On Actively-Secure Elementary MPC Reductions[*]

Benny Applebaum[1] and Aarushi Goel[2]

[1]Tel Aviv University, benny.applebaum@gmail.com
[2]Johns Hopkins University, aarushig@cs.jhu.edu

Abstract

We introduce the notion of *elementary MPC* reductions that allow us to securely compute a functionality f by making a single call to a constant-degree "non-cryptographic" functionality g without requiring any additional interaction. Roughly speaking, "non-cryptographic" means that g does not make use of cryptographic primitives, though the parties can locally call such primitives.

Classical MPC results yield such elementary reductions in various cases including the setting of passive security with full corruption threshold $t < n$ (Yao, FOCS'86; Beaver, Micali, and Rogaway, STOC'90), the setting of full active security against a corrupted minority $t < n/2$ (Damgård and Ishai, Crypto'05), and, for NC^1 functionalities, even for the setting of full active (information-theoretic) security with full corruption threshold of $t < n$ (Ishai and Kushilevitz, FOCS'00). This leaves open the existence of an elementary reduction that achieves full active security in the dishonest majority setting for all efficiently computable functions.

Our main result shows that such a reduction is unlikely to exist. Specifically, the existence of a computationally secure elementary reduction that makes black-box use of a PRG and achieves a very weak form of partial fairness (e.g., that holds only when the first party is not corrupted) would allow us to realize any efficiently-computable function by a *constant-round* protocol that achieves a non-trivial notion of information-theoretic passive security. The existence of the latter is a well-known 3-decade old open problem in information-theoretic cryptography (Beaver, Micali, and Rogaway, STOC'90).

On the positive side, we observe that this barrier can be bypassed under any of the following relaxations: (1) non-black-box use of a pseudorandom generator; (2) weaker security guarantees such as security with identifiable abort; or (3) an additional round of communication with the functionality g.

[*] An extended abstract of this paper appears in the proceedings of TCC2021. The first author is supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under Grant Agreement No. 639813 ERC-CLC, by the ISRAEL SCIENCE FOUNDATION (grant No. 2805/21), and by the Check Point Institute for Information Security and the second author is supported in part by an NSF CNS grant 1814919, NSF CAREER award 1942789 and Johns Hopkins University Catalyst award.

1 Introduction

The design and analysis of secure multiparty computation (MPC) protocols crucially rely on the notion of *secure reductions*. For example, the classical completeness results of Yao [47] and Goldreich, Micali and Wigderson [25] can be interpreted as saying that the problem of securely computing a general n-party functionality f efficiently reduces to the problem of securely computing the elementary finite 2-party Oblivious Transfer (OT) functionality [44, 21]. This paradigm of reducing a complicated f to a "simpler" randomized functionality g is especially useful when the reduction is *non-interactive*. That is, the parties compute f by making a non-interactive call to the n-party functionality g without any additional interaction. For example, Yao's celebrated garbled circuit technique [47] can be viewed as a non-interactive passively-secure reduction from any 2-party functionality f to a functionality g that can be represented by a vector of constant-degree polynomials. Extensions to the multiparty setting and to the information-theoretic setting for NC^1 functionalities were presented by Beaver, Micali and Rogaway [14] and by Ishai and Kushilevitz [35, 36]. Overall for passively secure protocols, we have the following satisfying picture.

Theorem 1.1 (Non-Interactive Passive Reductions [47, 14, 35, 36])**.** *Let f be an n-party functionality.*

- *If f is in NC^1, then there exists a non-interactive reduction from f to a constant-degree functionality g. The reduction preserves information-theoretic passive security against an adversary that corrupts up to $n-1$ parties.*

- *If f is efficiently computable (e.g., by a polynomial-size circuit family) and pseudorandom generators (PRG) exist, then there exists a non-interactive reduction from f to a constant-degree functionality g. The reduction preserves computational passive security against an adversary that corrupts up to $n-1$ parties.*

The above theorem and its many variants form the basis of most known general-purpose constant-round MPC protocols. Indeed, Theorem 1.1 non-interactively reduces the task of securely computing f to the task of securely computing a *constant-degree* function – a problem that can be solved within a constant number of rounds via standard protocols (e.g., [25, 15, 18]). Theorem 1.1 has also found, under the framework of randomized encoding (RE) [8], several surprising applications beyond MPC and even beyond cryptography. (See the surveys [34, 4].)

Elementary reductions. One important feature of the above theorem is the fact that, even in the computational setting, the reduction makes a black-box use of the PRG and the functionality g is *completely independent* of the PRG. In more detail, a non-interactive reduction consists of a preprocessing phase where each party P_i applies some

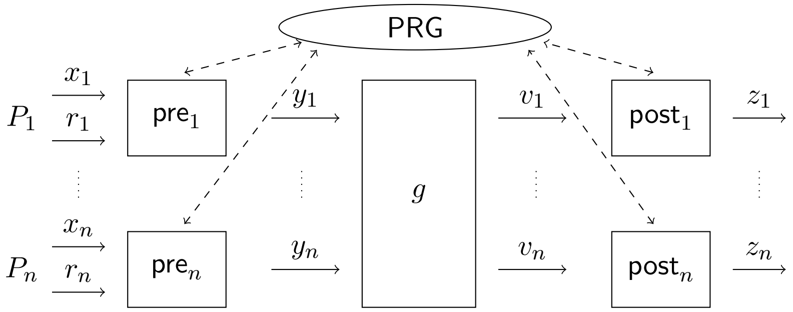

Figure 1: Elementary non-interactive reduction (dotted arrows represent oracle queries).

local preprocessing computation pre_i to its input x_i and its random tape r_i, and sends the result $y_i = \mathsf{pre}_i(x_i; r_i)$ to the g-oracle, which, in turn, computes a vector of outputs $(v_1, \dots, v_n) = g(y_1, \dots, y_n)$ and delivers v_i to the i-th party. Each party then applies some local postprocessing function post_i to its local view, and generates the final the output z_i (See Figure 1). The computational non-interactive reduction of Theorem 1.1 is *elementary* in the following sense: The parties make only black-box calls to the PRG as part of pre_i and post_i, and the constant-degree functionality g is independent of the PRG and depends only on f.[1] We refer to such a reduction as *elementary* and point out that this feature and especially the fact that g is independent of the PRG, makes the reduction highly efficient (since one does not have to "garble" the PRG). Moreover, from a theoretical point of view, elementary reductions enable to base constant-round protocols on PRGs via fully black-box reductions which forms an important feasibility result.

Actively-secure elementary reductions? The status of elementary reductions in the active (aka malicious) setting, where parties may deviate from the protocol, is less clear. For NC^1 functionalities, the information-theoretic part of Theorem 1.1 holds even in the active setting, and it provides an information-theoretic actively-secure reduction against an arbitrary coalition. In fact, the reduction preserves *fairness*, (namely, if one party receives its output then all parties do) and even *guaranteed output delivery* (i.e., honest parties are guaranteed to successfully complete the computation). That is, assuming an ideal realization of g with fairness (resp., guaranteed output delivery) against an adversary that actively corrupts a coalition T, we get a protocol for f with the same security guarantees. Adopting the perspective of Gordon et al. [28], the above result can be casted as a completeness result for fair secure computation of NC^1 functionalities: The only resource that is needed in order to fairly compute an NC^1 functionality is the ability to fairly compute a

[1] For the reader who is familiar with garbled circuits, we point out that g computes the "encrypted tables" of each gate which can be written as a degree-3 (multi-output function) over random mask bits, and pairs of the form (s, z) where s is a random seed that is chosen locally by some party and $z = \mathsf{PRG}(s)$ is pre-computed as part of the preprocessing phase.

constant-degree functionality.

Unfortunately, no such result is known for the more general case of efficiently-computable functionalities (even for computational security), instead we only have weaker results. In particular, assuming PRGs in NC^1, the work of [9] shows that every efficiently computable function f, non-interactively reduces to a constant-degree function g with fairness (and even guaranteed output delivery). However, the reduction is non-elementary since the function g depends (in a non-black-box way) on the code of the PRG. Elementary reductions that achieve weaker forms of security are implicit in the literature. This includes protocols that achieve guaranteed output delivery in the presence of *honest majority* [20], or dishonest-majority protocols that achieve (unfair) *security with abort* for single-receiver functionalities in the two-party setting [40] and in the multiparty setting [46, 32]. This leaves open the following natural question:

> *Is it possible to reduce every efficiently computable functionality to a constant-degree functionality via an elementary actively-secure reduction that preserves fairness, or even guaranteed output delivery, against arbitrary corruptions?*

1.1 Our Results

Our main result shows that it is unlikely to obtain an elementary fair reduction for general efficiently computable functions in the dishonest majority setting. Of course, we do not expect to obtain an unconditional negative result, since such a result would rule out the existence of information-theoretic elementary reductions for efficiently computable functions (a longstanding open problem in information-theoretic cryptography), and would imply complexity-theoretic groundbreaking results such as $\mathsf{P} \neq \mathsf{NC}^1$. Instead, we show that the existence of computationally-secure elementary fair reductions is essentially equivalent to the question of *information-theoretic* elementary reduction. That is, if f fairly-reduces to a non-cryptographic functionality g via a non-interactive reduction that makes a black-box use of a PRG, then these calls can be essentially removed. In fact, this holds even if the elementary reduction works only in the 2-party setting and only achieves fairness against an active adversary that only corrupts the second party (aka *partial fairness* [26]).[2] Furthermore, this result holds even if the PRG is modeled as a random oracle.

Theorem 1.2. *Suppose that every efficiently-computable 2-party functionality f, reduces to some constant-degree 2-party functionality g via an elementary reduction that makes black-box calls to a random oracle while providing partial fairness. Then, every efficiently-computable 2-party functionality f reduces to a constant-degree functionality g via a non-interactive reduction in the CRS model with inverse-polynomial average-case* information-theoretic *privacy against passive adversaries.*

[2] Note that a multiparty fair elementary reduction implies a 2-party elementary fair reduction, which in turn implies a 2-party elementary reduction with partial fairness. Therefore if we rule out the latter, we also rule out the former.

The theorem's hypothesis is, in a sense, minimal – if the parties are allowed to have an additional single round of interaction after calling g then one can achieve partial fairness by using an elementary reduction that delivers an output to the second party (e.g., based on the appendix of [40]) and then ask the second party to deliver the outcome to the first party. Let us elaborate on the "implication part" of Theorem 1.2. The notion of *inverse-polynomial average-case information-theoretic privacy* relaxes the standard notion of information-theoretic privacy by considering a scenario in which the honest party's input is chosen at random (and the adversary's input may be arbitrary) and by requiring only a fixed inverse polynomial simulation error as opposed to negligible.[3] In addition, the derived reduction is not completely non-interactive since the parties need an access to a common reference string (that can be removed at the expense of making the reduction non-uniform). These caveats can be removed under some circumstances (e.g., if the preprocessing algorithms make only random queries to the random-oracle which is the case in all existing constructions that employ a PRG – See Section 4).

Even with these minor caveats, the theorem's implication is highly non-trivial since it implies constant-round information-theoretic MPC protocols that are far beyond the current state-of-the-art. For example, the implication of Theorem 1.2 allows us to compute every efficiently-computable two-party functionality using a constant-round protocol with inverse-polynomial average-case information-theoretic security in the OT-hybrid model. This, in turn, leads to a constant-round protocol for any efficiently-computable 3-party functionality with inverse-polynomial average-case information-theoretic security.[4] The existence of such protocols is a well-known 3-decade old open problem in information-theoretic cryptography that goes back to the seminal work of Beaver, Micali and Rogaway [14]. (See also the discussions in [37, 34, 45].) While the original question is typically formulated with respect to standard security, the relaxation to inverse-polynomial average-case security does not seem to make it more tractable.

We complement our main result with some positive results. First, we observe that any passively-secure elementary reduction can be compiled into an actively-secure non-interactive, yet non-elementary, reduction.

Observation 1. *Suppose that the functionality f reduces to a constant degree functionality g via a non-interactive passively-secure reduction Π. Then, f reduces to a constant-degree functionality g' via a non-interactive actively-secure reduction Π' with guaranteed output delivery. Moreover, if Π is information-theoretically secure then so is Π'. The description of g' and Π' depends on the description of the preprocessing part of Π. Specifically, if Π makes use of a PRG in the preprocessing phase, then Π' and g' depend on the code of the PRG.*

[3] This relaxation applies only to privacy, and correctness holds for arbitrary inputs except with negligible probability. Also, we can support an arbitrary inverse polynomial privacy error α, at the expense of a $\mathsf{poly}(1/\alpha)$ slow-down in the running-time of the protocol.

[4] Both results can be lifted to the active setting as well.

This simple observation (whose proof is deferred to the Section A) shows that an elementary information-theoretic passively-secure reduction can be upgraded "for-free" to an information-theoretic actively-secure elementary reduction. Specifically, it can used to upgrade the implication in Theorem 1.2 to the active setting (however the reduction still achieves only inverse polynomial average-case security).

Getting back to the computational setting, by combining Observation 1 with the passively-secure elementary reductions from Theorem 1.1, we derive the following corollary.

Corollary 1.1. *Assuming the existence of pseudorandom generators, every efficiently-computable functionality f reduces to a constant-degree functionality g via a non-interactive computationally-secure reduction with guaranteed output delivery against an active adversary. The reduction and the functionality g make a non-black-box use of the PRG.*

Previously, such a result was known only based on an NC^1-computable PRG [9] (or equivalently NC^1-computable one-way function; see [3, Chapter 5]). The combination of Theorems 1.2 and Corollary 1.1, provides another interesting example for a gap between a black-box use of a primitive and a non-Black-Box use of a primitive. (See Section 1.3 for further discussion.)

Finally, we ask what level of active security can be obtained via elementary reductions. It turns out that it is possible to obtain the following notion of *security with identifiable abort* [11]: Upon abort every honest party learns the identity of some corrupted party. (This additional feature provides several advantages – see [39] for a discussion.)

Theorem 1.3. *Every efficiently-computable functionality reduces to a constant-degree functionality via an elementary computationally-secure reduction that achieves active security with identifiable abort.*

This result can be extracted from the recent constant-round protocol of Baum et al. [13]. We provide a self-contained proof that highlights the main ingredients needed for elementary reductions. Our construction also has a minor advantage: It natively supports fairness at the expense of an additional round of interaction. That is, if the parties are allowed to interact with g twice (or, equivalently, replace g with two sequential calls to memoryless constant-degree functionalities g_1 and g_2) then full fairness can be obtained! (See Remarks 3 and 4.)

1.2 Technical Overview

Let us start by examining the computationally-secure passive elementary reductions from Theorem 1.1 and see why they fail to achieve active security.

Attacking existing protocols. At a high-level, the garbled circuit technique non-interactively reduces the computation of the target functionality f to the computation of some form of a distributed encryption scheme. To make the reduction efficient for high-depth circuits, one has to use an encryption scheme whose keys are shorter than the message. The latter can be based on a PRG. Since the PRG should be employed locally, we ask each party P_i to compute, in the preprocessing phase, the PRG values on many random seeds $(s_1, \dots, s_t)$ and send each of these seeds, s_j, together with the outcome, $\mathsf{PRG}(s_j)$, to the functionality g. The functionality g then combines these values together (via a low-degree operation) and outputs a bunch of ciphertexts for each gate of the circuit, together with some keys (seeds) for the input wires. In the postprocessing phase, each party P_i decodes the output of f by decrypting some of the ciphertexts (according to the computation path of the garbled circuit). As part of this decryption operation, P_i computes the PRG on seeds that were selected locally during the preprocessing phase by each of the participating parties. This structure ensures that this non-interactive reduction is indeed elementary: The functionality g is independent of the PRG, and the PRG is being invoked only locally and only in a black-box way.

Unfortunately, in the active setting, this independence can be exploited. An active adversary can send an invalid pair of the from $(s, y \neq \mathsf{PRG}(s))$, and the functionality g, being "unaware" of the PRG, will not be able to detect such a cheating. As a result, honest parties are likely to get garbage values during the postprocessing phase and the decoding is likely to fail. Moreover, if the adversary knows an actual seed s' for which $y = \mathsf{PRG}(s')$, then the adversary can, in principle, recover the correct value of $f(x_1, \dots, x_n)$, violating the fairness of the protocol. Our main theorem shows that this problem is not specific to the current instantiations of garbled circuits: Such an attack is inherent in the setting of elementary reductions and it can be avoided only if the PRG is not "really needed".

Attacking general protocols. For simplicity let us focus on the two party case. We assume that the PRG is instantiated with a random oracle H and, for now, let us further assume that the parties invoke the oracle on randomly chosen seeds. Loosely speaking, we apply a variant of the above attack in which the second (corrupted) party samples a local independent random oracle G and uses this oracle in the preprocessing phase instead of using the publicly available oracle H. We argue that the reduction cannot detect such a cheating. Moreover, we note that the second party can still correctly recover the final outcome of the protocol. Indeed, except with negligible probability, the parties do not query the oracle on the same input, and so one can pretend that they honestly invoked the protocol on a new random oracle that is obtained by combining the oracles H and G. Both oracles are available to the second party and so she can use them in the postprocessing phase to correctly recover the output $f(x_1, x_2)$. Now if the protocol is indeed fair, then, intuitively, the first party should also be able to recover the output correctly (by accessing only H and without an access to the local oracle G). Furthermore, it can be shown that even under this attack the honest party, P_1, learns nothing on the input of the corrupted party

P_2.[5] Therefore, we get a modified protocol in which the preprocessing and postprocessing computation of P_1 depends only on H, the preprocessing part of P_2 depends only on G, and its postprocessing computation depends on both G and H. We can therefore further modify the protocol by asking P_1 (resp., P_2) to locally sample its own oracle H (resp., G), while removing the postprocessing phase of P_2 and replacing it with an empty output. (Formally, we also change the functionality g so that it hands to P_2 an empty value $\perp$.) This gives us an information-theoretic passively-secure non-interactive reduction that delivers an output only to the first party. We can easily fix this caveat and distribute an output to both parties, by running two copies of the reduction in parallel where P_1 is the receiver in one copy and P_2 is the receiver in the other copy. We can further make the reduction actively-secure by invoking Observation 1.

