a "best-possible" notion of security, as formalized in [24] for the case of a "chain" pattern and in [23, Definition 6] for general patterns. In [23], it was shown that the star pattern is "complete" in the sense that any pattern can be reduced to it with no additional cryptographic assumptions. However, their reduction inherently requires a correlated randomness setup and thus is not applicable in our setting.

Instead, we show how to directly modify our scheme to guarantee best-possible security for any interaction pattern. A representative example is that of a chain pattern considered in [24,20], namely a simple directed path traversing all nodes. The "best-possible" security notion that can be achieved in this case is typically stronger than the security notion for NI-MPC which corresponds to a star pattern. To see this, notice that in the star pattern the adversary can reset the input of each party it controls, whereas in the chain pattern he cannot reset the input of a malicious party that is followed by an honest party along the chain. Our results for the chain pattern provide the first extension of the positive results in the HLP model [24] to general functions with a similar PKI setup, at the (necessary) price of relying on stronger assumptions.[11]

1.2 Our Techniques

We now explain the main ideas in our construction of collusion-resistant NI-MPC. As explained earlier, this suffices to obtain multiparty iO.

Initial Challenges. Recall that in an NI-MPC protocol, each party sends a *single* message to the evaluator. Then, a starting idea to construct an NI-MPC protocol is to "compile" an interactive MPC protocol into a non-interactive one using program obfuscation. That is, let f be the function that the parties wish to compute and let Π be an interactive MPC protocol for computing f. Then, each party i simply computes an obfuscation $\overline{\mathsf{NMF}_i}$ of its next-message function NMF_i in Π, where NMF_i contains its input x_i and random tape r_i hardwired. It then sends $\overline{\mathsf{NMF}_i}$ to the evaluator. Upon receiving all the obfuscated programs, the evaluator computes a transcript of Π by acting as the communication channel between the "virtual parties" $\overline{\mathsf{NMF}_1}, \ldots, \overline{\mathsf{NMF}_n}$. At the end, it obtains the output $f(x_1, \ldots, x_n)$.

The main problem with the above approach is that an adversarial evaluator (that colludes with one or more parties) can perform resetting attacks to break the security of the underlying MPC protocol Π. Indeed, since the obfuscated programs $\overline{\mathsf{NMF}_i}$ are computing a "reactive" functionality, an evaluator can simply reset them to an earlier state and feed them different messages. Since the input of each honest party i is fixed in the obfuscated program $\overline{\mathsf{NMF}_i}$, this means that the adversary can now execute multiple "sessions" of Π w.r.t. the same fixed inputs of the honest parties. The security of standard MPC protocols completely breaks down in such a case.

To address this problem, a natural idea is to replace Π with a resettably secure MPC protocol [8,22,21]. Roughly speaking, resettable MPC guarantees that the only advantage an adversary can gain by performing a resetting attack is to change its input. As such, it can learn multiple function outputs w.r.t. a fixed input vector of the honest parties. No information beyond this set of outputs is leaked to the adversary. While resettably secure MPC for general functions is impossible in the plain model [21], it can be realized in the CRS model by

[11] Indeed, MPC for computing general functions on a chain implies indistinguishability obfuscation (iO). The positive results of [24,20] based on weaker assumptions are only for weaker classes of functions where positive results do not imply (general) iO.

compiling a UC-secure MPC protocol [10] using pseudorandom functions à la [8].

The security guarantee of resettable MPC coincides with our definition of NI-MPC, with the only difference that our definition is indistinguishability-based while resettable MPC security is defined w.r.t. simulation. Nevertheless, it is not immediately clear how to compile a resettably secure MPC protocol into NI-MPC when using iO. In particular, note that the natural approach to argue security in this context is to hardwire the simulated transcript in the obfuscated programs so as to "force" the simulated execution on the adversary. This strategy was previously used in [14] for constructing two-round MPC protocols. In our context, however, since the adversary can legitimately perform resetting attacks, the number of possible transcripts we may need to hardwire is *unbounded*.

A Starting Point: Obfuscation Combiners. In order to find a path to a solution, let us first consider a weaker problem: suppose we are given N candidates for program obfuscation, many of which are possibly "bad", i.e., either they do not hide the program or do not preserve correctness. We do not know which of the candidates are bad. The goal is to use these candidates to obfuscate a function in a secure manner. This problem is referred to as obfuscation combiner [1].

To see why this problem is relevant to us, note that the "bad" candidate obfuscations can be thought of as the corrupted parties in our setting. The role of the evaluator is the same. Furthermore, in this setting, similar to ours, resetting attacks are unavoidable. The key difference, however, is that while our setting is inherently distributed, the above setting is centralized, in that a common entity performs all the obfuscations.

Nevertheless, we use obfuscation combiner as a starting point for our construction. Specifically, we build on ideas used in the obfuscation combiner of Ananth et al. [1]. In their construction, they use a special-purpose MPC protocol [27] based on multi-key fully homomorphic encryption [26,11,27,12] (see Section 2 for the definition). Our solution also uses this MPC scheme as a starting basis. In particular, our construction implicitly compiles the MPC scheme of [27] into a resettably secure one in an "iO-friendly" manner. In order to develop a full solution for our setting, however, we have to address several additional challenges that do not arise in the setting of obfuscation combiners. Below, we elaborate on the details.

Our Approach. We first recall the notion of a multi-key FHE. A multi-key FHE allows for generating individual encryption/decryption-key pairs ek_i, dk_i such that they can be later combined to obtain a joint public key ek. To be more precise, given a ciphertext with respect to ek_i, there is an Expand operation that transforms it into a ciphertext with respect to a joint public key ek. Once this done, the resulting ciphertext can be homomorphically evaluated just like any FHE scheme. The resulting ciphertexts can then be individually decrypted using the dk_i's to obtain partial decryptions. Finally, there is a mechanism to combine the partial decryptions to obtain the final output.

Given a multi-key FHE, a first idea for a non-interactive MPC protocol for a function f is described below.[12]

1. **Setup of party $\boldsymbol{i}$**: Sample an encryption/decryption key pair ek_i, dk_i and sets its private-key to be $\mathsf{SK}_i = (dk_i)$ and published the public-key $\mathsf{PK}_i = (ek_i)$.
2. **Encryption of party $\boldsymbol{i}$ on input $\boldsymbol{x_i}$**: Compute an encryption of x_i with respect to the key ek_i: $\mathsf{CT}_i = \mathsf{Enc}(ek_i, x_i)$ and compute an obfuscation $\overline{G_i}$ of the circuit given in Figure 1. The message of this party is the pair $(\overline{G_i}, \mathsf{CT}_i)$.

$$G_i\left[dk_i, \{ek_j\}_{j\in[n]}\right]$$

Hardwired: the decryption key of party i, dk_i, and all the encryption keys $\{ek_j\}_{j\in[n]}$.
Input: n ciphertexts: $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$.

- For all $j \in [n]$ compute $\widehat{\mathsf{CT}_j} \leftarrow \mathsf{Expand}((ek_1, \ldots, ek_n), j, \mathsf{CT}_j)$.
- Perform $\widehat{\mathsf{CT}_{\mathsf{out}}} \leftarrow \mathsf{Eval}\left(f, \widehat{\mathsf{CT}_1}, \ldots, \widehat{\mathsf{CT}_n}\right)$.
- Output $p_i \leftarrow \mathsf{PartDec}\left(\widehat{\mathsf{CT}_{\mathsf{out}}}, ek_1, \ldots, ek_n, i, dk_i\right)$.

Fig. 1. The circuit G_i. The parameters in the square brackets are hardwired values and the inputs are specified separately (this is the convention throughout the paper).

3. **Evaluation given the messages of the parties:** Given $(\overline{G_i}, \mathsf{CT}_i)$ for each $i \in [n]$, the evaluator executes each $\overline{G_i}$ on $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ to get p_i. Then, it combines all partial decryptions $p_1, \ldots, p_n$ to get y

Correctness of the protocol is immediate by the correctness of the underlying obfuscator and the correctness of the multi-key FHE scheme. However, proving security of this scheme runs into several obstacles. First, we have to deal with the fact that the Eval operation is randomized. Second, there seems to be a very simple attack for the evaluator stemming from the fact that the encryption keys can be used to encrypt arbitrary values. Namely, the evaluator, given ek_i, can simulate ciphertexts generated by the i-th party and potentially fully recover all inputs.

The first issue is (by now) quite standard in the iO literature and we overcome it by using a puncturable PRF family. The second issue is more complicated to handle. Our idea is to augment the setup procedure to generate a signature/verification key pair and let each party sign its input. The circuit that each party will obfuscate will verify each such signature to make sure it was honestly generated. Intuitively, it seems to solve the problem, but formally arguing

[12] For simplicity of notation we assume a single public function.

security using iO and an arbitrary signature scheme seems to be problematic.[13] To overcome this, we construct a special kind of signatures with which we are able to complete the proof. Specifically, our signatures have a property that allows us to replace the verification key with a "punctured verification key" in an indistinguishable way such that the punctured key (statistically) verifies *only a single signature*.

Having these signatures, the proof proceeds by a sequence of hybrid experiments in which we slowly replace each partial decryption with a simulation which does not require the decryption key. The number of hybrids is proportional to the input domain of the circuit G_i. Throughout the proof, we extensively use the probabilistic iO of [9] and the partition-programming technique of [1].

The malicious case. Notice that in the malicious case, since our protocol is non-interactive, the evaluator can plug in any value it wishes in the corrupted parties and obtain a value for the function f. Thus, in this case, if the evaluator colludes with a subset $A \subset [n]$ of the parties (i.e. $H = [n] \setminus A$ is the honest set of parties), then we could hope to get security only for functions f and challenge $(\{x_i^0\}_{i \in H}, \{x_i^1\}_{i \in H})$ that satisfy

$$f(\langle \{x_i^0\}_{i \in H}, \{x_i\}_{i \in A} \rangle) = f(\langle \{x_i^1\}_{i \in H}, \{x_i\}_{i \in A} \rangle)$$

for every $\{x_i\}_{i \in A}$ and where $\langle \cdot \rangle$ sorts the inputs according to the index i.

The proof and construction above do not guarantee security in the case where the evaluator colludes with a subset of the parties. Specifically, there is no guarantee as to what happens when one of the (malicious) parties sends a malformed ciphertext CT_i. There, the partial decryption would still output some string, but we have no guarantee on whether the simulation of this partial decryption is going to be indistinguishable.

To overcome this we use non-interactive zero-knowledge proofs (NIZKs). Specifically, during encryption of the value x_i, party i will also provide (in addition to the obfuscated circuit, the ciphertext and the signature) a NIZK proof asserting that the ciphertext is a legal well-formed ciphertext. As in most works that combine NIZKs and iO, we need a special kind of NIZKs called statistically-simulation-sound NIZKs (SSS-NIZKs) [15]. However, this is still not enough for us. The reason is that for the partial-decryption simulation we need to know the final output, but we have no control over the inputs coming from the malicious parties (recall that they are encrypted!). To solve this we use statistical-simulation-extractable zero-knowledge proofs (SSE-NIZKs) which are a special kind of non-interactive zero-knowledge proofs of knowledge (NIWI-POKs) which allow us to *extract* from the proof of the malicious parties their decryption key, decrypt their ciphertexts and compute the expected final output of the computation. We construct such SSE-NIZKs in the CRS model starting with any NIWI-POK and any one-way permutation.

