Adaptively Secure MPC with Sublinear Communication Complexity[*]

Ran Cohen[†] abhi shelat[‡] Daniel Wichs[§]

February 14, 2023

Abstract

A central challenge in the study of MPC is to balance between security guarantees, hardness assumptions, and resources required for the protocol. In this work, we study the cost of tolerating adaptive corruptions in MPC protocols under various corruption thresholds.

In the strongest setting, we consider adaptive corruptions of an arbitrary number of parties (potentially all) and achieve the following results:

- A two-round secure function evaluation (SFE) protocol in the CRS model, assuming LWE and indistinguishability obfuscation (iO). The communication, the CRS size, and the online-computation are sublinear in the *size* of the function. The iO assumption can be replaced by secure erasures. Previous results required either the communication or the CRS size to be polynomial in the function size.
- Under the same assumptions, we construct a "Bob-optimized" 2PC (where Alice talks first, Bob second, and Alice learns the output). That is, the communication complexity and total computation of Bob are sublinear in the function size and in Alice's input size. We prove impossibility of "Alice-optimized" protocols.
- Assuming LWE, we bootstrap adaptively secure NIZK arguments to achieve proof size sublinear in the circuit size of the NP-relation.

On a technical level, our results are based on *laconic function evaluation (LFE)* (Quach, Wee, and Wichs, FOCS'18) and shed light on an interesting duality between LFE and FHE.

Next, we analyze adaptive corruptions of all-but-one of the parties and show a two-round SFE protocol in the threshold PKI model (where keys of a threshold FHE scheme are pre-shared among the parties) with communication complexity sublinear in the circuit size, assuming LWE and NIZK. Finally, we consider the honest-majority setting, and show a two-round SFE protocol with guaranteed output delivery under the same constraints.

Our results highlight that the asymptotic cost of adaptive security can be reduced to be comparable to, and in many settings almost match, that of static security, with only a little sacrifice to the concrete round complexity and asymptotic communication complexity.

[*]A preliminary version of this work appeared at *CRYPTO 2019*.

[†]Reichman University. E-mail: `cohenran@runi.ac.il`. Some of this work was done while the author was working at Boston University and Northeastern University, and supported by the Northeastern University Cybersecurity and Privacy Institute Post-doctoral fellowship, NSF grant TWC-1664445, NSF grant 1422965, and by the NSF MACS project.

[‡]Northeastern University. E-mail: `abhi@neu.edu`. Research supported by NSF grant TWC-1664445 and a Google Faculty fellowship.

[§]Northeastern University and NTT Research. E-mail: `wichs@ccs.neu.edu`. Research supported by NSF grants CNS-1314722, CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

Contents

1 Introduction

After establishing feasibility in the 1980s [93, 58, 11, 31, 87], the rich literature of multiparty computation (MPC) has focused on several performance aspects of the problem. These aspects include: (a) studying the resources required in terms of communication rounds, total amount of communication, and total amount of computation, (b) minimizing the required complexity assumptions under the various notions, and most importantly, (c) enhancing the notion of security, starting from the simplest notion of static corruptions with semi-honest adversaries in a stand-alone model, to sequential, and concurrent composition, to adaptive corruptions of parties by a malicious adversary.

Recent results have considered a few of these questions simultaneously. Despite several decades of progress, many basic questions about feasibility and asymptotic optimality of MPC protocols remain. The focus of this paper is to study *the price of adaptive security* in light of recent round-optimal and low-communication protocols for the static-security setting.

Recall that *adaptive security* [10, 23] for an MPC protocol models the realistic threat in which the adversary can corrupt a party during the execution of a protocol—in particular, after seeing some of the transcript of a protocol. In contrast, with *static corruptions*, the adversary must choose which parties to corrupt *before* the protocol begins. In this simpler static case, the security argument relies on the fact that the inputs of the corrupted parties are known, and thus the simulator can "work around" these parties to generate a reasonable, and consistent transcript for the remaining parties. Indeed, adaptive security is known to be strictly stronger than static security [23, 25].

While the idea of allowing an adversary to corrupt parties at anytime during protocol executions seems natural, its technical formulation is captured by obliging the simulator in the security definition to support some specific tasks. In particular, the technical difficulty in achieving adaptive security is that the simulator must produce a transcript for the execution *before* knowing which parties are corrupted. In an extreme case, the protocol can already be completed, and the adversary can then *begin* to corrupt all of the parties, one by one.

Two main models are considered for adaptive corruptions. In the first and simpler one, it is assumed that parties can *securely erase* certain parts (and even all) of their random tapes.[1] In this setting, when simulating a party who gets corrupted, the simulator may not be required to provide random coins explaining all the messages previously sent by that party. In the second, *erasures-free* model, there are no assumptions about the ability to erase local information. When a party is corrupted in this adaptive-security notion, the adversary can learn all of that party's inputs and internal random coins. In this case, a secure protocol requires a simulator that, after producing the transcript, can "explain" the transcript by generating the coins and inputs for a given party after they are corrupted. In particular, the simulator only learns the input of that party after the corruption (e.g., after the entire execution), and then must "explain" the transcript it produced beforehand in a way that is consistent with the given input.

As a result of these difficulties, most of the literature shows that achieving adaptive security is notoriously harder than achieving static security; in some cases, there are outright impossibility results such as the case of fully homomorphic encryption [76], public-key encryption which cannot exist for arbitrary messages [82], constant-round MPC in the plain model (under black-box simulation) [51], MPC protocols with non-expander communication graphs [20], and composable

[1]We note that in certain cases it is reasonable to erase the random coins, e.g., when encrypting a message it is normally fine not to store the encryption randomness; however, in some cases one cannot erase all of its random tape, e.g., when sending a public encryption key it is normally essential to store the decryption key. We refer the reader to [23, 21] for further discussion on secure erasures.

broadcast protocols without an honest majority [70, 38]. All of these lower bounds, with the exception of [51], hold also for the weaker adaptive setting with secure erasures.

1.1 Full Adaptivity: Adaptive Corruptions of All the Parties

We start by considering the strongest adversary that can adaptively corrupt, and arbitrarily control, any subset of the participating parties. We will focus on the resources required for securely evaluating a function, balancing between the number of rounds, the communication complexity, and the online-computational complexity (the work performed between the first and last messages).

The feasibility of *adaptively secure* MPC was established in the seminal CLOS protocol [24] in a resoundingly strong manner in the UC framework [22]. This paper established the notion of fully adaptive security as described above, in the stronger, erasures-free setting, when the adversary can corrupt *all* protocol parties after execution. They then achieved this notion with a brilliant, yet complicated protocol that worked in the *uniform random string* (URS) model.[2] However, that protocol's round complexity depended on the circuit *depth*, and its communication was polynomially larger than the *size* of the circuit being computed. Roughly 15 years later, Canetti et al. [30] constructed a constant-round protocol under standard assumptions, and recently Benhamouda et al. [13] constructed a two-round protocol assuming two-round adaptively secure oblivious transfer (OT). But again, both of these recent results require communication that is larger than the circuit size, and thus come at a larger cost than recent protocols for static corruptions that require two rounds and sublinear communication in the circuit size [80].

Another line of recent work overcomes the communication bottleneck, but at the cost of stronger assumptions and a large *common reference string* (CRS). Constant-round [41, 27] and two-round [50, 29] protocols for adaptively secure MPC are known assuming indistinguishability obfuscation (iO) for circuits and one-way functions (OWF). These protocols have sublinear communication ([41, 27] in the semi-honest model, [50, 29] in the malicious setting[3]), but require a large CRS (at least linear in the circuit size). In particular, the approach of these results is to place an obfuscated universal circuit into the structured reference string which can compute any function of a given size. Thus, these results are more aptly described as *bounded-circuit-size* adaptively secure MPC.[4] In contrast, we aim to study a setup model in which the reference string is smaller (preferably independent) of the size of the evaluated function.

Lastly, recent advances in the static setting [86, 4] presented protocols with online-computation that only depends on the function's *depth* but not on its *size*. In the adaptive setting, the only known protocols achieving this goal are [50, 29], which, as discussed above, rely on iO and a large structured reference string.

We now present three results in the fully adaptive setting: in Section 1.1.1, a resource-efficient MPC protocol; in Section 1.1.2, feasibility and infeasibility results regarding one-sided-optimized two-party protocols; and in Section 1.1.3, NIZK protocols with a short proof.

[2]In the *uniform random string* model (a.k.a. the *common random string* model), all parties receive a uniformly random string generated in a trusted setup phase. In the *structured random string* model (a.k.a. the *common reference string* model), the common string is sampled according to some pre-defined distribution.

[3]The protocols in [41, 27] use the CLOS compiler [24] to get malicious security. Since the communication of previously known adaptively secure ZK protocols depends on the NP relation (see [78, 64, 47] and references therein), the communication of the maliciously secure protocols depended on the CRS. Our short NIZK (Theorem 1.3) can be used to reduce the communication of [41, 27] in the malicious setting as well.

[4]The protocol in [29] considers the RAM complexity of the computation; hence, the CRS depends on the size of the RAM program. See further discussion in Section 1.3.

1.1.1 Two-Round MPC with Low Communication and Online-Computation

Thus, the first result of this paper is to present a two-round fully adaptively secure MPC that requires only sublinear communication (i.e., depends only on the inputs, outputs, and depth of the function), sublinear online-computation, and that uses a sublinear common reference string. To achieve our result, we combine the techniques from the recent work on *Laconic Function Evaluation* (LFE) [86] (that can be instantiated under a natural variant of the learning with errors assumption, called *adaptive LWE*, or ALWE for short.[5]) and *explainability compilers* [41]. In this sense, our answer to the main question regarding the cost of adaptive security versus static security shows a minimal cost to the communication complexity in the secure-erasures model, and the addition of complexity assumptions in the erasures-free setting: namely the need for sub-exponentially secure iO in order to implement the explainability compiler. Table 1 summarizes the performance characteristics of prior work in comparison with our new result.

Theorem 1.1 (adaptively secure MPC with sublinear communication, informal)**.** *Assuming ALWE and secure erasures (respectively, sub-exponential iO), every function can be securely computed by a two-round protocol tolerating a malicious adversary that can adaptively corrupt all of the parties, such that the communication complexity, the online-computational complexity, and the size of the URS (respectively, CRS) are sublinear in the function size.*

To explain the key bottleneck in achieving our result, note that almost all known methods for succinct MPC in the static setting rely on *fully homomorphic encryption* [53].[6] The general template is for parties to encrypt and broadcast their inputs, independently evaluate the function on said inputs, and then jointly decrypt the output. The problem in the case of adaptive security is that the simulator must produce a transcript for such a protocol, consisting of the input ciphertexts and the output ciphertext, without knowing the inputs of any parties; later after corruption, the simulator would need to provide a decryption key that explains the ciphertexts for any given input and for the final output. Unfortunately, Katz et al. [76] showed that this exact task is not possible for all functions, even assuming secure erasures, since the existence of such a simulator would imply a compact circuit that can be used to compute the function.

To get around the impossibility of adaptively secure FHE, the key insight of our approach is to instead use a recent technique of laconic function evaluation (LFE) [86], itself an extension of the idea of laconic OT [32]. At a high level, LFE allows a party to publish a short *digest* of a function; later any party can encrypt an input to that function such that the resulting ciphertext is still small with respect to the *size* of the function. In particular, both the digest and the ciphertext size are proportional to the *depth* of the function. Because the computational cost of the decryption algorithm is proportional to evaluating the function, LFE avoids the impossibility argument for adaptive security from [76], while preserving the succinct communication pattern. LFE is in some sense a dual notion to FHE. We extend on this duality in the discussion on the two-party case in Section 1.1.2.

Our starting point follows the statically secure protocol from [86]. The idea is for the parties to each locally compute a digest of the function f (this is done deterministically, using a URS for

[5]The basic construction in [86] holds under the standard LWE assumption; however, for the purpose of (semi-) malicious MPC, in which the inputs to the protocol can be chosen adaptively, after the URS is published, the stronger variant is required.

[6]Another approach for compact MPC is using *function secret sharing (FSS)* [18, 19]. This approach does not seem to support adaptive corruptions.

LFE parameters), and then use an MPC protocol (possibly not communication efficient) to jointly compute the encryption of the inputs $(x_1, \ldots, x_n)$. The communication and online-computation required are naturally proportional only to the encryption algorithm, which depends on the depth of the original function but not on its size. Finally, each of the parties can then locally decrypt the ciphertext with respect to the digest to recover the output.

Nonetheless, for adaptive security, it is unclear how to simulate the output ciphertext when possibly all n parties can be corrupted. To circumvent this barrier, we first observe that the protocol from [86] achieves adaptive security in the *erasures* model, without any additional assumptions, and then remove the erasures using the explainability compiler technique from [41]. Loosely speaking, an explainability compiler takes a randomized circuit C and compiles it to a circuit $\widetilde{C}$, computing the same function, along with an additional program $\mathsf{Explain}$, such that given any input/output pair (x, y) the program $\mathsf{Explain}$ can produce coins r satisfying $y = \widetilde{C}(x; r)$.

Overall, this framework achieves all of the round, communication, and online-computational complexity goals, but it still requires a URS/CRS whose size is related to the *depth* of the function being computed, and further in the erasures-free setting, it relies on iO. In contrast, in the static corruption setting, only LWE is required.

Protocol	**Security (erasures)**	**Rounds**	**Communication**	**Online Computation**	**Setup size**	**Setup type**	**Assumption**
MW [80]	static	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	$\mathrm{poly}(\kappa, d)$	URS	LWE, NIZK
QWW [86] ABJMS [4]	static	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	$\mathrm{poly}(\ell_{\mathsf{in}}, d, \kappa, n)$	URS	ALWE LWE
CLOS [24]	adaptive(no)	$O(d)$	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	$\mathrm{poly}(\kappa)$	URS	TDP, NCE dense-crypto
GS [51]*	adaptive(no)	$O(d)$	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	-	-	CRH TDP, NCE dense-crypto
DKR [41] CGP [27]	adaptive(no)	$O(1)$	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	CRS	OWF, iO
GP [50] CPV [29]†	adaptive(no)	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, \kappa, n)$	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, \kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$ $\mathrm{poly}(\lvert \mathrm{RAM}\rvert, \kappa)$	CRS	OWF, iO
CPV [30]	adaptive(no)	$O(1)$	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	$\mathrm{poly}(\kappa)$	URS	NCE dense-crypto
BLPV [13]	adaptive(no)	2	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	$\mathrm{poly}(\lvert C\rvert, \kappa)$	$\mathrm{poly}(\kappa)$	CRS	adaptive 2-round OT
This work	adaptive(yes) adaptive(no)	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	$\mathrm{poly}(\ell_{\mathsf{in}}, d, \kappa, n)$ $\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	URS CRS	ALWE ALWE, iO

Table 1: Round, communication, and online-computation of MPC tolerating any number of corruptions, for $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ represented by a circuit C of depth d. *URS* refers to a uniform random string, whereas *CRS* refers to a common reference string whose sampling coins are secret. $(*)$ The results in [51] only hold in the stand-alone setting. $(\dagger)$ The results in [29] consider the RAM complexity of the computation, see Section 1.3 for a detailed comparison.