The above description is over-simplistic since we assumed that the parties call the PRG on uniform independent seeds. While this is a reasonable assumption (especially if the random oracle models a PRG), it can be removed via standard techniques. Specifically, the above argument holds as long as the calls to G and H in the preprocessing phase do not intersect. To avoid such an event, we identify "heavy" queries [12], and let P_2 use its local oracle G only on non-heavy queries. This modification introduces several technicalities, which eventually allow us to obtain only inverse-polynomial average-case security. We can make sure that this problem does not affect the correctness of the protocol by adding to the functionality g a "detect-and-reveal" mechanism that identifies a "collision event" and releases, when such an event occurs, the private inputs of the parties. Thus, even when the event happens correctness remains unaffected. This additional mechanism increases the complexity of g to NC^1, and we can reduce its degree to a constant, by replacing the functionality with an NC^0 information-theoretic RE.

About the proofs of positive results. As already mentioned, there are non-interactive reductions that preserve full active security (including guaranteed output delivery) either with information-theoretic security for NC^1 functionalities [35, 36] or with computational security for polynomial-size circuits assuming a non-black-box access to a PRG in NC^1 [9]. These results are based on constant-degree information-theoretic/computational REs. Interestingly, REs essentially correspond to extremely simple non-interactive reductions in which the parties do not apply any preprocessing computation and submit their inputs directly to the randomized functionality g. This feature is obtained by "pushing all the computation" to functionality g; For NC^1 functionalities this can be done by plugging an

[5] Both statements regarding the view of P_1 ("fairness leads to correct output for P_1" and "P_1 learns nothing on P_2's input") are not immediate. First, the formal simulation-based definition of fairness only ensures that P_1 generates an output $f(x_1, x_2')$ with respect to some "effective input" x_2' of P_2 and not necessarily with respect to the "real" input x_2 that is given to P_2. This technicality is solved by working with "authenticated functionalities". Second, the standard MPC definition provides no guarantees on the privacy of a party P_2 that deviates from the protocol. So the fact that P_1 learns nothing requires concrete justification.

information-theoretic encryption scheme (e.g., one-time pad) into the garbled circuit construction, and in the computational setting, this is essentially done by relying on a PRG whose complexity is low enough so that it can be computed by g.

In order to derive Observation 1, we note that instead of pushing the computation of the preprocessing part to g, it suffices to let g *verify* that the preprocessing of P_i was done properly and to replace P_i's input to g with some default value if this condition fails. It is well known that such a verification procedure can be implemented via an NC^1 functionality by asking each party to supply all the intermediate values of the preprocessing circuit, and by checking a list of degree-2 constraints, one for each gate. We can further replace this NC^1 functionality with its constant-degree randomized encoding and derive Observation 1.

Finally, in order to obtain an elementary reduction with identifiable abort (Theorem 1.3), we consider the garbled circuit construction and abstract the syntax of distributed encryption scheme that suffices for deriving elementary reduction. Roughly, the underlying encryption scheme should satisfy the following non-standard syntactic properties. Despite being a symmetric-key cryptosystem, the key-generation algorithm (that will be run locally by each party) generates pairs of encryption/decryption keys, and is allowed to access a PRG. The encryption algorithm (which will be embedded inside g) uses multiple encryption keys, one from each party, and some internal randomness, to encrypt a message. This algorithm should be in NC^1 and should be PRG-independent. Finally, the decryption algorithm takes a ciphertext and decryption keys, and recovers the plaintext or outputs an error flag. This algorithm will be embedded in the postprocessing phase and is therefore allowed to call the PRG. We further require an expansion property, that is, the decryption keys should be shorter than the messages. We define different forms of robustness against malicious key-generation, and show how they affect the security of the resulting elementary reduction. While similar ideas have appeared implicitly in previous works (e.g., [14, 20, 38, 41, 31, 13]), we believe that our formulation clarifies the necessary conditions that enable elementary reductions. We present a new instantiation of distributed encryption whose security suffices for "identifiable abort", and for fairness given an additional round of interaction. The construction is based on information-theoretic MACs and cut-and-choose ideas.

1.3 Discussion and other related works

MPC reductions and Fairness. There is a rich body of works studying fairness in MPC. (A summary of some central lower-bounds and upper-bounds can be found in [29] and in [27, 43].) Most relevant to us is the work of Gordon et al. [28] on complete primitives for fairness. This work studies the ability to reduce fair protocols for general functionalities f to fair protocols for simpler functionalities g where "simplicity" is measured with respect to the input length of g. Our work complements this study by considering a different form of simplicity (low degree) and more restricted type of reductions (non-interactive).

Elementary reductions to degree-2 functionalities. Recent results regarding the exact round complexity of MPC, have motivated the study of the exact degree of the functionality g for which f reduces to. While the original randomized encoding tools achieve degree-3, the breakthrough results of [23, 16] suggested that it may be possible to non-interactively reduce every efficiently-computable function to a degree-2 function. Indeed, such reductions were obtained explicitly in [6, 7, 10], and implicitly in [22, 1, 2], both for active and passive adversaries. For the honest majority setting, these reductions are elementary (with computational security for polynomial-size circuits and information-theoretic security for NC^1 circuits). However, for the dishonest majority setting, these reductions rely on a non-black-box use of oblivious transfer. The latter non-black-box dependency was shown to be necessary in [5] even if one is allowed to treat the oblivious transfer as a two-round functionality and allow the reduction to have two rounds of interaction.

Other limitations of random oracles in secure computation. Haitner, Omri and Zarosim [30] and Mahmoody, Maji and Prabhakaran [42] showed that random oracles are essentially "useless" for secure 2-party computation of various functionalities. Specifically, they extended the Impagliazzo-Rudich separation [33] (and its tighter analysis given in [12]) and showed that, under mild conditions on the underlying functionality, any passively-secure protocol that makes use of a random oracle (RO) can be compiled into an information-theoretic protocol that does not depend on the random oracle. (In the active setting, the random oracle can be traded with an ideal commitment scheme.) While this result somewhat resembles our main theorem there are several important differences that suggest that our work captures a different limitation than the one captured in [30, 42]. Details follow.

Firstly, the results of [30, 42] work in the plain model while our results operate in a hybrid model in which the parties have access to a trusted party that implements a multiparty functionality. The power and usefulness of RO significantly changes in the presence of such a trusted party and one cannot easily transfer results from the plain model to the hybrid model. Indeed, our theorem is very sensitive to the exact notion of security that is being used and to the non-interactive nature of the reduction. In particular, Theorem 1.2 becomes incorrect if one allows an additional round of interaction or if one relaxes security to passive or even to security with abort as shown in Theorem 1.3. Secondly, from a technical point of view, our proof strongly exploits the fact that the functionality g is unaware of the structure of the PRG, together with the fairness properties of the protocol. These issues are unique to our setting and they do not appear in [30, 42]. On the other hand, while the proofs of [30, 42] tackle the challenging task of locally simulating the correlation that is induced in a RO-based interactive protocol, in our *non-interactive* setting this issue is handled easily based on simple machinery. (Specifically, we employ an extremely degenerate version of heavy-query learners. See Section 4.) Finally, and perhaps most importantly, we know that a non-black-box access to PRGs *does help* in our setting, and it can be used to bypass the negative result (i.e., to obtain computationally-secure fair non-interactive reductions as shown in Corollary 1.1). No similar result is known in the plain model.

Taking a more general perspective, our work provides an interesting example for a cryptographic task for which (1) we cannot rule out the existence of an information-theoretic solution; (2) we can show that a black-box use of a given primitive is useless; but (3) a non-black-box use of the primitive allows us to solve the problem. While we are aware of examples in which (1) and (2) hold (e.g., [30, 42]), and examples where (2) and (3) hold (most closely to our work, the impossibility of elementary reductions to oblivious transfer [5]), the current combination of (1), (2), and (3) seems rather unique to our setting.

2 Standard Preliminaries

Definition 2.1 (Pseudorandom Generator (PRG))**.** *A psedudorandom generator (PRG) is a determinstic poly-time algorithm* PRG*, satisfying two conditions:*

— ***Expansion:*** *There exists a polynomial* $\ell(\lambda) : \mathbb{N} \to \mathbb{N}$ *satisfying that* $\ell(\lambda) > \lambda$ *for all* $\lambda \in \mathbb{N}$*, such that* $|\mathsf{PRG}(x)| = \ell(|x|)$ *for all* $x \in \{0,1\}^*$*.*

— ***Pseudorandomness:*** *The distribution ensembles* $\{\mathsf{PRG}(U_\lambda)\}_{\lambda\in\mathbb{N}}$ *and* $\{U_{\ell(\lambda)}\}_{\lambda\in\mathbb{N}}$ *are computationally indistinguishable, where* U_n *denoted the uniform distribution over* n*-bit strings.*

The **stretch** *of a PRG is defined as* $|\mathsf{PRG}(x)| - |x|$*.*

Definition 2.2 (Randomized Encoding of Functions [35, 8])**.** *Let* $f : X \to Y$ *be a function. We say that a function* $\hat{f} : X \times R \to Z$ *is a perfect randomized encoding of* f *if there exist a pair of randomized algorithms, decoder* Dec *and simulator* Sim*, for which the following hold:*

— ***Correctness:*** *For any input* $x \in X$*, and* $r \in R$*, it holds that* $\mathsf{Dec}(\hat{f}(x; r)) = f(x)$*.*

— ***Privacy:*** *For every* $x \in X$ *the distribution* $\mathsf{Sim}\,(f(x))$ *is identical to the distribution* $\hat{f}(x; r)$ *induced by sampling* $r \leftarrow\$ R$*.*

The definition naturally extends to functions over infinite domains and to ensembles of functions (see [35, 8]).

3 Elementary Reductions

In this section, we formalize the notion of elementary reductions. At a high level, an elementary reduction from an n-input functionality f to another n-input, constant degree, non-cryptographic functionality g, is a non-interactive reduction that yields a non-interactive secure MPC protocol realizing f, where the parties make a single query to a trusted implementation of g. We refer to such protocols as elementary g-oracle protocols. We start by formally defining the syntax of such protocols. A pictorial representation of an elementary g-oracle protocol appears in Figure 1.

Definition 3.1 (Non-interactive g-oracle Protocol)**.** *A non-interactive g-oracle protocol is an n-party MPC protocol in the g-hybrid model that makes a single non-interactive call to the functionality g. Such a protocol is defined by a tuple of PPT algorithms $(\mathsf{pre}_1, \ldots, \mathsf{pre}_n, \mathsf{post}_1, \ldots, \mathsf{post}_n)$, where the parties make a single call to the n-input ideal functionality g as follows:*

1. ***Pre-processing:*** *Each party P_i (for $i \in [n]$) runs the pre-processing algorithm pre_i on its input x_i and randomness r_i to obtain y_i.*

2. ***g-oracle:*** *Each party P_i (for $i \in [n]$) invokes the functionality g using its pre-processed input y_i. The functionality returns output v_i to party P_i.*

3. ***Post-processing:*** *Each party P_i (for $i \in [n]$) runs the post-processing algorithm post_i on the output v_i received from g to obtain the final output z_i.*[6]

The protocol is elementary *if the preprocessing and postprocessing algorithms make only black-box calls to a PRG and g is a constant-degree functionality (i.e., each of its outputs can be written as a constant degree polynomial over the binary field) whose description is independent of the PRG.*

The security of g-oracle Protocol is defined by following the standard real/ideal paradigm using the standard extension to the hybrid model (see, for example, [24, Chapter 7] and [17]), and can be instantiated with different variants of security (e.g., computational/information-theoretic, passive/active, guaranteed-output-delivery/fairness/security-with-abort/security-with-identifiable abort). We postpone the exact specification to the technical sections.

Remark 1 (Non-Interactive Reductions and Randomized Encoding)**.** *Given a non-interactive g-oracle protocol for f and a randomized encoding $\hat{g}(y;\rho)$ of $g(y)$, one can always obtain a non-interactive g'-oracle protocol where $g'(y, \rho_1, \ldots, \rho_n) := \hat{g}(x; \sum_i \rho_i)$, the original preprocessing functions are extended by letting each party send a random string ρ_i to the functionality g' (in addition to the original (y_i, r_i) parts) and the postprocessing algorithm is extended with the RE decoder. This transformation preserves all the types of security that are considered in this paper (including information-theoretic security), and does not rely on computational assumptions/tools. Since NC^1 functionalities can be encoded by constant-degree functionalities [36] or even by NC^0 functionalities [8], we can*

[6] In principle, one could allow post_i to depend on the entire view of P_i which consists of v_i as well as x_i and r_i. However, one can always remove this dependency by assuming that P_i sends (x_i, r_i) to g as part of y_i and that g delivers these values back to P_i as part of v_i.

freely move from a liberal definition of elementary protocols in which g is allowed to be an NC^1 *(possibly randomized) functionality, to a more restricted definition in which g should be a constant-degree deterministic functionality or even an* NC^0 *functionality.*

4 Lower Bound for Elementary Reduction with Fairness

4.1 The Set-up

In this section, we focus for simplicity on two-party functionalities $f : X_1 \times X_2 \to Z_1 \times Z_2$ where $f_i : X_1 \times X_2 \to Z_i$ denote the restriction of f to its i-th output. Let $\mathcal{H}$ be a probability distributions over functions from $\mathcal{D}$ to $\mathcal{R}$. An elementary non-interactive reduction Π from f to a two-party functionality $g : Y_1 \times Y_2 \to V_1 \times V_2$ in the $\mathcal{H}$ model consists of four oracle aided algorithms $(\mathsf{pre}_1, \mathsf{post}_1, \mathsf{pre}_2, \mathsf{post}_2)$ which are sometimes grouped as $P_1 = (\mathsf{pre}_1, \mathsf{post}_1)$ and $P_2 = (\mathsf{pre}_2, \mathsf{post}_2)$. Formally, all the above objects are parameterized by a security parameter λ, and they are required to be computationally-efficient with respect to this parameter. For simplicity (and without loss of generality), we assume that the f_λ is defined over $\{0,1\}^\lambda \times \{0,1\}^\lambda$ and so we can think of the input length as the security parameter (which will be kept implicit most of the time). For concreteness, we also think of the domain $\mathcal{D}_\lambda$ and range $\mathcal{R}_\lambda$ of $\mathcal{H}_\lambda$ as $\{0,1\}^\lambda$. (Though, our results are insensitive to this choice, and any domain and range can be used as long as there exists a $\mathsf{poly}(\lambda)$-time algorithm that uniformly samples an element from these sets.)

We assume that when the parties are honest the outputs generated by the reduction are correct except with negligible error. The reduction should also achieve passive security against an adversary that corrupts the first party (aka *privacy*) and a weak form of partial fairness against an active adversary that corrupts the second party. We formalize these notions below for general protocols in the $\mathcal{H}$-model. (These definitions will be applied to MPC reductions, and specifically, to non-interactive g-oracle protocols.)

Passive security in the $\mathcal{H}$ model. Following the standard convention, when working with respect to random oracles we assume that adversaries are computationally unbounded but make a polynomially-bounded number of queries to the oracle (see, e.g., [30, 42, 5]). We also introduce an average-case version of privacy.

Definition 4.1 (privacy and AVG-privacy)**.** *A protocol Π in the $\mathcal{H}$-hybrid model realizes f with α-*privacy *against party i if there exists an efficient randomized simulator $\mathsf{Sim}(x_i, z_i)$ whose output consists of a view w and a stateful randomized oracle H such that for every computationally-unbounded distinguisher $\mathcal{A}$ that makes at most $\mathsf{poly}(\lambda)$ queries to its oracle and for every λ-bit inputs (x_1, x_2) it holds that*

$$\Delta_{\mathcal{A},\Pi,i}(x_1,x_2) :=$$

$$\left| \Pr_{H \leftarrow\$\mathcal{H}_\lambda}[\mathcal{A}^H(x_1,x_2,\mathsf{view}^H_{\Pi,P_i}(x_1,x_2)) = 1] - \Pr_{(w,H)\leftarrow\$\mathsf{Sim}(x_i,f_i(x_1,x_2))}[\mathcal{A}^H(x_1,x_2,w) = 1] \right|,$$

is upper-bounded by $\alpha(\lambda)$*, where* $\mathsf{view}^H_{\Pi,P_i}(x_1, x_2)$ *denotes the view of* P_i *in an execution of* $\Pi^H(x_1, x_2)$ *with fresh randomness for both parties and oracle* H*. If* α *is negligible, we say that* Π *is* IND-private *against party* i*.*

We say that the protocol is α-AVG-private *against the first party if for every computationally-unbounded distinguisher* $\mathcal{A}$ *that makes at most* $\mathsf{poly}(\lambda)$ *queries to its oracle and for every* λ*-bit input* x_1*, it holds that*

$$\mathbb{E}_{x_2}[\Delta_{\mathcal{A},\Pi,1}(x_1, x_2)] \leq \alpha(\lambda).$$

The notion of α*-AVG-privacy against the* second *party is defined analogously, i.e., for every* x_2*, we require that* $\mathbb{E}_{x_1}[\Delta_{\mathcal{A},\Pi,2}(x_1, x_2)] \leq \alpha(\lambda)$*.*

We mention that our proof goes through even if one uses a slightly weaker definition in which the oracle queries of the distinguisher $\mathcal{A}$ may depend only on the input of the corrupted party. We prefer the current definition due its simplicity.

The *random oracle* model corresponds to the case where $\mathcal{H}$ is distributed uniformly over all functions from $\mathcal{D}_\lambda$ to $\mathcal{R}_\lambda$. Also, observe that the notion of information-theoretic IND-privacy in the *standard model* can be derived from the above definition by instantiating $\mathcal{H}$ with some fixed simple function (like the all zero function or the identity function.)

Fairness against P_2. Following Goldwasser and Lindell [26], we require *fairness* against an active adversary that corrupts only the *second party* (aka *partial fairness*).[7] Roughly speaking, this means that when P_1 is honest, complete fairness is essentially achieved (i.e., either all parties abort or all parties receive correct outputs). But when P_1 is actively corrupt, fairness may be violated. In fact, unlike the notion of partial fairness from [26], we make no requirement at all in this case (i.e., an actively corrupted P_1 can learn the input of P_2). Again, this makes the result stronger.