[13] One standard way to handle this is to assume a differing-input obfuscator (diO) rather than plain iO. A diO guarantees that if an adversary distinguishes two obfuscated circuits, then it must know a differing input.

Reusable PKI with a session ID. The solution we described above only gives a non-reusable scheme. Namely, the PKI cannot be used for more than one encryption. We consider a stronger model where the PKI can be reused but this requires a more delicate security definition since the evaluator can mix-and-match ciphertexts corresponding to honest parties which makes the security definition much less meaningful (as it applies to a much smaller set of functions). We consider a hybrid security definition in which we support a reusable PKI but introduce session IDs and require correctness only for ciphertexts that are generated with the same session ID.

To support such functionality we construct puncturable signatures that can be punctured such that only a single message with a specific prefix is allowed. The session ID is used as a prefix and after puncturing no signature exists for messages agreeing on the prefix, except the one generated by the honest party. Our construction uses NIZKs and statistically-binding commitments.

1.3 Related Work

The problem of devising non-interactive protocols was first addressed in [13], where it was shown how to compute any function assuming that all parties share common private randomness which is unknown to the adversary.

As observed in [23], collusion-resistant non-interactive MPC in the correlated randomness model follows from multi-input functional encryption [19]. Thus, the MIFE scheme of Goldwasser et al. [19] gives an NI-MPC protocol in the correlated randomness model assuming iO and one-way functions. As noted already in [19], this is in fact tight since MIFE implies iO.

Non-interactive MPC protocols in the information-theoretic setting for a simple class of functions were constructed by [3] (see also improvements in [23]), again using correlated randomness.

2 Preliminaries

We use the following primitives in our constructions.

1. We use the notion of **indistinguishability obfuscation** (iO), as defined in [2,15].
2. We use the notion of **puncturable PRFs**, as defined in [6,7,25,30]. As observed in these works, the GGM construction [17] of PRFs from one-way functions yields puncturable PRFs.
3. We will also use the notion of **threshold multi-key FHE** [26,11,27]. The initial constructions of threshold multi-key FHE based on learning with errors assumption relied on a common random string. Recently, Dodis et al. [12] constructed a multi-key FHE scheme based on iO and DDH. We will use their scheme in our constructions.

The definitions of the above primitives are provided in the full version.

3 Building Blocks

3.1 Statistical Simulation-Extractable Zero Knowledge Proofs

Let $\mathcal{R}$ be an efficiently computable relation that consists of pairs (x, w), where x is called the statement and w is the witness. Let L denote the language consisting of statements in $\mathcal{R}$. A statistical simulation-extractable non-interactive zero-knowledge (SSE-NIZK) proof system for a language L consists of a tuple of algorithms $(\mathcal{K}, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2, \mathcal{E})$. We start by describing the basic algorithms $\mathcal{K}, \mathcal{P}, \mathcal{V}$ below:

- $\sigma \leftarrow \mathcal{K}(1^\lambda)$: On input the security parameter, it outputs a common random string (CRS) σ.
- $\pi \leftarrow \mathcal{P}(\sigma, x, w)$: On input a CRS σ, a statement x and a witness w s.t. $(x, w) \in \mathcal{R}$, it outputs a proof string π.
- $b \leftarrow \mathcal{V}(\sigma, x, \pi)$: On input a CRS σ, a statement x and a proof π, it outputs 1 or 0, denoting accept or reject.

Perfect Completeness. A non-interactive proof system is complete if an honest prover with a valid witness for a statement can convince a verifier of the validity of the statement. Formally, for every $(x, w) \in \mathcal{R}$,

$$\Pr[\sigma \leftarrow \mathcal{K}(1^\lambda); \pi \leftarrow \mathcal{P}(\sigma, x, w) : \mathcal{V}(\sigma, x, \pi) = 1] = 1$$

Statistical Soundness. A non-interactive proof system is sound if it is infeasible to convince a verifier if the statement is false. Formally, for all (possibly unbounded) adversaries $\mathcal{A}$,

$$\Pr[\sigma \leftarrow \mathcal{K}(1^k); (x, \pi) \leftarrow \mathcal{A}(\sigma) : \mathcal{V}(\sigma, x, \pi) = 1 : x \notin L] \leq \mathsf{negl}(\lambda)$$

Computational Zero Knowledge. A non-interactive proof system is computational zero knowledge if the proof does not reveal any information about the witness to the adversary. Formally, we require that for all non-uniform PPT adversaries $\mathcal{A}$, for all $(x, w) \in \mathcal{R}$,

$$\begin{aligned}&\Pr[\sigma \leftarrow \mathcal{K}(1^\lambda) : \pi \leftarrow \mathcal{P}(\sigma, x, w) : \mathcal{A}(\sigma, x, \pi) = 1] \approx \\ &\Pr[(\sigma, \tau) \leftarrow \mathcal{S}_1(1^\lambda, x) : \pi \leftarrow \mathcal{S}_2(\sigma, \tau, x) : \mathcal{A}(\sigma, x, \pi) = 1]\end{aligned}$$

Statistical Simulation-Extractability. A NIZK proof system is statistical simulation-extractable if under a simulated CRS, no proof for false statement exists, except for simulated proof for a fixed statement fed into $\mathcal{S}_1$ to generate the simulated CRS. Furthermore, using an efficient extractor algorithm $\mathcal{E}$, it is possible to extract a witness from any accepting proof generated by an unbounded adversary using the simulated CRS. Formally, for all statements x and all unbounded adversaries $\mathcal{A}$,

$$\begin{aligned}\Pr[(\sigma, \tau) \leftarrow &\mathcal{S}_1(1^\lambda, x) : \pi \leftarrow \mathcal{S}_2(\sigma, \tau, x) : (x^*, \pi^*) \leftarrow \mathcal{A}(\sigma, x, \pi) : (x^* \neq x) : \\ &1 \leftarrow \mathcal{V}(\sigma, x^*, \pi^*) : w^* \leftarrow \mathcal{E}(\sigma, \tau, x^*, \pi^*) : (x^*, w^*) \in \mathcal{R}] \geq 1 - \mathsf{negl}(\lambda)\end{aligned}$$

The Construction

Let $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ be a non-interactive witness-indistinguishable proof of knowledge (NIWI-POK) system in the common random string model. Let $\mathsf{Com}(\cdot, \cdot)$ be a non-interactive perfectly binding string commitment scheme with pseudorandom commitments. Using these ingredients, we will construct an SSE-NIZK proof system $(\mathcal{K}', \mathcal{P}', \mathcal{V}')$.

Let ℓ be the upper bound on the length of the statements to be proven and let $L = L(\ell)$ denote the length of commitments output by Com.

- $\mathcal{K}'(1^\lambda)$: Generate $\sigma \leftarrow \mathcal{K}(1^\lambda)$ and sample random strings $c_1, \ldots, c_\lambda \xleftarrow{\$} \{0,1\}^L$. Output the common random string as $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ (notice that σ' is a uniformly random string).
- $\mathcal{P}'(\sigma', x, w)$: Parse $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ and generate $\pi' \leftarrow \mathcal{P}(\sigma, x', w)$ where x' is the statement:

$$\exists \tilde{w}, r_1, \ldots, r_\lambda : \ (x, \tilde{w}) \in \mathcal{R} \vee \big(c_1 = \mathsf{Com}\,(x; r_1) \wedge \ldots \wedge c_\lambda = \mathsf{Com}\,(x; r_\lambda)\,\big). \quad (1)$$

- $\mathcal{V}'(\sigma', x, \pi')$: Parse $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ and output $\mathcal{V}(\sigma, x', \pi')$, where x' is as defined in Eq. (1).

Completeness. The completeness property of the scheme $(\mathcal{K}', \mathcal{P}', \mathcal{V}')$ follows directly from the completeness property of the underlying NIWI-POK scheme $(\mathcal{K}, \mathcal{P}, \mathcal{V})$.

Statistical Soundness. Let $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ be a CRS sampled at random by $\mathcal{K}$. With overwhelming probability, there does not exist $x, r_1, \ldots, r_\lambda$ s.t. for every i, $c_i = \mathsf{Com}(x; r_i)$.

Now, for any $x \notin L$, let x' be the corresponding statement as defined in Eq. (1). It follows from above that the second part of the statement x' is false. Then, from the statistical soundness of $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, it follows that there does not exist any accepting proof for x'.

Zero-Knowledge and Statistical Simulation-Extractability. We first describe the simulator and extractor algorithms $(\mathcal{S}_1', \mathcal{S}_2', \mathcal{E}')$ below. Let $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ denote the extractor for the NIWI-POK scheme $(\mathcal{K}, \mathcal{P}, \mathcal{V})$.

- $\mathcal{S}_1'(1^\lambda, x)$: On input a statement x, it first computes $(\sigma, \tau) \leftarrow \mathcal{E}_1(1^\lambda)$. Next, for every $i \in [\lambda]$, it samples a random string r_i and computes $c_i \leftarrow \mathsf{Com}(x; r_i)$. It sets $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$, $\tau_1' = (r_1, \ldots, r_\lambda)$, $\tau_2' = \tau$ and outputs $(\sigma', \tau_1', \tau_2')$.
- $\mathcal{S}_2'(\sigma', \tau_1', x)$: It sets $w = (r_1, \ldots, r_\lambda)$ and computes $\pi \leftarrow \mathcal{P}(\sigma', x', w)$ where x' is as defined in Eq. (1). Note that here the honest prover algorithm uses w to prove the second part of x'.
- $\mathcal{E}'(\sigma', \tau_2', x^*, \pi^*)$: It parses $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ and outputs the value returned by the extractor $\mathcal{E}_2(\sigma, \tau_2', x^*, \pi^*)$.

We first argue computational zero-knowledge property of our scheme. Let $x \in L$ be any statement. Consider the following sequence of hybrid experiments:

– H_0: In this experiment, the CRS $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ is honestly generated and we compute a proof π' for x using the honest prover algorithm.
– H_1: Same as above, except that the CRS $\sigma' = (\sigma, c_1, \ldots, c_\lambda)$ is computed using the simulator algorithm $\mathcal{S}'_1(1^\lambda, x)$ as described above. Let (τ'_1, τ'_2) be the trapdoor computed by $\mathcal{S}'_1$.
– H_2: Same as above, except that we now compute π' using the simulator algorithm $\mathcal{S}'_2(\sigma', x, \tau'_1)$.

In order to prove the zero knowledge property of $(\mathcal{K}', \mathcal{P}', \mathcal{V}')$, it suffices to show that H_0 and H_2 are computationally indistinguishable. The indistinguishability of H_0 and H_1 follows immediately from the hiding property of the commitment scheme Com. Further, the indistinguishability of H_1 and H_2 follows from the witness indistinguishability property of the underlying NIWI-POK $(\mathcal{K}, \mathcal{P}, \mathcal{V})$. Put together, we have that H_0 and H_2 are computationally indistinguishable.