1.1.2 Alice/Bob-Optimized protocols

Consider a two-message protocol for two parties, where Alice sends the first message, Bob replies with the second, and only Alice learns the output. In this setting, it is possible for one party's total *computation* (and thus also total communication) to depend on the size of their input and output, while the other party (whose input may be *much* larger) "does all of the work" of securely evaluating the function. These protocol variants are designated as "optimized for Alice" or "optimized for Bob," depending on which party saves the work.

In the static-corruption setting, Alice-optimized protocols can be constructed assuming FHE, where Alice encrypts her input, Bob homomorphically evaluates the circuit and returns the encrypted result. Quach et al. [86] showed that Bob-optimized protocols can be constructed from LFE, where Alice compresses the function with her input hard-wired, sends the digest to Bob who replies with the encryption of his input. Therefore, in the static setting, FHE and LFE are dual notions with respect to the workload of the computation. We next show that in the adaptive setting this duality breaks. On the one hand, we extend the impossibility result of FHE [76] to rule out adaptively secure two-round Alice-optimized protocols (even assuming secure erasures), in situations where Alice's input is significantly smaller than Bob's input. On the other hand, we construct an adaptively secure, semi-malicious,[7] Bob-optimized protocol from LFE and explainability compilers (alternatively, just from LFE assuming secure erasures). We note that any two-round Bob-optimized protocol can be converted into a three-round Alice-optimized protocol, which is the best one could hope for. Table 2 summarizes our results vis-a-vis prior work.

Theorem 1.2 (Alice/Bob-optimized protocols, informal)**.**

1. *Assuming ALWE and secure erasures (alternatively, sub-exponential iO), there exists an adaptively secure semi-malicious 2PC, where the total communication and Bob's computation are sublinear in the function size and in Alice's input size.*

2. *There exists two-party functions such that in any adaptively secure, semi-honest, two-round protocol realizing them, Bob's message must grow linearly in his input, even assuming secure erasures.*

The key idea behind our Bob-optimized protocol is to use the same LFE approach put forth in [86] for static security, and strengthen it to tolerate adaptive corruptions. To support an adaptive corruption of Alice, the simulator will need to produce an *equivocal* first message, i.e., to simulate the digest without knowing the input value of Alice, and upon a later corruption of Alice generate appropriate random coins explaining the message. Our first technical contribution is to create an equivocal version of the LFE scheme of [86]. Similarly, to support an adaptive corruption of Bob, the simulator should be able to generate an equivocal second message, i.e., generate the ciphertext without knowing the input of Bob, and upon a later corruption of Bob provide appropriate random coins. This can be handled either assuming secure erasures, or using explainability compilers.

1.1.3 Succinct Adaptively Secure NIZK

Next, we consider the problem of constructing an adaptively secure non-interactive zero-knowledge protocol (NIZK) that is "succinct," i.e., the size of the proof and of the common reference string should be smaller than the size of the circuit relation. The best we can hope for is for the proof to be the size of the witness (as otherwise, the lower bound of Gentry and Wichs [54] requires a non-standard complexity assumption). The first adaptively secure NIZK was constructed by Groth

[7] In the semi-malicious setting, the adversary follows the protocol as in the semi-honest case, but he can choose arbitrary random coins for corrupted parties.

Approach	**Security**	**Setup**	**Communication**		**Computation**		**Assump.**
	(erasures)		Alice	Bob	Alice	Bob	
GC [93]	static	-	poly(ℓ_A)	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	static OT
LOT [32]	static	$O(1)$	$O(1)$	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	DDH, etc.
FHE [53]	static	-	poly(ℓ_A)	poly(ℓ_{out})	poly$(\ell_A, \ell_{\mathsf{out}})$	poly$(\lvert f\rvert)$	LWE
LFE [86]	static	poly(ℓ_B)	$O(1)$	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	poly$(\lvert f\rvert)$	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	ALWE
equivocal GC [30]	adaptive (no)	-	poly(ℓ_A)	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	adaptive OT
This work	adaptive (yes)	poly(ℓ_B)	$O(1)$	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	poly$(\lvert f\rvert)$	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	ALWE
	adaptive (no)	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	$O(1)$	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	poly$(\lvert f\rvert)$	$\ell_{\mathsf{out}} \cdot$ poly(ℓ_B)	ALWE, iO
	adaptive (yes)	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	poly$(\ell_{\mathsf{out}}, \ell_A) \cdot o(\ell_B)$	poly$(\lvert f\rvert)$	poly$(\lvert f\rvert)$	impossible

Table 2: Comparison of two-message semi-honest protocols for $f : \{0,1\}^{\ell_A} \times \{0,1\}^{\ell_B} \to \{0,1\}^{\ell_{\mathsf{out}}}$. Alice talks first, Bob the second, and only Alice learns the output. For simplicity, multiplicative factors that are polynomial in the security parameter κ or the circuit depth d are suppressed. Setup refers to a URS in the static/adaptive-with-erasures cases, and a CRS in the adaptive-without-erasures case; the impossibility holds for any common reference string.

et al. [62]; however it was not succinct. Gentry [53] and later Gentry et al. [57] combined FHE with a standard NIZK system to construct such schemes that are secure against static corruptions, and as observed in [57] also against adaptive corruptions in the secure-erasures setting. However, these schemes are not secure against adaptive corruptions in the erasure-free setting. In particular, they run into the FHE bottleneck for adaptive security by Katz et al. [76] described above.

Our main technique to overcome this lower bound is to use *homomorphic trapdoor functions* (HTDFs) [59]. HTDF schemes are a primitive that conceptually unites homomorphic encryption and homomorphic signatures. In our usage, HTDF can be thought of as fully homomorphic commitment schemes which are equivocal (hence, statistically hiding), where a trapdoor can be used to open any commitment to any desired value. Using HTDF, the prover can commit to the witness (instead of encrypting it), evaluate the circuit over the commitments, and use adaptive but non-succinct NIZK (e.g., from [62]) to prove knowledge of the witness and that the result commits to 1. The verifier evaluates the circuits over the committed witness, and verifies the NIZK to ensure that the result is a commitment to 1. A summary of our results in comparison with prior work appears in Table 3.

Theorem 1.3 (short NIZK, informal)**.** *Assuming LWE, if there exists adaptively secure NIZK arguments for NP, then there exists adaptively secure NIZK arguments for NP with proof size sublinear in the circuit size of the NP relation.*

1.2 Adaptive Corruptions of a Strict Subset of the parties

Recall that the notion of fully adaptive security allows the adversary to corrupt *all* of the parties in the execution—in which case the protocol offers no privacy of inputs. A criticism of this notion is that it may be too strong for certain applications. In fact, the motivation behind this strong notion arises mainly from its application to *composition* of protocols. Namely, in a larger protocol that involves more parties, participants of a subprotocol may eventually *all* become corrupted, and thus security of the larger protocol will depend on the fully adaptive security of the subprotocol.

It is equally justifiable, however, to consider other protocol-design tasks in which the protocol needs only withstand a weaker adversary who can corrupt either all-but-one of the participants,

Protocol	**Security (erasures)**	**CRS size**	**Proof size**	**Assumptions**
Groth [61]	static	$\|C\| \cdot \text{poly}(\kappa)$	$\|C\| \cdot \text{poly}(\kappa)$	TDP
Groth [61]	static	$\|C\| \cdot \text{polylog}(\kappa) + \text{poly}(\kappa)$	$\|C\| \cdot \text{poly}(\kappa)$	Naccache–Stern
GOS [62]	adaptive (no)	$\text{poly}(\kappa)$	$\|C\| \cdot \text{poly}(\kappa)$	pairing based
Gentry [53]	adaptive (yes)	$\text{poly}(\kappa)$	$\|w\| \cdot \text{poly}(\kappa, d)$	LWE, NIZK
GGIPSS [57]	adaptive (yes)	$\text{poly}(\kappa)$	$\|w\| + \text{poly}(\kappa, d)$	LWE, NIZK
This work	adaptive (no)	$\text{poly}(\kappa)$	$\|w\| \cdot \text{poly}(\kappa, d)$	LWE, NIZK

Table 3: NIZK arguments for security parameter κ, circuit size $|C|$, depth d, and witness size $|w|$.

or—weaker still—only a minority of the players. We next consider adaptive security in these two settings.

1.2.1 All-But-One Corruptions

When considering adaptive security for all-but-one corruptions, Ishai et al. [71] constructed a constant-round, information-theoretically secure protocol in the OT-hybrid model. Garg and Sahai [51] showed an elegant way to instantiate the trusted setup required for [71] using non-black-box techniques and thus constructed a constant-round MPC protocol in the plain model, under standard cryptographic assumptions. The communication in both of these protocols is super-linear in the circuit size.

In contrast, for the weaker notion of static security, Asharov et al. [5] presented a two-round protocol with sublinear communication, albeit in the *threshold-PKI model*. The threshold-PKI model is a setup in which all the participants of the protocol are privately given individualized key shares corresponding to a public key. A single-round protocol for threshold PKI was also given in [5], yielding a three-round protocol in a standard URS setup. Mukherjee and Wichs [80] removed the need for this extra round, thereby presented a two-round MPC with sublinear communication in the uniform random string model.

We can thus pose our main question regarding the *cost* of adaptive security for communication-optimal protocols. Recently, Damgård et al. [46] constructed an adaptively secure three-round MPC protocol with sublinear communication complexity in the threshold-PKI model assuming LWE. Their main idea is to use a special threshold FHE scheme that enables equivocating encryptions of 0 to encryptions of 1. Initially, the parties broadcast encryptions of their inputs. Next, each party locally evaluates the circuit, and the parties re-randomize the evaluated ciphertext in the second round by broadcasting (special) encryptions of 0. The third round is a single-round threshold decryption protocol.

To simulate this protocol, the simulator uses the equivocal mode of the public key. This way, all ciphertexts in the first round are simulated as encryptions of 0. After extracting corrupted parties' inputs, and obtaining the output value, the simulator uses the re-randomizing round to carefully add nonzero encryptions, and force the joint ciphertext to be an encryption of the output. Finally, the threshold decryption protocol is simulated. We note that using the approach of [46] (which is based on [44]), the re-randomization round seems to be inherent, and so it is unclear how to obtain optimal two rounds using this technique.

Our result in this setting is to construct an adaptively secure two-round MPC assuming non-committing encryption (NCE) and threshold equivocal FHE in the threshold-PKI setup model. The setup assumption can be instantiated using the recent two-round protocol of [13], assuming two-round adaptively secure OT, resulting in a four-round variant in the CRS model. All of the necessary primitives can be instantiated from LWE in the semi-malicious setting, and security in the malicious case follows using NIZK. Table 4 summarizes the prior work and our contribution in this model.

Theorem 1.4 (all-but-one corruptions, informal)**.** *Assuming LWE and adaptively secure NIZK, every function can be securely computed by a two-round protocol in the threshold-PKI model tolerating a malicious adversary that can adaptively corrupt all-but-one of the parties such that the communication complexity is sublinear in the function size.*

Protocol	**Security**	**Rounds**	**Communication**	**Assumptions**	**Setup**
AJLTVW [5]	static	2 3	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	threshold PKI URS
MW [80]	static	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	URS
IPS [71]	adaptive	$O(1)$	$\|C\| + \mathrm{poly}(d, \log \|C\|, \kappa, n)$	OT-hybrid	-
GS [51]	adaptive	$O(1)$	$\|C\| + \mathrm{poly}(d, \log \|C\|, \kappa, n)$	CRH, TDP, NCE dense crypto	-
DPR [46]	adaptive	3	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	threshold PKI
This work	adaptive	2 4	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	threshold PKI CRS

Table 4: Comparison of maliciously secure MPC for $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ represented by a circuit C of depth d, tolerating $n-1$ corruptions. $(*)$ The results in [51] only hold in the stand-alone model.

Our protocol follows the template of [5], where every party encrypts his input in the first round, locally evaluates the circuit over the ciphertexts, uses its key-share to partially decrypt the result, and broadcasts the decrypted share (some additional "smudging" noise is sometimes required to protect the decryption share). The technical challenges are: (1) the ciphertexts in the first round must be created in an equivocal way, and (2) the simulation strategy used for the threshold decryption in [5] (and similarly in [80]) is inherently static, and does not translate in a straightforward way to the adaptive setting.

We overcome the first challenge by constructing a novel *threshold equivocal FHE* scheme. The scheme is equipped with an equivocal key-generation algorithm. All ciphertexts encrypted in this mode are "meaningless" and carry no information about the plaintext; a trapdoor can be used to equivocate any ciphertext to any message. We instantiate this FHE scheme using the *dual-mode* HTDF scheme of Gorbunov et al. [59] that can generate the homomorphic trapdoor functions in an extractable mode, corresponding to the standard (meaningful) mode of the FHE, and an equivocal mode, corresponding to the meaningless mode.

We proceed to explain the second challenge. As observed in [5, 80], the threshold decryption protocol may leak some information about the shares of the secret key, and the simulator for the decryption protocol can be used to protect *exactly* one party. In the static setting, when the set of corrupted parties is known ahead of time, the simulator can choose one of the honest parties P_h as a special party for simulating the threshold decryption. This approach does not work in the adaptive setting since the party P_h may get corrupted after simulating the decryption protocol. The simulator cannot know in advance which party will be the last to remain honest. For this reason, we use a different simulation strategy which allows the simulator to "correct" his choice of the party that is simulated as honest for the decryption protocol. Technically, this is done by having each party send shares of zero to each other party over a secure channel (that can be instantiated via NCE). These shares are used to hide the partial decryptions without changing their values. Since shares exchanged between pairs of honest parties remain hidden from the eyes of the adversary, the simulator has more freedom to replace the special party P_h upon corruption, by another honest party, even after simulating the decryption protocol.

Thus, as it stands, the cost of adaptive security with respect to the best statically secure protocols is either the threshold-PKI setup assumption, or the requirement of 2 additional rounds. Removing either of these costs remains an interesting open question.

1.2.2 Honest-Majority Setting

In the honest-majority setting, it is possible to guarantee output delivery to all honest parties. Damgård and Ishai [42] demonstrated the feasibility of constructing adaptively secure protocols that use a constant number of rounds and only require one-way functions. However, the communication of their protocol is super-linear in the circuit size.

In the static-corruption setting, Asharov et al. [5] constructed the first protocol with sublinear communication using threshold FHE; their protocol requires 4 rounds in the threshold-PKI model and 5 rounds in the URS model. Gordon et al. [60] reduced the round complexity to 2 in the threshold-PKI model or 3 in the URS model. Recently, Ananth et al. [3] showed a three-round protocol in the plain model with communication polynomial in the circuit size, and Badrinarayanan et al. [6] showed a similar result with sublinear communication. Moreover, this round complexity is tight because it is known that two-round fair protocols are impossible in the URS/CRS model [52, 60, 85].[8]

Our result in this setting is to construct an adaptively secure analog of [5, 6]. In particular, we construct a two-round adaptively secure MPC with guaranteed output delivery and the same communication complexity as in the static case, assuming NCE and threshold equivocal FHE in the threshold-PKI model in the semi-malicious setting (all assumptions can be based on LWE). Security in the malicious case follows using NIZK. We can compile our two-round protocol into a constant-round protocol in the plain with the same communication complexity by computing the threshold-PKI setup using the protocol of Damgård and Ishai [42].