Definition 4.2 (Fairness against P_2)**.** *A protocol* Π *in the* $\mathcal{H}$*-hybrid model realizes* f *with fairness against an adversary that corrupts* P_2 *if for every computationally-unbounded distinguisher* $\mathcal{A}$ *that makes at most* $\mathsf{poly}(\lambda)$ *queries to its oracle, there exists a simulator* Sim *whose running-time is polynomial in the running time of* $\mathcal{A}$*, such that for every pair of inputs* $x_1, x_2 \in \{0,1\}^\lambda$ *the distributions* $\mathsf{Ideal}_{f,\mathsf{Sim}}(x_1, x_2)$*and* $\mathsf{Real}^H_{\Pi,\mathcal{A}}(x_1, x_2)$*, where* $H \xleftarrow{\$} \mathcal{H}_\lambda$ *cannot be distinguished by any computationally-unbounded distinguisher* D *with advantage better than negligible.*

As usual, the random variable $\mathsf{Real}^H_{\Pi,\mathcal{A}}(x_1, x_2)$ *corresponds to the joint outputs of* P_1 *and the* $\mathcal{A}$ *(who corrupts* P_2*) in the execution of* Π *over the inputs* x_1 *and* x_2 *and with respect to the oracle* H*. The random variable* $\mathsf{Ideal}_{f,\mathsf{Sim}}(x_1, x_2) = (z_1, z_2)$ *corresponds to the joint outputs of the first party and* Sim *in an ideal-world execution in which the parties access*

[7] As mentioned in the introduction, unlike full fairness which is impossible in the plain model without honest majority [19], partial fairness can be achieved (using multiple rounds) in the plain model assuming the existence of OT.

to an f-oracle that either computes f or sends an abort symbol to both parties (depending on the choice of the simulator). That is, $\mathsf{Sim}(x_2)$ *computes some* $x_2' \in X_2 \cup \{\perp\}$ *and sends it to* f*. If* $x_2' = \perp$ *the functionality sets the output* z_1 *of* P_1 *to* $\perp$ *and returns* $z_2' = \perp$ *to the simulator, and if* $x_2' \neq \perp$ *the functionality returns* $z_1 = f_1(x_1, x_2')$ *and* $z_2' = f_2(x_1, x_2')$*. The simulator terminates with the output* z_2 *which is computed based on* z_2' *and its internal state.*

Remark 2 (weak fairness)**.** *We remark that for our purposes, it suffices to assume that fairness holds only for* computationally-bounded *adversary* $\mathcal{A}$ *and even if the corresponding simulator is allowed to be computationally unbounded. Even more importantly, we only need to consider the following specific (computationally-bounded) distinguisher* D *that given a pair* (z_1, z_2) *(supposedly sampled from* $\mathsf{Ideal}^H_{f,\mathsf{Sim}}(x_1, x_2)$ *or from* $\mathsf{Real}^H_{\Pi,\mathcal{A}}(x_1, x_2)$*) outputs 1 if*

$$z_1 \neq f_1(x_1, x_2) \bigwedge z_2 = f_2(x_1, x_2).$$

Equivalently, one can think of a game (that can be played both in the ideal world and real world) in which the adversary $\mathcal{A}(x_2)$ *(resp., simulator* $\mathsf{Sim}(x_2)$*) wins when interacting with* $P_1(x_1)$*, if she outputs the "right" value* $f_2(x_1, x_2)$ *while the honest party* P_1 *errs and outputs some* $z_1 \neq f_1(x_1, x_2)$*. We say that a protocol achieves weak-fairness against* P_2 *if for any such adversary* $\mathcal{A}$*, there exists an ideal-world simulator* Sim *such that for every* x_1, x_2*, the winning probability in the real execution is upper-bounded by the winning probability in the ideal execution (plus some negligible quantity).*

Weak fairness is implied by fairness, and it can be viewed as an extension of the correctness property of the protocol. (Indeed, if, for example, the honest party outputs the "right" value, we do not care whether an active adversary gets to learn the honest party's input.)

Authenticated functionality. Fairness will be mostly useful when it is applied to so-called authenticated functionalities. Formally, given an arbitrary 2-party functionality $f_1(x_1, x_2)$ that delivers an output only to P_1, we define a 2-party functionality $f((x_1, k), x_2)$ that delivers $f_1(x_1, x_2)$ to P_1 and delivers $\mathsf{MAC}_k(x_2)$ to P_2 where k is a λ-bit key for a one-time information-theoretic secure message authentication-code MAC. We refer to f as the P_1-authenticated version of f_1.

4.2 Main Results

We can now state our key theorem whose proof will be deferred to the following subsections. In the following, we say that a reduction Π in the random-oracle model makes *input-independent queries* if the calls to the oracle H that a party P_i makes are statistically-independent of its input x_i. We say that the reduction makes uniform queries if, in addition, each query to H in the preprocessing phase is sampled uniformly and independently of all the other queries. Known elementary reductions (that make use of PRG's) satisfy these additional properties.

Theorem 4.1. *Let f be the P_1-authenticated version of some functionality f_1. Assume the existence of an elementary non-interactive reduction Π from f to a constant-degree two-party functionality $g : Y_1 \times Y_2 \to V_1 \times V_2$ in the random-oracle model that achieves privacy against P_1 and weak fairness against P_2. Then, for every inverse polynomial $\alpha(\lambda)$, there exists an efficient two-round reduction Σ from f_1 to an NC^1 functionality g' with the following properties.*

1. *(Syntax) At the first round P_1 sends a message to P_2 that consists of random coins. Then, both parties make a call to g' (who delivers output only to P_1) and then P_1 applies some postprocessing computation and terminates with an output. (The other party terminates with an empty output.)*
2. *(Correctness) For every input x_1, x_2 the output of P_1 is $f_1(x_1, x_2)$ except with negligible probability.*
3. *(Privacy against P_2) The reduction achieves information-theoretic privacy against the second party.*
4. *(AVG-privacy against P_1) The reduction achieves information-theoretic α-AVG-privacy against the first party. Moreover, if the original reduction Π makes only input-independent queries then the reduction Σ is $O(\alpha)$-private, and if the reduction makes uniform queries then Σ is private (with negligible privacy error).*[8]

Since the first message sent from P_1 to P_2 in Σ consists of only random coins, we can think of Σ as a non-interactive reduction in a CRS model where the parties get an access to a shared random string. (For passively-secure protocols, the CRS model is equivalent to a two-round model in which the first message contains random coins.)

By repeating the reduction twice (while replacing the roles of P_1 and P_2) we can make sure that both parties receive an output. Moreover, we can reduce the complexity of g' from NC^1 to a an NC^0 functionality (following Remark 1), and derive Theorem 1.2. Furthermore, by exploiting the fact that NC^0 functionalities can be evaluated by making a single round of parallel calls to an ideal OT-functionality with passive information-theoretic privacy (using a variant of Yao's protocol), we get the following corollary.

Corollary 4.1. *Suppose that every efficiently-computable two-party functionality f can be reduced to some constant-degree two-party functionality in the random-oracle model via an elementary reduction that achieves privacy against P_1 and weak fairness against P_2.*

Then, every such functionality can be computed in the CRS model by making only parallel calls to Oblivious-Transfer with information-theoretic $O(\alpha)$-AVG-privacy for any

[8] In fact, it suffices to assume that only the preprocessing algorithm of P_2 makes input-independent/uniform queries.

a-priory given inverse polynomial α. Furthermore, if the hypothesis holds with respect to reductions that make input-independent queries then the resulting protocol is $O(\alpha)$-private, and if the hypothesis holds with respect to uniform queries then the resulting protocol is private (with negligible privacy error).

We note that the above yields an efficient constant-round multiparty protocol for $n \geq 3$ in the plain model with passive $O(\alpha)$-AVG-privacy against any single party. Indeed, consider, wlog, the single-output functionality $f(x_1, \ldots, x_n)$ that delivers its output to P_1. Then each party i shares its input x_i via 2-out-of-2 secret-sharing and hands one share r_i to P_1 and another share $x'_i = r_i \oplus x_i$ to P_2. Then, P_1 and P_2 run the OT-based protocol (promised by Corollary 4.1) for the two-party functionality $f'((r_1, \ldots, r_n), (x'_1, \ldots, x'_n))$ where the OT-channel is being replaced by a constant-round protocol (e.g., BGW) for computing the degree-2 OT functionality.

Notation. The following notation will be extensively used throughout the proof of Theorem 4.1. For a pair of oracles $G, H : \mathcal{D} \to \mathcal{R}$ and a set $S \subset \mathcal{D}$, we define the oracle $G[S] \cup H$ to be the oracle that given q returns $G(q)$ if $q \in S$ and otherwise, returns $H(q)$. For an oracle-aided algorithm $A(x; r)$ with input x and randomness r we let $Q(A^H(x; r))$ denote the tuple of queries that $A(x; r)$ makes to H. When r is omitted, $Q(A^H(x))$ denotes the random variable that contains all the queries that A makes when executed with x, H and a uniformly chosen r.

4.3 Tools: Finding Heavy queries

To prove Theorem 4.1, we will need the following simple lemma that can be viewed as a (very) degenerate version of the Barak-Mahmoody [12] heavy-query learner.

Lemma 4.1. *Let A be a randomized input-less oracle-aided algorithm that makes at most T queries to a random oracle $H : \mathcal{D} \to \mathcal{R}$ and runs in time t. Then, there exists an oracle-aided randomized "heavy-query finder" algorithm $F^{\cdot}_A(\epsilon, \delta)$ with the following properties:*

1. *(efficiency) $F^H_A(\epsilon, \delta)$ runs in time $\mathsf{poly}(1/\epsilon, \log(1/\delta), t)$ and makes at most $\mathsf{poly}(T, 1/\epsilon, \log(1/\delta))$ queries to its oracle H. The output of $F^H_A(\epsilon, \delta)$, denoted by Q_h, is the list of queries that F issued to its oracle H.*

2. *(Hitting heavy queries) For every fixing of the oracle H, with probability $1 - \delta$ over the internal coins of F the output Q_h of $F^H_A(\epsilon, \delta)$ satisfies the following*

$$\forall w \notin Q_h, \quad \Pr_{r_A, G}[w \in Q(A^{H[Q_h] \cup G}(r_A))] \leq \epsilon, \tag{1}$$

   where r_A denotes the random tape of A.

That is, in a random execution of A in which queries that belong to Q_h are answered by H and other queries are answered by an independent random oracle G, every string $w \notin Q_h$ will be hit by a query of A with probability at most ϵ.

Proof. Let us assume without loss of generality that A never makes the same query twice. For $i \in [T]$, we let A_i denote a modified version of the algorithm A which is halted after i queries. (Thus $A_T = A$.) The algorithm F proceeds as follows.

1. Take $\epsilon' = \epsilon/T$ and let $\ell = O((1/\epsilon')\log(T/(\delta \cdot \epsilon))$, and initializes an empty list Q_h.
2. For $i = 1$ to T do:
   (a) Sample ℓ random inputs, $r^1, \ldots, r^\ell$ for A, use "lazy sampling" to sample ℓ random oracles $G^1, \ldots, G^\ell$ and invoke ℓ executions $A_i(r^1), \ldots, A_i(r^\ell)$ where the first $i-1$ queries of the j-th instance are answered according to $H[Q_h] \cup G^j$.
   (b) Given the i-th tuple of queries $(q^1, \ldots, q^\ell)$ (hereafter referred to as the *i-queries*) we mark every string w that appears in at least $\epsilon'/2$ locations as "heavy" and add it to Q_h.

For $i \in [T]$ let $Q_h[i]$ denote the set Q_h at the end of the i-th iteration and let $Q_h[0]$ denote the empty set. Call $Q_h[i]$ *good* if (1) holds wrt to $Q_h[i]$, the algorithm A_i, and the error parameter $\epsilon_i = i \cdot \epsilon/T$. Since $Q_h[0]$ is trivially good, it suffices to show that for every $i \in [T]$,

$$\Pr[Q_h[i] \text{ is good} \mid Q_h[i-1] \text{ is good }] \geq 1 - \delta/T,$$

and conclude, via a union-bound, that $Q_h = Q_T$ is good with probability $1-\delta$, as required.

Fix some i, and condition on $Q_h[i-1]$ being good. We prove a lower-bound on the probability that $Q_h[i]$ is good. Call a string $w \notin Q_h[i-1]$ *heavy* if $\Pr[q_i = w] > \epsilon'$, where q_i is the random variable that represents the i-th query of A in a random execution where all the queries in $Q_h[i-1]$ are answered by H and all other queries are answered by a random oracle G. (So the probability is taken over the randomness of A and the randomness of G.) Such a heavy query is expected to appear in at least ϵ'-fraction of the i-queries, and therefore, by a Chernoff bound, a fixed heavy string w will be marked and added to $Q_h[i]$ except with probability of $\exp(-\Omega(\epsilon'\ell))$. Since there are at most $1/\epsilon'$ heavy strings, we conclude that, except we probability $\exp(-\Omega(\epsilon'\ell))/\epsilon' \leq \delta/T$, *all* heavy strings are added to $Q_h[i]$. In this case, $Q_h[i]$ is indeed good, since the probability (over r_A and G) that some $w \notin Q_h[i]$ is being queried by $A^{H[Q_h]\cup G}(r_A)$ is at most $\epsilon_{i-1} + \epsilon' = (i-1)\epsilon/T + \epsilon/T = \epsilon_i$, as required. □

4.4 Proof of Theorem 4.1

Let $\Pi = (P_1 = (\mathsf{pre}_1, \mathsf{post}_1), P_2 = (\mathsf{pre}_2, \mathsf{post}_2))$ be an elementary non-interactive reduction from f to $g(y_1, y_2) = (v_1, v_2)$ in the random-oracle model. Let $T = T(\lambda)$ be an upper-bound on the number of queries that are made by both P_1 and P_2. For an inverse polynomial parameter $\alpha := \alpha(\lambda)$ set $\epsilon = \alpha/(2T)$ and $\delta = \alpha/2$, and let $F = F^{\cdot}_{\mathsf{pre}_2}(\epsilon, \delta)$ denote the heavy-query finder applied to pre_2, viewed as a randomized algorithm. (That is, we concatenate the input x_2 of pre_2 to its random tape r_1.) Observe that the running time of F is $\mathsf{poly}(\lambda)$.

Recall that f denotes the P_1-authenticated version of f_1. Accordingly, the input of P_1 consists of a pair (x_1, k) where k is a MAC-key. Throughout this section, we assume that P_1 chooses k at random. Most of the time we will keep k implicit as part of the random tape r_1 of P_1. That is, we let $r_1 = (k, r'_1)$ where r'_1 denotes the random tape of P_1 that is consumed during the execution of the protocol.

4.4.1 An intermediate protocol Π_1 and the no-collision event

We consider the following modified reduction $\Pi_1 = (P_1, P'_2)$ in which the second party gets an access to the standard random oracle H and an additional private random oracle G. (Jumping ahead, the oracle G will be sampled locally by the second party via lazy sampling.) Specifically, the party P'_2 acts as follows:

— (preprocessing step) First P'_2 calls the heavy-query finder F^H with randomness r_F, and gets a list of queries Q_h. Then P'_2 invokes $\mathsf{pre}_2(x_2; r_2)$ with the oracle $H[Q_h] \cup G$ where G is a private oracle defined by coins r_G; That is, all the queries that are issued to the RO are answered by using the private oracle G unless they belong to Q_h.

— (postprocessing phase) P'_2 first generates the list L of all queries that were issued to G in the preprocessing step. Next, P'_2 invokes $\mathsf{post}_2(v_2; r_2)$ with the oracle $G[L] \cup H$. That is, whenever a query q is issued to the random oracle, she answers the query with $G(q)$ if $q \in L$, and answers it with $H(q)$ otherwise.

Our main goal is to show that P_1 is likely to output the correct value. This will follow from a sequence of claim, but first it will be useful to define some "good" event under which correctness holds. (This event will also serves us in the following sections.)

The good "no-collision" event E. Informally, the "no collision" event happens if in the preprocessing phase P_1 does not query any point that is sent in the preprocessing phase of P'_2 to G. Formally, let Q_G denote the list of oracle queries that the preprocessing phase of P'_2 sends to G when it is invoked with input x_2, r_2 and r_F and with the oracles H and G, and let $Q_1 = Q(\mathsf{pre}_1^H(x_1; r_1))$ denote the list of oracle queries that P_1 sends to H when it is invoked with inputs x_1, r_1 and oracle H, we define the "no-collision" event to be

$$E := \qquad Q_1 \quad \text{and} \quad Q_G \quad \text{are disjoint.}$$

Observe that E depends on the values $(H, x_1, r_1, x_2, r_2, r_F, G)$. The following claim follows from the properties of the algorithm F.

Claim 1 (Collisions are rare)**.** *For every H, x_1 and r_1, we have $\Pr_{x_2,r_2,G,r_F}[\neg E] \leq \alpha$.*

Proof. By definition, for every fixing of H, x_1 and r_1, it holds that

$$\Pr[\neg E] = \Pr_{x_2,r_2,G,r_F}[\exists q \in Q_1 \text{ s.t. } q \in Q_G]$$

which, by a union-bound, is upper-bounded by

$$\Pr_{r_F}[F^H \text{ fails}] + \sum_{q \in Q_1} \Pr_{x_2,r_2,G}[q \in Q_G] \leq \delta + T\epsilon \leq \alpha.$$

Here we say that F^H fails if it does not satisfy (1). The claim follows. □

Let $\beta(x_1, x_2) := \Pr_{H,G,r_1,r_2,r_F}[\neg E(H, x_1, r_1, x_2, r_2, r_F, G)]$. By Claim 1, for every x_1, the expectation $\mathbb{E}_{x_2}[\beta(x_1, x_2)]$ is at most α. If the reduction makes input-independent queries or uniform queries, a stronger bound can be easily derived.