We now argue statistical simulation-extractability. We first note that if $x \notin L$, then in experiment H_2 described above, x is the only false statement for which an accepting proof exists. This follows from the perfectly binding property of Com and the statistical soundness property of the scheme. Now, let (x^*, π^*) be a statement and proof output by an unbounded adversary $\mathcal{A}$ who is given σ' s.t. $\mathcal{V}(\sigma', x^*, \pi^*) = 1$. Since $x^* \neq x$, it follows from above that $x^* \in L$. We now run the extractor $\mathcal{E}'(\sigma', \tau'_2, x^*, \pi^*)$ to compute w^*. From the proof of knowledge property of the underlying NIWI-POK $(\mathcal{K}, \mathcal{P}, \mathcal{V})$, it follows that w^* is a valid witness for x^*, except with negligible probability.

3.2 Puncturable Signatures

We define a special kind of signature scheme which is puncturable at any prefix, such that after puncturing no signature exists for messages agreeing on the prefix. A puncturable signature scheme is a tuple of efficient algorithms $\mathsf{PuncSig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Puncture})$ described as follows:

– $(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: is a randomized algorithm which takes as input the security parameter and outputs a key pair.
– $\sigma \leftarrow \mathsf{Sign}(sk, m)$: is a randomized algorithm which takes as input the signing key sk, some message m and outputs a signature σ.
– $b \leftarrow \mathsf{Verify}(vk, m, \sigma)$: is a deterministic algorithm which takes as input the verification key, a message m, and a signature σ and outputs a bit b.
– $vk_{m,s} \leftarrow \mathsf{Puncture}(sk, m, s)$: is a randomized algorithm which takes as input the signing key, a message m, a prefix s of m, and outputs a punctured verification key $vk_{m,s}$.

We require following properties from the scheme.

1. **Correctness:** For any message $m \in \{0,1\}^\lambda$ it holds that

$$\Pr[\mathsf{Verify}(vk, m, \sigma) = 1 : (sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda), \sigma \leftarrow \mathsf{Sign}(sk, m)] = 1$$

2. **Security:** For any message m, and prefix s of m it holds that:

$$\{vk_{m,s}, m, \sigma\} \approx_c \{vk, m, \sigma\},$$

   where $(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\sigma \leftarrow \mathsf{Sign}(sk, m)$, $vk_{m,s} \leftarrow \mathsf{Puncture}(sk, m, s)$.

3. **Punctured functionality:** There exist a negligible function $\mathsf{negl}(\cdot)$ such that for any message m and for any $m' \neq m$ such that s is a prefix of m' it holds that

$$\Pr[\exists \sigma : \mathsf{Verify}(vk_{m,s}, m', \sigma) = 1] \leq \mathsf{negl}(\lambda),$$

   where $(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $vk_{m,s} \leftarrow \mathsf{Puncture}(sk, m, s)$.

The Construction

We show how to construct a puncturable signature scheme from NIZK proofs and a statistically binding commitment scheme. The construction is as follows:

– $\mathsf{KeyGen}(1^\lambda)$: Sample a PRF key K, sample a crs, compute $c \leftarrow \mathsf{Com}(C_K; r)$ where C is a circuit that on input x outputs $\mathsf{PRF}_K(x)$ (padded to be large enough). Set $sk = C_K, r$ and $vk = c, \mathsf{crs}$.
– $\mathsf{Sign}(sk, m)$: Compute $y = C_K(m)$ and a proof π that y is indeed that output of the circuit committed in c on the input m.
– $\mathsf{Verify}(vk, m, \sigma)$: Parse σ as y and π. Verify that proof π to the instance (m, y) and that $y \neq \perp$.
– $\mathsf{Puncture}(sk, m, s)$: Compute PRF key K^* that is punctured at the prefix s except the message m. That is, using K^* one can compute the PRF value on m and on all inputs that do not begin with s. In the verification key, replace the PRF key in the committed circuit to $C_{K^*,s,m}$. In the secret key, replace K with K^*.

Correctness is immediate from the construction. Security follows from the hiding property of the commitment scheme. Punctured security follows from the statistical soundness of the NIZK proof. After the circuit C is altered to output $\perp$ on messages with prefix s, we know that (with high probability over the crs) no valid proof exists for messages with prefix s, and thus no valid signatures.

4 Non-Interactive Multiparty Computation

A non-interactive multiparty computation protocol (NI-MPC) for a function $f\colon \mathcal{X}^n \to \mathcal{Y}$ is a protocol between n parties and a single evaluator. Each party holds an input x_i and sends exactly one message to an evaluator, who computes the output. The correctness requirement of the protocol is that the evaluator, given one message from each party, is able to compute the value $y = f(x_1, \ldots, x_n)$ correctly.

Our security guarantee is formalized as an indistinguishability game in which the evaluator commits on a set of parties with whom he is colluding. Specifically, since our setting is non-interactive, the evaluator can use the controlled parties to make resetting attacks that allow him to evaluate the function on different inputs of his choice (but he has no control over the inputs of the honest parties). Thus, our security is assuming that the "residual" function determined after fixing the challenge inputs of the honest parties is functionally equivalent in the view of the evaluator [24].

Definition 1 (Admissible inputs). *Let $f\colon \mathcal{X}^n \to \mathcal{Y}$ be a function. We say that $H \subseteq [n]$, $\{x_i^0\}_{i\in H}$ and $\{x_i^1\}_{i\in H}$ are* admissible *for f if for any $\{x_i\}_{i\notin H}$ it holds that*

$$f(\langle\{x_i^0\}_{i\in H}, \{x_i\}_{i\notin H}\rangle) = f(\langle\{x_i^1\}_{i\in H}, \{x_i\}_{i\notin H}\rangle),$$

where $\langle\cdot,\dots,\cdot\rangle$ sorts the inputs according to the index i.

Definition 2 (Non-interactive multiparty computation). *A non-interactive multiparty computation protocol Π for the function $f\colon \mathcal{X}^n \to \mathcal{Y}$ consists of a probabilistic setup procedure* Setup*, a probabilistic encryption procedure* Enc *and a probabilistic evaluation procedure* Eval *that satisfy the following requirements:*

1. $\mathsf{Setup}(1^\lambda, \mathsf{crs}, i)$ *takes as input a security parameter λ, a common random string* crs*, a party index $i \in [n]$, computes a private key SK_i and outputs a public key PK_i.*
2. $\mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i, \mathsf{PK}_1, \dots, \mathsf{PK}_n, \mathsf{SK}_i, i)$ *takes as input a security parameter λ, a common random string* crs*, an input $x_i \in \mathcal{X}$, public keys $\mathsf{PK}_1, \dots, \mathsf{PK}_n$, a secret key SK_i and a party index $i \in [n]$, and outputs a ciphertext $\widehat{x_i}$.*
3. $\mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_1}, \dots, \widehat{x_n}, \mathsf{PK}_1, \dots, \mathsf{PK}_n)$ *takes as input a security parameter λ, a common random string* crs*, n ciphertexts $\widehat{x_1}, \dots, \widehat{x_n}$, n public keys $\mathsf{PK}_1 \dots, \mathsf{PK}_n$, and outputs a value $y \in \mathcal{Y}$.*
4. ***Correctness:*** *This property states that the evaluation on n ciphertexts of $x_1, \dots, x_n$ gives $f(x_1, \dots, x_n)$. Specifically, for every $\lambda \in \mathbb{N}$ and every $x_1, \dots, x_n \in \mathcal{X}$, it holds that*

$$\begin{aligned}\Pr\big[y = f(x_1, \dots, x_n);\\ &y = \mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_1}, \dots, \widehat{x_n}, \mathsf{PK}_1, \dots, \mathsf{PK}_n),\\ &\forall i \in [n]\colon\ \widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i, \mathsf{PK}_1, \dots, \mathsf{PK}_n, \mathsf{SK}_i, i),\\ &\forall i \in [n]\colon\ (\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{crs}, i),\\ &\mathsf{crs} \leftarrow \{0,1\}^\lambda\big] = 1,\end{aligned}$$

   where the probability is over the internal randomness of $\mathsf{Setup}, \mathsf{Enc}$ *and* Eval.
5. ***Security:*** *Informally speaking, this property states that an adversary that controls a subset of parties does not learn anything about the underlying plaintexts from the ciphertexts of the remaining parties, besides the output of the function. Specifically, for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, any admissible $H \subseteq [n]$, $\boldsymbol{x}^0 = \{x_i^0\}_{i\in H}$ and $\boldsymbol{x}^1 = \{x_i^1\}_{i\in H}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1}(\lambda) \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1}(\lambda) = 1\right] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the random variable $\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1}(\lambda)$ *is defined via the following experiment:*

1. *Sample* $\mathsf{crs} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ *and* $b \leftarrow \{0,1\}$ *at random.*
2. *For* $i \in H$*:* $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{crs}, i)$
3. $(\{\mathsf{PK}_i\}_{i\notin H}, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{crs}, \{\mathsf{PK}_i\}_{i\in H})$
4. *For* $i \in H$*:* $\widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i^b, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i)$*.*
5. $b' \leftarrow \mathcal{A}_2(1^\lambda, \mathsf{crs}, \{\mathsf{PK}_i\}_{i\in H}, \{\widehat{x_i}\}_{i\in H}, \mathsf{st})$*.*
6. *If* $b' = b$ *then output* 1*, and otherwise output* 0*.*

A reusable PKI and session IDs. The above definition is not reusable. Namely, it cannot be used across multiple ciphertexts generated by the honest parties. A definition that support multiple encryption sessions is given below. We remark that in our non-interactive setting, session IDs are required to prevent an adversary for combining ciphertexts from one session with another (otherwise, the definition of an admissible function becomes very weak and makes sense only for a very restricted set of functions). Our construction and proof given in Section 5 satisfy the weaker definition above but we sketch in Section 5.2 how to prove that our construction satisfies the stronger reusable-PKI with session IDs notion as well.