Theorem 1.5 (honest majority, informal)**.** *Assuming LWE and adaptively secure NIZK, every function can be securely computed with guaranteed output delivery by a two-round protocol in the threshold-PKI model tolerating a malicious adversary that can adaptively corrupt a minority of the parties such that the communication complexity is sublinear in the function size.*

The two-round protocol is based on the protocol from the all-but-one case, described in Section 1.2.1. The challenge lies in overcoming aborting parties to guarantee output delivery. We

[8]We emphasize that the lower bounds hold given a public-coin setup, where all parties get the same information, and does not hold given a private-coin setup such as threshold PKI.

combine techniques from the threshold FHE of [60] that required $n/2$ decryption shares to reconstruct the output into our threshold equivocal FHE. The main idea is to share the decryption key using Shamir's secret sharing instead of additive secret sharing. Both Shamir's reconstruction and the decryption algorithm consist of linear operations, which make them compatible with each other. As observed by Gordon et al. [60] (see also [17]), the problem with a naïve use of this technique is that the "smudging noise" (used to protect partial decryptions from leakage) is multiplied by the Lagrange coefficients, which may cause an incorrect decryption. Following [60], we have each party secret shares his smudging noise using Shamir's scheme, in a way that is compatible with the reconstruction procedure. We show that this technique can support adaptive corruptions.

To conclude, in the threshold-PKI model, the price of adaptive security is *the same* as of static security in terms of assumptions, number of rounds, and communication complexity. In the plain model, the cost is an additional constant number of rounds. Table 5 summarizes prior work and our results.

Protocol	**Security**	**Rounds**	**Communication**	**Assumptions**	**Setup**
AJLTVW [5]	static	4	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	threshold PKI
		5			URS
GLS [60]	static	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	threshold PKI
		3			URS
ACGJ [3]	static	3	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	PKE and zaps	-
BJMS [6]	static	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, zaps, dense crypto	threshold PKI
		3			-
DI [42]	adaptive	$O(1)$	$\lvert C\rvert \cdot \mathrm{poly}(\kappa, n)$	OWF	-
This work	adaptive	2	$\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$	LWE, NIZK	threshold PKI
		$O(1)$			-

Table 5: Comparison of maliciously secure MPC for $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ represented by a circuit C of depth d, in the honest-majority setting.

1.3 Additional Related Work

Adaptive security tolerating an arbitrary number of corruptions has been considered in various models, including protocols in the common string model [24, 30, 13], the sunspot model [26], the key-registration model [8], the tamper-proof hardware model [67], the super-polynomial simulation model [7, 65], and more generally, based on UC-puzzles [40, 92]. All of these protocols require super-linear communication complexity.

Adaptive security in the secure-erasures model was considered in [10, 77, 72, 12, 83, 45, 66], and in the erasures-free model tolerating all-but-one corruptions in [74, 71, 63, 46] as well as in the honest-majority setting [39, 44, 42]. With the exception of [46], all of these protocols also require super-linear communication complexity.

Garay et al. [49] considered information-theoretic MPC in the client–server setting, where a constant number of clients use n servers that assist with the computation, and studied sublinear communication in the number of *servers*. They gave a complete characterization for semi-honest security with static corruptions and adaptive corruptions with or without erasures.

In the static setting, MPC with sublinear communication complexity over eventual-delivery asynchronous channels was constructed in [35]. We conjecture that our techniques can also be applied in the asynchronous setting to obtain adaptive security with low communication.

We note that since the protocol of Garg and Polychroniadou [50] has low communication complexity, and its CRS size depends on the circuit size, it is possible to use a more compact representation of the function, e.g., by a Turing machine (TM) (or a RAM program as considered in [29]), and obfuscate it using iO for Turing machines. Nonetheless, the solution provided in this paper is different in several qualitative aspects. First, to make the CRS independent of the computation at hand, it is preferred to obfuscate a *universal* TM, which receives the description of the concrete TM on its input tape; while iO for TM with *bounded* inputs exists under the same assumptions as iO for circuits [14, 28, 15], iO for TM with *unbounded* inputs is only known under the stronger assumption of public-coin differing-inputs obfuscation [73]. Second, it is not clear how to replace the iO for TM assumption by secure erasures. Third, the computation may require a large auxiliary information, e.g., access to a large database, whose description is independent of the TM; this may result with a large description of the function. In our solution, the obfuscated circuit is sublinear in the computation size of the function to be computed, even when a large auxiliary information is used.

1.4 Open Questions

Our main question is to study the price of adaptive security. Dramatic improvements in the answer to this question have emerged over the past 15 years, and this paper is able to establish almost zero cost in terms of round or communication. Our results, however, leave the following questions as future work.

- **Reducing setup assumptions.** Our results for fully adaptive, two-round, protocols without erasures require a *common reference string.* Are there fully adaptively secure protocols with sublinear communication complexity in the uniform random string (even with super-constant number of rounds)?

- **Reducing hardness assumptions.** Are there fully adaptively secure protocols with sublinear communication without assuming secure erasures or explainability compilers/iO?

- **Improving setup assumptions/round complexity for all-but-one.** Our optimal-round protocol requires a pre-distribution of the FHE keys. We show a four-round protocol in the CRS model (equivalently, in the plain model for semi-honest). Are there two or three round protocols with sublinear communication in the CRS model to match the results for static adversaries?

Paper Organization

The preliminaries are presented in §2. In §3, we present our adaptively secure NIZK construction. In §4, we present our results on fully adaptive security, and in §5.1 and §5.2, we present our results on Bob- and Alice-optimized protocols. In §6, we consider the all-but-one corruption case, and in §7, the honest-majority case. Some of the preliminaries (Appendix A), the model definition (Appendix B), and the construction of threshold equivocal FHE from LWE (Appendix C) are deferred to the appendix.

2 Preliminaries

Basic notations. For $n \in \mathbb{N}$ let $[n] = \{1, \cdots, n\}$. We denote by κ the security parameter. Let poly denote the set all positive polynomials and let PPT denote a probabilistic algorithm that runs in *strictly* polynomial time. A function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}$ is *negligible* if $\mathsf{negl}(\kappa) < 1/p(\kappa)$ for every $p \in$ poly and sufficiently large κ. Given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X. Given a PPT Turing machine M, denote by $T_M(x)$ the polynomial that bounds the number of random coins used by M on input x; by abuse of notation, instead of denoting the sampling the random coins for M on input x by $r \leftarrow \{0,1\}^{T_M(x)}$ we will often write $r \leftarrow \{0,1\}^*$. The statistical distance between two random variables X and Y over a finite set $\mathcal{U}$, denoted $\mathrm{SD}(X, Y)$, is defined as $\frac{1}{2} \cdot \sum_{u \in \mathcal{U}} |\Pr[X = u] - \Pr[Y = u]|$. The $\mathsf{entropy}$ of X is denoted by $H(X) = -\sum_{x \in \mathcal{U}} \Pr[X = x] \log_2 \Pr[X = x]$.

Two distribution ensembles $X = \{X(a, \kappa)\}_{a \in \{0,1\}^*, \kappa \in \mathbb{N}}$ and $Y = \{Y(a, \kappa)\}_{a \in \{0,1\}^*, \kappa \in \mathbb{N}}$ are $\mathsf{computationally\ indistinguishable}$ (denoted $X \overset{\mathrm{c}}{\equiv} Y$) if for every non-uniform polynomial-time distinguisher $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\kappa)$, such that for every $a \in \{0,1\}^*$ and every κ,

$$|\Pr[\mathcal{A}(X(a, \kappa), 1^\kappa) = 1] - \Pr[\mathcal{A}(Y(a, \kappa), 1^\kappa) = 1]| \leq \mathsf{negl}(\kappa).$$

The distribution ensembles X and Y are $\mathsf{statistically\ close}$ (denoted $X \overset{\mathrm{s}}{\equiv} Y$) if for every $a \in \{0,1\}^*$ and every κ it holds that $\mathrm{SD}(X(a, \kappa), Y(a, \kappa)) \leq \mathsf{negl}(\kappa)$. We represent elements in $\mathbb{Z}_q$ as integers in the range $(-q/2, q/2]$. We denote by λ the empty string.

Cryptographic primitives. In this work, we consider secure protocols in various security settings that require different cryptographic primitives. We present formal definitions for all primitives in Appendix A. An informal description of every primitive is given before it is used in the main body.

Security model. We present our results in the UC framework. In Appendix B, we provide a high-level description of the framework, and refer the reader to [22] for a more detailed description.

In our secure function evaluation (SFE) protocols, we will consider two security notions. In the honest-majority setting, we will consider security with *guaranteed output delivery*, informally meaning that all honest parties will receive the correct output from the computation; we denote by $\mathcal{F}_{\mathsf{sfe\text{-}god}}$ the SFE functionality with guaranteed output delivery. In general, when an honest majority is not assumed this cannot be achieved [34], and the standard requirement is for *security with abort*, informally meaning that the adversary has the capability to first learn the output from the computation and later force all honest parties to output $\perp$; we denote by $\mathcal{F}_{\mathsf{sfe\text{-}abort}}$ the SFE functionality with abort.

All parties are connected by authenticated communication channels; that is, the adversary learns the content of messages sent between honest parties, but cannot change them. We denote by $\mathcal{F}_{\mathsf{auth}}$ the authenticated-communication functionality. In some of our results, specifically in Sections 5 to 7 we further require private channels, where the adversary learns the lengths of messages sent between honest parties, but not their content. We denote by $\mathcal{F}_{\mathsf{smt}}$ the secure message transmission functionality. Private channels can be built over authenticated channels using non-committing encryption. We denote by $\mathcal{F}_{\mathsf{bc}}$ the broadcast channel functionality.

Guaranteed output delivery and security with abort are not to be confused with *guaranteed termination*, which means that the honest parties actually finish the protocol. We emphasize that

UC protocols cannot provide guaranteed termination since the adversary has full control over the communication channels, and he can simply "hang" the computation. Therefore, following the convention of Canetti et al. [24], we exclude trivial protocols, and require that the properties of guaranteed output delivery or security with abort will hold when the environment provides sufficiently many activations to the parties, and the adversary delivers all messages.[9] In particular, unlike the stand-alone model, in the UC model even when a protocol guarantees output delivery, we allow the adversary to learn the output from the computation while the honest parties do not; however, if an honest party terminates it is guaranteed to receive the output. An alternative, is to work in the $\mathcal{F}_{\mathsf{sync}}$-hybrid model [34] or to consider the framework of [75], which ensures guaranteed termination regardless of the adversary's actions (but, still, as long as the environment provides sufficiently many activations to the parties).

3 Adaptively Secure NIZK

Informally speaking, universally composable non-interactive zero-knowledge arguments (UC-NIZK) are single-message zero-knowledge protocols, that remain secure independently of the environment in which they are run, i.e., regardless of the protocol that is calling them. Groth et al. [62] formalized UC-NIZK as a protocol that realizes the NIZK ideal functionality, and constructed adaptively secure UC-NIZK protocols (that remain secure even if all the participants are dynamically corrupted) under hardness assumptions in bilinear groups. The size of the proof in [62] is $|C| \cdot \mathrm{poly}(\kappa)$, where C is the circuit computing the NP-relation R, i.e., $C(x, w) = 1$ if and only if $(x, w) \in R$.

Gentry [53] showed that assuming LWE the communication complexity can be reduced to be sublinear in the circuit size, namely $|w| \cdot \mathrm{poly}(\kappa, d)$. Gentry et al. [57] further reduced the proof size to $|w| + \mathrm{poly}(\kappa, d)$ and proved adaptive security of the scheme assuming secure erasures. The underlying idea in [53, 57] is for the prover to encrypt the witness using an FHE scheme and prove knowledge of the plaintexts. In addition, the prover homomorphically evaluates the circuit C and proves that the resulting ciphertext decrypts to 1. The verifier homomorphically evaluates the circuit C over the encrypted witness, makes sure that the result matches the one sent by the prover, and validates the proof sent by the prover (that he knows the witness and the result decrypts to 1). Adaptive security is obtained by having the prover erase all of his coins before sending the proof.

We next present a new adaptively secure UC-NIZK with communication complexity that only depends on the witness size and does *not* rely on secure erasures. The main obstacles in translating the ideas from [53, 57] to the erasures-free adaptive setting are: (1) the encryption of the witness must be first simulated (without knowing the witness) and later, upon a corruption of the prover, the encryption coins must be properly explained, and (2) the impossibility result of Katz et al. [76] rule-out a standard use of FHE schemes in the protocol. We get around these barriers by using *homomorphic trapdoor functions* (HTDFs) [59] (see Appendix A.2.3).

3.1 The UC-NIZK Ideal Functionality

The NIZK functionality, as defined in [62, 57], is based on the zero-knowledge functionality from [22], and is adjusted to capture the special properties of a non-interactive proof. In particular, as opposed to interactive ZK protocols, the verifiers are not always known in advance, and therefore should not be defined as part of the interface to the functionality. Furthermore, since the NIZK argument

[9]Other properties such as *privacy* and *independence of inputs* are always required to hold.

is simply a bit-string, anybody can use this string to verify the validity of the statement and can use it to convince others.

Functionality $\mathcal{F}_{\mathsf{nizk}}^{R}$

$\mathcal{F}_{\mathsf{nizk}}^{R}$ proceeds as follows, interacting with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$, and parameterized by an NP-relation R.

Proof: On input $(\mathsf{prove}, \mathsf{sid}, x, w)$ from party P_i, ignore if $(x, w) \notin R$. Send $(\mathsf{prove}, P_i, \mathsf{sid}, x)$ to $\mathcal{S}$ and wait for $(\mathsf{proof}, \mathsf{sid}, \pi)$. Upon receiving the answer store (sid, x, π) and send $(\mathsf{proof}, \mathsf{sid}, \pi)$ to P_i.

Verification: On input $(\mathsf{verify}, \mathsf{sid}, x, \pi)$ from a party P_j check whether (sid, x, π) is stored. If not send $(\mathsf{verify}, P_j, \mathsf{sid}, x, \pi)$ to $\mathcal{S}$ and wait for an answer $(\mathsf{witness}, w)$. Upon receiving the answer, check whether $(x, w) \in R$ and in that case, store (sid, x, π). If (sid, x, π) has been stored return $(\mathsf{verification}, \mathsf{sid}, x, \pi, 1)$ to P_j, else return $(\mathsf{verification}, \mathsf{sid}, x, \pi, 0)$ to P_j.

Figure 1: The ideal NIZK proof functionality

The NIZK ideal functionality (formally described in Figure 1) is parametrized by an NP-relation R and a set of parties $P_1, \ldots, P_n$. Every party P_i can send a prove request which consists of a pair (x, w). The functionality verifies that $(x, w) \in R$, and asks the adversary to generate a proof π for the statement x. The functionality stores (x, π) and returns the proof to P_i. Similarly, every party P_j can send a verify request, consisting of a statement x and a proof π. In case the pair (x, π) is stored, the functionality returns 1. Otherwise, the functionality asks the adversary to come up with a witness w. If $(x, w) \in R$ the functionality returns 1 to P_j; otherwise, return 0.

3.2 Cryptographic Primitives Used in the Protocol

We informally describe the cryptographic primitives used in the construction. Formal definitions appear in the appendix.