Claim 2. *If pre_2 makes input-independent queries (resp., uniform queries) then for every x_1, x_2, it holds that $\beta(x_1, x_2)$ is upper-bounded by α (resp., by a negligible function in λ).*

Proof. For input-independent queries, $\beta(x_1, x_2)$ is independent of x_2. Therefore, for every x_1, x_2 we have that $\beta(x_1, x_2) = \mathbb{E}_{x_2}[\beta(x_1, x_2)]$ which, by Claim 1, is at most α. In the case of uniform queries, for every x_1, x_2, the probability that Q_G intersects with Q_1 is at most $|Q_G| \cdot |Q_1| \cdot 2^{-\lambda} = \mathsf{neg}(\lambda)$. □

4.4.2 Correctness of Π_1

We move on and show that, conditioned on E, the party P_2' is likely to output the "correct" value. Let $\mathsf{out}^H_{P_1,P_2'}(x_1, x_2; (r_1, r_2, r_F, G)) = (z_1', z_2')$ denote the outputs of P_1 and P_2' on inputs (x_1, x_2), oracle H, and random tapes r_1, r_2 and r_F and local oracle G.

Claim 3 (P_2' is typically correct)**.** *For every inputs x_1, x_2, consider the output distribution $\mathsf{out}^H_{P_1,P_2'}(x_1, x_2; (r_1, r_2, r_F, G))) = (z_1', z_2')$ induced by randomly chosen $r_1 = (k, r_1'), r_2, r_F$ and H, G. Then,*

$$\Pr[z_2' \neq f_2((x_1, k), x_2) | E] \leq \mathsf{neg}(\lambda). \tag{2}$$

We remark that the claim actually holds over a worst-case choice of k (though we will not need this property).

Proof. Fix x_1, r_1, x_2, r_2, r_F. We will show that conditioned on the event E, the random variable z_2' (defined above) is distributed identically to the output distribution of P_2 in an execution of the original reduction Π over x_1, r_1, x_2, r_2 and relative to a random oracle U. Thus the claim will follow from the correctness of the original reduction.

Define a mapping ρ_{x_2,r_2,r_F} that maps a pair of oracles $G, H : \mathcal{D} \to \mathcal{R}$ to an oracle $U : \mathcal{D} \to \mathcal{R}$ as follows. Let Q_G denote the list of oracle queries that the preprocessing phase of P_2' sends to G when it is invoked with input x_2, r_2 and r_F and with the oracles H and G. Then, set $U = G[Q_G] \cup H$; That is, for every possible $q \in \mathcal{D}$, set $U(q)$ to $G(q)$ if $q \in Q_G$ and otherwise set it to $H(q)$. We claim that, for any fixed (x_2, r_2, r_F), when the mapping ρ_{x_2,r_2,r_F} is applied to a pair of random oracles G and H the resulting oracle U is uniformly distributed over all functions from $\mathcal{D}$ to $\mathcal{R}$. To see this observe that U is defined by a procedure that (adaptively) send queries to G and H with the property that a query q is never issued twice, and whenever q is sent to $O \in \{G, H\}$ the value of $U(q)$ is being committed to $O(q)$.

Conditioned on the good (no-collision) event E, the joint output of the preprocessing phase, $(\mathsf{pre}_1^H(x_1; r_1), \mathsf{pre}_2^{H[Q_h]\cup G}(x_2; r_2))$, in the Π_1 execution, is distributed identically to the joint output, $(\mathsf{pre}_1^U(x_1; r_1), \mathsf{pre}_1^U(x_2; r_2))$, of the preprocessing phase in an honest execution of Π relative to the oracle U. Moreover, the postprocessing of P_2' takes $v_2 = g_2((\mathsf{pre}_1^H(x_1; r_1), \mathsf{pre}_2^{H[Q_h]\cup G}(x_2; r_2)))$ and outputs exactly the value $\mathsf{post}_2^U(v_2; r_2)$ that an honest P_2 would output in Π relative to the oracle U. We conclude that, under E, the final output z_2' of P_2' and the final output z_2 of P_2 in $\mathsf{out}_{P_1,P_2}^U(x_1, x_2; (r_1; r_2)) = (z_1, z_2)$ are identically distributed, and so the claim follows by the correctness of the original reduction Π. □

By using partial fairness (and the security of the MAC), we will show that P_1 is likely to output the correct value whenever P_2 does it. Formally, we prove the following claim.

Claim 4 (P_1 is correct when P_2' is correct)**.** *For every input x_1 and x_2,*

$$\Pr_{H,r_1,r_2,r_F,G}[z_1' \neq f_1(x_1, x_2) \bigwedge z_2' = f_2((x_1, k), x_2)] \leq \mathsf{neg}(\lambda), \tag{3}$$

where $\mathsf{out}_{P_1,P_2'}^{H,G}(x_1, x_2; (r_F, r_1, r_2))) = (z_1', z_2')$ *and* $r_1 = (k, r_1')$.

Proof. Fix some (x_1, x_2) and assume that $r_1 = (k, r_1')$, r_2, r_F, G and H are uniformly distributed and that (z_1', z_2') denote the corresponding output of the protocol. We can think of Π_1 as an execution of Π under an adversary P_2' that corrupts the second party. Therefore, there exists a simulator Sim, promised by partial fairness of Π, that makes a single query to the ideal functionality f with some input x_2' that depends only on x_2, on the simulator's randomness and on H. Denote by (z_1, z_2) the outputs of the simulator and P_1 under an ideal execution when the input of the simulator is x_2 and the input of P_1 is (x_1, k). By the security of the simulator, it holds that the probability of the "ideal-world" event

$$B: \quad z_1 \neq f_1(x_1, x_2) \bigwedge z_2 = f_2((x_1, k), x_2)$$

and the probability of the "real-world" event

$$B': \quad z_1' \neq f_1(x_1, x_2) \bigwedge z_2' = f_2((x_1, k), x_2)$$

are within a $\mathsf{neg}(\lambda)$ difference. (Otherwise a distinguisher can efficiently distinguish between the two worlds.) Therefore, to prove (3) it suffices to show that the "ideal-world" event B happens with negligible probability.

First, observe that B can happen only if the simulator submits to f an input $x'_2 \neq x_2$. (Otherwise, if $x'_2 = x_2$ then the output z_1 of P_1 is $f_1(x_1, x_2)$, and the first condition of B is violated.) However, when $x'_2 \neq x_2$ the functionality f_2 returns to the simulator a MAC-tag over x'_2 (or a $\perp$ symbol) and the probability that the simulator can guess the authentication tag on x_2 is negligible. It follows that, except with negligible probability, the output of the simulator z_2 is not equal to $\mathsf{MAC}_k(x_2)$, which violates the second part of B. □

By combining all three claims we derive the following lemma.

Lemma 4.2 (P_1 is typically correct)**.** *For every input x_1 and x_2,*

$$\Pr_{r_1=(r'_1,k),r_2,r_F,G,H}[z'_1 \neq f_1(x_1,x_2)|E] \leq \mathsf{neg}(\lambda) \tag{4}$$

where $\mathsf{out}^{H,G}_{P_1,P'_2}(x_1, x_2; (r_F, r_1, r_2))) = (z'_1, z'_2)$.

Proof. The LHS is upper-bounded by

$$\Pr_{H,r_1,r_2,r_F,G}[z'_1 \neq f_1(x_1,x_2) \bigwedge z'_2 = f_2((x_1,k),x_2)|E] + \Pr_{H,r_1,r_2,r_F,G}[z'_2 \neq f_2((x_1,k),x_2)|E].$$

By Claims 1 and (4), the first summand is negligible, and by Claim 3 the second summand is also negligible. □

4.4.3 Privacy of Π_1

Our next goal is to show that Π_1 preserves privacy against a passive adversary that corrupts the first party P_1. Recall that the view of P_1 (both in Π and in Π_1) consists of the following values: the input x_1, the random tape r_1, a list of the preprocessing queries Q_1 and a list of all the oracle responses R_1, the value v_1 given to the first party by the functionality g, and a list S_1 of queries that are performed in the postprocessing step and the list of the corresponding oracle answers T_1.

We relate the P_1-privacy of Π_1 to the P_1-privacy of Π by showing that the view in both experiments is statistically close. In fact, it will be useful to prove this even when P_1 gets to see the randomness r_F used by P'_2 for sampling Q_h. Let us refer to this modified protocol as Π_2. (E.g., think of r_F as being taken from a shared random tape.)

Claim 5. *There exists an efficent randomized oracle-aided procedure $\rho^{(\cdot)}(\cdot)$ that takes a P_1-view w under Π and outputs a P_1-view w' under Π_2 and lazy-samples an oracle H such that the following holds. For every (x_1, x_2) the distribution of*

$$(H, w') = \rho^V(\mathsf{view}^V_{\Pi,P_1}(x_1, x_2; r_1, r_2)) \qquad \textit{where } V, r_1, r_2 \textit{ are uniform,}$$

is $O(\beta(x_1, x_2))$-statistically close to

$$(H, \mathsf{view}^H_{\Pi_2, P_1}(x_1, x_2; r_1, r_2, r_F, G)) \qquad \textit{where } H, r_1, r_2, r_F, G \textit{ are uniform.}$$

Recall that $\beta(x_1, x_2) := \Pr_{H,G,r_1,r_2,r_F}[\neg E(H, x_1, r_1, x_2, r_2, r_F, G)]$.

Proof. Given $w = (x_1, r_1, Q_1, R_1, v_1, S_1, T_1)$ (supposedly a P_1-view under Π) and an oracle V, the mapping ρ does the following:

— Sample a random tape r_F for F, let $Q_h = F^V(r_F)$, and let H be a fresh random oracle that is consistent with V on the queries $Q_1 \cup Q_h$.

— Invoke $\mathsf{post}^H_1(v_1; r_1)$ with the oracle H and collect all the queries and their answers in the lists (S'_1, T'_1).

— Output $w' = (x_1, r_1, r_F, Q_1, R_1, v_1, S'_1, T'_1)$.

Fix x_1 and x_2 and let w be a random view of P_1 that corresponds to the experiment $\Pi^V(x_1, r_1, x_2, r_2)$ with randomly chosen tapes r_1 and r_2 and random oracle V. We will show that w' is statistically-close to $\mathsf{view}^H_{\Pi_2, P_1}(x_1, x_2; r_1, r_2, G)$ where the oracle G is defined as follows.

For every fixing of x_2, r_2 and r_F, let us denote by L the list of queries that P'_2 sends to its "local" oracle when its global oracle is set to V and its local oracle is also set to V. Formally, $L = Q_2 \setminus Q_h$ where $Q_h = F^V(r_F)$ and $Q_2 = Q(\mathsf{pre}^V_2(x_2; r_2))$. Then, the oracle G is defined to be $V[L] \cup U$ where U is a fresh oracle.

Let $\beta = \beta(x_1, x_2)$. We begin by showing that the joint distribution

$$(H, G) \text{ is } 3\beta\text{-close to uniform.} \tag{$*$}$$

Since E happens with probability $1 - \beta$, it suffices to show that the conditional distribution $((H, G)|E)$ is β-close to uniform. Next, note that under E the oracles G and H are statistically independent (since G and H are based on disjoint entries of V), and therefore it suffices to show that $(G|E)$ is β-close to uniform and that $(H|E)$ is β-close to uniform. Indeed, without conditioning, the marginal distribution of G (resp., H) is uniform and so the conditional distribution deviates from uniform by at most $\Pr[\neg E] = \beta$, and ($*$) follows.

From now on, fix the random tapes r_1, r_2, r_F and the oracles V, H and G. We begin by showing that the response that the oracle g sends to P_1 in an execution of $\Pi^H_2(x_1, x_2; r_1, r_2, r_F, G)$ is equal to v_1. Indeed, the values (x_1, r_1, Q_1, R_1) in the execution of $\Pi^V(x_1, r_1, x_2, r_2)$ are identical to the ones that are computed in the execution $\Pi^H_2(x_1, r_1, x_2, r_2, r_F, G)$. (Since $Q_1 = Q(\mathsf{pre}^V_1(x_1; r_1)) = Q(\mathsf{pre}^H_1(x_1; r_1))$ and $R_1 = V(Q_1) = H(Q_1)$.) Consequently, in both experiments, P_1 sends the same message y_1 to the oracle g. Furthermore, in both experiments, P_2 sends the same message y_2 to the oracle g since $y_2 = \mathsf{pre}^V_2(x_2; r_2) = \mathsf{pre}^{H[Q_h] \cup G}_2(x_2; r_2)$. We conclude that the ideal functionality g is applied to the same input in both experiments and therefore, conditioned on the above, the message, v_1, that is being delivered by g to P_1 is distributed identically in both experiments. Finally, the postprocessing queries S'_1 and their answers T'_1 are generated in both experiments by applying the deterministic procedure $\mathsf{post}_1^H(v_1; r_1)$. $\square$

We can now prove the following lemma.

Lemma 4.3 (P_1-privacy)**.** *There exists a simulator against a passive P_1 adversary such that for every (x_1, x_2) the statistical deviation of the simulator from the real distribution, as defined in Definition (4.1), is upper-bounded by $O(\beta(x_1, x_2)) + \mathsf{neg}(\lambda)$.*

Consequently, by Claim 1 the protocol Π_2 is $O(\alpha)$-AVG-private against the first party. Moreover, by Claim 2, if pre_2 makes input-independent queries then the protocol is $O(\alpha)$-private, and if pre_2 makes uniform queries then Π_1 is private (with negligible privacy error).

Proof. Given an input/output pair (x_1, z_1) for P_1, we define a P_1-simulator $\mathsf{Sim}_2(x_1, z_1)$ for Π_2 as follows. Call the P_1-simulator $\mathsf{Sim}(x_1, z_1)$ of Π and generate a view $w = (x_1, r_1, Q_1, R_1, v_1, S_1, T_1)$ together with a simulated random oracle V. Generate a view w' of the new protocol together with an oracle H, by calling the procedure $\rho^V(w)$ promised in Claim 5.

We show that for every (x_1, x_2), the simulated view deviates from the real view by $O(\beta(x_1, x_2)) + \mathsf{neg}(\lambda)$. Indeed, fix an oracle-aided distinguisher D^H that distinguishes with advantage Δ between the Π_2-simulated view, $\mathsf{Sim}_2(x_1, z_1)$, to the real view $\mathsf{view}^H_{\Pi_2,P_1}(x_1, x_2)$ where H (and the local random tapes) are random. We construct a distinguisher D' against Π as follows: Given w and an oracle access to V, compute $\rho^V(w) = (H, w')$ and output $D^H(w')$.

By definition, it holds that

$$\Pr_{(V,w)\leftarrow\$\mathsf{Sim}(x_1,z_1)}[D'^V(w) = 1] = \Pr_{(H,w')\leftarrow\$\mathsf{Sim}_2(x_1,z_1)}[D^H(w') = 1].$$

Also, by Claim 5,

$$\left|\Pr[D'^V(\mathsf{view}^V_{\Pi,P_1}(x_1, x_2)) = 1] - \Pr[D^H(\mathsf{view}^H_{\Pi_2,P_1}(x_1, x_2)) = 1]\right| \leq 3\beta(x_1, x_2).$$

By the privacy of Π_1, we conclude that $\Delta + 3\beta(x_1, x_2) \leq \mathsf{neg}(\lambda)$ and the lemma follows. $\square$

4.4.4 Deriving Theorem 4.1

We can now remove the random oracle and derive Theorem 4.1. Let g' denote a modified version of g whose input, in addition to (y_1, y_2), consists of the query list Q_1 that the first party issued to H in the preprocessing phase, and the list L that the second party in Π_2 issued to its local oracle during the preprocessing stage. In addition, the functionality takes the private input x_2 from the second party. The functionality g' checks if there is a collision between the lists Q_1 and L (i.e., if $\neg E$ happens) and if this is the case it sends a flag $e = 0$ together with x_2 to the first party. Otherwise, it computes $g(y_1, y_2) = (v_1, v_2)$, sends v_1 to the first party with the flag $e = 1$. In any case, g' delivers a $\perp$ to the second party.

Consider the following reduction Σ from f_1 to g'.

1. The first party lazy samples a random oracle H and samples random tapes r_F and r_1. She computes $Q_h = F^H(r_F)$ and sends to the second party the list of query/response pairs $(q, H(q))_{q \in Q_h}$. (Note that it suffices to send r_F and all the H-answers that are provided to $F^H(r_F)$ and therefore the first message consists of a sequence of random coins.)

2. The two parties generate g-queries (y_1, y_2) like in Π_1. Namely, the first party computes $y_1 = \mathsf{pre}_1^H(x_1; r_1)$ and the second party computes $y_2 = \mathsf{pre}_2^{H[Q_h] \cup G}(x_2; r_2)$ where r_2 is a fresh random tape and G is a private oracle that is sampled locally. The parties call g' with the inputs $(y_1, Q_1 = Q(\mathsf{pre}_1^H(x_1; r_1)))$ for the first party and the inputs y_2, x_2 and $L = Q(\mathsf{pre}_2^{H[Q_h] \cup G}(x_2; r_2)) \setminus Q_h$ for the second party.

3. In the postprocessing phase, the first party checks the received flag e. If $e = 1$ it retrieves the value v_1 from the oracle g' and outputs $\mathsf{post}_1^H(v_1; r_1)$. Otherwise, it retrieves the value x_2 and outputs $f_1(x_1, x_2)$. In any case, the second party aborts with an empty output.

The syntax of Σ satisfies the syntax promised in Theorem 4.1 (item 1). Indeed, to see that g' can be implemented in NC^1, we observe that each pair of strings (q_1, q_2) in $Q_1 \times Q_G$ are of bit length $\ell = \mathsf{poly}(\lambda)$ and so we can check if $q_1 = q_2$ by an $O(\ell)$-size circuit of depth $\log(\ell) + O(1)$. Since the number of pairs in $|Q_1 \times Q_G|$ is polynomial in λ, we can check all pairs in parallel and aggregate the result via an "OR-tree". Overall we can detect if E happens by an NC^1 circuit, and combine it with the original NC^1 circuit of g (promised by the fact that it is a constant-degree functionality).

Also, the only value that P_2 receives is $(q, H(q))_{q \in Q_h}$ which is distributed independently of P_1's input. Therefore, the protocol is private against the second party. To analyze privacy against the first party, observe that, for every x_1, x_2 and oracle H, conditioned on the event E, the view of P_1 in Σ is distributed identically to its view under Π_2. Hence, by Lemma 4.3, there exists a simulator that on (x_1, x_2) has a statistical deviation of $O(\beta(x_1, x_2)) + \mathsf{negl}(n)(\lambda)$. Consequently, by Claim 1 the protocol Σ is $O(\alpha)$-AVG-private against the first party. Moreover, by Claim 2, if pre_2 makes input-independent queries then the protocol is $O(\alpha)$-private, and if pre_2 makes uniform queries then Σ is private (with negligible privacy error).