Definition 3 (Non-interactive multiparty computation with session IDs). *A non-interactive multiparty computation protocol Π for the function $f\colon \mathcal{X}^n \to \mathcal{Y}$ consists of a probabilistic setup procedure* Setup*, a probabilistic encryption procedure* Enc *and a probabilistic evaluation procedure* Eval*. The setup procedure and evaluation procedure have the same syntax as in Definition 2 but* Enc *also received a session ID. Moreover, the correctness and security are modified as follows:*

1. $\mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i, \mathit{session})$ *takes as input a security parameter λ, a common random string* crs*, an input $x_i \in \mathcal{X}$, public keys* $\mathsf{PK}_1, \ldots, \mathsf{PK}_n$*, a secret key* SK_i*, a party index $i \in [n]$, and a session ID $\mathit{session} \in \{0,1\}^\lambda$, and outputs a ciphertext $\widehat{x_i}$.*
2. ***Correctness:*** *This property states that the evaluation on n ciphertexts of $x_1, \ldots, x_n$ encrypted in the same session gives $f(x_1, \ldots, x_n)$. Specifically, for every $\lambda \in \mathbb{N}$, every session ID $\mathit{session} \in \{0,1\}^\lambda$, and every $x_1, \ldots, x_n \in \mathcal{X}$, it holds that*

$$\begin{aligned}\Pr\big[&y = f(x_1, \ldots, x_n);\\ &y = \mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_1}, \ldots, \widehat{x_n}, \mathsf{PK}_1, \ldots, \mathsf{PK}_n),\\ &\forall i \in [n]\colon\ \widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i, \mathit{session}),\\ &\forall i \in [n]\colon\ (\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{crs}, i),\\ &\mathsf{crs} \leftarrow \{0,1\}^\lambda\big] = 1,\end{aligned}$$

where the probability is over the internal randomness of $\mathsf{Setup}, \mathsf{Enc}$ *and* Eval*.*

3. ***Security:*** *For any probabilistic polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, any admissible* $H \subseteq [n]$, $\boldsymbol{x}^0 = \{x_i^0\}_{i \in H}$ *and* $\boldsymbol{x}^1 = \{x_i^1\}_{i \in H}$*, and any session ID* $\mathsf{session} \in \{0,1\}^\lambda$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1,\mathsf{session}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1,\mathsf{session}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where* $\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1,\mathsf{session}}(\lambda)$ *is a random variable defined via the following experiment:*
   1. *Sample* $\mathsf{crs} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ *and* $b \leftarrow \{0,1\}$ *at random.*
   2. *For* $i \in H$*:* $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{crs}, i)$
   3. $(\{\mathsf{PK}_i\}_{i \notin H}, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{crs}, \{\mathsf{PK}_i\}_{i \in H})$
   4. *For* $i \in H$*:* $\widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i^b, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i, \mathsf{session})$*.*
   5. $b' \leftarrow \mathcal{A}_2^{\mathsf{Enc}(\cdot,\cdot,\cdot)}(1^\lambda, \mathsf{crs}, \{\mathsf{PK}_i\}_{i \in H}, \{\widehat{x_i}\}_{i \in H}, \mathsf{st})$*, where the encryption oracle* $\mathsf{Enc}(\cdot,\cdot,\cdot)$ *on input triple* $(x_i, i, \mathsf{session}')$ *produces an encryption of* x_i *with respect to party* $i \in H$ *as long as* $\mathsf{session}' \neq \mathsf{session}$ *(by running* $\mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i, \mathsf{session}')$*).*
   6. *If* $b' = b$ *then output* 1*, and otherwise output* 0*.*

The fully honest case. A particularly interesting case is when $H = [n]$ and the evaluator does not collude with any party. In this case, which we call the fully honest case, we can assume without loss of generality that one of the parties will also generate the CRS as part of the PKI and thus we get a construction in the PKI model.

Selective vs. adaptive security. Our security definitions are selective in the sense that the adversary commits on the challenge before the challenger published the PKI. By standard "random guessing" we can turn any selectively secure scheme into an adaptively secure one. This is done by guessing ahead of time all the adaptive choices made by the adversary throughout the game and reducing to the selective case. By setting the security parameter to be large enough and using the sub-exponential security of the scheme, we can tolerate this exponential loss. (Recall that our main result assumes sub-exponentially secure primitives to begin with so we can get adaptive security for free.)

4.1 Communication on a Chain

We follow Halevi et al. [23] and consider the case of more general interaction patterns described by a directed acyclic graph (DAG), where each node represents a party who expects to receive messages from all of its parents and can then send messages to all of its children, and where the sinks of the graph compute outputs. The setting of Definitions 2 and 3 is a special case of the above where the communication pattern is a *star*, a graph connecting all nodes to a single central node. Another special case is a chain, a simple directed path traversing all nodes.

Here, we focus on a chain and define NI-MPC for this pattern. We assume that there are n parties with corresponding inputs x_i. The first party is the source of the chain and party n is the last party in the chain which sends its message to the evaluator.

Definition 4 (Chain admissible inputs). *Let $f\colon \mathcal{X}^n \to \mathcal{Y}$ be a function. Let $H \subseteq [n]$ with maximal index i^*. We say that H, $\{x_i^0\}_{i\in H}$, $\{x_i^1\}_{i\in H}$, and $\{x_i\}_{i\notin H \wedge i\le i^*}$ are chain-admissible for f if for any $\{x_i\}_{i\notin H\wedge i>i^*}$ it holds that*

$$f(\langle\{x_i^0\}_{i\in H}, \{x_i\}_{i\notin H}\rangle) = f(\langle\{x_i^1\}_{i\in H}, \{x_i\}_{i\notin H}\rangle),$$

where $\langle\cdot,\ldots,\cdot\rangle$ sorts the inputs according to the index i.

Definition 5 (Non-interactive multiparty computation on a chain). *A non-interactive multiparty computation protocol Π for the function $f\colon \mathcal{X}^n \to \mathcal{Y}$* on a chain *consists of a probabilistic setup procedure* Setup*, a probabilistic encryption procedure* Enc *and a probabilistic evaluation procedure* Eval *that satisfy the following requirements:*

1. $\mathsf{Setup}(1^\lambda, \mathsf{crs}, i)$ *takes as input a security parameter λ, a common random string* crs*, a party index $i \in [n]$, computes a private key SK_i and outputs a public key PK_i.*
2. $\mathsf{Enc}(1^\lambda, \mathsf{crs}, \widehat{x_{i-1}}, x_i, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i)$ *takes as input a security parameter λ, a common random string* crs*, the message sent by party $i-1$ an input $x_i \in \mathcal{X}$, public keys $\mathsf{PK}_1, \ldots, \mathsf{PK}_n$, a secret key SK_i and a party index $i \in [n]$, and outputs a ciphertext $\widehat{x_i}$. We assume that $\widehat{x_0} = \perp$.*
3. $\mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_n}, \mathsf{PK}_1, \ldots, \mathsf{PK}_n)$ *takes as input a security parameter λ, a common random string* crs*, the ciphertexts of the last party $\widehat{x_n}$, n public keys $\mathsf{PK}_1 \ldots, \mathsf{PK}_n$, and outputs a value $y \in \mathcal{Y}$.*
4. ***Correctness:*** *This property states that the evaluation on n ciphertexts of $x_1, \ldots, x_n$ gives $f(x_1, \ldots, x_n)$. Specifically, for every $\lambda \in \mathbb{N}$ and every $x_1, \ldots, x_n \in \mathcal{X}$, it holds that*

$$\begin{aligned}\Pr\big[&y = f(x_1, \ldots, x_n);\\ &y = \mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_n}, \mathsf{PK}_1, \ldots, \mathsf{PK}_n),\\ &\forall i \in [n]\colon\ \widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, \widehat{x_{i-1}}, x_i, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i),\\ &\forall i \in [n]\colon\ (\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{crs}, i),\\ &\mathsf{crs} \leftarrow \{0,1\}^\lambda\big] = 1,\end{aligned}$$

   where the probability is over the internal randomness of $\mathsf{Setup}, \mathsf{Enc}$ *and* Eval.
5. ***Security:*** *For any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, any chain-admissible $H \subseteq [n]$, $\boldsymbol{x}^0 = \{x_i^0\}_{i\in H}$, $\boldsymbol{x}^1 = \{x_i^1\}_{i\in H}$, and $\boldsymbol{x} = \{x_i\}_{i\notin H\wedge i<i^*}$, where i^* is the maximal index in H, there exists a negligible function $\mathsf{negl}(\lambda)$ such that*

$$\mathsf{Adv}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1,\boldsymbol{x}}(\lambda) \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1,\boldsymbol{x}}(\lambda) = 1\right] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where* $\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1,\boldsymbol{x}}(\lambda)$ *is a random variable defined via the following experiment:*

1. *Sample* $\mathsf{crs} \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$ *and* $b \leftarrow \{0,1\}$ *at random.*
2. *For* $i \in H$*:* $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{crs}, i)$
3. $(\{\mathsf{PK}_i\}_{i \notin H}, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{crs}, \{\mathsf{PK}_i\}_{i \in H})$
4. *For* $i = 1 \dots i^*$*:*
   i. *If* $i \in H$*:* $\widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, \widehat{x_{i-1}}, x_i^b, \mathsf{PK}_1, \dots, \mathsf{PK}_n, \mathsf{SK}_i, i)$.
   ii. *If* $i \notin H$*:* $\widehat{x_i} \leftarrow \mathsf{Enc}(1^\lambda, \mathsf{crs}, \widehat{x_{i-1}}, x_i, \mathsf{PK}_1, \dots, \mathsf{PK}_n, \mathsf{SK}_i, i)$
5. $b' \leftarrow \mathcal{A}_2(1^\lambda, \mathsf{crs}, \{\mathsf{PK}_i\}_{i \in H}, \widehat{x_{i^*}}, \mathsf{st})$.
6. *If* $b' = b$*, then output* 1*, and otherwise, output* 0.

5 Our Construction

We now present our construction of NI-MPC as in Definition 2. The main ingredients in our construction are:

- A threshold multi-key FHE scheme $\mathsf{MFHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Expand}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{PartDec}, \mathsf{FinDec})$.
- An SSE-NIZK proof system $\mathsf{NIZK} = (\mathcal{K}, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2, \mathcal{E})$ for **NP**.
- A puncturable PRF family $\mathcal{F}$.
- A puncturable signature scheme $\mathsf{PuncSig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Puncture})$
- An indistinguishability obfuscator iO for general circuits.

Let $f\colon \mathcal{X}^n \to \mathcal{Y}$ be a deterministic function that takes n inputs $x_1, \dots, x_n \in \mathcal{X}$ and outputs $y \in \mathcal{Y}$. We construct a non-interactive multiparty computation protocol $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Eval})$ for f below.

$\underline{\mathsf{Setup}(1^\lambda, \mathsf{crs}, i)}$: It takes as input security parameter λ, a common random string crs, and does the following.

1. Execute the key generation procedure of the signature scheme, $(sk_i, vk_i) \leftarrow \mathsf{PuncSig.KeyGen}(1^\lambda)$.
2. Execute $\{(dk_i, ek_i) \leftarrow \mathsf{MFHE.KeyGen}(1^\lambda)\}_{i \in [n]}$.
3. Output public key $\mathsf{PK}_i = (vk_i, ek_i)$ and private key $\mathsf{SK}_i = (sk_i, dk_i)$.

$\underline{\mathsf{Enc}(1^\lambda, \mathsf{crs}, x_i, \mathsf{PK}_1, \dots, \mathsf{PK}_n, \mathsf{SK}_i, i)}$: It takes as input security parameter λ, a common random string crs, an input $x_i \in \mathcal{X}$, a list of n public keys $\mathsf{PK}_1 \dots, \mathsf{PK}_n$, one private key SK_i, an index $i \in [n]$, and does the following.