3.2.1 Homomorphic Trapdoor Functions (HTDFs)

We briefly recall the notion of homomorphic trapdoor functions, formally described in Appendix A.2.3. The key-generation algorithm outputs a public key and a secret key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d)$. The public key along with some bit $x \in \{0,1\}$ define an efficiently computable function $f_{\mathsf{pk},x} : \mathcal{U} \to \mathcal{V}$, and the secret key acts as a trapdoor that enables inverting the function, i.e., defines a function $\mathsf{HTDF.Inv}_{\mathsf{sk},x} : \mathcal{V} \to \mathcal{U}$. Two homomorphic evaluation algorithms are defined given a function $g : \{0,1\}^\ell \to \{0,1\}$ of depth d: the *inner* evaluation $u^* = \mathsf{HTDF.Eval}^{\mathsf{in}}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ and the *outer* evaluation $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(g, v_1, \ldots, v_\ell)$.

Informally, we require the following properties from an HTDF scheme:

- **Correctness.** Let $x_1, \ldots, x_\ell \in \{0,1\}$ and $v_i = f_{pk,x_i}(u_i)$ for $i \in [\ell]$. Then, for $u^* = \mathsf{HTDF.Eval}^{\mathsf{in}}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ and $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(g, v_1, \ldots, v_\ell)$ it holds that $f_{pk,y}(u^*) = v^*$, where $y = g(x_1, \ldots, x_\ell)$.
- **Distributional equivalence of inversion.** For a bit $x \in \{0,1\}$, the tuple (pk, x, u, v) computed as $v = f_{\mathsf{pk},x}(u)$ for a random $u \leftarrow \mathcal{U}$ is statistically close to sampling $v \leftarrow \mathcal{V}$ at random and computing $u = \mathsf{HTDF.Inv}_{\mathsf{sk},x}(v)$.

- **Claw-free security.** Given the public key, no efficient adversary can come up with u and u' such that $f_{\mathsf{pk},0}(u) = f_{\mathsf{pk},1}(u')$ with more than a negligible probability.

For the remaining of the section, it will be useful to think of an HTDF scheme as a fully homomorphic, statistically-hiding commitment scheme with a trapdoor sk.

3.2.2 Strong One-Time Signatures

Strong one-time signatures are formally defined in Appendix A.2.4. Loosely peaking, the scheme consists of three algorithms $(\mathsf{Sig.Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. The key-generation algorithm outputs a public verification key and a secret signing key $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.Gen}(1^\kappa)$. The signing algorithm uses the signing key to sign an arbitrary message m as $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sigk}}(m)$. The verification algorithm uses the verification key to accept or reject the signature as $b = \mathsf{Vrfy}_{\mathsf{vk}}(\sigma, m)$. We require that the scheme be correct and strongly existentially unforgeable under a single chosen message attack, meaning that except for negligible probability, no efficient adversary that chooses a message m and receives a signature σ can come up with $(m', \sigma') \neq (m, \sigma)$ such that $1 = \mathsf{Vrfy}_{\mathsf{vk}}(\sigma', m')$.

3.3 The UC-NIZK Protocol

We proceed to present a non-interactive protocol that securely realizes $\mathcal{F}^{R}_{\mathsf{nizk}}$ for an arbitrary NP-relation R. Denote by d the depth of the circuit computing the relation. A proof for a statement x with witnesses of size $|w|$ consists of $|w| \cdot \mathrm{poly}(\kappa, d)$ bits (we denote the bit-length of the witness by $\ell = |w|$ below, i.e., $w = w_1, \ldots, w_\ell$). We make two assumptions: the existence of an HTDF scheme and that $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$ can be securely realized for the relation R_{htdf} defined as follows:

$$R_{\mathsf{htdf}} = \left\{ ((\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), (w_1, \ldots, w_\ell, u_1, \ldots, u_\ell, u^*)) \;\middle|\; \ell \in \mathbb{N}, \forall i \in [\ell] : v_i = f_{\mathsf{pk}, w_i}(u_i) \text{ , and } v^* = f_{\mathsf{pk},1}(u^*) \right\}.$$

We define the protocol in the common reference string model, with the distribution D_{nizk} that is parametrized by an HTDF scheme Π. The distribution D_{nizk} samples a key-pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d)$, and outputs pk. Note that since the public key in the construction of [59] is a random string, we can consider $\mathcal{F}^{D_{\mathsf{nizk}}}_{\mathsf{crs}}$ as a uniform random string functionality.

The main idea of the protocol is to use the HTDF scheme to commit to the witness, bit by bit, and use the homomorphic properties to evaluate the relation on the committed witness. The prover can use a NIZK for the relation R_{htdf} to prove knowledge of the witness, and that the result of the homomorphic evaluation commits to 1. The simulator can use the equivocal properties of the HTDF scheme to generate an equivocal commitment that can be opened to any witness.

Theorem 3.1 (Theorem 1.3 restated)**.** *Assume the existence of HTDF schemes and let R be an NP-relation. Protocol π_{nizk}, defined in Figure 2, UC-realizes $\mathcal{F}^{R}_{\mathsf{nizk}}$ in the $(\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}, \mathcal{F}^{D_{\mathsf{nizk}}}_{\mathsf{crs}})$-hybrid model tolerating an adaptive, malicious adversary. The proof size is $\ell \cdot \mathrm{poly}(\kappa, d)$, where ℓ is a bound on the witness size and d is the multiplicative depth of R.*

Proof. Let $\mathcal{A}$ be an adaptive, malicious adversary attacking π_{nizk} in the $(\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{crs}})$-hybrid model. We will construct an ideal-process adversary $\mathcal{S}$, interacting with the ideal functionality $\mathcal{F}^{R}_{\mathsf{nizk}}$ and with ideal (dummy) parties $\widetilde{P}_1, \ldots, \widetilde{P}_n$,[10] such that no environment can distinguish between $\mathcal{S}$ and

[10]Recall that a dummy party in UC acts as a "router" passing every message it receives from the environment to the ideal functionality and vice versa, see Appendix B.2.

$\mathcal{A}$. Let $\mathcal{Z}$ be an environment. Following the lines of [57], we will construct $\mathcal{S}$ gradually, starting with a simulator $\mathcal{S}_{\text{REAL}}$ with "extended capabilities" that can perfectly simulate an execution of π_{nizk} with $\mathcal{A}$ and adjusting it in steps to a simulator $\mathcal{S}_{\text{SIM}}$ that has the same power as in the ideal computation; we will prove that the environment will have a negligible probability of distinguishing between consecutive steps.

Protocol π_{nizk}

- **Common Input:** An HTDF scheme, a one-time signature scheme, and an NP-relation R.
- **Notation:** Assume every witness to relation R is ℓ-bits long, i.e., $w = (w_1, \dots, w_\ell) \in \{0,1\}^\ell$. For a statement x denote by C_x the circuit that on input $y \in \{0,1\}^\ell$ computes $C_x(y) := R(x, y)$.
- **Hybrid model:** The parties have access to the NIZK functionality $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{htdf}}}$ and to the CRS functionality $\mathcal{F}_{\mathsf{crs}}^{D_{\mathsf{nizk}}}$ that outputs an HTDF public key pk.
- **The Protocol:**

1. Upon receiving $(\mathsf{prove}, \mathsf{sid}, x, w)$, party P_i proceeds as follows:
   (a) If $(x, w) \notin R$ ignore the message. Otherwise, denote $w = (w_1, \dots, w_\ell)$.
   (b) For $i \in [\ell]$ compute $v_i = f_{\mathsf{pk}, w_i}(u_i)$ for a uniformly distributed $u_i \leftarrow \mathcal{U}$.[a]
   (c) Compute $u^* = \mathsf{HTDF.Eval}^{\mathsf{in}}(C_x, (w_1, u_1), \dots, (w_\ell, u_\ell))$.
   (d) Compute $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(C_x, v_1, \dots, v_\ell)$.
   (e) Generate signature keys $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.Gen}(1^\kappa)$.
   (f) Send $(\mathsf{prove}, \mathsf{sid}, (\mathsf{pk}, v_1, \dots, v_\ell, v^*, \mathsf{vk}), (w_1, \dots, w_\ell, u_1, \dots, u_\ell, u^*))$ to $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{htdf}}}$.
   (g) Wait for the answer $(\mathsf{proof}, \mathsf{sid}, (\mathsf{pk}, v_1, \dots, v_\ell, v^*, \mathsf{vk}), \pi)$ from $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{htdf}}}$.
   (h) Compute $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sigk}}((x, \mathsf{pk}, v_1, \dots, v_\ell, v^*, \pi, \mathsf{vk}))$.
   (i) Return $(\mathsf{proof}, \mathsf{sid}, x, (\mathsf{pk}, v_1, \dots, v_\ell, v^*, \pi, \mathsf{vk}, \sigma))$.
2. Upon receiving $(\mathsf{verify}, \mathsf{sid}, x, \Pi)$, party P_j proceeds as follows:
   (a) Parse Π as $(\mathsf{pk}, v_1, \dots, v_\ell, v^*, \pi, \mathsf{vk}, \sigma)$.
   (b) Verify that $\mathsf{Vrfy}_{\mathsf{vk}}(\sigma, (x, \mathsf{pk}, v_1, \dots, v_\ell, v^*, \pi, \mathsf{vk})) = 1$. If not return $(\mathsf{verification}, \mathsf{sid}, x, \Pi, 0)$.
   (c) Verify that $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(C_x, v_1, \dots, v_\ell)$. If not return $(\mathsf{verification}, \mathsf{sid}, x, \Pi, 0)$.
   (d) Send $(\mathsf{verify}, \mathsf{sid}, (\mathsf{pk}, v_1, \dots, v_\ell, v^*, \mathsf{vk}), \pi)$ to $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{htdf}}}$.
   (e) Wait for the answer $(\mathsf{verification}, \mathsf{sid}, (\mathsf{pk}, v_1, \dots, v_\ell, v^*, \mathsf{vk}), \pi, b)$ from $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{htdf}}}$.
   (f) Return $(\mathsf{verification}, \mathsf{sid}, x, \Pi, b)$.

[a]Recall that an HTDF scheme defines a function $f_{\mathsf{pk},x} : \mathcal{U} \to \mathcal{V}$, see Definition A.8.

Figure 2: Adaptively secure UC-NIZK protocol

Simulator $\mathcal{S}_{\text{REAL}}$. In the first mental experiment, the simulator $\mathcal{S}_{\text{REAL}}$ can learn the inputs of the ideal functionality $\mathcal{F}_{\mathsf{nizk}}^{R}$ and control its output. It can therefore run a perfect simulation of the parties $P_1, \dots, P_n$ and the adversary $\mathcal{A}$ running π_{nizk}. The simulator $\mathcal{S}_{\text{REAL}}$ starts by running $\mathcal{A}$.

- To simulate the communication with $\mathcal{Z}$, every input value that $\mathcal{S}_{\text{REAL}}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}_{\text{REAL}}$'s own output tape.

- To simulate the common reference string, the simulator $\mathcal{S}$ first computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d)$, sets the reference string to be pk, and stores sk.
- When $\mathcal{S}_{\text{REAL}}$ receives $(\mathsf{prove}, P_i, \mathsf{sid}, x)$ from $\mathcal{F}^R_{\mathsf{nizk}}$, it is because an honest party $\widetilde{P}_i$ has input $(\mathsf{prove}, \mathsf{sid}, x, w)$ with $(x, w) \in R$. The simulator $\mathcal{S}_{\text{REAL}}$ has access to the input of $\mathcal{F}^R_{\mathsf{nizk}}$, and so it can simulate P_i running π_{nizk}. In particular, for simulating $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$ the simulator may send $(\mathsf{prove}, P_i, \mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}))$ to $\mathcal{A}$ and get back the answer $(\mathsf{proof}, \mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \pi)$. Finally, $\mathcal{S}_{\text{REAL}}$ computes a signature to prepare the proof $\Pi = (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \pi, \mathsf{vk}, \sigma)$ and answers $(\mathsf{proof}, \mathsf{sid}, x, \Pi)$ to $\mathcal{F}^R_{\mathsf{nizk}}$.
- When $\mathcal{S}_{\text{REAL}}$ receives $(\mathsf{verify}, P_j, \mathsf{sid}, x, \Pi)$ from $\mathcal{F}^R_{\mathsf{nizk}}$, it is because an honest party $\widetilde{P}_j$ has input $(\mathsf{verify}, \mathsf{sid}, x, \Pi)$ and (sid, x, Π) was not stored in $\mathcal{F}^R_{\mathsf{nizk}}$, hence has not been created by an honest party. The simulator simulates P_j running the verification protocol on input $(\mathsf{verify}, \mathsf{sid}, x, \Pi)$ until obtaining output $(\mathsf{verification}, \mathsf{sid}, x, \Pi, b)$. Next, $\mathcal{S}_{\text{REAL}}$ instructs $\mathcal{F}^R_{\mathsf{nizk}}$ to return the message $(\mathsf{verification}, \mathsf{sid}, x, \Pi, b)$ to $\widetilde{P}_j$, and to store (sid, x, Π) if $b = 1$. (Note that $\mathcal{S}_{\text{REAL}}$ does not send $(\mathsf{witness}, w)$ to $\mathcal{F}^R_{\mathsf{nizk}}$.)
- Upon a corruption request for a party P_i, the simulator corrupts the dummy party $\widetilde{P}_i$. For every input message of the form $(\mathsf{prove}, \mathsf{sid}, x, w)$ that $\widetilde{P}_i$ received, set the random coins corresponding to the proof as $(u_1, \ldots, u_\ell)$ that were used for computing $(v_1, \ldots, v_\ell)$ along with the random coins that were used to generate the signature.

Claim 3.2. $\text{REAL}_{\pi_{\mathsf{nizk}}, \mathcal{A}, \mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{S}_{\text{REAL}}, \mathcal{F}^R_{\mathsf{nizk}}, \mathcal{Z}}$.

Proof. The only difference between an execution of π_{nizk} in the $(\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}, \mathcal{F}^{D_{\mathsf{nizk}}}_{\mathsf{crs}})$-hybrid model and the simulation by $\mathcal{S}_{\text{REAL}}$, lies in the proving protocol. In the simulation, it is guaranteed that a proof that is generated by an honest party will always be accepting (as it is generated, and stored by $\mathcal{F}^R_{\mathsf{nizk}}$, when invoked with $(\mathsf{proof}, \mathsf{sid}, x, w)$) and that accepted proofs will always be accepted again. This holds also in the execution of π_{nizk} by the correctness of the signature scheme and of the HTDF scheme, and by the properties if $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$.

It follows that to the environment, a real execution of π_{nizk} in the $(\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}, \mathcal{F}^{D_{\mathsf{nizk}}}_{\mathsf{crs}})$-hybrid model with adversary $\mathcal{A}$ is perfectly indistinguishable from the simulation of $\mathcal{S}_{\text{REAL}}$ running with $\mathcal{F}^R_{\mathsf{nizk}}$. $\square$

Simulator $\mathcal{S}_{\text{EXT}}$. The simulator $\mathcal{S}_{\text{EXT}}$ runs like $\mathcal{S}_{\text{REAL}}$ when the proofs are constructed, but changes the way proofs are verified. When $\mathcal{S}_{\text{EXT}}$ receives $(\mathsf{verify}, P_j, \mathsf{sid}, x, \Pi)$ from $\mathcal{F}^R_{\mathsf{nizk}}$, it simulates P_j running the verification protocol on input $(\mathsf{verify}, \mathsf{sid}, x, \Pi)$. If P_j outputs $(\mathsf{verification}, \mathsf{sid}, x, \Pi, 0)$, then return $(\mathsf{witness}, \perp)$ to $\mathcal{F}^R_{\mathsf{nizk}}$. If P_j outputs $(\mathsf{verification}, \mathsf{sid}, x, \Pi, 1)$, then $\mathcal{S}_{\text{EXT}}$ will try to extract a witness w such that $(x, w) \in R$, return $(\mathsf{witness}, w)$ to $\mathcal{F}^R_{\mathsf{nizk}}$, and send the resulting message $(\mathsf{verification}, \mathsf{sid}, x, \Pi, 1)$ to $\widetilde{P}_j$ (in case the extraction fails, the simulator aborts).