As for correctness, observe that, conditioned on E, the output distribution of $\Sigma(x_1, x_2)$ is identical to the output distribution of $\Pi_1^H(x_1, x_2)$. Therefore, Lemma 4.2 guarantees correctness under E (except with negligible error probability). When E does not happen, correctness holds trivially since x_2 is being revealed.

5 Distributed Encryption

5.1 Definitions

In this section, we define a new notion of a multi-party symmetric-key encryption scheme, which we call a *distributed encryption scheme.* In this primitive, each party samples a key independent of the other parties, and the message is simultaneously encrypted under each party's key. Although this is a symmetric-key primitive, we distinguish between two types of keys: "encryption" keys that are being used by the encryption algorithm, and "decryption" keys that are being used by the decryption algorithm. This will allow us to define the encryption algorithm in a way that is independent of the underlying cryptographic primitive (e.g., a PRG), without giving-up on the benefits of computational security; Notably, it will be crucial to have decryption keys whose bit-length is shorter than the message's length. We define different forms of privacy and correctness properties which hold as long as the adversary is efficient (i.e., runs in non-uniform probabilistic polynomial-time in the security parameter λ) and corrupts at most $t(n)$ out of the n parties. We refer to $t = t(n)$ as the corruption threshold. (The reader may think of $t = n - 1$ as the default value of t.)

Definition 5.1 (Distributed Encryption: Syntax and Correctness)**.** *A* distributed encryption scheme *is defined by a triple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *as follows:*

— $\mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell) \to (\mathsf{ek}, \mathsf{dk})$*: On input the security parameter λ, the number of parties $n = n(\lambda)$ and the bit-length of the messages ℓ, the randomized key generation algorithm outputs a decryption key* dk *and an encryption key* ek*. The length of* dk *is required to be $p(\lambda)$ for some fixed polynomial p that is independent of n and ℓ. We further assume, by default, that* dk *is simply the coins used by λ.*

— $\mathsf{Enc}(1^\lambda, 1^n, 1^\ell, m, \mathsf{ek}_1, \dots, \mathsf{ek}_n) \to \mathsf{ct}$*: On input the security parameter λ, a message $m \in \{0,1\}^\ell$ and a set of encryption keys $\mathsf{ek}_1, \dots, \mathsf{ek}_n$, the randomized encryption algorithm outputs a ciphertext* ct*.*

— $\mathsf{Dec}(\mathsf{ct}, \mathsf{dk}_1, \dots, \mathsf{dk}_n) \to (b, m, \mathsf{bad})$*: Given a ciphertext* ct *and a set of decryption keys $(\mathsf{dk}_1, \dots, \mathsf{dk}_n)$ as inputs, the deterministic decryption algorithm outputs a validity bit $b \in \{0,1\}$, the message $m \in \{0,1\}^\ell \cup \{\perp\}$ and a set $\mathsf{bad} \subset [n]$.*

The scheme should be perfectly correct*: For every λ, n, ℓ, every vector of keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i \in [n]}$, where $\mathsf{ek}_i, \mathsf{dk}_i$ is in the support of $\mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell)$, and every message $m \in \{0,1\}^\ell$ it holds that*

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, 1^n, 1^\ell, m, \mathsf{ek}_1, \dots, \mathsf{ek}_n), \mathsf{dk}_1, \dots, \mathsf{dk}_n) = (1, m, \emptyset)] = 1.$$

Our syntax for the encryption algorithm is somewhat degenerate since ℓ and n can be extracted from m and $\mathsf{ek}_1, \dots, \mathsf{ek}_n$. We therefore typically omit these parameters except for special cases where we wish to emphasize the restriction of Enc to concrete lengths.

We now proceed to define three properties of a distributed encryption scheme: *privacy, security with abort*, and *security with identifiable abort.*

Privacy. An adversary who is allowed to choose a subset of the encryption keys, should not be able distinguish between encryptions of any two messages of its choice.

Definition 5.2 ($t(n)$-Privacy)**.** *A distributed encryption scheme is said to have one-time* $t(n)$-privacy *(where $t(n) < n$) if every n.u PPT adversary $\mathcal{A}$ cannot win the following game with more than $\frac{1}{2} + \mu(\lambda)$ probability where μ is some negligible function:*

- *$\mathcal{A}(1^\lambda)$ selects the following values and sends them to the challenger: parameters 1^n and 1^ℓ, a pair of messages $m_0, m_1 \in \{0,1\}^\ell$, a $t(n)$-subset $\mathcal{I} \subset [n]$, and a tuple of arbitrary encryption keys $\{\mathsf{ek}_i\}_{i\in\mathcal{I}}$ for the corrupted parties.*
- *The challenger samples keys for the remaining parties, i.e., for each $i \in [n] \setminus \mathcal{I}, (\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell)$ and samples a bit $b \leftarrow\!\$ \{0,1\}$ uniformly at random. It then sends the ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m_b, \mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ to $\mathcal{A}$.*
- *The adversary responds with a bit b' and wins if $b' = b$.*

Security with Abort. In addition to privacy, we require different flavors of correctness which hold even against active adversaries that corrupt the keys. Our first variant requires security-with-abort. Formally,

Definition 5.3 ($t(n)$-Security with Abort)**.** *A distributed encryption scheme is said to have one-time* $t(n)$-security with abort *(where $t(n) < n$) if it satisfies $t(n)$-privacy and satisfies the following additional property. There exists an efficiently computable randomized predicate P that outputs "pass" (1) or "abort" (0), such that for every polynomials $n = n(\lambda), \ell = \ell(\lambda)$, every $t(n)$-subset $\mathcal{I} \subset [n]$, every message $m \in \{0,1\}^\ell$, every tuple of corrupted keys $\{\mathsf{ek}_i, \mathsf{dk}_i\}_{i\in\mathcal{I}}$ and every tuple of honest keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]\setminus\mathcal{I}}$ that are in the support of $\mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell)$, the following* detection *property holds.*

Let $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m, \mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ and $\vec{\mathsf{dk}} = (\mathsf{dk}_1, \ldots, \mathsf{dk}_n)$, then

$$\Pr[\mathsf{Dec}(\mathsf{ct}, \vec{\mathsf{dk}}) = (1, m, \emptyset) \vee \mathsf{Dec}(\mathsf{ct}, \vec{\mathsf{dk}}) = (0, \perp, \emptyset)] \geq 1 - \epsilon(\lambda),$$

and

$$\Pr[\mathsf{Dec}_1(\mathsf{ct}, \vec{\mathsf{dk}}) = \mathsf{P}((\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]})] \geq 1 - \epsilon(\lambda),$$

where Dec_1 denotes the first output of the decryption algorithm (the validity bit), the probability is taken over the choice of ct and the internal randomness of P, and $\epsilon(\cdot)$ is some negligible function.

Few comments are in place. First, observe that the adversary wins if it can make the decryption algorithm err (output a value different than m or $\perp$). In addition, the above definition essentially implies that the event of aborting cannot depend on the message m (since the predicate P should simulate it without knowing m). Moreover, this simulation should succeed even when we condition on a fixed value of the honestly generated decryption/encryption keys.

We construct a distributed encryption scheme satisfying $(n-1)$-security with abort based on message-authentication codes in Section 5.2.

Security with Identifiable Abort. In the following we strengthen our requirement so that when abort happens a bad party is identified.

Definition 5.4 ($t(n)$-Security with Identifiable Abort)**.** *A distributed encryption scheme is said to have one-time $t(n)$-*security with identifiable abort *(where $t(n) < n$) if it has $t(n)$-privacy and there exists an efficiently computable randomized algorithm P that given a vector of encryption/decryption keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]}$, outputs a a validity bit $a \in \{0,1\}$ (where zero indicates "abort" and one indicates "pass") and a subset $\mathsf{bad} \subset [n]$ such that for every polynomials $n = n(\lambda), \ell = \ell(\lambda)$, every $t(n)$-subset $\mathcal{I} \subset [n]$, every message $m \in \{0,1\}^\ell$, every tuple of corrupted keys $\{\mathsf{ek}_i, \mathsf{dk}_i\}_{i\in\mathcal{I}}$ and every tuple of honest keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]\setminus\mathcal{I}}$ that are in the support of $\mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell)$, the following* identification *property holds.*

Let $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m, \mathsf{ek}_1, \ldots, \mathsf{ek}_n)$ and $\vec{\mathsf{dk}} = (\mathsf{dk}_1, \ldots, \mathsf{dk}_n)$, then

$$\Pr[\mathsf{Dec}(\mathsf{ct}, \vec{\mathsf{dk}}) = (1, m, \emptyset) \vee \Big((\mathsf{Dec}(\mathsf{ct}, \vec{\mathsf{dk}}) = (0, \perp, \mathsf{bad}) \wedge \emptyset \neq \mathsf{bad} \subset \mathcal{I}\Big)] \geq 1 - \epsilon(\lambda), \quad (5)$$

and

$$\Pr[\mathsf{Dec}_{1,3}(\mathsf{ct}, \vec{\mathsf{dk}}) = \mathsf{P}((\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]})] \geq 1 - \epsilon(\lambda), \quad (6)$$

where $\mathsf{Dec}_{1,3}$ denotes the first and last outputs of the decryption algorithm (the validity bit and the set), the probability is taken over the choice of ct and the internal randomness of P, and $\epsilon(\cdot)$ is some negligible function.

Identification is a strictly stronger property than detection which ensures identification of malformed key pairs. In other words, if the decrypted message is $\perp$, the decryption algorithm also outputs a non-empty set $\mathsf{bad} \subseteq [n]$ consisting of at least one corrupted party. Additionally, the algorithm P can perfectly simulate the validity bit and the set bad given only the vector of encryption/decryption keys. Moreover, this simulation should succeed even when we condition on a fixed value of the honestly generated decryption/encryption keys.

In Section 5.3 we show how to upgrade any distributed encryption scheme satisfying $(n-1)$-security with abort into an $(n-1)$-secure scheme with identifiable abort.

5.2 Distributed Encryption Satisfying Security with Abort

In this section we present our construction of a distributed encryption scheme that satisfies $(n-1)$-security with abort. The construction naturally follows the MAC-and-Encrypt paradigm while exploiting the *internal randomness* of the encryption algorithm for the use of MAC.

Theorem 5.1. *Assuming the existence of a pseudorandom generator, there exists a distributed encryption scheme that satisfies $(n-1)$-security with abort. Moreover, the key-generation and decryption algorithms make a black-box use of the PRG and the encryption algorithm is independent of the PRG and is in NC^1 whenever n and ℓ are polynomial in λ.*[9]

Information-theoretic MAC. For the proof of the theorem, we will need to employ an information theoretic one-time secure MAC scheme $(\mathsf{M.KeyGen}, \mathsf{MAC})$ where $\mathsf{M.KeyGen}(1^\ell, 1^\lambda)$ samples a key mk such that $\mathsf{MAC}_{\mathsf{mk}} : \{0,1\}^\ell \to \{0,1\}^\lambda$ is pair-wise independent hash function. That is, for every pair of inputs $x \neq y \in \{0,1\}^\ell$, for $\mathsf{mk} \leftarrow \mathsf{M.KeyGen}(1^\ell, 1^\lambda)$, the random variable $(\mathsf{MAC}_{\mathsf{mk}}(x), \mathsf{MAC}_{\mathsf{mk}}(y))$ is uniform over $\{0,1\}^\lambda \times \{0,1\}^\lambda$. In standard implementation (e.g., using Toeplitz-based affine transformation) the key-length is of size $\ell + 2\lambda$, and both $(\mathsf{M.KeyGen}, \mathsf{MAC})$ can be computed by an NC^1 circuit of size polynomial in ℓ and λ.

Lemma 5.1. *The construction in Figure 2 satisfies $(n-1)$-security with abort.*

Proof. We prove privacy and detection separately.

Privacy. For privacy, assume for the sake of contradiction, that the construction in Figure 2 is not private. This implies that there exist messages $m_0, m_1 \in \{0,1\}^\ell$, keys $(\mathsf{ek}_i)_{i \in \mathcal{I}}$ and a polynomial time adversary $\mathcal{A}$ corrupting an $(n-1)$-subset $\mathcal{I} \subset [n]$ of parties, that can distinguish between $\mathsf{Enc}(1^\lambda, m_0, (\mathsf{ek}_i)_{i\in[n]})$ and $\mathsf{Enc}(1^\lambda, m_1, (\mathsf{ek}_i)_{i\in[n]})$ with non-negligible probability $\epsilon(\lambda)$ where $(\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$ for every $i \notin \mathcal{I}$. We use a hybrid argument to get a contradiction. Let $j \notin \mathcal{I}$ be some honest party. Fix arbitrary values for the keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i \neq j}$, and consider the following hybrids:

$$\begin{aligned}
H_1 &: \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m_0, (\mathsf{ek}_i)_{i\in[n]}) && \text{where } \mathsf{ek}_j = \mathsf{PRG}(\mathsf{dk}_j), \mathsf{dk}_j \leftarrow U_\lambda \\
H_2 &: \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m_0, (\mathsf{ek}_i)_{i\in[n]}) && \text{where } \mathsf{ek}_j \leftarrow\$ \{0,1\}^{\lambda+\ell} \\
H_3 &: \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m_1, (\mathsf{ek}_i)_{i\in[n]}) && \text{where } \mathsf{ek}_j \leftarrow\$ \{0,1\}^{\lambda+\ell} \\
H_4 &: \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, m_1, (\mathsf{ek}_i)_{i\in[n]}) && \text{where } \mathsf{ek}_j = \mathsf{PRG}(\mathsf{dk}_j), \mathsf{dk}_j \leftarrow U_\lambda
\end{aligned}$$

[9] More precisely, for every fixed polynomials $n(\lambda)$ and $\ell(\lambda)$, the encryption algorithm, viewed as an infinite sequence of functions $\{\mathsf{Enc}_\lambda\}_{\lambda\in\mathbb{N}}$ that is parameterized solely by λ, can be realized by a poly-time uniform NC^1 circuit family. This notion of efficiency, that will be also used in the next subsection, suffices for our needs.

DE Secure with Abort

— $\mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell)$: Sample a random decryption key $\mathsf{dk} \leftarrow\!\$\ \{0,1\}^\lambda$. Expand dk into a pseudorandom string $\mathsf{ek} \in \{0,1\}^{\lambda+\ell}$ using the pseudorandom generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\lambda+\ell}$, i.e., set $\mathsf{ek} = \mathsf{PRG}(\mathsf{dk})$ and output $(\mathsf{ek}, \mathsf{dk})$. (Note that the key-generation is independent of the number of keys n.)

— $\mathsf{Enc}(1^\lambda, m, \mathsf{ek_1}, \ldots, \mathsf{ek_n})$: Sample a random MAC key $\mathsf{mk} \leftarrow \mathsf{M.KeyGen}(1^\ell, 1^\lambda)$, and output the ciphertext:

$$\mathsf{ct} = \left[\left((m||\mathsf{MAC}\,(m, \mathsf{mk})) \oplus \bigoplus_{i\in[n]} \mathsf{ek}_i\right), \mathsf{mk}\right].$$

— $\mathsf{Dec}(\mathsf{ct}, \mathsf{dk_1}, \ldots, \mathsf{dk_n})$: Parse $\mathsf{ct} = (\mathsf{ct}', \mathsf{mk})$ and compute $(m'||\mathsf{tag}') = \mathsf{ct}' \oplus \bigoplus_{i\in[n]} \mathsf{PRG}(\mathsf{dk}_i)$. If $\mathsf{MAC_{mk}}(m') = \mathsf{tag}'$ output $(1, m', \emptyset)$, else output $(0, \perp, \emptyset)$, where $\emptyset$ denotes the "null" set.

Figure 2: A distributed encryption scheme that satisfies $(n-1)$-security with abort

Since there are 4 hybrids, there must be a $j \in [3]$ such that $\mathcal{A}$ can distinguish between neighboring hybrids H_j and H_{j+1} with probability $\epsilon(\lambda)/4$. However, if this is the case, then we can show that $\mathcal{A}$ can be used to distinguish a randomly chosen string from an output of PRG. Since H_2 and H_3 are identically distributed, therefore $j \in \{1, 3\}$. Let $j = 1$ (the case for $j = 3$ is similar), i.e., $\mathcal{A}$ can distinguish between neighboring hybrids H_1 and H_2 with probability $\epsilon(\lambda)/4$. Given a string z of length $\lambda + \ell$ (that is either sampled from $\mathsf{PRG}(U_\lambda)$ or from $U_{\lambda+\ell}$), we can set $\mathsf{ek}_j = z$ and compute $\mathsf{ct} = \mathsf{Enc}(1^\lambda, m_0, \{\mathsf{ek}_i\}_{i\in[n]})$. If z is sampled from $\mathsf{PRG}(U_\lambda)$, then this is distributed exactly as H_1, else if it is sampled from $U_{\lambda+\ell}$, then this is identically distributed to H_2. Hence, we can now use $\mathcal{A}$ to break the pseudorandomness of PRG, which is a contradiction.

Detection. For detection, let us start by describing the predicate P. Given the set of all encryption/decryption key pairs, $(\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]}$ the predicate P computes, for each $i \in [n]$ the values $\Delta_i = \mathsf{ek}_i \oplus \mathsf{PRG}(\mathsf{dk}_i)$ and checks if $\Delta = \bigoplus_{i\in[n]} \Delta_i$ is the all-zero string. If this check goes through, it outputs 1 (pass), and otherwise it outputs 0 (abort).