1. For every $j \in [n]$, parse $\mathsf{PK}_j = (vk_j, ek_j)$ and let $\mathsf{SK}_i = (sk_i, dk_i)$.
2. Encrypt the input x_i using ek_i: $\mathsf{CT}_i \leftarrow \mathsf{MFHE.Enc}(ek_i, x_i)$.
3. Sign on (CT_i, i) using sk_i: $\psi_i \leftarrow \mathsf{PuncSig.Sign}(sk_i, (\mathsf{CT}_i, i))$.
4. Sample puncturable PRF key $K_i \xleftarrow{\$} \{0,1\}^\lambda$.
5. Generate a proof π_i that CT_i and ek_i are valid ciphertext and encryption key, respectively. Namely, compute $\pi_i \leftarrow \mathsf{NIZK}.\mathcal{P}(\mathsf{crs}, (\mathsf{CT}_i, ek_i), w = (r, r', x_i, dk_i))$, where r and r' are the randomness used for the computation of CT_i and (dk_i, ek_i), respectively. The exact statement is given in Figure 2.

$$G\left[i, \mathsf{crs}, K_i, dk_i, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}\right]$$

Input: $(\mathsf{CT}_1, \psi_1, \pi_1), \ldots, (\mathsf{CT}_n, \psi_n, \pi_n)$.

- If $\mathsf{PuncSig.Verify}(vk_j, (\mathsf{CT}_j, j), \psi_j)$ is false for some $j \in [n]$ then output $\perp$.
- For every $j \in [n]$, verify $\mathsf{NIZK}.\mathcal{V}(\mathsf{crs}, (ek_j, \mathsf{CT}_j), \pi_j)$ for the statement:

  $$\exists r, r', M_j, dk_j : \mathsf{CT}_j = \mathsf{MFHE.Enc}(ek_j, M_j; r) \ \wedge\ (ek_j, dk_j) = \mathsf{MFHE.KeyGen}(1^\lambda; r').$$

  If this check fails for some $j \in [n]$ then output $\perp$.
- For all $j \in [n]$ compute $\widehat{\mathsf{CT}}_j \leftarrow \mathsf{MFHE.Expand}((ek_1, \ldots, ek_n), j, \mathsf{CT}_j)$.
- Perform $\widehat{\mathsf{CT}_{\mathsf{out}}} \leftarrow \mathsf{MFHE.Eval}\left(f, \widehat{\mathsf{CT}_1}, \ldots, \widehat{\mathsf{CT}_n}\right)$.
- $r_i \leftarrow \mathsf{PRF}_{K_i}(\mathsf{CT}_1 \| \pi_1 \| \ldots \| \mathsf{CT}_n \| \pi_n)$.
- $p_i \leftarrow \mathsf{MFHE.PartDec}\left(\widehat{\mathsf{CT}_{\mathsf{out}}}, ek_1, \ldots, ek_n, i, dk_i;\ r_i\right)$.
- Output p_i.

Fig. 2. The circuit G.

6. Obfuscate the circuit $\overline{G_i} \leftarrow \mathsf{iO}\big(1^\lambda, G\left[i, \mathsf{crs}, K_i, dk_i, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}\right]\big)$ as described in Figure 2.
7. Output the tuple $\widehat{x_i} = (\overline{G_i}, \mathsf{CT}_i, \psi_i, \pi_i)$.

<u>$\mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_1}, \ldots, \widehat{x_n}, \mathsf{PK}_1, \ldots, \mathsf{PK}_n)$</u>: It takes as input security parameter λ, a common random string crs, n strings $\widehat{x_1}, \ldots, \widehat{x_n}$, a list of n public keys $\mathsf{PK}_1 \ldots, \mathsf{PK}_n$, and does the following:

1. Parse each $\widehat{x_i} = (\overline{G_i}, \mathsf{CT}_i, \psi_i, \pi_i)$.
2. Evaluate each obfuscation $\overline{G_i}$ on the input $\left((\widehat{\mathsf{CT}}_1, \psi_1, \pi_1), \ldots, (\widehat{\mathsf{CT}}_n, \psi_n, \pi_n)\right)$ to get the partial decryption p_i.
3. Execute the final decryption algorithm, $y \leftarrow \mathsf{MFHE.FinDec}(p_1, \ldots, p_n)$ and output y.

We proceed with the proof of correctness and security of the scheme. Correctness follows by the correctness of the underlying building blocks. Specifically, let $x_1, \ldots, x_n \in \mathcal{X}$ be inputs such that party i holds x_i. By the correctness of the signature scheme and the threshold multi-key FHE scheme, each obfuscated circuit will output a partial decryption p_i such that $\mathsf{MFHE.FinDec}(p_1, \ldots, p_n)$ must be equal to $f(x_1, \ldots, x_n)$.

5.1 Proof of Security

We show that our scheme is secure for a restricted set of input vector pairs.

Lemma 1. *Assume the existence of a indistinguishability obfuscator, a threshold multi-key FHE scheme, an SSE-NIZK in the common random string model, a* *puncturable PRF, and a puncturable signature, all of which are sub-exponentially secure. Then, for any $H \subseteq [n]$, any $\mathsf{ind} \in H$ and any $\boldsymbol{x}^0 = \{x_i\}_{i\in H}$, $\boldsymbol{x}^1 = \{x_i\}_{i\in H\setminus\{\mathsf{ind}\}} \cup \{x'_{\mathsf{ind}}\}$, such that $H, \boldsymbol{x}^0, \boldsymbol{x}^1$ are admissible, it holds that*

$$\mathsf{Adv}^{\mathsf{NI\text{-}MPC}}_{\Pi,f,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1}(\lambda) \leq \mathsf{negl}(\lambda).$$

Given this lemma, we can construct a scheme that is secure as in Definition 2.

Lemma 2. *Assume the existence of a indistinguishability obfuscator, a threshold multi-key FHE scheme, an SSE-NIZK in the common random string model, a puncturable PRF, and a puncturable signature, all of which are sub-exponentially secure. Then, there exists a secure non-interactive multiparty computation scheme for any efficiently computable function f.*

The proof of Lemma 2 (given Lemma 1) is given in the full version of the paper.

Threshold multi-key FHE scheme exists based on the Learning with Errors assumption [27] in the CRS model or on indistinguishability obfuscation and DDH [12]. In Section 3.1 we constructed an SSE-NIZK scheme in the common random string model assuming any NIWI-POK and one-way permutations. A NIWI-POK can be constructed from any NIWI together with a standard encryption scheme. NIWI and one-way permutations exist based on iO and DDH [4]. Finally, we constructed a puncturable signature based on NIZKs and statistically-binding commitments. In total, we can instantiate our construction based on indistinguishability obfuscation and DDH, both with sub-exponential security.

Proof (of Lemma 1). Fix $H \subseteq [n]$, $\mathsf{ind} \in H$, $\boldsymbol{x}^0 = \{x_i\}_{i\in H}$, and $\boldsymbol{x}^1 = \{x_i\}_{i\in H\setminus\{\mathsf{ind}\}} \cup \{x'_{\mathsf{ind}}\}$, such that $H, \boldsymbol{x}^0, \boldsymbol{x}^1$ are admissible. The lemma is proved by a sequence of hybrid experiments. Denote by $\ell(\lambda)$ a polynomial bounding the total length of n ciphertexts and n NIZK proofs computed with security parameter λ.

$\underline{\mathsf{Hyb}_1}$: This experiment corresponds to the original experiment $\mathsf{Exp}^{\mathsf{NI\text{-}MPC}}_{\Pi,f,\mathcal{A},H,\boldsymbol{x}^0,\boldsymbol{x}^1}(\lambda)$:

1. Sample a random bit $b \in \{0,1\}$.
2. Do the following for party $i \in H$:
   - 2.1. Sample $(dk_i, ek_i) \leftarrow \mathsf{MFHE.KeyGen}(1^\lambda; r)$, where r is a random string.
   - 2.2. $\mathsf{CT}_i \leftarrow \mathsf{MFHE.Enc}(ek_i, x_i^b; r')$, where r' is a random string.
   - 2.3. Sample $(sk_i, vk_i) \leftarrow \mathsf{PuncSig.KeyGen}(1^\lambda)$.
   - 2.4. Set $\mathsf{PK}_i = (vk_i, ek_i)$ and $\mathsf{SK}_i = (sk_i, dk_i)$.
3. The adversary, given $\{\mathsf{PK}_i\}_{i\in H}$ and crs, publishes a public key PK_i for every $i \notin H$ of the form $\mathsf{PK}_i = (vk_i, ek_i)$.
4. Do the following for party $i \in H$:
   - 4.1. $\psi_i \leftarrow \mathsf{PuncSig.Sign}(sk_i, (\mathsf{CT}_i, i))$.
   - 4.2. Sample a PRF key K_i.
   - 4.3. Compute $\pi_i \leftarrow \mathsf{NIZK}.\mathcal{P}(\mathsf{crs}, (\mathsf{CT}_i, ek_i), w = (r, r', x_i^b, dk_i))$.
   - 4.4. Compute $\overline{G_i}$ as the obfuscation of the following circuit defined in Figure 2:
$$G\left[i, \mathsf{crs}, K_i, dk_i, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}\right].$$

4.5. Let $\widehat{x_i} = (\overline{G_i}, \mathsf{CT}_i, \psi_i, \pi_i)$.
5. The challenge: $b' \leftarrow \mathcal{A}(\mathsf{crs}, \{\mathsf{PK}_i\}_{i\in H}, \{\widehat{x_i}\}_{i\in H})$ and output b'.

<u>Hyb_2</u>: This experiment corresponds to experiment Hyb_1 except that now the published verification keys of parties in H are modified to verify only one precomputed message and nothing else. Specifically, now between Item 2.3. and Item 2.4. we puncture the verification key of each $i \in H$ and set

$$vk_i \leftarrow \mathsf{PuncSig.Puncture}(sk_i, (\mathsf{CT}_i, i), \perp),$$

where the $\perp$ corresponds to an empty prefix. That is, the new verification key accepts the precisely one message which is (CT_i, i). The indistinguishability of this experiment and experiment Hyb_1 follows directly from the security of the signature scheme (see Item 2 at Section 3.2).

<u>Hyb_3</u>: This experiment corresponds to experiment Hyb_2 except that now the NIZK proof of the honest party ind is generated via the zero-knowledge simulator.