To extract the witness, $\mathcal{S}_{\text{EXT}}$ parses $\Pi = (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \pi, \mathsf{vk}, \sigma)$. We can assume that the signature is valid and that $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(C_x, v_1, \ldots, v_\ell)$, since π_{nizk} will reject the proof otherwise. As part of the verification protocol, the functionality $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$ may send $(\mathsf{verify}, P_j, \mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \pi)$ to $\mathcal{A}$ (if $(\mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \pi)$ is not stored), who responds with $(\mathsf{witness}, w_1, \ldots, w_\ell, u_1, \ldots, u_\ell, u^*)$. In that case, the simulator sets $w = (w_1, \ldots, w_\ell)$ and sends $(\mathsf{witness}, w)$ back to $\mathcal{F}^R_{\mathsf{nizk}}$, and aborts otherwise.

Claim 3.3. $\text{IDEAL}_{\mathcal{S}_{\textsc{real}},\mathcal{F}^{R}_{\mathsf{nizk}},\mathcal{Z}} \overset{\text{c}}{\equiv} \text{IDEAL}_{\mathcal{S}_{\textsc{ext}},\mathcal{F}^{R}_{\mathsf{nizk}},\mathcal{Z}}$.

Proof. The simulated $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$ will only return $(\mathsf{verification}, \mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \pi, 1)$ if an honest party has created a proof π or if the adversary $\mathcal{A}$ supplies a witness $(w_1, \ldots, w_\ell, u_1, \ldots, u_\ell, u^*)$ such that $v_i = f_{\mathsf{pk},w_i}(u_i)$ and $v^* = f_{\mathsf{pk},1}(u^*)$. In the latter case, the simulator sets $w = (w_1, \ldots, w_\ell)$. By the correctness of the HTDF scheme, it holds that $(x, w) \in R$; therefore, $\mathcal{S}_{\text{EXT}}$ can return $(\mathsf{witness}, w)$ to $\mathcal{F}^{R}_{\mathsf{nizk}}$. In case an honest party has previously created the proof π on $(\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk})$ (and so it was stored in the simulated $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$), then the strong existential unforgeability of the one-time signature scheme implies that the adversary is able to produce a different valid signature σ using vk only with a negligible probability. Therefore, $\mathcal{S}_{\text{EXT}}$ fails to produce a witness only with a negligible probability. $\square$

Simulator $\mathcal{S}_{\textbf{SIM}}$. The simulator $\mathcal{S}_{\text{SIM}}$ runs exactly like $\mathcal{S}_{\text{EXT}}$, with the following modification. Instead of running a perfect simulation of $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$, the simulator $\mathcal{S}_{\text{SIM}}$ allows simulated honest parties to submit $(\mathsf{prove}, \mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \perp)$, even if $(x, w) \notin R$. This means that the simulated ideal functionality $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$ may ask $\mathcal{A}$ for a proof π for a statement $(\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk})$, store $(\mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \pi)$ as being a valid proof, and return $(\mathsf{proof}, \mathsf{sid}, (\mathsf{pk}, v_1, \ldots, v_\ell, v^*, \mathsf{vk}), \pi)$ to the calling party P_i.

When $\mathcal{S}_{\text{SIM}}$ receives $(\mathsf{prove}, P_i, \mathsf{sid}, x)$ from $\mathcal{F}^{R}_{\mathsf{nizk}}$, it changes the way the proof is simulated. Instead of computing the values $v_1, \ldots, v_\ell$ as $v_i = f_{\mathsf{pk},w_i}(u_i)$ for $u_i \leftarrow \mathcal{U}$, the simulator samples uniformly distributed values $v_1, \ldots, v_\ell \leftarrow \mathcal{V}$. Upon a corruption request for a party P_i, the simulator corrupts the dummy party $\widetilde{P}_i$. For every input message of the form $(\mathsf{prove}, \mathsf{sid}, x, w)$ that $\widetilde{P}_i$ received, set the random coins corresponding to the proof as $(u_1, \ldots, u_\ell)$, where $u_i = \mathsf{HTDF.Inv}_{\mathsf{sk},w_i}(v_i)$.

Claim 3.4. $\text{IDEAL}_{\mathcal{S}_{\textsc{ext}},\mathcal{F}^{R}_{\mathsf{nizk}},\mathcal{Z}} \overset{\text{s}}{\equiv} \text{IDEAL}_{\mathcal{S}_{\textsc{sim}},\mathcal{F}^{R}_{\mathsf{nizk}},\mathcal{Z}}$.

Proof. This follows from the security of the HTDF scheme by using a standard hybrid argument. First, we show that the simulated proofs of $\mathcal{S}_{\text{EXT}}$ are statistically close to the simulated proofs of $\mathcal{S}_{\text{SIM}}$. We define $\ell + 1$ experiments, where in the i'th experiment the first $i - 1$ values v_j are computed as $v_j \leftarrow \mathcal{V}$ and are explained as $u_j \leftarrow \mathsf{HTDF.Inv}_{\mathsf{sk},w_j}(v_j)$. The remaining $\ell - i + 1$ values are computed as $v_j = f_{\mathsf{pk},w_j}(u_j)$ for $u_j \leftarrow \mathcal{U}$. The first experiment is exactly the way $\mathcal{S}_{\text{EXT}}$ simulates the proof and the last experiment is exactly the way $\mathcal{S}_{\text{SIM}}$ simulates the proof. By the properties of the HTDF scheme neighboring experiments are statistically close, and therefore also the proof simulated by $\mathcal{S}_{\text{EXT}}$ and $\mathcal{S}_{\text{SIM}}$.

Second, since the number of prove requests is bounded by a polynomial number, we can use a second hybrid argument to replace the way the proofs are simulated and explained one by one, and conclude that the simulation of $\mathcal{S}_{\text{EXT}}$ is statistically close to the simulation of $\mathcal{S}_{\text{SIM}}$. $\square$

Note that $\mathcal{S}_{\text{SIM}}$ does not require any access to the internals of $\mathcal{F}^{R}_{\mathsf{nizk}}$. Therefore, we can now define the simulator $\mathcal{S}$ that does not have any access to the internals of $\mathcal{F}^{R}_{\mathsf{nizk}}$, and operates exactly like $\mathcal{S}_{\text{SIM}}$. As we have shown, $\mathcal{S}$ running with $\mathcal{F}^{R}_{\mathsf{nizk}}$ is computationally indistinguishable from the protocol π_{nizk} running with $\mathcal{A}$ in the $(\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{crs}})$-hybrid model. The protocol π_{nizk} therefore securely realizes $\mathcal{F}^{R}_{\mathsf{nizk}}$ in the $(\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}, \mathcal{F}_{\mathsf{crs}})$-hybrid model. $\square$

By instantiating the $\mathcal{F}^{R_{\mathsf{htdf}}}_{\mathsf{nizk}}$ with the UC-NIZK protocol from [62], we obtain the following corollary.

Corollary 3.5. *Under the decision linear assumption (equivalently, the subgroup decision problem) in bilinear groups, and the existence of HTDF schemes, the following holds. For every NP-relation R, the functionality $\mathcal{F}_{\mathsf{nizk}}^{R}$ can be UC-realized in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model tolerating an adaptive, malicious adversary such that the size of the CRS is* $\mathrm{poly}(\kappa, d)$ *and the size of the proofs is* $|w| \cdot \mathrm{poly}(\kappa, d)$*.*

4 Sublinear Communication in the Fully Adaptive Setting

In this section, we consider the fully adaptive setting (where the adversary can corrupt all parties) and construct two-round secure protocols with sublinear communication and online-computational complexity (in the circuit size). Our starting point is the protocol of Quach et al. [86] that is based on *laconic function evaluation (LFE)* (see Appendix A.2.1).

4.1 Cryptographic Primitives used in the Protocol

We informally describe the cryptographic primitives used in the construction. Formal definitions appear in the appendix.

Laconic function evaluation. We formally define laconic function evaluation (LFE) in Appendix A.2.1. Informally, an LFE scheme consists of 4 algorithms. The CRS generation algorithm generates a common reference string given the security parameter and function parameters (e.g., function depth and input length) $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^{\kappa}, \mathsf{params})$. The compression algorithm produces a small digest of a circuit $\mathsf{digest}_C = \mathsf{LFE.Compress}(\mathsf{crs}, C; r)$. The encryption algorithm encrypts the input based on the digest $\mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_C, x)$. The decryption algorithm decrypts the ciphertext using the random coins used in the compression $y = \mathsf{LFE.Dec}(\mathsf{crs}, C, r, \mathsf{ct})$.

We require the LFE to be $\mathsf{correct}$, i.e., using the notation above it holds that $y = C(x)$, and secure, meaning that the ciphertext can be simulated given the output value y without knowing the input x. LFE can be constructed with the $\mathsf{function\text{-}hiding}$ property, which ensures that the digest can be simulated based on the function parameters without knowing the function itself. If function hiding is not required (as is the case in this section) the compression algorithm can be made deterministic. We consider the "adaptive" version of LFE, where the inputs to the computation can be chosen *after* the CRS has been sampled. Quach et al. [86] constructed LFE schemes satisfying this property assuming *adaptive LWE* (see Appendix A.1.2), where the CRS is *uniform* random string.

For a function $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \rightarrow \{0,1\}^{\ell_{\mathsf{out}}}$ of depth d, the crs length is $\mathrm{poly}(\kappa, n, d, \ell_{\mathsf{in}})$, the digest is $\mathrm{poly}(\kappa)$, and the ciphertext ct is $\mathrm{poly}(\kappa, n, d, \ell_{\mathsf{in}}, \ell_{\mathsf{out}})$,

Explainability compilers. We formally define explainability compilers in Appendix A.2.2. Informally, an explainability compiler takes as input a description of a randomized algorithm Alg, and outputs two algorithms: $\widetilde{\mathsf{Alg}}$ and $\mathsf{Explain}$. The first algorithm $\widetilde{\mathsf{Alg}}$ computes the same functionality as Alg. The second algorithm $\mathsf{Explain}$ takes an input/output pair (x, y) and produces random coins r such that $y = \widetilde{\mathsf{Alg}}(x; r)$.

Assuming iO for circuits and OWF, Dachman-Soled et al. [41] constructed explainability compilers with *selective* security, where the challenge input is selected independently of the compiled circuit. Explainability compilers with *adaptive* security, where the challenge input is selected based on the compiled circuit, follow via complexity leveraging [16] assuming iO and OWF with sub-exponential security (see also [27]). Looking ahead, to support adaptive inputs from the environment, our protocol requires the latter variant.

4.2 Adaptive Security with Sublinear Communication: Secure-Erasures Setting

We will show that assuming LFE with a uniform random string and a deterministic compression algorithm, every function can be securely realized in the uniform *random* string model with secure erasures, by a two-round protocol tolerating an arbitrary number of adaptive corruptions with sublinear communication, online-computation, and CRS size. In Section 4.3, we will show how to replace the secure-erasures assumption by assuming explainability compilers, in which case the protocol requires a common *reference* string.

The basis of our protocol is the two-round protocol of Quach et al. [86, Thm. 6.2] in the uniform random string model, that is secure against $n-1$ static corruptions and achieves sublinear communication and online-computation assuming the existence of LFE. The protocol from [86] is specified in a hybrid model with an ideally secure computation (with abort) of the function $\mathsf{LFE.Enc}$ (i.e., the $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{\mathsf{LFE.Enc}}$-hybrid model). That is, the ideal functionality receives $(\mathsf{crs}, \mathsf{digest}_f, x_i, r_i)$ from each party P_i and computes

$$\mathsf{ct} = \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n; \oplus_{i\in[n]} r_i).$$

In case of inconsistent inputs, or if the adversary sends abort, the functionality outputs $\perp$.

Given a circuit C_f computing the function $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$, denote $f.\mathsf{params} = (1^{n\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, 1^d)$ where $n\ell_{\mathsf{in}}$ is the input size, ℓ_{out} the output length, and d is the depth of C_f. The protocol of [86] is defined as follows:

- The uniform random string is computed as $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, f.\mathsf{params})$.
- Upon receiving $(\mathsf{input}, \mathsf{sid}, x_i)$, every party P_i computes $\mathsf{digest}_f = \mathsf{LFE.Compress}(\mathsf{crs}, C_f)$, samples a uniformly random $r_i \leftarrow \{0,1\}^*$, and invokes the ideal functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{\mathsf{LFE.Enc}}$ with $(\mathsf{input}, \mathsf{sid}, (\mathsf{crs}, \mathsf{digest}_f, x_i, r_i))$.
- Upon receiving $(\mathsf{output}, \mathsf{sid}, \mathsf{ct})$ from the ideal functionality, party P_i checks that $\mathsf{ct} \neq \perp$ (otherwise, P_i outputs $(\mathsf{output}, \mathsf{sid}, \perp)$), computes $y = \mathsf{LFE.Dec}(\mathsf{crs}, C_f, \mathsf{ct})$, and outputs $(\mathsf{output}, \mathsf{sid}, y)$.

Proving security of the protocol against a static adversary corrupting all-but-one of the parties is straightforward. Namely, by definition of LFE schemes, the simulator can simulate the ciphertext ct based on the output y, and without knowing the input values, as $\mathsf{ct} \leftarrow \mathsf{Sim}_{\mathrm{LFE}}(\mathsf{crs}, C_f, \mathsf{digest}_f, y)$. Furthermore, by the properties of LFE, the size of the circuit computing $\mathsf{LFE.Enc}$ is $\mathrm{poly}(\kappa, \ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, n)$. By instantiating the ideal functionality using a statically secure two-round protocol (e.g., the one from [80]), Quach et al. [86] achieved a statically secure protocol with sublinear communication and online-computational complexity.

A closer look at the protocol of [86] shows that it remains secure even facing adaptive corruptions of all-but-one of the parties, since a single honest party suffices to keep the randomness used for $\mathsf{LFE.Enc}$ hidden from the adversary. Furthermore, under the additional assumption of secure erasures, each party can erase his random coins r_i immediately after invoking $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{\mathsf{LFE.Enc}}$, and the protocol can satisfy adaptive corruptions of all the parties. By instantiating the functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{\mathsf{LFE.Enc}}$ with the two-round adaptively secure MPC from [13], we obtain the following theorem.