We prove detection with respect to the predicate P. Fix a vector of keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]}$ and let Δ be the sum of differences as defined above. Fix a message m and observe that the corresponding ciphertext is distribute as $\mathsf{ct} = (\mathsf{ct}', \mathsf{mk})$ where

$$\mathsf{ct}' = \left((m||\mathsf{tag}) \oplus \bigoplus_{i\in[n]} \mathsf{ek}_i\right), \mathsf{mk} \leftarrow \mathsf{M.KeyGen}(1^\ell, 1^\lambda), \text{ and } \mathsf{tag} = \mathsf{MAC}\,(m, \mathsf{mk})\,.$$

We prove that if the predicate output "pass" then decryption succeeds, and if the predicate outputs "abort", the decryption aborts, except with negligible probability. The first step of the decryption algorithm outputs the value

$$(m'||\mathsf{tag}') = \mathsf{ct}' \oplus \bigoplus_{i\in[n]} \mathsf{PRG}(\mathsf{dk}_i) = (m||\mathsf{tag}) \oplus \Delta.$$

If $\mathsf{P}((\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]}) = 1$ then Δ is the all-zero string, $\mathsf{tag}' = \mathsf{tag}$, and therefore $\mathsf{Dec}(\mathsf{ct}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n) = (1, m, \emptyset)$, as required. On the other hand, if Δ is not the all zero string then, by the pairwise independence of the MAC, the probability that $\mathsf{tag}' = \mathsf{tag}$ is at most $2^{-\lambda}$. We conclude that in this case, the decryption algorithm outputs $(0, \perp, \emptyset)$, except with negligible probability.

This completes the proof of Lemma 5.1. □

5.3 Distributed Encryption Satisfying Security with Identifiable Abort

Theorem 5.2. *Assuming the existence of a distributed encryption scheme* $(\overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ *that satisfies one-time $(n-1)$-security with abort there exists a distributed encryption scheme that satisfies $(n-1)$-security with identifiable abort. Moreover, if the key-generation and decryption algorithms of the original scheme make a black-box use of the PRG, then so do the new key-generation and decryption algorithms, and if the orignal encryption algorithm is independent of the PRG and is in* NC^1 *(whenever n and ℓ are polynomial in λ) then so is the new encryption algorithm.*

The proof is based on a simple cut-and-choose approach. We let each party generate $k = O(\lambda n)$ many keys for the underlying encryption (that satisfies security-with abort) and let the encryption algorithm sample a random subset of the encryption keys. The selected keys are used for encrypting many copies of the message under the selected keys, and the keys that were not selected are published as part of the ciphertext. The decryption algorithm checks that the revealed encryption keys are well-formed (with respect to the corresponding decryption keys), and if they are all well-formed, it decrypts all the ciphertexts under the corresponding keys and takes the majority value. Roughly, we show that, except with negligible probability, either decryption reveals a mal-formed key (and can identify a cheater) or most of the ciphertexts decrypt well and the final outcome is correct. The scheme is formally described in Figure 3, and its security is analyzed in Lemma 5.2.[10]

Remark 3 (Secrecy-preserving identifiable abort)**.** *Our construction (Figure 3) provides a useful feature of "Secrecy-preserving identifiable abort". That is, we can augment the encryption algorithm to output the decryption keys $(\mathsf{dk}_i^j)_{i\in[n],j\in S}$ as part of the ciphertext, and think of the ciphertext as being composed of two parts: a "tag" that consists of $S, (\mathsf{ek}_i^j, \mathsf{dk}_i^j)_{i\in[n],j\in S}$ and a "data" $\{\mathsf{ct}^j\}_{j\in[2n\lambda]\setminus S})$. Note that the secrecy of the message is*

[10] The construction presented here simplifies the one that appears in the conference version of this paper.

DE Secure with Identifiable Abort

— $\mathsf{KeyGen}(1^\lambda, 1^n, 1^\ell)$: Let $s = \lambda n$ and $k = 2s$. A key-pair consists of k key-pairs of the underlying secure with abort distributed encryption scheme. That is, for $j \in [k]$, sample $(\mathsf{ek}^j, \mathsf{dk}^j) \leftarrow \overline{\mathsf{KeyGen}}(1^\lambda, 1^n, 1^\ell)$ and output $(\mathsf{ek} = (\mathsf{ek}^1, \ldots, \mathsf{ek}^k), \mathsf{dk} = (\mathsf{dk}^1, \ldots, \mathsf{dk}^k))$.

— $\mathsf{Enc}(1^\lambda, m, \mathsf{ek}_1, \ldots, \mathsf{ek}_n)$: For each $i \in [n]$, parse $\mathsf{ek}_i = (\mathsf{ek}_i^1, \ldots, \mathsf{ek}_i^k)$. Sample a random s-subset $S \subset [k]$ and let $\overline{S} = [k] \setminus S$ denote its complement. For each $j \in S$ compute $\mathsf{ct}^j \leftarrow \overline{\mathsf{Enc}}(1^\lambda, m, \mathsf{ek}_1^j, \ldots, \mathsf{ek}_n^j)$ and output $\mathsf{ct} = (S, (\mathsf{ek}_i^j)_{i \in [n], j \in \overline{S}}, (\mathsf{ct}^j)_{j \in S})$.

— $\mathsf{Dec}(\mathsf{ct}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n)$: For each $i \in [n]$, parse $\mathsf{dk}_i = (\mathsf{dk}_i^1, \ldots, \mathsf{dk}_i^k)$ and $\mathsf{ct} = (S, (\mathsf{ek}_i^j)_{i \in [n], j \in \overline{S}}, (\mathsf{ct}^j)_{j \in S})$. Initialize $\mathsf{bad} = \emptyset$ and for each $i \in [n]$ do:

  - For each $j \in \overline{S}$ if $(\mathsf{ek}_i^j, \mathsf{dk}_i^j)$ are inconsistent with $\overline{\mathsf{KeyGen}}$, insert i to bad.[a]

  If bad is non-empty, output $(0, \perp, \mathsf{bad})$. Else, for each $j \in S$, compute $m^j = \overline{\mathsf{Dec}}(\mathsf{ct}^j, \mathsf{dk}_1^j, \ldots, \mathsf{dk}_n^j) \in \{0,1\}^\ell \cup \{\perp\}$. Set m' to be the majority of $(m^j : m^j \neq \perp)_{j \in S}$ and output $(1, m', \emptyset)$.

[a] Recall that we assume by default that dk is simply the coins of the key-generation algorithm and so the above condition can be checked.

Figure 3: A distributed encryption scheme that satisfies $(n-1)$-security with identifiable abort

still preserved even given the tag information, and so it is safe to append it to ciphertext. Moreover, the abort decision of the decryption algorithm is performed solely based on the "tag". That is, the decryption algorithm is composed of two stages: detection and recovery. The detection algorithm decides whether to abort or not (and in case of abort it identifies a bad party) solely based on the tag information. If detection passes without abort, then the recovery algorithm should be able to decrypt successfully. This feature allows us to check the abort event without leaking information on the encrypted message, and can be eventually used in order to get a fair protocol at the expense of adding an extra round of interaction.

Lemma 5.2. *The construction in Figure 3 satisfies $(n-1)$-security with identifiable abort.*

Proof of Lemma 5.2. We prove privacy and identification separately.

Privacy. The privacy follows from the privacy of the underlying scheme in a straight forward way. For completeness, we sketch the proof. Assume, towards a contradiction, that the construction in Figure 3 is not private. Fix some polynomials $n = n(\lambda)$ and

$\ell = \ell(\lambda) > \log \lambda$, and let $\mathcal{A}$ be an efficient adversary corrupting an $(n-1)$-subset $\mathcal{I} \subset [n]$ of parties, that chooses a vector of corrupted keys $\{\mathsf{ek}_i\}_{i \in \mathcal{I}}$ and a pair of messages $m_0, m_1 \in \{0,1\}^\ell$, such that $\mathcal{A}$ can distinguish with non-negligible probability $\epsilon(\lambda)$ between

$$\mathsf{Enc}(1^\lambda, m_0, (\mathsf{ek}_i)_{i\in[n]}) \quad \text{and} \quad \mathsf{Enc}(1^\lambda, m_1, (\mathsf{ek}_i)_{i\in[n]}) \tag{7}$$

where for every $i \notin \mathcal{I}$ it holds that

$$\mathsf{ek}_i = (\mathsf{ek}_i^j)_{j\in[k]}, \quad \mathsf{dk}_i = (\mathsf{dk}_i^j)_{j\in[k]} \quad \text{and } (\mathsf{ek}_i^j, \mathsf{dk}_i^j) \leftarrow \overline{\mathsf{KeyGen}}(1^\lambda, 1^n, 1^\ell).$$

We use a hybrid argument to get a contradiction. Fix an arbitrary s-subset $S \subset [k]$ and let $\overline{S}$ denote its complement. For $0 \leq i \leq \lambda$, let $S[:i]$ denote the first i elements of S and let $S[i+1:]$ denote the last $s-i$ elements of λ. For every $i \in [s]$, define the ith hybrid encryption $\mathsf{Enc}_i(1^\lambda, m_1, (\mathsf{ek}_i)_{i\in[n]})$ to output $(S, (\mathsf{ek}_i^j)_{i\in[n], j\in\overline{S}}$ together with the vector of ciphertexts

$$(\overline{\mathsf{Enc}}(1^\lambda, m_1, \mathsf{ek}_1^j, \ldots, \mathsf{ek}_n^j))_{j\in S[:i]}, (\overline{\mathsf{Enc}}(1^\lambda, m_0, \mathsf{ek}_1^j, \ldots, \mathsf{ek}_n^j))_{j\in S[i+1:]}).$$

By the $n-1$ privacy of the original scheme, every pair of neighboring hybrids are indisitinguishable. Since there are polynomially many hybrids, this implies that the extreme hybrids, that correspond to the distributions in Eq. (7), are indistinsguishable, as required.

Identification. For identification, we start by describing the predicate P. For each $i \in [n]$, P parses $\mathsf{ek}_i = (\mathsf{ek}_i^j)_{j\in[k]}$ and $\mathsf{dk}_i = (\mathsf{dk}_i^j)_{j\in[k]}$. The predicate samples a random s-subset $S_\mathsf{P} \subset [k]$ and computes a set bad by running the first part of the decryption algorithm (up to the "If bad is non-empty" part) over S_P, $(\mathsf{ek}_i^j)_{i\in[n], j\in\overline{S}_\mathsf{P}}$ and $(\mathsf{dk}_i^j)_{i\in[n], j\in\overline{S}_\mathsf{P}}$. If the outcome set bad is non-empty, the predicate outputs $(0, \mathsf{bad})$ and otherwise it outputs $(1, \emptyset)$.

We prove that P satisfies the $(n-1)$-identification property. Fix a message m, a set $\mathcal{I}$ of corrupted parties, and a vector of encryption/decryption keys $(\mathsf{ek}_i, \mathsf{dk}_i)_{i\in[n]}$. Let S be the random set chosen by the encryption algorithm. First observe that Eq. (6) holds. Indeed, S and S_P are identically distributed, and when $S = S_\mathsf{P}$, the decryption "fails" (outputs $(0, \perp, \mathsf{bad})$) if and only if the predicate outputs $(0, \mathsf{bad})$.

To establish Eq. (5), we begin by noting that when decryption "fails" bad must be a (non-empty) subset of $\mathcal{I}$ since an honest party $i \notin \mathcal{I}$ prepares all its sub-keys $(\mathsf{ek}_i^j, \mathsf{dk}_i^j)_{j\in[2\lambda]}$ honestly by calling $\overline{\mathsf{KeyGen}}$, and so it never enters into bad. It remains to show that when the decryption succeeds and outputs $(1, m', \emptyset)$ the resulting message m' equals to m, except with negligible probability over the choice of S. Let N_i denote the number of sub-keys of $(\mathsf{ek}_i, \mathsf{dk}_i)$ that are inconsistent with with $\overline{\mathsf{KeyGen}}$ and let $N = \sum_{i\in[n]} N_i$ denote the total number of inconsistent sub-keys. We distinguish between two cases:

1. $N < s$. In this case, there exists at least a single index $j \in S$ such that all the jth sub-keys $(\mathsf{ek}_i^j, \mathsf{dk}_i^j)_{i\in[n]}$ are honestly generated and so, the value m^j as defined by

the decryption algorithm will be equal to m (by the correctness of the underlying decryption algorithm). Moreover, by the detection property of the underlying scheme, except for negligible probability, it holds that for every $j \in S$, the decrypted value m^j is either m or $\perp$, and therefore, except with negligible probability, the output of the decryption algorithm is m.

2. $N \geq s$. In this case, by an averaging argument, there exists $i \in \mathcal{I}$, such that $N_i \geq s/n$. We show that, in this case, except with negligible probability, the decryption (and predicate) output $(0, \perp, \mathsf{bad})$. Indeed, the probability that a random s-subset S of $[k]$ completely misses all the N_i bad sub-keys of $(\mathsf{ek}_i, \mathsf{dk}_i)$ is at most

$$\frac{\binom{k-N_i}{s-N_i}}{\binom{k}{s}} \leq O\left(\frac{2^{k-N_i}}{2^k/k}\right) \leq O(\lambda n 2^{-\lambda}) \leq 2^{-\Omega(\lambda)},$$

where in the first inequality we make use of the fact that $k = 2s$ and of the bound $\binom{k}{k/2} = \Omega(2^k/k)$, the second inequality follows by plugging-in the values $k = 2\lambda n$ and $N_i \geq s/n \geq \lambda$, and the third inequality follows by recalling that n is polynomial in λ.

The lemma follows. □

6 Elementary Reduction using Distributed Encryption

In this section, we prove our main positive result. We show how to convert a distributed encryption scheme that satisfies security with identifiable abort into a non-interactive reduction from any efficiently computable functionality f to a constant-degree functionality g that satisfies security with identifiable abort.

Theorem 6.1. *Let $f = \{f_\lambda\}$ be an n-party efficiently-computable functionality where $n = n(\lambda)$ is polynomial in the security parameter. Assuming the existence of a distributed encryption scheme, $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, with $(n-1)$ security with identifiable abort (resp., with abort), there exists an n-input functionality g such that f elementary reduces to g with $(n-1)$ security with identifiable abort (resp., with abort). Moreover, g can be computed by an NC^0-circuit with oracle gates to $\mathsf{Enc}(1^\lambda, 1^n, 1^\ell, \cdot)$, where $\ell = O(\lambda, n)$.*

The encryption algorithm of the distributed encryption scheme from Theorem 5.2 is in NC^1, and so, by Remark 1, we derive Theorem 1.3. The elementary reduction promised in Theorem 6.1 is described in Section 6.1 and its analysis appears in Section B.

6.1 Proof of Theorem 6.1: The Reduction

Notation. Let f be an n-party functionality where $n = n(\lambda)$. We assume for the sake of simplicity (and without loss of generality) that the functionality returns the same output to all parties. Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a distributed encryption scheme that achieves security with abort (or identifiable abort). We further assume, without loss of generality, that the scheme supports messages of bit-length $\ell = 2n\lambda + 2$ with decryption keys of bit-length λ.

Looking ahead, for our elementary reduction we will encrypt messages under two key vectors. For this we consider a *double encryption gadget* denoted by $(\mathsf{Doub.Enc}, \mathsf{Doub.Dec})$, defined as follows:

- $\mathsf{Doub.Enc}(1^\lambda, m, \overrightarrow{\mathsf{ek}}_1, \overrightarrow{\mathsf{ek}}_2)$: The double encryption algorithm samples a random $r \leftarrow\$ \{0,1\}^\ell$, computes two ciphertexts, $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(1^\lambda, m \oplus r, \overrightarrow{\mathsf{ek}}_1)$ and $\mathsf{ct}_2 \leftarrow \mathsf{Enc}(1^\lambda, r, \overrightarrow{\mathsf{ek}}_2)$ and outputs $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- $\mathsf{Doub.Dec}(\mathsf{ct}, \overrightarrow{\mathsf{dk}}_1, \overrightarrow{\mathsf{dk}}_2)$: The double decryption algorithm parses $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$, and sets $(b_1, m_1, \mathsf{bad}_1) = \mathsf{Dec}(\mathsf{ct}_1, \overrightarrow{\mathsf{dk}}_1)$ and $(b_2, m_2, \mathsf{bad}_2) = \mathsf{Dec}(\mathsf{ct}_2, \overrightarrow{\mathsf{ek}}_2)$. If either b_1 or b_0 equal to zero, we say that decryption "aborts" with the output $\mathsf{bad} = \mathsf{bad}_1 \cup \mathsf{bad}_2$. Otherwise, we say that decryption succeeds and output $m = m_1 \oplus m_2$.

Let C be a boolean circuit that represents f. We assume without loss of generality that the fan-in and fan-out of every gate in C is two, since any circuit can be transformed to satisfy this condition with constant multiplicative overhead in the size. Let W be the set of all wires in this circuit C. Let $W_{\mathsf{in}} \subset W$ be the set of input wires and $W_{\mathsf{out}} \subset W$ be the set of output wires. We let $\Gamma = \Gamma(\lambda)$ denote the number of gates in C and assume that the gates are indexed from 1 to Γ according to some topological order. We represent each such gate as a tuple (G, a, b, c, d) where $G : \{0,1\}^2 \to \{0,1\}$ is the gate's operator, a, b are the indices of the incoming wires and c, d are the indices of the outgoing wires.