1. Sample $(dk_{\mathsf{ind}}, ek_{\mathsf{ind}}) \leftarrow \mathsf{MFHE.KeyGen}(1^\lambda; r)$, where r is a random string.
2. $\mathsf{CT}_{\mathsf{ind}} \leftarrow \mathsf{MFHE.Enc}(ek_{\mathsf{ind}}, x^b_{\mathsf{ind}}; r')$, where r' is a random string.
3. <u>Sample $(\mathsf{crs}, \tau) \leftarrow \mathsf{NIZK}.\mathcal{S}_1(1^\lambda, (\mathsf{CT}_{\mathsf{ind}}, ek_{\mathsf{ind}}))$</u>.
4. Sample random a bit $b \in \{0,1\}$.
5. Proceed as before (Item 2) for party $i \in H \setminus \{\mathsf{ind}\}$.
6. Do the following for party ind:
   6.1. Sample $(sk_{\mathsf{ind}}, vk_{\mathsf{ind}}) \leftarrow \mathsf{PuncSig.KeyGen}(1^\lambda)$.
   6.2. $vk_{\mathsf{ind}} \leftarrow \mathsf{PuncSig.Puncture}(sk_{\mathsf{ind}}, (\mathsf{CT}_{\mathsf{ind}}, i), \perp)$.
   6.3. Set $\mathsf{PK}_i = (vk_{\mathsf{ind}}, ek_{\mathsf{ind}})$ and $\mathsf{SK}_{\mathsf{ind}} = (sk_{\mathsf{ind}}, dk_{\mathsf{ind}})$.
7. The adversary, given $\{\mathsf{PK}_i\}_{i\in H}$ and crs, publishes a public key PK_i for every $i \notin H$ of the form $\mathsf{PK}_i = (vk_i, ek_i)$.
8. Proceed as before (Item 4) for party $i \in H \setminus \{\mathsf{ind}\}$.
9. Do the following for party ind:
   9.1. Sample a PRF key K_{ind}.
   9.2. <u>$\pi_{\mathsf{ind}} \leftarrow \mathsf{NIZK}.\mathcal{S}_2(\mathsf{crs}, \tau, (\mathsf{CT}_{\mathsf{ind}}, ek_{\mathsf{ind}}))$</u>.
   9.3. Compute $\overline{G_{\mathsf{ind}}}$ as the obfuscation of the following circuit defined in Figure 2:
   $$G\left[\mathsf{ind}, \mathsf{crs}, K_{\mathsf{ind}}, dk_{\mathsf{ind}}, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in H}\right].$$
   9.4. Let $\widehat{x_{\mathsf{ind}}} = (\overline{G_{\mathsf{ind}}}, \mathsf{CT}_{\mathsf{ind}}, \psi_{\mathsf{ind}}, \pi_{\mathsf{ind}})$.
10. The challenge: $b' \leftarrow \mathcal{A}(\mathsf{crs}, \{\mathsf{PK}_i\}_{i\in H}, \{\widehat{x_i}\}_{i\in H})$ and output b'.

The indistinguishability of this experiment and experiment Hyb_2 follows directly from the zero-knowledge property of the NIZK (see definition in Section 3.1), since the adversary does not get the trapdoor τ as input. The proof here relies on the fact that the adversary has to commit to his challenge input vectors before seeing the common random string.

<u>$\mathsf{Hyb}_{4,1,\{\mathsf{CT}^*_i,\pi^*_i\}_{i\notin H}}$ for $\{\mathsf{CT}^*_i, \pi^*_i\}_{i\notin H} \in \{0,1\}^{\ell(\lambda)}$:</u>

1. Sample $(dk_{\mathsf{ind}}, ek_{\mathsf{ind}}) \leftarrow \mathsf{MFHE.KeyGen}(1^\lambda; r)$, where r is a random string.
2. $\mathsf{CT}_{\mathsf{ind}} \leftarrow \mathsf{MFHE.Enc}(ek_{\mathsf{ind}}, x^b_{\mathsf{ind}}; r')$, where r' is a random string.
3. Sample $(\mathsf{crs}, \tau) \leftarrow \mathsf{NIZK}.\mathcal{S}_1(1^\lambda, (\mathsf{CT}_{\mathsf{ind}}, ek_{\mathsf{ind}}))$.
4. Sample random a bit $b \in \{0, 1\}$.
5. Proceed as before (Item 2) for party $i \in H \setminus \{\mathsf{ind}\}$.
6. Do the following for party ind:
   6.1. Sample $(sk_{\mathsf{ind}}, vk_{\mathsf{ind}}) \leftarrow \mathsf{PuncSig.KeyGen}(1^\lambda)$.
   6.2. $vk_{\mathsf{ind}} \leftarrow \mathsf{PuncSig.Puncture}(sk_{\mathsf{ind}}, (\mathsf{CT}_{\mathsf{ind}}, i), \perp)$.
   6.3. Set $\mathsf{PK}_{\mathsf{ind}} = (vk_{\mathsf{ind}}, ek_{\mathsf{ind}})$ and $\mathsf{SK}_{\mathsf{ind}} = (sk_{\mathsf{ind}}, dk_{\mathsf{ind}})$.
7. The adversary, given $\{\mathsf{PK}_i\}_{i \in H}$ and crs, publishes a public key PK_i for every $i \notin H$ of the form $\mathsf{PK}_i = (vk_i, ek_i)$.
8. Proceed as before (Item 4) for party $i \in H \setminus \{\mathsf{ind}\}$.
9. Do the following for party ind:
   9.1. $\psi_{\mathsf{ind}} \leftarrow \mathsf{PuncSig.Sign}(sk_{\mathsf{ind}}, (\widehat{\mathsf{CT}_{\mathsf{ind}}}, i))$.
   9.2. Sample a PRF key K_{ind}.
   9.3. $\pi_{\mathsf{ind}} \leftarrow \mathsf{NIZK}.\mathcal{S}_2(\mathsf{crs}, \tau, (\mathsf{CT}_{\mathsf{ind}}, ek_{\mathsf{ind}}))$.
   9.4. <u>Compute $\overline{G^{(2)}_{\mathsf{ind}}}$ as the obfuscation of the circuit defined in Figure 3:</u>

$$G^{(2)}\Big[\mathsf{ind}, \mathsf{crs}, \{\tau^j\}_{j \notin H}, K_{\mathsf{ind}}, \{dk_j\}_{j \in H}, \{ek_j\}_{j \in [n]}, \{vk_j\}_{j \in [n]}, \{x^b_j\}_{j \in H}, \{\mathsf{CT}^*_j, \pi^*_j\}_{j \notin H}\Big]$$

   9.5. Let $\widehat{x_{\mathsf{ind}}} = (\overline{G_{\mathsf{ind}}}, \mathsf{CT}_{\mathsf{ind}}, \psi_{\mathsf{ind}}, \pi_{\mathsf{ind}})$.
10. The challenge: $b' \leftarrow \mathcal{A}(\mathsf{crs}, \{\mathsf{PK}_i\}_{i \in H}, \{\widehat{x_i}\}_{i \in H})$ and output b'.

Notice that when $\{\mathsf{CT}^*_i, \pi^*_i\}_{i \notin H}$ are all equal to the all zero string, then this hybrid is identical to Hyb_3.

<u>$\mathsf{Hyb}_{4,2,\{\mathsf{CT}^*_i,\pi^*_i\}_{i \notin H}}$ for $\{\mathsf{CT}^*_i, \pi^*_i\}_{i \notin H} \in \{0,1\}^{\ell(\lambda)}$</u>: Proceed as in the previous hybrid, except that now hardwire in the circuit $G^{(2)}_i$ a punctured PRF key $K^* = \mathsf{Puncture}(K, \{\mathsf{CT}_i\}_{i \in H} \| \{\mathsf{CT}^*_i\}_{i \notin H})$, sample r^*_i at random instead of via a PRF.

1. Repeat steps 1–9.1. from the previous experiment.
2. Modify step 9.2.–9.4. by doing the following for party ind:
   2.1. <u>Compute punctured key $K^*_{\mathsf{ind}} = \mathsf{PRF.Puncture}(K, \{\mathsf{CT}_i\}_{i \in H} \| \{\mathsf{CT}^*_i\}_{i \notin H})$</u>.
   2.2. <u>For $j \notin H$, compute $\widehat{\mathsf{CT}^*_j} = \mathsf{MFHE.Expand}((ek_1, \ldots, ek_n), j, \mathsf{CT}^*_j)$</u> .
   2.3. <u>Compute $\widehat{\mathsf{CT}^*_{\mathsf{out}}} \leftarrow \mathsf{MFHE.Eval}\left(f, \{\widehat{\mathsf{CT}_j}\}_{j \in H}, \{\widehat{\mathsf{CT}^*_j}\}_{j \in H}\right)$.</u>
   2.4. <u>$r_{\mathsf{ind}} \leftarrow \mathsf{PRF}_{K_{\mathsf{ind}}}(\mathsf{CT}_1 \| \pi_1 \| \ldots \| \mathsf{CT}_n \| \pi_n)$.</u>
   2.5. <u>Compute $p^*_{\mathsf{ind}} \leftarrow \mathsf{MFHE.PartDec}\left(\widehat{\mathsf{CT}_{\mathsf{out}}}, ek_1, \ldots, ek_n, \mathsf{ind}, dk_{\mathsf{ind}};\ r_{\mathsf{ind}}\right)$</u>.
   2.6. $\pi_{\mathsf{ind}} \leftarrow \mathsf{NIZK}.\mathcal{S}_2(\mathsf{crs}, \tau, (\mathsf{CT}_{\mathsf{ind}}, ek_{\mathsf{ind}}))$.
   2.7. <u>Compute $\overline{G^{(3)}_{\mathsf{ind}}}$ as the obfuscation of the circuit defined in Figure 4:</u>

$$G^{(3)}\Big[\mathsf{ind}, \mathsf{crs}, \{\tau^j\}_{j \notin H}, \underline{K^*_{\mathsf{ind}}}, \{dk_j\}_{j \in H}, \{ek_j\}_{j \in [n]}, \{vk_j\}_{j \in [n]}, \{x^b_j\}_{j \in H}, \{\mathsf{CT}^*_j, \pi^*_j\}_{j \notin H}, \underline{p^*_{\mathsf{ind}}}\Big]$$

$$G^{(2)}\left[i, \mathsf{crs}, \tau, K_i, \{dk_j\}_{j\in H}, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}, \{x_j\}_{j\in H}, \underline{\{\mathsf{CT}_j^*, \pi_j^*\}_{j\notin H}}\right]$$

```
Input: $(\mathsf{CT}_1, \psi_1, \pi_1), \ldots, (\mathsf{CT}_n, \psi_n, \pi_n)$.

- If $\mathsf{PuncSig.Verify}(vk_j, (\mathsf{CT}_j, j), \psi_j)$ is false for some $j \in [n]$ then output $\bot$.
- For every $j \in [n]$, verify $\mathsf{NIZK}.\mathcal{V}(\mathsf{crs}, (ek_j, \mathsf{CT}_j), \pi_j)$ for the statement:

  $\exists r, r', x_j, dk_j \colon \mathsf{CT}_j = \mathsf{MFHE.Enc}(ek_j, x_j; r) \ \wedge\ (ek_j, dk_j) = \mathsf{MFHE.KeyGen}(1^\lambda; r')$.

  If this check fails for some $j \in [n]$ then output $\bot$.
- For all $j \in [n]$ compute $\widehat{\mathsf{CT}}_j \leftarrow \mathsf{MFHE.Expand}((ek_1, \ldots, ek_n), j, \mathsf{CT}_j)$.
- Perform $\widehat{\mathsf{CT}}_{\mathsf{out}} \leftarrow \mathsf{MFHE.Eval}\left(f, \widehat{\mathsf{CT}}_1, \ldots, \widehat{\mathsf{CT}}_n\right)$.
- $r_i \leftarrow \mathsf{PRF}_{K_i}(\mathsf{CT}_1 \| \pi_1 \| \ldots \| \mathsf{CT}_n \| \pi_n)$.
- If $\{\mathsf{CT}_i, \pi_i\}_{i\in H} < \{\mathsf{CT}_i^*, \pi_i^*\}_{i\in H}$:
    - For all $j \notin H$, let $(r, r', x_j, dk_j) \leftarrow \mathsf{NIZK}.\mathcal{E}(\mathsf{crs}, \tau, (\mathsf{CT}_j, ek_j), \pi_j)$.
    - $p_i \leftarrow \mathsf{MFHE.Sim}\left(y, \widehat{\mathsf{CT}}_{\mathsf{out}}, \{dk_j\}_{j\in[n]\setminus\{i\}};\ r_i\right)$, where $y = f(x_1, \ldots, x_n)$.
- Otherwise:
    - $p_i \leftarrow \mathsf{MFHE.PartDec}\left(\widehat{\mathsf{CT}}_{\mathsf{out}}, ek_1, \ldots, ek_n, i, dk_i;\ r_i\right)$
- Output $p_i$.
```

Fig. 3. The circuit $G^{(2)}$

  2.8. Let $\widehat{x_{\mathsf{ind}}} = (\overline{G^{(3)}_{\mathsf{ind}}}, \widehat{\mathsf{CT}}_{\mathsf{ind}}, \psi_{\mathsf{ind}}, \pi_{\mathsf{ind}})$.