Theorem 4.1 (Theorem 1.1, secure-erasures version, restated)**.** *Assume the existence of compact LFE schemes for* P/poly *with a uniform random string and deterministic compression, of two-round adaptively and maliciously secure OT, and of secure erasures, and let* $f : (\{0,1\}^{\ell_{in}})^n \to \{0,1\}^{\ell_{out}}$ *be an n-party function of depth d.*

Then, $\mathcal{F}^{f}_{\mathsf{sfe\text{-}abort}}$ *can be UC-realized tolerating a malicious, adaptive PPT adversary by a two-round protocol in the uniform random string model. The size of the uniform random string is* $\mathrm{poly}(\kappa, \ell_{in}, d, n)$*, whereas the communication and online-computational complexity of the protocol are* $\mathrm{poly}(\kappa, \ell_{in}, \ell_{out}, d, n)$*.*

Note that following [86, 13], the assumptions in Theorem 4.1 hold under the adaptive LWE assumption.

4.3 Adaptive Security with Sublinear Communication: Erasures-Free Setting

In the erasures-free setting, it is unclear how to simulate the output ciphertext, and later upon learning all of the inputs values of the parties, explain the random coins that are used to generate it. We get around this barrier by using explainability compilers.

4.3.1 Two-Round Protocol Assuming Adaptive Explainability Compilers

We consider explainability compilers with adaptive security (where the challenge ciphertext is dynamically chosen) that can be realized by sub-exponentially secure iO and OWF. To define the common reference string for the protocol, we define the distribution $D_{\mathsf{lfe}}(\mathsf{params})$ that is parametrized by an LFE scheme and by the parameters of the function to be computed params. The distribution D_{lfe} computes $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$ and $(\widetilde{\mathsf{LFE.Enc}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{LFE.Enc})$, and outputs the reference string $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$.

We would like to define the protocol in the $\widetilde{\mathsf{LFE.Enc}}$-hybrid model; however, the function $\widetilde{\mathsf{LFE.Enc}}$ is only given in the CRS and is not known before the protocol begins. To get around this technicality, we define the function $f_C((C_1, x_1, r_1), \ldots, (C_n, x_n, r_n))$ that receives a circuit C_i, a value x_i, and random coins r_i from each party, and outputs $C_1(x_1, \ldots, x_n; \oplus r_i)$ in case $C_1 = \ldots = C_n$, or $\perp$ otherwise.

Theorem 4.2 (Theorem 1.1, erasures-free version, restated)**.** *Assume the existence of compact LFE schemes for* P/poly *with deterministic compression, of explainability compilers with adaptive security for* P/poly*, and of two-round adaptively and maliciously secure OT, and let* $f : (\{0,1\}^{\ell_{in}})^n \to \{0,1\}^{\ell_{out}}$ *be a deterministic n-party function of depth d.*

Then, $\mathcal{F}^{f}_{\mathsf{sfe\text{-}abort}}$ *can be UC-realized in the* $\mathcal{F}^{D_{\mathsf{lfe}}}_{\mathsf{crs}}$*-hybrid model tolerating a malicious, adaptive PPT adversary by a two-round protocol. The size of the common reference string, the communication complexity, and online-computational complexity of the protocol are* $\mathrm{poly}(\kappa, \ell_{in}, \ell_{out}, d, n)$*.*

The proof of the theorem follows from Lemma 4.3 by instantiating the functionality $\mathcal{F}^{f_C}_{\mathsf{sfe\text{-}abort}}$, that is used to compute $\widetilde{\mathsf{LFE.Enc}}$, using the two-round protocol from [13] that requires two-round adaptively and maliciously secure OT.

Protocol π_{full}

- **Common Input:** An LFE scheme and a circuit C_f computing the function f.
- **Hybrid model:** The parties have access to the CRS functionality $\mathcal{F}_{\mathsf{crs}}^{D_{\mathsf{lfe}}(f.\mathsf{params})}$ that outputs a crs for the LFE scheme and a circuit $\widetilde{\mathsf{LFE.Enc}}$, and to the SFE functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C}$.
- **The Protocol:**

1. Upon receiving $(\mathsf{input}, \mathsf{sid}, x_i)$, every party P_i invokes $\mathcal{F}_{\mathsf{crs}}^{D_{\mathsf{lfe}}(f.\mathsf{params})}$ to get $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$, computes $\mathsf{digest}_f = \mathsf{LFE.Compress}(\mathsf{crs}, C_f)$, samples a uniformly random $r_i \leftarrow \{0,1\}^*$, and invokes $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C}$ with $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$.
2. Upon receiving ct from the ideal functionality, party P_i checks that $\mathsf{ct} \neq \bot$ (if so P_i outputs $(\mathsf{output}, \mathsf{sid}, \bot)$), computes $y = \mathsf{LFE.Dec}(\mathsf{crs}, C_f, \mathsf{ct})$, and outputs $(\mathsf{output}, \mathsf{sid}, y)$.

Figure 3: Two-round SFE with adaptive, malicious security

Lemma 4.3. *Assume the existence of compact LFE schemes for* $\mathsf{P/poly}$ *with deterministic compression, and of explainability compilers with adaptive security for* $\mathsf{P/poly}$*, and let f be a deterministic n-party function. Then, the protocol π_{full} (Figure 3) UC-realizes $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f}$ tolerating a malicious, adaptive PPT adversary in the $(\mathcal{F}_{\mathsf{crs}}^{D_{\mathsf{lfe}}}, \mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C})$-hybrid model.*

Proof. The protocol π_{full} is defined in Figure 3 as an adjustment of the protocol from [86]. We now turn to prove its security. Let $\mathcal{A}$ be the dummy adversary attacking π_{full} in the $(\mathcal{F}_{\mathsf{crs}}^{D_{\mathsf{lfe}}}, \mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C})$-hybrid model. We will construct an ideal-process adversary $\mathcal{S}$, interacting with the ideal functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f}$ and with ideal (dummy) parties $\widetilde{P}_1, \ldots, \widetilde{P}_n$, such that no environment can distinguish between $\mathcal{S}$ and $\mathcal{A}$. Let $\mathcal{Z}$ be an environment. As there is no interaction between the parties, the simulation should only explain corruption requests and the hybrid functionalities.

- To simulate the CRS, $\mathcal{S}$ computes $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$ and $(\widetilde{\mathsf{LFE.Enc}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{LFE.Enc})$. Next, $\mathcal{S}$ gives $\mathcal{A}$ the string $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ and stores the function $\mathsf{Explain}$.
- Upon receiving from the adversary the message $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$ (by simulating the hybrid functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C}$), check if the messages thus far are consistent (meaning that all messages have the same $\widetilde{\mathsf{LFE.Enc}}, \mathsf{crs}$ and digest_f), and send $(\mathsf{input}, \mathsf{sid}, x_i)$ to $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f}$. Otherwise send abort and respond with $\bot$ to the adversary.
- Upon receiving $(\mathsf{output}, \mathsf{sid}, y)$ from $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f}$, compute $\mathsf{ct}^* \leftarrow \mathsf{Sim}_{\mathrm{LFE}}(\mathsf{crs}, C_f, \mathsf{digest}_f, y)$ and respond with $(\mathsf{output}, \mathsf{sid}, \mathsf{ct}^*)$ to all corrupted parties (on behalf of $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C}$).
- Upon a corruption request of a party P_i, corrupt the dummy party $\widetilde{P}_i$, learn his input x_i, and proceeds as follows:
  - If not all parties are corrupted, sample uniformly distributed random coins $r_i \leftarrow \{0,1\}^*$ and set the message to $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f_C}$ to be $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$. If the party P_i received output y from $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^{f}$, set the response from $\widetilde{\mathsf{LFE.Enc}}$ to be $(\mathsf{output}, \mathsf{sid}, \mathsf{ct}^*)$.

– To explain the last corruption request (say of party P_n), the simulator computes $r \leftarrow \mathsf{Explain}((\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n), \mathsf{ct}^*)$, and sets $r_n = r \oplus r_1 \oplus \ldots \oplus r_{n-1}$.

We prove the indistinguishability of the execution of the protocol with the dummy adversary $\mathcal{A}$ from the ideal process with $\mathcal{F}^{f}_{\mathsf{sfe\text{-}abort}}$ and $\mathcal{S}$ by defining a series of hybrid games. The output of each game is the output of the environment.

The game $\text{HYB}^1_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$**.** The first hybrid is defined as the execution of the protocol π_{full} in the $(\mathcal{F}^{D_{\mathsf{lfe}}}_{\mathsf{crs}}, \mathcal{F}^{f_C}_{\mathsf{sfe\text{-}abort}})$-hybrid model.

The game $\text{HYB}^2_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$**.** The first hybrid is defined as $\text{HYB}^1_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$ with the following modification. Upon a corruption of the n'th party (for convenience, say it is P_n) replace the actual coins that were used by P_n in the protocol with the following value r_n. Let $r_1, \ldots, r_{n-1}$ be the coins used by the first $n-1$ parties, then first compute $r \leftarrow \mathsf{Explain}((\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n), \mathsf{ct})$, and next compute P_n's coins as $r_n = r \oplus r_1 \oplus \ldots \oplus r_{n-1}$.

Claim 4.4. $\text{HYB}^1_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}} \overset{c}{\equiv} \text{HYB}^2_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$.

Proof. The claim holds by the security of the explainability compiler. More precisely, given a PPT environment $\mathcal{Z}$ and a polynomial-time non-uniform distinguisher $\mathcal{D}$ that can distinguish between $\text{HYB}^1_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$ and $\text{HYB}^2_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$, we construct an attacker $\mathcal{A}'$ to the game $\mathsf{Expt}^{\mathsf{Explain\text{-}Adapt}}_{\mathsf{Comp},\mathsf{LFE.Enc},\mathcal{A}'}(\kappa)$ (defined in Appendix A.2.2).

Upon receiving $(1^\kappa, \widetilde{\mathsf{LFE.Enc}})$ from the challenger, the attacker $\mathcal{A}'$ starts by computing $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa)$ and $\mathsf{digest}_f = \mathsf{LFE.Compress}(\mathsf{crs}, C_f)$, and sends $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ to the environment. Next, proceed to simulate the dummy honest parties receiving the input values $(\mathsf{input}, \mathsf{sid}, x_i)$ from $\mathcal{Z}$, and the functionality $\mathcal{F}^{f_C}_{\mathsf{sfe\text{-}abort}}$ receiving the messages $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$ from $\mathcal{Z}$ on behalf of corrupted parties (if there are inconsistent messages output a random bit b' and halt).

Once all honest parties have been activated with input and all corrupted parties have sent messages, set $x^* = (\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n)$ and send x^* to the challenger. Upon receiving back (y^*, r^*) where $y^* = \mathsf{ct}^*$, respond with $(\mathsf{output}, \mathsf{sid}, \mathsf{ct}^*)$ to $\mathcal{Z}$ for every corrupted party. Corruption requests are answered as in the simulation by choosing random coins r_i; however, for the n'th corruption (say of P_n), compute $r_n = r^* \oplus r_1 \oplus \ldots \oplus r_{n-1}$. Once the environment halts, invoke $\mathcal{D}$ on the output of the environment and output the bit b' returned by $\mathcal{D}$.

Notice that if the challenger chooses the bit $b = 0$, i.e., sets r^* to be the random coins r_0 used to compute $y^* = \widetilde{\mathsf{LFE.Enc}}(x^*; r_0)$, then the view of the environment is identically distributed as in an $\text{HYB}^1_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$, whereas if the challenger chooses the bit $b = 1$, i.e., sets r^* to be the random coins r_1 computed as $r_1 \leftarrow \mathsf{Explain}(x^*, y^*)$, then the view of the environment is identically distributed as in $\text{HYB}^2_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$. Therefore, the success probability of $\mathcal{D}$ is the same as of $\mathcal{A}'$, which is negligible by the security of the explainability compiler. □

The game $\text{HYB}^3_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$**.** The third hybrid is defined as the second hybrid, but the functionality $\mathcal{F}^{f_C}_{\mathsf{sfe\text{-}abort}}$ operates differently. Upon receiving inputs $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$ from all the parties, the functionality computes the function $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n)$ instead of $\widetilde{\mathsf{LFE.Enc}}(\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n)$.

Claim 4.5. $\text{HYB}^2_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}} \overset{\text{s}}{\equiv} \text{HYB}^3_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$.

Proof. The proof follows by the statistical functional equivalence of the explainability compiler. $\square$

The game $\text{HYB}^4_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$**.** The fourth hybrid is defined as the third hybrid, but the functionality $\mathcal{F}^{f_C}_{\mathsf{sfe\text{-}abort}}$ operates differently. Upon receiving inputs $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$ from all the parties, the functionality computes $y = f(x_1, \ldots, x_n)$, creates a simulated ciphertext $\mathsf{ct} \leftarrow \mathsf{Sim}_{\text{LFE}}(\mathsf{crs}, C_f, \mathsf{digest}_f, y)$, and returns ct to the parties.

Claim 4.6. $\text{HYB}^3_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \text{HYB}^4_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$.

Proof. The claim holds by the security of the LFE scheme. More precisely, given a PPT environment $\mathcal{Z}$ and a polynomial-time non-uniform distinguisher $\mathcal{D}$ that can distinguish between $\text{HYB}^3_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$ and $\text{HYB}^4_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$, we construct an adversary $\mathcal{A}'$ and a distinguisher $\mathcal{D}'$ to the games $\mathsf{Expt}^{\mathsf{LFE\text{-}real}}_{\Pi,\mathcal{A}'}(\kappa)$ and $\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}_{\Pi,\mathcal{A}'}(\kappa)$ (defined in Appendix A.2.1).

The adversary $\mathcal{A}'$, on input 1^κ, starts by sending $f.\mathsf{params}$ to the challenger. Upon receiving crs, the adversary computes $(\widetilde{\mathsf{LFE.Enc}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{LFE.Enc})$ and gives $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ to the environment. Next, $\mathcal{A}'$ proceeds by simulating the dummy honest parties receiving the input values $(\mathsf{input}, \mathsf{sid}, x_i)$ from $\mathcal{Z}$, and the functionality $\mathcal{F}^{f_C}_{\mathsf{sfe\text{-}abort}}$ receiving the messages $(\mathsf{input}, \mathsf{sid}, (\widetilde{\mathsf{LFE.Enc}}, (\mathsf{crs}, \mathsf{digest}_f, x_i), r_i))$ from $\mathcal{Z}$ on behalf of corrupted parties (if there are inconsistent messages output abort).

Once all honest parties have been activated with input and all corrupted parties have sent messages, set $x^* = (x_1, \ldots, x_n)$ and send (x^*, C_f) to the challenger (recall that since function-hiding is not required, there is no need to send the coins r that are defined in the experiment). Upon receiving back ct^* from the challenger, respond with $(\mathsf{output}, \mathsf{sid}, \mathsf{ct}^*)$ to $\mathcal{Z}$ for every corrupted party. Corruption requests are answered as in the simulation by choosing random coins r_i, and for the n'th corruption (say of P_n), compute $r \leftarrow \mathsf{Explain}((\mathsf{crs}, \mathsf{digest}_f, x_1, \ldots, x_n), \mathsf{ct}^*)$, and set $r_n = r \oplus r_1 \oplus \ldots \oplus r_{n-1}$. Once the environment halts, the adversary $\mathcal{A}'$ outputs the output of the environment. When the distinguisher $\mathcal{D}'$ is invoked with the output of $\mathcal{A}'$, it invokes $\mathcal{D}$ and outputs the bit b' returned by $\mathcal{D}$.