Reduction. The reduction proceeds as follows:

- **Pre-Processing.** For each wire $w \in W$ in the circuit C and for every $(a,b) \in \{0,1\}^2$, each party i, with input x_i, samples $\left(\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b}\right) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Each party then invokes the oracle implementing functionality g on its actual input x_i and on these keys $\{\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b}\}_{w \in W, (a,b) \in \{0,1\}^2}$.
- ***g*-Oracle.** Upon invocation, oracle g parses the inputs as

  $$\left\{x_i, \left\{\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b}\right\}_{w \in W, (a,b) \in \{0,1\}^2}\right\}_{i \in [n]}$$

  and sets $x = x_1 || \ldots || x_n$. For each wire $w \in W$, it samples a random mask $\delta^w \leftarrow\$ \{0,1\}$ and for each $(a,b) \in \{0,1\}^2$, it sets $\overrightarrow{\mathsf{ek}}^w_{a,b} = (\mathsf{ek}^w_{1,a,b}, \ldots, \mathsf{ek}^w_{n,a,b})$ and $\overrightarrow{\mathsf{dk}}^w_{a,b} = (\mathsf{dk}^w_{1,a,b}, \ldots, \mathsf{dk}^w_{n,a,b})$. Also, for each $w \in W$ and each $a \in \{0,1\}$, we use $\overrightarrow{\mathsf{ek}}^w_a$ to denote $\overrightarrow{\mathsf{ek}}^w_{a,0} || \overrightarrow{\mathsf{ek}}^w_{a,1}$ and $\overrightarrow{\mathsf{dk}}^w_a$ to denote $\overrightarrow{\mathsf{dk}}^w_{a,0} || \overrightarrow{\mathsf{dk}}^w_{a,1}$. It then proceeds as follows:

1. **Gates:** For every $j \in [\Gamma]$, the reduction considers the jth gate (G, a, b, c, d) and computes four ciphertexts defined as follows. For each $(\alpha, \beta) \in \{0,1\}^2$, let $\gamma_{\alpha,\beta} = G(\alpha \oplus \delta^a, \beta \oplus \delta^b)$ and define
$$m_{\alpha,\beta} = ((\overrightarrow{\mathsf{dk}}^c_{\gamma_{\alpha,\beta}\oplus\delta^c} || \gamma_{\alpha,\beta} \oplus \delta^c) || (\overrightarrow{\mathsf{dk}}^d_{\gamma_{\alpha,\beta}\oplus\delta^d} || \gamma_{\alpha,\beta} \oplus \delta^d))$$
$$\mathsf{ct}^j_{\alpha,\beta} \leftarrow \mathsf{Doub.Enc}(1^\lambda, m_{\alpha,\beta}, \overrightarrow{\mathsf{ek}}^a_{\alpha,\beta}, \overrightarrow{\mathsf{ek}}^b_{\beta,\alpha}).$$
2. **Output:** For every gate index $j \in [\Gamma]$, and for each $(\alpha, \beta) \in \{0,1\}^2$, output all the four ciphertexts $\mathsf{ct}^j_{\alpha,\beta}$. Let y_w denote the value induced by x on wire w. For each input wire w, output the masked input $\hat{y}_w := y_w \oplus \delta^w$ along with the decryption key $\overrightarrow{\mathsf{dk}}^w_{\hat{y}_w}$ corresponding to the masked input. For each output wire $w \in W_{\mathsf{out}}$, output the mask δ^w.

— **Post-Processing.** Upon receiving $\{\mathsf{ct}^j_{\alpha,\beta}\}_{(\alpha,\beta)\in\{0,1\}^2}$ for each gate's index $j \in [\Gamma]$, $(\overrightarrow{\mathsf{dk}}^w_{\hat{y}_w}, \hat{y}_w)$ for each input wire $w \in W_{\mathsf{in}}$, and δ^w for each output wire $w \in W_{\mathsf{out}}$ from the g-oracle, each party locally computes the following.

1. Traverse the circuit according to a topological order from inputs to outputs, while maintaining for each wire w, the pair $(\overrightarrow{\mathsf{dk}}^w_{\hat{y}_w}, \hat{y}_w)$. Specifically, for each gate $j \in [\Gamma]$, with incoming wires a, b and outgoing wires c, d, call $\mathsf{Doub.Dec}(\mathsf{ct}^j_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{dk}}^w_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{dk}}^w_{\hat{y}_b,\hat{y}_a})$. If the decryption fails with output bad, we halt with output $(\perp, \mathsf{bad})$. If the decryption succeeds with output $m_{\hat{y}_a,\hat{y}_b}$ we parse $m_{\hat{y}_a,\hat{y}_b} = ((\overrightarrow{\mathsf{dk}}^c_{\hat{y}_c} || \hat{y}_c) || (\overrightarrow{\mathsf{dk}}^d_{y_d} || \hat{y}_d))$, and continue to the next gate.
2. If all the calls to the decryption algorithm succeed, then, for each output wire $w \in W_{\mathsf{out}}$, output $\hat{y}_w \oplus \delta^w$.

Remark 4 (An interactive variant with fairness)**.** *By changing g into a reactive functionality to which the parties are allowed to make 2 sequential calls, one can obtain full fairness. First, let us assume that the ciphertexts of the underlying distributed encryption scheme are augmented with tags as explained in Remark 3. Next, break g to a pair of functionalities g_1 and g_2 as follows. The input to g_1 is the same as in g but its output consists of the ciphertext tuples that are computed for each gate, together with an encrypted version of all other outputs of g. The encryption is performed by using a one-time pad where the key K is chosen using the internal randomness of the functionality. Given these values each party checks if the ciphertexts of the gates are valid using the public detection algorithm. If a party P_i, detects a problem it sends a flag-bit $b_i = 1$ to the functionality g_2, if no problem occurs the flag is set to zero. The functionality g_2 releases the key K if and only if none of the flags indicate that there is a problem. The original postprocessing can be applied when g_2 releases the key; otherwise the parties abort.*

The current analysis can be easily adopted to show that the reduction achieves full fairness. While the current description assumes that the functionalities maintain a shared *state (the key K) one can easily remove it using secret sharing and message-authentication codes. Specifically, we augment g_1 and let it send to each party i, a share K_i of the key K (computed via n-out-of-n secret sharing), a sequence of authentication keys $A_i = (A_{i,1}, \dots, A_{i,n})$ and a vector of n authentication tags $T_i = (\mathsf{MAC}_{A_{1,i}}(K_i), \dots, \mathsf{MAC}_{A_{n,i}}(K_i)))$ where MAC is a one-time secure information-theoretic MAC. The functionality g_2 receives from each party P_i a flag bit, b_i, as before, together with (K_i, A_i, T_i). If all the validity bits are OK, and all the tags pass the authentication, the functionality recovers the key K and send it to all parties. Otherwise, it sends an abort signal.*

References

[1] Ananth, P., Choudhuri, A. R., Goel, A., and Jain, A. Round-optimal secure multiparty computation with honest majority. In *CRYPTO 2018, Part II* (Aug. 2018), H. Shacham and A. Boldyreva, Eds., vol. 10992 of *LNCS*, Springer, Heidelberg, pp. 395–424.

[2] Ananth, P., Choudhuri, A. R., Goel, A., and Jain, A. Two round information-theoretic MPC with malicious security. In *EUROCRYPT 2019, Part II* (May 2019), Y. Ishai and V. Rijmen, Eds., vol. 11477 of *LNCS*, Springer, Heidelberg, pp. 532–561.

[3] Applebaum, B. *Cryptography in Constant Parallel Time.* ISC. Springer, Heidelberg, 2014.

[4] Applebaum, B. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography*. 2017, pp. 1–44.

[5] Applebaum, B., Brakerski, Z., Garg, S., Ishai, Y., and Srinivasan, A. Separating two-round secure computation from oblivious transfer. In *ITCS 2020* (Jan. 2020), T. Vidick, Ed., vol. 151, LIPIcs, pp. 71:1–71:18.

[6] Applebaum, B., Brakerski, Z., and Tsabary, R. Perfect secure computation in two rounds. In *TCC 2018, Part I* (Nov. 2018), A. Beimel and S. Dziembowski, Eds., vol. 11239 of *LNCS*, Springer, Heidelberg, pp. 152–174.

[7] Applebaum, B., Brakerski, Z., and Tsabary, R. Degree 2 is complete for the round-complexity of malicious MPC. In *EUROCRYPT 2019, Part II* (May 2019), Y. Ishai and V. Rijmen, Eds., vol. 11477 of *LNCS*, Springer, Heidelberg, pp. 504–531.

[8] Applebaum, B., Ishai, Y., and Kushilevitz, E. Cryptography in NC^0. In *45th FOCS* (Oct. 2004), IEEE Computer Society Press, pp. 166–175.

[9] Applebaum, B., Ishai, Y., and Kushilevitz, E. Computationally private randomizing polynomials and their applications. In *20th Annual IEEE Conference on*

Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA (2005), pp. 260–274.

[10] Applebaum, B., Kachlon, E., and Patra, A. The round complexity of perfect MPC with active security and optimal resiliency. In *61st FOCS* (Nov. 2020), IEEE Computer Society Press, pp. 1277–1284.

[11] Aumann, Y., and Lindell, Y. Security against covert adversaries: Efficient protocols for realistic adversaries. In *TCC 2007* (Feb. 2007), S. P. Vadhan, Ed., vol. 4392 of *LNCS*, Springer, Heidelberg, pp. 137–156.

[12] Barak, B., and Mahmoody-Ghidary, M. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In *CRYPTO 2009* (Aug. 2009), S. Halevi, Ed., vol. 5677 of *LNCS*, Springer, Heidelberg, pp. 374–390.

[13] Baum, C., Orsini, E., Scholl, P., and Soria-Vazquez, E. (efficient constant-round MPC with identifiable abort and public verifiability. In *EUROCRYPT 2020, Part II* (May 2020), A. Canteaut and Y. Ishai, Eds., vol. 12106 of *LNCS*, Springer, Heidelberg, pp. 562–592.

[14] Beaver, D., Micali, S., and Rogaway, P. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC* (May 1990), ACM Press, pp. 503–513.

[15] Ben-Or, M., Goldwasser, S., and Wigderson, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC* (May 1988), ACM Press, pp. 1–10.

[16] Benhamouda, F., and Lin, H. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *EUROCRYPT 2018, Part II* (Apr. / May 2018), J. B. Nielsen and V. Rijmen, Eds., vol. 10821 of *LNCS*, Springer, Heidelberg, pp. 500–532.

[17] Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS* (Oct. 2001), IEEE Computer Society Press, pp. 136–145.

[18] Chaum, D., Crépeau, C., and Damgård, I. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC* (May 1988), ACM Press, pp. 11–19.

[19] Cleve, R. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC* (May 1986), ACM Press, pp. 364–369.

[20] Damgård, I., and Ishai, Y. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO 2005* (Aug. 2005), V. Shoup, Ed., vol. 3621 of *LNCS*, Springer, Heidelberg, pp. 378–394.

[21] Even, S., Goldreich, O., and Lempel, A. A randomized protocol for signing contracts. In *CRYPTO'82* (1982), D. Chaum, R. L. Rivest, and A. T. Sherman, Eds., Plenum Press, New York, USA, pp. 205–210.

[22] Garg, S., Ishai, Y., and Srinivasan, A. Two-round MPC: Information-theoretic and black-box. In *TCC 2018, Part I* (Nov. 2018), A. Beimel and S. Dziembowski, Eds., vol. 11239 of *LNCS*, Springer, Heidelberg, pp. 123–151.

[23] Garg, S., and Srinivasan, A. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT 2018, Part II* (Apr. / May 2018), J. B. Nielsen and V. Rijmen, Eds., vol. 10821 of *LNCS*, Springer, Heidelberg, pp. 468–499.

[24] Goldreich, O. *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge University Press, Cambridge, UK, 2004.

[25] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC* (May 1987), A. Aho, Ed., ACM Press, pp. 218–229.

[26] Goldwasser, S., and Lindell, Y. Secure computation without agreement. Cryptology ePrint Archive, Report 2002/040, 2002. http://eprint.iacr.org/2002/040.

[27] Gordon, S. D., Hazay, C., Katz, J., and Lindell, Y. Complete fairness in secure two-party computation. *J. ACM 58*, 6 (2011), 24:1–24:37.

[28] Gordon, S. D., Ishai, Y., Moran, T., Ostrovsky, R., and Sahai, A. On complete primitives for fairness. In *TCC 2010* (Feb. 2010), D. Micciancio, Ed., vol. 5978 of *LNCS*, Springer, Heidelberg, pp. 91–108.

[29] Haitner, I., and Karidi-Heller, Y. A tight lower bound on adaptively secure full-information coin flip. In *61st FOCS* (Nov. 2020), IEEE Computer Society Press, pp. 1268–1276.

[30] Haitner, I., Omri, E., and Zarosim, H. Limits on the usefulness of random oracles. In *TCC 2013* (Mar. 2013), A. Sahai, Ed., vol. 7785 of *LNCS*, Springer, Heidelberg, pp. 437–456.

[31] Hazay, C., Ishai, Y., and Venkitasubramaniam, M. Actively secure garbled circuits with constant communication overhead in the plain model. In *TCC 2017, Part II* (Nov. 2017), Y. Kalai and L. Reyzin, Eds., vol. 10678 of *LNCS*, Springer, Heidelberg, pp. 3–39.

[32] Hazay, C., Scholl, P., and Soria-Vazquez, E. Low cost constant round MPC combining BMR and oblivious transfer. In *ASIACRYPT 2017, Part I* (Dec. 2017), T. Takagi and T. Peyrin, Eds., vol. 10624 of *LNCS*, Springer, Heidelberg, pp. 598–628.

[33] Impagliazzo, R., and Rudich, S. Limits on the provable consequences of one-way permutations. In *CRYPTO'88* (Aug. 1990), S. Goldwasser, Ed., vol. 403 of *LNCS*, Springer, Heidelberg, pp. 8–26.

[34] Ishai, Y. Randomization techniques for secure computation. In *Secure Multi-Party Computation*, M. Prabhakaran and A. Sahai, Eds., vol. 10 of *Cryptology and Information Security Series*. IOS Press, 2013, pp. 222–248.

[35] Ishai, Y., and Kushilevitz, E. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS* (Nov. 2000), IEEE Computer Society Press, pp. 294–304.

[36] Ishai, Y., and Kushilevitz, E. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP 2002* (July 2002), P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, Eds., vol. 2380 of *LNCS*, Springer, Heidelberg, pp. 244–256.

[37] Ishai, Y., and Kushilevitz, E. On the hardness of information-theoretic multiparty computation. In *EUROCRYPT 2004* (May 2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *LNCS*, Springer, Heidelberg, pp. 439–455.

[38] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., and Sahai, A. Efficient non-interactive secure computation. In *EUROCRYPT 2011* (May 2011), K. G. Paterson, Ed., vol. 6632 of *LNCS*, Springer, Heidelberg, pp. 406–425.

[39] Ishai, Y., Ostrovsky, R., and Zikas, V. Secure multi-party computation with identifiable abort. In *CRYPTO 2014, Part II* (Aug. 2014), J. A. Garay and R. Gennaro, Eds., vol. 8617 of *LNCS*, Springer, Heidelberg, pp. 369–386.

[40] Ishai, Y., Prabhakaran, M., and Sahai, A. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO 2008* (Aug. 2008), D. Wagner, Ed., vol. 5157 of *LNCS*, Springer, Heidelberg, pp. 572–591.

[41] Lindell, Y., Pinkas, B., Smart, N. P., and Yanai, A. Efficient constant round multi-party computation combining BMR and SPDZ. In *CRYPTO 2015, Part II* (Aug. 2015), R. Gennaro and M. J. B. Robshaw, Eds., vol. 9216 of *LNCS*, Springer, Heidelberg, pp. 319–338.

[42] Mahmoody, M., Maji, H. K., and Prabhakaran, M. Limits of random oracles in secure computation. In *ITCS 2014* (Jan. 2014), M. Naor, Ed., ACM, pp. 23–34.

[43] Moran, T., Naor, M., and Segev, G. An optimally fair coin toss. *J. Cryptol. 29*, 3 (2016), 491–513.

[44] Rabin, M. O. How to exchange secrets with oblivious transfer. Tech. Rep. TR-81, Aiken Computation Lab, Harvard University, 1981.

[45] Vaikuntanathan, V. Some open problems in information-theoretic cryptography. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India* (2017), S. V. Lokam and R. Ramanujam, Eds., vol. 93 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 5:1–5:7.

[46] Wang, X., Ranellucci, S., and Katz, J. Authenticated garbling and efficient maliciously secure two-party computation. In *ACM CCS 2017* (Oct. / Nov. 2017), B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds., ACM Press, pp. 21–37.

[47] Yao, A. C.-C. How to generate and exchange secrets (extended abstract). In *27th FOCS* (Oct. 1986), IEEE Computer Society Press, pp. 162–167.

A Proof of Observation 1

Given a passively secure non-interactive reduction $\Sigma = (\mathsf{pre}_1, \ldots, \mathsf{pre}_n, \mathsf{post}_1, \ldots, \mathsf{post}_n)$ from f to a constant degree functionality g we define a new actively-secure non-interactive reduction $\Sigma' = (\mathsf{pre}'_1, \ldots, \mathsf{pre}'_n, \mathsf{post}'_1, \ldots, \mathsf{post}'_n)$ from f to a related NC^1 functionality g', and then reduce g' to a constant-degree functionality via Remark 1. The reduction Σ' and the functionality g' are defined as follows.

For every i, the preprocessing $\mathsf{pre}'_i(x_i, r_i)$ outputs the original output $y_i = \mathsf{pre}_i(x_i, r_i)$ together with all the intermediate values that are induced in the circuit that computes $\mathsf{pre}_i(x_i, r_i)$. Denote the concatenation of the latter values by π_i. The functionality g' first checks, for each i, whether the values that are given in π_i are self-consistent and that the output of the computation is indeed y_i. (This condition can be checked in NC^1 or even in AC^0.) If the condition passes set y'_i to y_i and if the condition fails, set y_i to $\mathsf{pre}_i(\mathbf{0}, \mathbf{0})$ (the latter value is hardwired into the reduction). Finally, g' send the outputs $(v_1, \ldots, v_n) = g(y'_1, \ldots, y'_n)$. The postprocessing part is identical to the postprocessing computation of the original reduction.

It is not hard to see that the new reduction realizes f with full active security and guaranteed output delivery. Indeed, any "cheating" during the preprocessing phase, translates into an invocation of the protocol with fixed zero inputs.

B Proof of Theorem 6.1: Security Analysis

For an arbitrary functionality f, we define $[f]_{\mathsf{id-abort}}$ to be the corresponding functionality with *identifiable abort*, which behaves as f with the following modifications. Each party (including the simulator) can send inputs to the functionality. In addition, the simulator can either send a $\mathsf{continue}$ command and in this case the outputs are delivered to all parties, or send a special $(\mathsf{abort}, \mathsf{bad})$ command. In the latter case, the functionality verifies that bad is a non-empty subset of the corrupted parties, and sets the output of all (honest) parties to $(\perp, \mathsf{bad})$, and delivers the output to the adversary. If the verification fails, the functionality treats this command as a $\mathsf{continue}$ command. The model of *security with abort* is defined similarly, except that the functionality $[f]_{\mathsf{abort}}$ drops the bad input (i.e., does not apply any verification and does not deliver it to the honest parties).[11]

Simulator. For an efficient adversary $\mathcal{A}$ corrupting a set of parties $\mathcal{I} \subset [n]$, the simulator, on input $x_{\mathcal{I}} = (x_i)_{i \in \mathcal{I}}$ and advice a, proceeds as follows.