3. The challenge: $b' \leftarrow \mathcal{A}(\mathsf{crs}, \{\mathsf{PK}_i\}_{i\in H}, \{\widehat{x_i}\}_{i\in H})$ and output b'.

This hybrid is indistinguishable from $\mathsf{Hyb}_{4,1,\{\mathsf{CT}_i^*, \pi_i^*\}}$ since the circuits $G^{(2)}$ and $G^{(3)}$ (with all the hardwired values) are functionally equivalent, and therefore, their obfuscation is indistinguishable.

$\underline{\mathsf{Hyb}_{4,3,\{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H}}\text{ for }\{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H} \in \{0,1\}^{\ell(\lambda)}}$: Proceed as in the previous hybrid, except that in line 2.5., sample r^*_{ind} at random instead of via a PRF. This hybrid is indistinguishable from $\mathsf{Hyb}_{4,2,\{\mathsf{CT}_i^*, \pi_i^*\}}$ from the security of the puncturable PRF.

$\underline{\mathsf{Hyb}_{4,4,\{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H}}\text{ for }\{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H} \in \{0,1\}^{\ell(\lambda)}}$: Proceed as in the previous hybrid, except that in line 2.5., p^*_{ind} is computed as follows:

- For all $j \notin H$, compute $(r, r', x_j, dk_j) \leftarrow \mathsf{NIZK}.\mathcal{E}(\mathsf{crs}, \tau, (\mathsf{CT}_j^*, ek_j), \pi_j^*)$.
- $p^*_{\mathsf{ind}} \leftarrow \mathsf{MFHE.Sim}\left(y, \widehat{\mathsf{CT}}^*_{\mathsf{out}}, \{dk_i\}_{i\neq\mathsf{ind}}\right)$ for $y = f(\{x_j^b\}_{j\in H}, \{x_j\}_{j\notin H})$.

This hybrid is indistinguishable from $\mathsf{Hyb}_{4,3,\{\mathsf{CT}_i^*, \pi_i^*\}}$ from the simulation security of the MFHE scheme. For the simulation security to hold, we need to show that the value y actually corresponds to the message underlying the ciphertext

```
$G^{(3)}\left[i, \mathsf{crs}, \tau, K_i, \{dk_j\}_{j\in H}, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}, \{x_j\}_{j\in H}, \{\mathsf{CT}_j^*, \pi_j^*\}_{j\notin H}, p_i^*\right]$

Input: $(\mathsf{CT}_1, \psi_1, \pi_1), \dots, (\mathsf{CT}_n, \psi_n, \pi_n)$.

- If $\mathsf{PuncSig.Verify}(vk_j, (\mathsf{CT}_j, j), \psi_j)$ is false for some $j \in [n]$ then output $\perp$.
- For every $j \in [n]$, verify $\mathsf{NIZK}.\mathcal{V}(\mathsf{crs}, (ek_j, \mathsf{CT}_j), \pi_j)$ for the statement:

  $\exists r, r', x_j, dk_j \colon \mathsf{CT}_j = \mathsf{MFHE.Enc}(ek_j, x_j; r) \ \wedge\ (ek_j, dk_j) = \mathsf{MFHE.KeyGen}(1^\lambda; r')$.

  If this check fails for some $j \in [n]$ then output $\perp$.
- For all $j \in [n]$ compute $\widehat{\mathsf{CT}}_j \leftarrow \mathsf{MFHE.Expand}((ek_1, \dots, ek_n), j, \mathsf{CT}_j)$.
- Perform $\widehat{\mathsf{CT}_{\mathsf{out}}} \leftarrow \mathsf{MFHE.Eval}\left(f, \widehat{\mathsf{CT}}_1, \dots, \widehat{\mathsf{CT}}_n\right)$.
- $r_i \leftarrow \mathsf{PRF}_{K_i}(\mathsf{CT}_1 \| \pi_1 \| \dots \| \mathsf{CT}_n \| \pi_n)$.
- If $\{\mathsf{CT}_i, \pi_i\}_{i\notin H} < \{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H}$ and $i = \mathsf{ind}$:
    - For all $j \notin H$, let $(r, r', x_j, dk_j) \leftarrow \mathsf{NIZK}.\mathcal{E}(\mathsf{crs}, \tau, (\mathsf{CT}_j, ek_j), \pi_j)$.
    - $p_i \leftarrow \mathsf{MFHE.Sim}\left(y, \widehat{\mathsf{CT}_{\mathsf{out}}}, \{dk_j\}_{j\in[n]\setminus\{\mathsf{ind}\}};\ r_i\right)$, where $y = f(x_1, \dots, x_n)$.
- If $\{\mathsf{CT}_i, \pi_i\}_{i\notin H} = \{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H}$, output $p_i^*$.
- Otherwise:
    - $p_i \leftarrow \mathsf{MFHE.PartDec}\left(\widehat{\mathsf{CT}_{\mathsf{out}}}, ek_1, \dots, ek_n, i, dk_i;\ r_i\right)$.
- Output $p_i$.
```

Fig. 4. The circuit $G^{(3)}$

$\widehat{\mathsf{CT}^*_{\mathsf{out}}}$ with respect to the public keys $ek_1, \dots, ek_n$. Notice that y is computed as $y = f(\{x_j^b\}_{j\in H}, \{x_j\}_{j\notin H})$. For all $j \in H$ we have that the signature ψ_j is punctured such that it can verify only CT_j, which is indeed an encryption of x_j^b. For all $j \notin H$ by the statistical simulation extractability of the NIZK scheme, we get the corresponding secret key and message x_j for the ciphertext CT_j, and thus indeed it is a valid encryption of x_j. Altogether, we get that y is the correct value underlying the expanded ciphertext $\mathsf{CT}^*_{\mathsf{out}}$ and thus the security of the simulation holds.

$\mathsf{Hyb}_{4,5,\{\mathsf{CT}_i^*,\pi_i^*\}_{i\notin H}}$ for $\{\mathsf{CT}_i^*, \pi_i^*\}_{i\notin H} \in \{0,1\}^{\ell(\lambda)}$: Proceed as in the previous hybrid, except that in line 2.5., sample r_i^* via the PRF instead of uniformly at random. Again, this hybrid is indistinguishable from the previous one from the security of the puncturable PRF.

Hyb_5: This experiment corresponds to experiment $\mathsf{Hyb}_{4,1,1^{\ell(\lambda)}}$ except that now in step 2.7. we obfuscate the circuit with the values $\{x_j^0\}_{j\in H}$ hardwired instead of $\{x_j^b\}_{j\in H}$.

Since f is admissible (see Definition 1), we get that the two circuits are equivalent, and thus the indistinguishability of this experiment and the previous one follows from the security of the obfuscation scheme.

$\underline{\mathsf{Hyb}_6}$: This experiment corresponds to experiment Hyb_5 except that now in step 2 we encrypt x^0_{ind} in $\mathsf{CT}_{\mathsf{ind}}$ and not x^b_{ind}. Notice that this experiment is independent of the bit b and thus the probability of any adversary in guessing b is $1/2$.

The indistinguishability of this experiment and the previous one follows from the semantic security of the MFHE scheme, since dk_{ind} is not used in this hybrid any more.

In conclusion, notice that each two experiments are indistinguishable and there are in total $5 \cdot 2^{\ell(\lambda)} + 3$ hybrids, where in each such hybrid we lose the security of the primitive in hand. Thus, if we start with sub-exponentially secure schemes and initialize the underlying primitives with a large enough security parameter $\mathsf{poly}(\lambda)$, our resulting scheme is secure, as required.

5.2 Reusable PKI

As discussed, the security we defined for the MPC protocol is a single challenge, for a single PKI instantiation. A protocol that works for many sessions using the same PKI instantiation can be achieved by using session ids for each computation. The only modification needed is to sign the ciphertexts with a prefix of the session id. The puncturable signature scheme is constructed to support creating verification keys that do not accept any message within a given prefix except one. Thus, the only change in the proof would be to puncture the signature at the given session id, instead of the prefix $\perp$ as currently performed.

6 The Chain Construction

We now present our construction of NI-MPC for the chain graph communication pattern. The construction is rather similar to the one of the start pattern, and share the same building blocks. The setup procedure is the same, the main difference is the encryption procedure. The obfuscated circuit will take as input only the future inputs on the chain, and the rest will be hard-wired in the circuit.

$\underline{\mathsf{Setup}(1^\lambda, \mathsf{crs}, i)}$: It takes as input security parameter λ, a common random string crs, and does the following.

1. Execute the key generation procedure of the signature scheme, $(sk_i, vk_i) \leftarrow \mathsf{PuncSig.KeyGen}(1^\lambda)$.
2. Execute $\{(dk_i, ek_i) \leftarrow \mathsf{MFHE.KeyGen}(1^\lambda)\}_{i\in[n]}$.
3. Output public key $\mathsf{PK}_i = (vk_i, ek_i)$ and private key $\mathsf{SK}_i = (sk_i, dk_i)$.

$\underline{\mathsf{Enc}(1^\lambda, \mathsf{crs}, \widehat{x_{i-1}}, x_i, \mathsf{PK}_1, \ldots, \mathsf{PK}_n, \mathsf{SK}_i, i)}$: It takes as input security parameter λ, a common random string crs, an input $x_i \in \mathcal{X}$, a list of n public keys $\mathsf{PK}_1 \ldots, \mathsf{PK}_n$, one private key SK_i, an index $i \in [n]$, the previous message m, and does the following.