Consider the experiment $\mathsf{Expt}^{\mathsf{LFE\text{-}real}}_{\Pi,\mathcal{A}'}(\kappa)$. In this case, the ciphertext ct^* is computed as $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_f, x^*)$ where $\mathsf{digest}_f = \mathsf{LFE.Compress}(\mathsf{crs}, C_f)$, exactly as done in $\text{HYB}^3_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$; hence, the view of the environment is identically distributed as in $\text{HYB}^3_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$. In the experiment $\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}_{\Pi,\mathcal{A}'}(\kappa)$, the ciphertext ct^* is computed as $\mathsf{ct} \leftarrow \mathsf{Sim}_{\text{LFE}}(\mathsf{crs}, C_f, \mathsf{digest}_f, y)$ (for $y = C_f(x_1, \ldots, x_n)$), and the view of the environment is identically distributed as in $\text{HYB}^4_{\pi_{\mathsf{full}},\mathcal{A},\mathcal{Z}}$. Therefore, the success probability of $\mathcal{D}$ is the same as of $\mathcal{D}'$, which is negligible by the security of the LFE scheme. $\square$

The fourth hybrid experiment operates identically as the simulator in the ideal model computing $\mathcal{F}^{f}_{\mathsf{sfe\text{-}abort}}$, and the lemma follows. $\square$

5 Adaptively Secure Alice/Bob-Optimized Protocols

In this section, we consider two-message protocols between Alice and Bob, with respective inputs $x_A \in \{0,1\}^{\ell_A}$ and $x_B \in \{0,1\}^{\ell_B}$, where only Alice learns the output $y = f(x_A, x_B)$. We say that a protocol is "Alice-optimized" if Alice's computation and the total communication of the protocol are proportional to $|x_A| + |y|$, while the computation complexity of Bob is proportional to $|f|$. We say that a protocol is "Bob-optimized" if Bob's computation and the total communication are proportional to $|x_B| + |y|$, while the computation complexity of Alice is proportional to $|f|$.

There exist insecure protocols which are Alice-optimized, where Alice sends her input to Bob who computes the function and returns the output to Alice. Similarly, there exist insecure protocols which are Bob-optimized, where Bob sends his input to Alice when she asks for it, and Alice computes the function on her own.

Assuming FHE [53], there exist statically secure Alice-optimized protocols, where Alice sends her encrypted input to Bob who homomorphically evaluates the function and returns the encrypted output to Alice. Alice's computation and the total communication of the protocol are $(|x_A| + |y|) \cdot \mathrm{poly}(\kappa)$. Assuming function-hiding LFE [86], there exist statically secure Bob-optimized protocols, where Alice sends $\mathsf{digest} \leftarrow \mathsf{LFE.Compress}(\mathsf{crs}, f_{x_A}(\cdot))$ to Bob, who replies with his encrypted input $\mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{digest}, x_B)$, and finally Alice recovers the output. Bob's computation and the total communication of the protocol are $(|x_B| + |y|) \cdot \mathrm{poly}(\kappa, d)$, where d is the depth of the function f.

The question we consider is whether there exist adaptively secure protocols which are Alice-optimized or Bob-optimized.

5.1 Adaptively Secure Bob-Optimized Protocol

The elegant protocol from [86] is secure in the uniform random string model tolerating a static corruption of one of the parties by a semi-malicious adversary (that can choose arbitrary random coins for the corrupted party, but acts honestly otherwise).

Adjusting this protocol to the adaptive setting requires overcoming a few obstacles. Namely, the simulator should be able to generate an equivocal first message, i.e., to simulate the digest without knowing the input value of Alice, and upon a later corruption of Alice generate appropriate random coins explaining the message. Similarly, the simulator should be able to generate an equivocal second message, i.e., generate the ciphertext without knowing the input of Bob, and upon a later corruption of Bob provide appropriate random coins.

To support an adaptive corruption of Alice, we enhance the LFE scheme to support an equivocal mode (see Section 5.1.1). In this mode, the CRS is generated along with a trapdoor information. The trapdoor can be used to explain a simulated digest as a compression of any circuit with the appropriate parameters. Similarly to Section 4, to support an adaptive corruption of Bob, we can use either secure erasures or explainability compilers (see Appendix A.2.2).

Theorem 5.1 (Part 1 of Theorem 1.2, restated)**.** *Assume the existence of equivocal, function-hiding, compact LFE schemes for* $\mathsf{P/poly}$ *and of explainability compilers with adaptive security for* $\mathsf{P/poly}$*, and let* $f : \{0,1\}^{\ell_A} \times \{0,1\}^{\ell_B} \rightarrow \{0,1\}^{\ell_{\mathsf{out}}}$ *be a deterministic two-party function computable by a depth-*d *circuit.*

Then, $\mathcal{F}^f_{\mathsf{sfe}}$ *can be UC-realized tolerating a semi-malicious, adaptive PPT adversary by a two-message protocol in the common reference string model with secure channels. The size of the common reference string, the communication complexity (of both parties), and the computational complexity of Bob are* $\ell_{\mathsf{out}} \cdot \mathrm{poly}(\kappa, \ell_B, d, n)$*.*

In Lemma 5.5, we show that protocol π_{bob} (defined in Figure 4) securely realizes the functionality $\mathcal{F}^f_{\mathsf{sfe}}$ with the required parameters; this, in turn, proves Theorem 5.1. In the secure-erasures setting, we can remove the explainability compilers assumption, and get the following corollary.

Corollary 5.2. *Assume the existence of equivocal, function-hiding, compact LFE schemes for* $\mathsf{P/poly}$ *and let* f *be a two-party function as above. Then,* $\mathcal{F}_{\mathsf{sfe}}^{f}$ *can be UC-realized in the secure-erasures model tolerating a semi-malicious, adaptive PPT adversary by a two-message protocol in the uniform random string model with secure channels. The size of the uniform random string is* $\mathrm{poly}(\kappa, \ell_B, d, n)$*, and the communication complexity and computational complexity of Bob are* $\ell_{\mathit{out}} \cdot \mathrm{poly}(\kappa, \ell_B, d, n)$.

The secure channels can be instantiated over authenticated channels assuming NCE [23, 43, 33, 37]; however, delivering Bob's public key to Alice requires either an additional communication round or a trusted setup.

5.1.1 Equivocal LFE

We start by extending the notion of LFE to support an equivocal mode.

Definition 5.3 (equivocal LFE)**.** *A function-hiding LFE scheme* Π *is* equivocal *if there exists a PPT simulator* $(\mathsf{Sim}^1_{\mathit{EQUIV\text{-}FH}}, \mathsf{Sim}^2_{\mathit{EQUIV\text{-}FH}})$ *for the scheme* Π *such that for all stateful PPT adversary* $\mathcal{A}$*, it holds that*

$$\left|\Pr\left[\mathsf{Expt}_{\Pi,\mathcal{A}}^{\mathsf{EquivFH\text{-}real}}(\kappa) = 1\right] - \Pr\left[\mathsf{Expt}_{\Pi,\mathcal{A}}^{\mathsf{EquivFH\text{-}ideal}}(\kappa) = 1\right]\right| \leq \mathsf{negl}(\kappa),$$

for the experiments $\mathsf{Expt}^{\mathsf{EquivFH\text{-}real}}$ *and* $\mathsf{Expt}^{\mathsf{EquivFH\text{-}ideal}}$ *defined below:*

$\mathsf{Expt}_{\Pi,\mathcal{A}}^{\mathsf{EquivFH\text{-}real}}(\kappa)$	$\mathsf{Expt}_{\Pi,\mathcal{A}}^{\mathsf{EquivFH\text{-}ideal}}(\kappa)$
$\mathsf{params} \leftarrow \mathcal{A}(1^\kappa)$ $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$ $C \leftarrow \mathcal{A}(\mathsf{crs})$ *s.t.* $C \in \mathcal{C}$ *and* $C.\mathsf{params} = \mathsf{params}$ $r \leftarrow \{0,1\}^*$ $\mathsf{digest} = \mathsf{LFE.Compress}(\mathsf{crs}, C; r)$ *Output* $\mathcal{A}(\mathsf{crs}, \mathsf{digest}, r)$	$\mathsf{params} \leftarrow \mathcal{A}(1^\kappa)$ $(\mathsf{crs}, \mathsf{digest}, \mathsf{state}) \leftarrow \mathsf{Sim}^1_{\mathit{EQUIV\text{-}FH}}(1^\kappa, \mathsf{params})$ $C \leftarrow \mathcal{A}(\mathsf{crs})$ *s.t.* $C \in \mathcal{C}$ *and* $C.\mathsf{params} = \mathsf{params}$ $r \leftarrow \mathsf{Sim}^2_{\mathit{EQUIV\text{-}FH}}(C, \mathsf{state})$ *Output* $\mathcal{A}(\mathsf{crs}, \mathsf{digest}, r)$

In the following lemma, we show that the generic construction of function-hiding LFE from standard LFE presented in [86] can be adjusted to provide equivocality.

Lemma 5.4. *Assuming the existence of standard LFE schemes and semi-malicious, adaptively secure, two-round OT, there exists a function-hiding, equivocal LFE scheme.*

Proof (sketch). Quach et al. [86, Sec. 5] showed how to construct a function-hiding LFE scheme given any standard LFE scheme. Recall that in an LFE scheme Alice sends digest_A to Bob, who replies with $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_A, x_B; r_B)$. Instead, Alice and Bob run a semi-malicious, two-message 2PC protocol, where Alice has input digest_A, Bob has input (x_B, r_B), and Alice learns $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_A, x_B; r_B)$. The message of Alice is the new digest and the message of Bob is the new ciphertext.

Concretely, we use the two-round adaptively secure protocol of Benhamouda et al. [13] that requires adaptively secure two-round OT. Note that any two-party two-round protocol π where only Alice learns the output can be adjusted into a two-message protocol π' as follows. Alice's message in π' consists of her first-round message in π. Bob's message in π' consists of both his first-round and second-round messages in π. The security of π' reduces to the security of π by the resiliency to rushing adversaries. Indeed, a corrupted Bob may choose his message in π' (consisting of the first-and second round messages in π) after seeing Alice's first-round message; this translates to an attack that a rushing adversary has the capability to execute in π.

By the security of the adaptively secure protocol, there exists a simulator that can simulate the first message of Alice, and later, given the input value, can provide random coins that explain this message accordingly. □

We note that both LFE [86] and adaptively and maliciously (hence, also semi-maliciously) secure two-round OT [13] can be instantiated assuming adaptive LWE. Hence, also equivocal FH-LFE can be instantiated assuming adaptive LWE.

5.1.2 Semi-Malicious Bob-optimized Protocol

We proceed to our Bob-optimized protocol. Recall that the distribution $D_{\mathsf{lfe}}(\mathsf{params})$ samples a crs for the LFE scheme, computes $(\widetilde{\mathsf{LFE.Enc}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{LFE.Enc})$, and outputs $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$.

Lemma 5.5. *Consider the notations and assumptions in Theorem 5.1. Then, protocol π_{bob}, defined in Figure 4, securely realizes the functionality $\mathcal{F}^f_{\mathsf{sfe}}$ tolerating a semi-malicious, adaptive PPT adversary in the $(\mathcal{F}_{\mathsf{smt}}, \mathcal{F}^{D_{\mathsf{lfe}}(f.\mathsf{params})}_{\mathsf{crs}})$-hybrid model.*

Protocol π_{bob}

- **Common Input:** An LFE scheme and a circuit C_f computing the function f.
- **Notation:** Define the algorithm $\mathsf{LFE.Compress}_{\mathsf{crs},C_f}(x; r)$ by hard-wiring crs and the circuit C_f to the compression algorithm $\mathsf{LFE.Compress}(\mathsf{crs}, C_f(x, \cdot))$, and given input x and randomness r, compress the circuit $C_f(x, \cdot)$ with the input x hard-wired.
- **The Protocol:**

1. Upon receiving $(\mathsf{input}, \mathsf{sid}, x_A)$, Alice samples uniformly at random $r_A \leftarrow \{0,1\}^*$, computes $\mathsf{digest} = \mathsf{LFE.Compress}_{\mathsf{crs},C_f}(x_A; r_A)$, and sends $(\mathsf{sid}, \mathsf{digest})$ to Bob.
2. Upon receiving $(\mathsf{sid}, \mathsf{digest})$ from Alice, and having received $(\mathsf{input}, \mathsf{sid}, x_B)$, Bob computes $\mathsf{ct} \leftarrow \widetilde{\mathsf{LFE.Enc}}(\mathsf{crs}, \mathsf{digest}, x_B)$, and sends $(\mathsf{sid}, \mathsf{ct})$ to Alice.
3. Upon receiving a message $(\mathsf{sid}, \mathsf{ct})$ from Bob, Alice computes $y = \mathsf{LFE.Dec}(\mathsf{crs}, C, r_A, \mathsf{ct})$ and outputs $(\mathsf{output}, \mathsf{sid}, y)$.

Figure 4: Two-round, Bob-optimized protocol with adaptive, semi-malicious security

Proof. Let $\mathcal{A}$ be an adaptive, semi-malicious adversary attacking π_{bob} in the $(\mathcal{F}_{\mathsf{smt}}, \mathcal{F}^{D_{\mathsf{lfe}}}_{\mathsf{crs}})$-hybrid model. We will construct an ideal-process adversary $\mathcal{S}$, interacting with the ideal functionality $\mathcal{F}^f_{\mathsf{sfe}}$ and with ideal (dummy) parties $\tilde{A}$ and $\tilde{B}$, such that no environment can distinguish between $\mathcal{S}$ and $\mathcal{A}$. Let $\mathcal{Z}$ be an environment. Let $\mathsf{Sim}_{\text{LFE}}$ and $\mathsf{Sim}_{\text{EQUIV-FH}}$ be the simulators guaranteed to exist by the security of the equivocal FH-LFE scheme. The simulator $\mathcal{S}$ constructs virtual parties Alice and Bob, and runs the adversary $\mathcal{A}$.

- To simulate the communication with $\mathcal{Z}$, every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape.

- To simulate the common reference string, the simulator $\mathcal{S}$ first computes $(\mathsf{crs}, \mathsf{digest}, \mathsf{state}) \leftarrow \mathsf{Sim}^1_{\text{EQUIV-FH}}(1^\kappa, C_f.\mathsf{params})$ and $(\widetilde{\mathsf{LFE.Enc}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(\mathsf{LFE.Enc})$. Next, $\mathcal{S}$ sets the reference string to be $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ and stores $(\mathsf{digest}, \mathsf{state}, \mathsf{Explain})$.