1. **Adversarial preprocessing:** The simulator invokes $\mathcal{A}(x_{\mathcal{I}}, a)$ and gets the values $(x'_i, (\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b})_{w \in W, (a,b) \in \{0,1\}^2})_{i \in \mathcal{I}}$ that $\mathcal{A}$ sends to the g-oracle.

2. **Honest preprocessing and local randomness:** For each wire $w \in W$, the simulator Sim samples a random value $\hat{y}_w \in \{0,1\}$. Let $\overline{\mathcal{I}} = [n] \setminus \mathcal{I}$ denote the set of honest parties. For each honest party $i \in \overline{\mathcal{I}}$ and for each $(a,b) \in \{0,1\}^2$, the simulator samples keys $(\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

3. **Preparing the ciphertexts:** The simulator traverses the gates as follows. For the jth gate (G, a, b, c, d), the simulator does the following:

   (a) Compute $m_{\hat{y}_a, \hat{y}_b} = ((\overrightarrow{\mathsf{dk}}^c_{\hat{y}_c} || \hat{y}_c) || (\overrightarrow{\mathsf{dk}}^d_{\hat{y}_d} || \hat{y}_d))$ and set the ciphertexts

   $$\begin{aligned}
   \mathsf{ct}^j_{\hat{y}_a, \hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, m_{\hat{y}_a, \hat{y}_b}, \overrightarrow{\mathsf{ek}}^a_{\hat{y}_a, \hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{\hat{y}_b, \hat{y}_a}) \\
   \mathsf{ct}^j_{1-\hat{y}_a, \hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, 0^\ell, \overrightarrow{\mathsf{ek}}^a_{1-\hat{y}_a, \hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{\hat{y}_b, 1-\hat{y}_a}) \\
   \mathsf{ct}^j_{1-\hat{y}_a, 1-\hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, 0^\ell, \overrightarrow{\mathsf{ek}}^a_{1-\hat{y}_a, 1-\hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{1-\hat{y}_b, 1-\hat{y}_a}) \\
   \mathsf{ct}^j_{\hat{y}_a, 1-\hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, 0^\ell, \overrightarrow{\mathsf{ek}}^a_{\hat{y}_a, 1-\hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{1-\hat{y}_b, \hat{y}_a}).
   \end{aligned}$$

   (b) Compute

   $$(\mathsf{P}_a, \mathsf{bad}'_a) = \mathsf{P}((\mathsf{ek}^a_{i,\hat{y}_a,\hat{y}_b}, \mathsf{dk}^a_{i,\hat{y}_a,\hat{y}_b})_{i \in [n]}) \quad \text{and} \quad (\mathsf{P}_b, \mathsf{bad}'_b) = \mathsf{P}((\mathsf{ek}^b_{i,\hat{y}_b,\hat{y}_a}, \mathsf{dk}^b_{i,\hat{y}_b,\hat{y}_a})_{i \in [n]}).$$

   If either $\mathsf{P}_a = 0$ or $\mathsf{P}_b = 0$ (i.e., failure) set $F_j = 0$ (“failure”) and $\mathsf{bad}_j = \mathsf{bad}'_a \cup \mathsf{bad}'_b$. Else, set $F_j = 1$.

[11] The more standard version of security with abort [39] allows the simulator to take its decision (abort/continue) after seeing the output of the computation. In contrast, in our notion the simulator has to take its decision (abort/continue) *before* seeing the output, and so it achieves a stronger form of security. This is possible since we are assuming an ideal g-hybrid model. Of course, if the oracle g is instantiated with an actual protocol that achieves the standard (weaker) form of security-with-abort then the resulting protocol for f will inherit the same level of security.

4. **Calling the ideal functionality:** If a failure has been recorded, the simulator takes j to be the smallest index for which $F_j = 0$, and queries the ideal functionality on inputs $(x'_i)_{i\in\mathcal{I}}$ and $(\mathsf{abort}, \mathsf{bad}_j)$ to instruct the ideal functionality to send $(\bot, \mathsf{bad}_j)$ as output to all the honest parties. Else, (if $F_j = 1$ for all $j \in [\Gamma]$), the simulator queries the ideal functionality on inputs $(x'_i)_{i\in\mathcal{I}}$ and $\mathsf{continue}$, to instruct the ideal functionality to send the correct output to all the honest parties.

5. **Output for the Adversary:** Upon receiving output z from the trusted functionality, the simulator retrieves, for every output wire w, the value y_w that is induced by the output z on the wth output wire, and sets

$$\delta^w = \hat{y}_w \oplus y_w.$$

   The simulator sends to the adversary the values

$$(\mathsf{ct}^j_{\alpha,\beta})_{j\in[\Gamma],\alpha,\beta\in\{0,1\}}, \quad (\hat{y}_w, \overrightarrow{\mathsf{dk}}^w_{\hat{y}_w})_{w\in W_{\mathsf{in}}}, \quad (\delta^w)_{w\in W_{\mathsf{out}}},$$

   and terminates with the output of the adversary.

Analysis. Fix an an efficient (non-uniform) adversary $\mathcal{A}$, and let Sim denote the corresponding simulator, as defined above. Let $\mathcal{D}$ be an efficient distinguisher and let $\Delta_{\mathcal{D}}(X; Y)$ denote the distinguishing advantage of $\mathcal{D}$ between the distributions X and Y. To complete the proof of security, we need to show that, for every input $x = (x_i)_{i\in[n]}$ and every advice a,

$$\Delta_{\mathcal{D}}(\mathsf{Ideal}_{f,\mathcal{I},\mathsf{Sim}(a)}(x) \ ; \ \mathsf{Real}_{\pi,\mathcal{I},\mathcal{A}(a)}(x))$$

is negligible in λ, where the former random variable consists of the output of $\mathsf{Sim}(a, (x_i)_{i\in\mathcal{I}})$ concatenated with the output of the honest parties as computed in an ideal execution over the input x (at the presence of an $[f]_{\mathsf{id-abort}}$-oracle or $[f]_{\mathsf{abort}}$-oracle), and the latter ensemble consists of the output of $\mathcal{A}(a, (x_i)_{i\in\mathcal{I}})$ concatenated to the output of honest parties in a real execution over the input x.

Fix an input x, an advice a, and a random tape of the adversary, and therefore also fix the preprocessing values, $(x'_i, (\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b})_{w\in W,(a,b)\in\{0,1\}^2})_{i\in\mathcal{I}}$, that are submitted by the adversary to the g-oracle. Denote by y_w the value that the inputs induce on the wth wire of C. Let us condition on the event that, for every wire w, the simulated values $\hat{y}_w$ and random mask δ^w selected by g in an ideal execution satisfy the equation

$$\delta^w = \hat{y}_w \oplus y_w.$$

Furthermore, let us condition on the event that for every honest party $i \in \overline{\mathcal{I}}$, and every pair of wires (a, b) that enter the same gate, the "on-path" keys $(\mathsf{ek}^w_{i,\hat{y}_a,\hat{y}_b}, \mathsf{dk}^w_{i,\hat{y}_a,\hat{y}_b})$ in the simulation and in the real world are equal. (Recall that in both cases these keys are selected by calling the key-generation algorithm.) From now on, we assume that all these values are fixed, and show that the $\mathcal{D}$ cannot distinguish between the conditional distribution $\mathsf{Ideal}'_{f,\mathcal{I},\mathsf{Sim}}(x)$ from the conditional distribution $\mathsf{Real}'_{\pi,\mathcal{I},\mathcal{A}}(x)$ with more than negligible probability. To this end, we define a sequence of hybrids as follows. Let Γ be the total number of gates in C, and assume that the gates are indexed from 1 to Γ under the same topological order that is being used in the protocol and in the simulation. Our hybrids H_k are indexed by $0 \leq k \leq \Gamma$ and are defined as follows. (We focus on the case of identifiable abort, the proof for the case of security with abort is almost identical.)

The hybrid H_k. For each honest party $i \in \overline{\mathcal{I}}$ and for each $(a, b) \neq (\hat{y}_a, \hat{y}_b)$, sample the "off-path" keys $(\mathsf{ek}^w_{i,a,b}, \mathsf{dk}^w_{i,a,b}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. For $j = 1$ to Γ do the following:

— Let (G, a, b, c, d) denote the jth gate. Compute $m_{\hat{y}_a,\hat{y}_b} = ((\overrightarrow{\mathsf{dk}}^c_{\hat{y}_c} || \hat{y}_c) || (\overrightarrow{\mathsf{dk}}^d_{\hat{y}_d} || \hat{y}_d))$ and set the ciphertext

$$\mathsf{ct}^j_{\hat{y}_a,\hat{y}_b} \leftarrow \mathsf{Doub.Enc}(1^\lambda, m_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{ek}}^a_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{\hat{y}_b,\hat{y}_a}).$$

— If $j \leq k$, for every $(\alpha, \beta) \neq (\hat{y}_a, \hat{y}_b)$ let

$$\gamma_{\alpha,\beta} = G(\alpha \oplus \delta^a, \beta \oplus \delta^b), \quad m_{\alpha,\beta} = ((\overrightarrow{\mathsf{dk}}^c_{\gamma_{\alpha,\beta}\oplus\delta^c} || \gamma_{\alpha,\beta} \oplus \delta^c) || (\overrightarrow{\mathsf{dk}}^d_{\gamma_{\alpha,\beta}\oplus\delta^d} || \gamma_{\alpha,\beta} \oplus \delta^d)),$$

and compute

$$\mathsf{ct}^j_{\alpha,\beta} \leftarrow \mathsf{Doub.Enc}(1^\lambda, m_{\alpha,\beta}, \overrightarrow{\mathsf{ek}}^a_{\alpha,\beta}, \overrightarrow{\mathsf{ek}}^b_{\beta,\alpha}). \tag{8}$$

In addition, call $\mathsf{Doub.Dec}(\mathsf{ct}^j_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{dk}}^w_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{dk}}^w_{\hat{y}_b,\hat{y}_a})$. If the decryption fails with output bad, set $F_j = 0$ and let $\mathsf{bad}_j = \mathsf{bad}$. Otherwise, set $F_j = 1$.

— If $j > k$, define:

$$\begin{aligned} \mathsf{ct}^j_{1-\hat{y}_a,\hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, 0^\ell, \overrightarrow{\mathsf{ek}}^a_{1-\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{\hat{y}_b,1-\hat{y}_a}) \\ \mathsf{ct}^j_{1-\hat{y}_a,1-\hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, 0^\ell, \overrightarrow{\mathsf{ek}}^a_{1-\hat{y}_a,1-\hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{1-\hat{y}_b,1-\hat{y}_a}) \\ \mathsf{ct}^j_{\hat{y}_a,1-\hat{y}_b} &\leftarrow \mathsf{Doub.Enc}(1^\lambda, 0^\ell, \overrightarrow{\mathsf{ek}}^a_{\hat{y}_a,1-\hat{y}_b}, \overrightarrow{\mathsf{ek}}^b_{1-\hat{y}_b,\hat{y}_a}). \end{aligned} \tag{9}$$

Compute $(\mathsf{P}_a, \mathsf{bad}'_a) = \mathsf{P}((\mathsf{ek}^a_{i,\hat{y}_a,\hat{y}_b}, \mathsf{dk}^a_{i,\hat{y}_a,\hat{y}_b})_{i\in[n]})$ and $(\mathsf{P}_b, \mathsf{bad}'_b) = \mathsf{P}((\mathsf{ek}^b_{i,\hat{y}_b,\hat{y}_a}, \mathsf{dk}^b_{i,\hat{y}_b,\hat{y}_a})_{i\in[n]})$. If $\mathsf{P}_a = 0$ or $\mathsf{P}_b = 0$ (i.e., failure) set $F_j = 0$ and $\mathsf{bad}_j = \mathsf{bad}'_a \cup \mathsf{bad}'_b$.

— If a failure has been recorded, the output of the honest parties is defined to be $(\perp, \mathsf{bad}_j)$ where j is the smallest index for which $F_j = 0$. Otherwise, the output of the honest parties is defined to be $(y_w)_{w\in W_{\mathsf{out}}}$. The output of the adversary is obtained by sending the values

$$(\mathsf{ct}^j_{\alpha,\beta})_{j\in[\Gamma],\alpha,\beta\in\{0,1\}}, \quad (\hat{y}_w, \overrightarrow{\mathsf{dk}}^w_{\hat{y}_w})_{w\in W_{\mathsf{in}}}, \quad (\delta^w)_{w\in W_{\mathsf{out}}},$$

to $\mathcal{A}$ and outputting its output.

We begin by noting that the extreme hybrids correspond to the ideal/real experiments. Let $\epsilon = \epsilon(\lambda)$ be a negligible function that upper-bounds the failure probabilities of the identification property of the distributed encryption. (See Definition 5.4.)

Claim 6. *The statistical distance between the hybrid H_0 and $\mathsf{Ideal}'_{f,\mathcal{I},\mathsf{Sim}}(x)$ is at most $O(\epsilon)$ and the statistical distance between the hybrid H_Γ and $\mathsf{Real}'_{\pi,\mathcal{I},\mathcal{A}}(x)$ is at most $O(\Gamma \cdot \epsilon)$.*

Consequently, $\mathcal{D}$ cannot distinguish H_0 from $\mathsf{Ideal}'_{f,\mathcal{I},\mathsf{Sim}}(x)$ (resp., H_Γ from $\mathsf{Real}'_{\pi,\mathcal{I},\mathcal{A}}(x)$) with more than negligible advantage.

Proof. The hybrid H_0 is computed identically to the $\mathsf{Ideal}'_{f,\mathcal{I},\mathsf{Sim}}(x)$ except for the way that the output to the honest parties is computed. Let j denote the minimal index for which $F_j = 0$. (If no such j exist then in both cases the output of the honest parties is $(y_w)_{w \in W_{\mathsf{out}}}$.) By the "security with Identifiable Abort" property of the underlying encryption (Definition 5.4), it holds that, except with probability $O(\epsilon)$, the set bad_j is a non-empty subset of $\mathcal{I}$. Therefore, except with probability $O(\epsilon)$, when the simulator in the ideal execution sends an abort command $(\mathsf{abort}, \mathsf{bad}_j)$ to $[f]_{\mathsf{id-abort}}$, this command will be accepted, and the output of the honest party will be $(\perp, \mathsf{bad}_j)$ just like in H_0.

We move on to analyze the hybrid H_Γ. Here, too, the hybrid H_Γ is computed identically to the computation in $\mathsf{Real}'_{\pi,\mathcal{I},\mathcal{A}}(x)$ except for the way that the output to the honest parties is computed. Let j denote the minimal index for which $F_j = 0$. If no such index exist, set $j = \Gamma + 1$. Consider the post-processing phase of the real execution and let E denote the event that for each wire w that is visited before the jth gate is visited it holds that the value $(\overrightarrow{\mathsf{dk}}^{w}_{\hat{y}_w}, \hat{y}_w)$ computed by the honest parties equals to the corresponding "on-path" keys/masked-values (that were previously fixed). We claim that this event E happens with all but probability $O(\Gamma \cdot \epsilon)$ over the randomness of the encryption. Indeed, this follows by induction (on the depth of the wire) and by the "security with Identifiable Abort" property of the underlying encryption (Definition 5.4). Finally, observe that, conditioned on the event E, the output of the honest parties in both experiments is identical. That is, if $j = \Gamma + 1$, the output is $(y_w)_{w \in W_{\mathsf{out}}}$ and, otherwise, the output is $(\perp, \mathsf{bad}_j)$ where bad_j is the set outputted by $\mathsf{Doub.Dec}(\mathsf{ct}^{j}_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{dk}}^{w}_{\hat{y}_a,\hat{y}_b}, \overrightarrow{\mathsf{dk}}^{w}_{\hat{y}_b,\hat{y}_a})$. □

We continue by showing that any pair of neighboring hybrids, H_{k-1} and H_k are computationally indistinguishable. Let $\delta(\lambda)$ be a negligible function that upper-bounds on the distinguishing advantage of an efficient adversary that tries to break the privacy of the distributed encryption scheme. (It suffices to consider non-uniform adversaries whose circuit-size is upper-bounded by the size of $\mathcal{A}$ plus the size of $\mathcal{D}$.)

Claim 7. *For every $k \in [\Gamma]$, it holds that $\Delta_{\mathcal{D}}(H_{k-1}; H_k) \leq O(\delta + \epsilon)$.*

Sketch. The first difference between these hybrids is in the way the off-path ciphertexts of the kth gate,

$$\mathsf{ct}^{k}_{\alpha,\beta} \quad \text{where } (\alpha, \beta) \neq (\hat{y}_a, \hat{y}_b),$$

are computed. A standard argument shows that, by the privacy of the distributed encryption, the ciphertext $\mathsf{ct}^k_{\alpha,\beta}$ computed for the kth gate in H_{k-1} per Eq. (9), is δ-indistinguishable from the ciphertext $\mathsf{ct}^k_{\alpha,\beta}$ computed for the kth gate in H_k per Eq. (8). Moreover, for this it suffices to assume that, for each $i \in \overline{\mathcal{I}}$, the keys $(\mathsf{ek}^a_{i,\alpha,\beta}, \mathsf{dk}^w_{i,\alpha,\beta})$ are sampled independently according to $\mathsf{KeyGen}(1^\lambda)$. (Therefore, one can treat each ciphertext independently.)

The second difference between the hybrids H_{k-1} and H_k is in the way the values (F_k, bad_k) are computed. By the security with identifiable abort property of the distributed encryption scheme, the values (F_k, bad_k) as computed in H_{k-1} (based on the decryption algorithm) and the values (F_k, bad_k) as computed in H_k based on the predicate algorithm) are ϵ-indistinguishable. (Here the probability is taken only over the randomness of the encryption algorithm and over the randomness of the predicate.) □

Overall, $\mathcal{D}$ cannot distinguish between $\mathsf{Ideal}'_{f,\mathcal{I},\mathsf{Sim}}(x)$ and $\mathsf{Real}'_{\pi,\mathcal{I},\mathcal{A}}(x)$ with more than negligible advantage of $O(\Gamma(\epsilon + \delta))$. This completes the proof of Theorem 6.1. □