1. For every $j \in [n]$, parse $\mathsf{PK}_j = (vk_j, ek_j)$ and let $\mathsf{SK}_i = (sk_i, dk_i)$.
2. Parse $\widehat{x_{i-1}} = (\overline{G_1}, \mathsf{CT}_1, \psi_1, \pi_1), \ldots, (\overline{G_i}, \mathsf{CT}_{i-1}, \psi_{i-1}, \pi_{i-1})$, and let $\sigma = (\mathsf{CT}_1, \psi_1, \pi_1), \ldots, (\mathsf{CT}_{i-1}, \psi_{i-1}, \pi_{i-1})$.
3. Encrypt the input x_i using ek_i: $\mathsf{CT}_i \leftarrow \mathsf{MFHE.Enc}(ek_i, x_i)$.
4. Sign on (CT_i, i) using sk_i: $\psi_i \leftarrow \mathsf{PuncSig.Sign}(sk_i, (\mathsf{CT}_i, i))$.
5. Sample puncturable PRF key $K_i \xleftarrow{\$} \{0,1\}^\lambda$.
6. Generate a proof π_i that CT_i and ek_i are valid ciphertext and encryption key, respectively. Namely, compute $\pi_i \leftarrow \mathsf{NIZK}.\mathcal{P}(\mathsf{crs}, (\mathsf{CT}_i, ek_i), w = (r, r', x_i, dk_i))$, where r and r' are the randomness used for the computation of CT_i and (dk_i, ek_i), respectively. The exact statement is given in Figure 5.
7. Obfuscate the circuit $\overline{G_i} \leftarrow \mathsf{iO}\big(1^\lambda, G\left[i, \mathsf{crs}, K_i, dk_i, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}, \sigma\right]\big)$ as described in Figure 5.
8. Output $\widehat{x_i} = \widehat{x_{i-1}}, (\overline{G_i}, \mathsf{CT}_i, \psi_i, \pi_i)$.

$$G\left[i, \mathsf{crs}, K_i, dk_i, \{ek_j\}_{j\in[n]}, \{vk_j\}_{j\in[n]}, \sigma\right]$$

Input: $(\mathsf{CT}_i, \psi_i, \pi_i), \ldots, (\mathsf{CT}_n, \psi_n, \pi_n)$.

- Parse $\sigma = (\mathsf{CT}_1, \psi_1, \pi_1), \ldots, (\mathsf{CT}_{i-1}, \psi_{i-1}, \pi_{i-1})$.
- If $\mathsf{PuncSig.Verify}(vk_j, (\mathsf{CT}_j, j), \psi_j)$ is false for some $j \in [n]$ then output $\perp$.
- For every $j \in [n]$, verify $\mathsf{NIZK}.\mathcal{V}(\mathsf{crs}, (ek_j, \mathsf{CT}_j), \pi_j)$ for the statement:

  $$\exists r, r', M_j, dk_j : \mathsf{CT}_j = \mathsf{MFHE.Enc}(ek_j, M_j; r) \ \wedge \ (ek_j, dk_j) = \mathsf{MFHE.KeyGen}(1^\lambda; r').$$

  If this check fails for some $j \in [n]$ then output $\perp$.
- For all $j \in [n]$ compute $\widehat{\mathsf{CT}}_j \leftarrow \mathsf{MFHE.Expand}((ek_1, \ldots, ek_n), j, \mathsf{CT}_j)$.
- Perform $\widehat{\mathsf{CT}_{\mathsf{out}}} \leftarrow \mathsf{MFHE.Eval}\left(f, \widehat{\mathsf{CT}}_1, \ldots, \widehat{\mathsf{CT}}_n\right)$.
- $r_i \leftarrow \mathsf{PRF}_{K_i}(\mathsf{CT}_1 \| \pi_1 \| \ldots \| \mathsf{CT}_n \| \pi_n)$.
- $p_i \leftarrow \mathsf{MFHE.PartDec}\left(\widehat{\mathsf{CT}_{\mathsf{out}}}, ek_1, \ldots, ek_n, i, dk_i;\ r_i\right)$.
- Output p_i.

Fig. 5. The circuit G.

$\underline{\mathsf{Eval}(1^\lambda, \mathsf{crs}, \widehat{x_n}, \mathsf{PK}_1, \ldots, \mathsf{PK}_n)}$: It takes as input security parameter λ, a common random string crs, a string $\widehat{x_n}$, a list of n public keys $\mathsf{PK}_1 \ldots, \mathsf{PK}_n$, and performs:

1. Parse $\widehat{x_n} = (\overline{G_1}, \mathsf{CT}_1, \psi_1, \pi_1), \ldots, (\overline{G_n}, \mathsf{CT}_n, \psi_n, \pi_n)$.
2. Evaluate each obfuscation $\overline{G_i}$ on the input $\left((\widehat{\mathsf{CT}}_i, \psi_i, \pi_i), \ldots, (\widehat{\mathsf{CT}}_n, \psi_n, \pi_n)\right)$ to get the partial decryption p_i.
3. Execute the final decryption, $y \leftarrow \mathsf{MFHE.FinDec}(p_1, \ldots, p_n)$ and output y.

The correctness of the scheme is immediate and follows by the correctness of the underlying building blocks. Specifically, let $x_1, \ldots, x_n \in \mathcal{X}$ be inputs such that party i holds x_i. By the correctness of the signature scheme and the threshold multi-key FHE scheme, each obfuscated circuit will output a partial decryption p_i such that $\mathsf{MFHE.FinDec}(p_1, \ldots, p_n)$ must be equal to $f(x_1, \ldots, x_n)$.

For security, we use the same overall strategy we used in the star case in Section 5. Note however that the admissibility condition in the case of a chain (see Definition 4) is *weaker* than the admissibility condition in the case of a star. Indeed, in the chain case, the "free" inputs for which the adversary can reset their values are only those appearing after the last honest party, whereas in the star case he can reset the input of any dishonest party.

Consider first a function f that is admissible for a star (and thus also chain admissible). Every (obfuscated) circuit in the chain construction is exactly like the circuit in the star construction except that it has hardwired a subset of the inputs (appearing earlier in the chain). Then, the same sequence of hybrids from the star case applies and proves the security for f.

For the general case, where f is admissible for a chain pattern, we follow the same sequence of hybrids, but make appropriate modifications. Instead of hardwiring the inputs of the honest parties $\{x_i^b\}_{i\in H}$, we hardwire the whole set of inputs of parties up to i^* (where i^* is the index of the last honest party), namely, $\{x_i^b\}_{i\in H} \cup \{x_i\}_{[i^*]\setminus H}$. Notice that we know the inputs $\{x_i\}_{[i^*]\setminus H}$ in advance. Then, the loop of hybrids from $\mathsf{Hyb}_{4,1,\{\mathsf{CT}_i^*,\pi_i^*\}_{i>i^*}}$ to $\mathsf{Hyb}_{4,5,\{\mathsf{CT}_i^*,\pi_i^*\}_{i>i^*}}$ is only over the inputs of parties whose index is larger than i^* (rather than $i \notin H$).

Notice that in the original proof the only hybrid that uses the fact that f is admissible (for a star) is for indistinguishability (based on iO) between $\mathsf{Hyb}_{4,1,1^{\ell(\lambda)}}$ (the last hybrid in the loop) and Hyb_5. Thus, here we use a similar argument for the indistinguishability based on iO and the fact that f is admissible. Since , we know the input of all parties before i^* and since f is chain admissible (with the hardwired inputs of the parties before i^*), we get that the circuit with $\{x_i^b\}_{i\in H} \cup \{x_i\}_{[i^*]\setminus H}$ hardwired is functionally equivalent to the circuit with $\{x_i^0\}_{i\in H} \cup \{x_i\}_{[i^*]\setminus H}$ hardwired. Thus, indistinguishability of the above hybrids follows from the security of iO.

General interaction patterns. One can generalize the above idea, support arbitrary interaction patterns and achieve the "best-possible" security. The formalization of the "best-possible" security per interaction pattern is given in [24,23]. The modification of our construction is: each party will forward all its input messages to the next party in the DAG in addition to its own message that includes an obfuscated circuit and another ciphertext. The new obfuscated circuit, as in the chain pattern, will have hardwired all ciphertexts it received from previous parties in the chain. The contruction, other than this change, remains the same.

The security proof is also easily modified. Specifically, the only part in the proof that relies on the interaction pattern is the part where we use the admissibility of the function we compute (notice that the definition of admissible functions varies per interaction pattern). We follow the modification we did for the chain case and obtain a security proof for every pattern.

Acknowledgments

Shai Halevi was supported by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office(ARO) under Contract No. W911NF-15-C-0236. Yuval Ishai was supported in part by NSF-BSF grant 2015782, BSF grant 2012366, ISF grant 1709/14, ERC grant 742754, DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the DARPA through the ARL under Contract W911NF-15-C-0205. Abhishek Jain was supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213 and a sub-award from NSF CNS-1414023. Ilan Komargodski is supported in part by Elaine Shi's Packard Foundation Fellowship. Most of this work done while he was a Ph.D student at the Weizmann Institute of Science, supported in part by grants from the Israel Science Foundation and by a Levzion Fellowship. Amit Sahai was supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. Eylon Yogev is supported in part by a grant from the Israel Science Foundation. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1. Ananth, P., Jain, A., Naor, M., Sahai, A., Yogev, E.: Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In: CRYPTO. pp. 491–520 (2016)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO (2001)
3. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: CRYPTO. pp. 387–404 (2014)
4. Bitansky, N., Paneth, O.: Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In: TCC. pp. 401–427 (2015)
5. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: TCC. pp. 253–273 (2011)
6. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: ASIACRYPT. pp. 280–300 (2013)
7. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: PKC. pp. 501–519 (2014)
8. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC. pp. 235–244 (2000)

9. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: TCC. pp. 468–497 (2015)
10. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC. pp. 494–503 (2002)
11. Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: CRYPTO (2015)
12. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: CRYPTO. pp. 93–122 (2016)
13. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: STOC (1994)
14. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: TCC (2014)
15. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
16. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. 45(3), 882–929 (2016)
17. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) (1986)
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC (1987)
19. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F., Sahai, A., Shi, E., Zhou, H.: Multi-input functional encryption. In: EUROCRYPT. pp. 578–602 (2014)
20. Gordon, S.D., Malkin, T., Rosulek, M., Wee, H.: Multi-party computation of polynomials and branching programs without simultaneous interaction. In: EUROCRYPT. pp. 575–591 (2013)
21. Goyal, V., Maji, H.K.: Stateless cryptographic protocols. In: FOCS. pp. 678–687 (2011)
22. Goyal, V., Sahai, A.: Resettably secure computation. In: EUROCRYPT. pp. 54–71 (2009)
23. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: ITCS. pp. 157–168 (2016)
24. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: CRYPTO. pp. 132–150 (2011)
25. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM SIGSAC (2013)
26. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC (2012)
27. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: EUROCRYPT. pp. 735–763 (2016)
28. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive 2010, 556 (2010)
29. Sahai, A., Waters, B.: Slides on functional encryption. Available at http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt (2008)
30. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC (2014)
31. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS (1986)