Since the protocol is defined over secure channels, there is no need to simulate anything as long as both parties are honest. Upon the first corruption of one of the parties, the simulator learns his input and can simulate the internal state as if the party was corrupted from the beginning. We can therefore simplify the proof and consider two cases: the case where Alice is statically corrupted at the beginning and Bob is dynamically corrupted, and the case where Bob is statically corrupted at the beginning and Alice is dynamically corrupted.[11]

Case 1: Alice is corrupted first. The simulator corrupts $\tilde{A}$ and proceeds as follows:

- Upon receiving $(\mathsf{input}, \mathsf{sid}, x_A)$ from the environment, $\mathcal{S}$ invokes $\mathcal{A}$ with $(\mathsf{input}, \mathsf{sid}, x_A)$ and receives back the message $(\mathsf{sid}, \mathsf{digest})$ from $\mathcal{A}$. The simulator reads the random coins r_A that were used by $\mathcal{A}$ from the witness tape of the semi-malicious $\mathcal{A}$.
- Next, $\mathcal{S}$ sends $(\mathsf{input}, \mathsf{sid}, x_A)$ to $\mathcal{F}^f_{\mathsf{sfe}}$ and receives back the output $(\mathsf{output}, \mathsf{sid}, y)$.
- To simulate the second message, $\mathcal{S}$ computes $\mathsf{ct} \leftarrow \mathsf{Sim}_{\text{LFE}}(\mathsf{crs}, C_f(x_A, \cdot), r_A, \mathsf{digest}, y)$ and sends $(\mathsf{sid}, \mathsf{ct})$ to $\mathcal{A}$.
- To explain a corruption request of Bob, the simulator corrupts $\tilde{B}$, learns its input x_B, and computes the random coins as $r_B \leftarrow \mathsf{Explain}((\mathsf{crs}, \mathsf{digest}, x_B), \mathsf{ct})$. (In case Bob was not activated with an input yet, set r_B to be a uniformly random string.)

Proposition 5.6. *Assuming the first corruption request is of Alice, it holds that*

$$\text{REAL}_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}} \overset{c}{\equiv} \text{IDEAL}_{\mathcal{F}^f_{\mathsf{sfe}}, \mathcal{S}, \mathcal{Z}}.$$

Proof. We prove the indistinguishability of the real and ideal worlds by defining a series of hybrid games. The output of each game is the output of the environment.

The game $\text{HYB}^1_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}}$**.** In this game, we modify the experiment $\text{REAL}_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}}$ as follows. Bob first computes $\mathsf{ct} = \widetilde{\mathsf{LFE.Enc}}(\mathsf{digest}, x_B; r)$ followed by $r_B \leftarrow \mathsf{Explain}((\mathsf{crs}, \mathsf{digest}, x_B), \mathsf{ct})$. Next, the originals coins r are ignored, and the new coins r_B are set as the random tape of Bob.

Claim 5.7. $\text{REAL}_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}} \overset{c}{\equiv} \text{HYB}^1_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}}$.

Proof. The claim holds by the security of the explainability compiler. More precisely, given a PPT environment $\mathcal{Z}$ and a polynomial-time non-uniform distinguisher $\mathcal{D}$ that can distinguish between $\text{HYB}^1_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}}$ and $\text{REAL}_{\pi_{\mathsf{bob}}, \mathcal{A}, \mathcal{Z}}$, we construct an attacker $\mathcal{A}'$ to the game $\mathsf{Expt}^{\mathsf{Explain\text{-}Adapt}}_{\mathsf{Comp}, \mathsf{LFE.Enc}, \mathcal{A}'}(\kappa)$ (defined in Appendix A.2.2).

Upon receiving $(1^\kappa, \widetilde{\mathsf{LFE.Enc}})$ from the challenger, the attacker $\mathcal{A}'$ starts by computing $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, C_f.\mathsf{params})$, and sends $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ to the environment. Next, proceed to simulate

[11]This is effectively the compiler from semi-adaptive security to adaptive security of Garay et al. [48].

the dummy honest Bob receiving the input value $(\mathsf{input}, \mathsf{sid}, x_B)$ from $\mathcal{Z}$, and receiving $(\mathsf{sid}, \mathsf{digest})$ from $\mathcal{Z}$ on behalf of the corrupted Alice.

Next, set $x^* = (\mathsf{crs}, \mathsf{digest}, x_B)$ and send x^* to the challenger. Upon receiving back (y^*, r^*) where $y^* = \mathsf{ct}^*$, respond with $(\mathsf{sid}, \mathsf{ct}^*)$ as the second message of Bob. Upon a corruption request of Bob, set his random coins as r^*. Once the environment halts, invoke $\mathcal{D}$ on the output of the environment and output the bit b' returned by $\mathcal{D}$.

Notice that if the challenger chooses the bit $b = 0$, i.e., sets r^* to be the random coins r_0 used to compute $y^* = \widetilde{\mathsf{LFE.Enc}}(x^*; r_0)$, then the view of the environment is identically distributed as in an $\textsc{real}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$, whereas if the challenger chooses the bit $b = 1$, i.e., sets r^* to be the random coins r_1 computed as $r_1 \leftarrow \mathsf{Explain}(x^*, y^*)$, then the view of the environment is identically distributed as in $\textsc{hyb}^1_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$. Therefore, the success probability of $\mathcal{D}$ is the same as of $\mathcal{A}'$, which is negligible by the security of the explainability compiler. □

The game $\textsc{hyb}^2_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$**.** The second hybrid is defined as the first hybrid, but instead of computing $\mathsf{ct} \leftarrow \widetilde{\mathsf{LFE.Enc}}(\mathsf{crs}, \mathsf{digest}, x_B)$, Bob computes $\mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}, x_B)$.

Claim 5.8. $\textsc{hyb}^1_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{s}}{\equiv} \textsc{hyb}^2_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$.

Proof. The proof follows by the statistical functional equivalence of the explainability compiler. □

The game $\textsc{hyb}^3_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$**.** The third hybrid is defined as the second hybrid, but instead of computing $\mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}, x_B)$, Bob computes $\mathsf{ct} \leftarrow \mathsf{Sim}_{\textsc{lfe}}(\mathsf{crs}, C_f(x_A, \cdot), r_A, \mathsf{digest}, y)$ (where r_A are the coins used to compute digest, read from the tape of the semi-malicious $\mathcal{A}$).

Claim 5.9. $\textsc{hyb}^2_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{c}}{\equiv} \textsc{hyb}^3_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$.

Proof. The claim holds by the security of the LFE scheme. More precisely, given a PPT environment $\mathcal{Z}$ and a polynomial-time non-uniform distinguisher $\mathcal{D}$ that can distinguish between $\textsc{hyb}^2_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$ and $\textsc{hyb}^3_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$, we construct an adversary $\mathcal{A}'$ and a distinguisher $\mathcal{D}'$ to the games $\mathsf{Expt}^{\mathsf{LFE\text{-}real}}_{\Pi,\mathcal{A}'}(\kappa)$ and $\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}_{\Pi,\mathcal{A}'}(\kappa)$ (defined in Appendix A.2.1).

The adversary $\mathcal{A}'$, on input 1^κ, starts by sending $f.\mathsf{params}$ to the challenger. Upon receiving crs, the adversary computes $(\widetilde{\mathsf{LFE.Enc}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{LFE.Enc})$ and gives $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ to the environment. Next, $\mathcal{A}'$ proceeds by simulating the dummy honest Bob receiving the input values $(\mathsf{input}, \mathsf{sid}, x_B)$ from $\mathcal{Z}$, and receiving $(\mathsf{sid}, \mathsf{digest})$ from $\mathcal{Z}$ on behalf of the corrupted Alice. When receiving Alice's message, $\mathcal{A}'$ reads the input x_A and random coins r_A from the witness tape of the semi-malicious adversary.

Next, set $x^* = x_B$, set $C_A = C_f(x_A, \cdot)$, and send (x^*, C_A, r_A) to the challenger. Upon receiving back ct from the challenger, respond with $(\mathsf{sid}, \mathsf{ct})$ as the second message of Bob. Upon a corruption request of Bob, set his random coins as $r_B \leftarrow \mathsf{Explain}((\mathsf{crs}, \mathsf{digest}, x_B), \mathsf{ct})$. Once the environment halts, the adversary $\mathcal{A}'$ outputs the output of the environment. When the distinguisher $\mathcal{D}'$ is invoked with the output of $\mathcal{A}'$, it invokes $\mathcal{D}$ and outputs the bit b' returned by $\mathcal{D}$.

Consider the experiment $\mathsf{Expt}^{\mathsf{LFE\text{-}real}}_{\Pi,\mathcal{A}'}(\kappa)$. In this case, the ciphertext ct is computed as $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}, x^*)$ where $\mathsf{digest} = \mathsf{LFE.Compress}(\mathsf{crs}, C_A; r_A)$, exactly as done in $\textsc{hyb}^2_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$; hence, the view of the environment is identically distributed as in $\textsc{hyb}^2_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$. In the experiment $\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}_{\Pi,\mathcal{A}'}(\kappa)$, the ciphertext ct is computed as $\mathsf{ct} \leftarrow \mathsf{Sim}_{\textsc{lfe}}(\mathsf{crs}, C_A, r_A, \mathsf{digest}, y)$ (for

$y = C_A(x_B)$), and the view of the environment is identically distributed as in $\textsc{hyb}^3_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$. Therefore, the success probability of $\mathcal{D}$ is the same as of $\mathcal{D}'$, which is negligible by the security of the LFE scheme. $\square$

The game $\textsc{hyb}^4_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$**.** In the fourth game, we modify the third game $\textsc{hyb}^3_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of computing the common reference string $(\mathsf{crs}, \widetilde{\mathsf{LFE.Enc}})$ using $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, C_f.\mathsf{params})$, it is computed using $(\mathsf{crs}, \mathsf{digest}, \mathsf{state}) \leftarrow \mathsf{Sim}^1_{\text{EQUIV-FH}}(1^\kappa, C_f.\mathsf{params})$.

Claim 5.10. $\textsc{hyb}^3_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \textsc{hyb}^4_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$.

Proof. This follows by the definition of equivocal LFE. $\square$

Claim 5.11. $\textsc{hyb}^4_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \equiv \textsc{ideal}_{f,\mathcal{S},\mathcal{Z}}$.

Proof. This follows since in the experiment $\textsc{hyb}^3$ behavior of Bob is identical to the simulation done by $\mathcal{S}$. In particular, the second message ct is computed via the LFE simulator without knowing the input value x_B, and the random coins are set using $\mathsf{Explain}$. $\square$

Combining Claims 5.7 to 5.10, we get that $\textsc{real}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \textsc{ideal}_{\mathcal{F}^f_{\mathsf{sfe}},\mathcal{S},\mathcal{Z}}$ when the first corruption is of Alice. This concludes the proof of Proposition 5.6. $\square$

Case 2: Bob is corrupted first. The simulator corrupts $\tilde{B}$ and proceeds as follows:

- Upon receiving $(\mathsf{input}, \mathsf{sid}, x_B)$ from the environment, and having received the notification $(\mathsf{input}, \mathsf{sid}, \tilde{A})$ from $\mathcal{F}^f_{\mathsf{sfe}}$, the simulator sends $(\mathsf{sid}, \mathsf{digest})$ to $\mathcal{A}$ (where the simulated digest was computed at the beginning of the simulation).
- Next, $\mathcal{S}$ sends $(\mathsf{input}, \mathsf{sid}, x_B)$ to $\mathcal{F}^f_{\mathsf{sfe}}$ and receives back the output $(\mathsf{output}, \mathsf{sid}, y)$.
- Finally, $\mathcal{S}$ receives the second message $(\mathsf{sid}, \mathsf{ct})$ from $\mathcal{A}$.
- To explain a corruption request of Alice, the simulator corrupts $\tilde{A}$, learn her input x_A, denotes by $C_1 = C_f(x_A, \cdot)$ the circuit C_f with the value x_A hard-wired, and computes the random coins as $r_A \leftarrow \mathsf{Sim}^2_{\text{EQUIV-FH}}(C_1, \mathsf{state})$.

Proposition 5.12. *Assuming the first corruption request is of Bob, it holds that*

$$\textsc{real}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \textsc{ideal}_{\mathcal{F}^f_{\mathsf{sfe}},\mathcal{S},\mathcal{Z}}.$$

Proof. We prove the indistinguishability of the real and ideal worlds by defining a hybrid game. The output of the game is the output of the environment.

The game $\text{HYB}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify the real-model $\widetilde{\text{experiment}}$ $\text{REAL}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of computing the common reference string $(\mathsf{crs}, \mathsf{LFE.Enc})$ using $\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, C_f.\mathsf{params})$, it is computed using $(\mathsf{crs}, \mathsf{digest}, \mathsf{state}) \leftarrow \mathsf{Sim}^1_{\text{EQUIV-FH}}(1^\kappa, C_f.\mathsf{params})$. Instead of computing $\mathsf{digest} \leftarrow \mathsf{LFE.Compress}(\mathsf{crs}, C_1; r_A)$ for $C_1 = C_f(x_A, \cdot)$, Alice uses digest as computed by $\mathsf{Sim}^1_{\text{EQUIV-FH}}$ followed by $r_A \leftarrow \mathsf{Sim}^2_{\text{EQUIV-FH}}(C_1, \mathsf{state})$. Next, the originals coins r are ignored, and the new coins r_A are set as the random tape of Alice.

Claim 5.13. $\text{REAL}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \text{HYB}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$.

Proof. This follows by the security of the equivocal LFE scheme, since any distinguisher between $\text{REAL}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$ and $\text{HYB}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}}$ immediately translates into an adversary violating Definition 5.3. □

Claim 5.14. $\text{HYB}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \equiv \text{IDEAL}_{\mathcal{F}^f_{\mathsf{sfe}},\mathcal{S},\mathcal{Z}}$.

Proof. This follows since in the experiment HYB the behavior of Alice is identical to the simulation done by $\mathcal{S}$. In particular, the first message digest is computed using $\mathsf{Sim}^1_{\text{EQUIV-FH}}$ without knowing the input value x_A, and the random coins are set using $\mathsf{Sim}^2_{\text{EQUIV-FH}}$. □

It follows that $\text{REAL}_{\pi_{\mathsf{bob}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \text{IDEAL}_{\mathcal{F}^f_{\mathsf{sfe}},\mathcal{S},\mathcal{Z}}$ when Bob is corrupted first. This concludes the proof of Proposition 5.12. □

This concludes the proof of Lemma 5.5. □

5.2 Impossibility of Adaptively Secure Alice-Optimized Protocol

We now turn to show that the impossibility of adaptively secure FHE from [76] can be extended to rule out adaptively secure Alice-optimized protocols. In fact, we prove a stronger impossibility showing that for some functions the size of Bob's message cannot be smaller than his input, even if Alice's message and the CRS are long. Intuitively, if the output of the function is simply Bob's input, then clearly Bob's message cannot be compressing. We show that this is the case even if the output is short.

For $n \in \mathbb{N}$, we define the two-party functionality $f_n(x_A, g_B) = (g_B(x_A), \lambda)$,[12] where Alice has input $x_A \in \{0,1\}^{\log n}$, Bob has input a function $g_B : \{0,1\}^{\log n} \to \{0,1\}$, represented by its truth table as an n-bit string, and Alice learns the output $g_B(x_A)$. Intuitively, by adaptively corrupting Alice and equivocating her input, we can essentially recover $g_B(x_A)$ in any choice of x_A from the protocol transcript. This means that the Bob's response must encode the entire truth table of g_B, which is of size n.

Theorem 5.15 (Part 2 of Theorem 1.2, restated)**.** *Let π_n be a two-message protocol in the common reference string model for computing f_n, where Alice sends first the message m_1 and Bob replies with the message m_2. If the protocol tolerates a semi-honest, adaptive adversary in the secure-erasures model, then $|m_2| \geq n$.*

Proof. Assume toward contradiction that $|m_2| < n$. The protocol π_n can be described as follows:

- The common reference string is set as $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\kappa, 1^n)$ according to some distribution.

[12]Recall that λ denotes the empty string.

- Alice generates the first message as $m_1 = \mathsf{Alice}_1(\mathsf{crs}, x_A; r_A)$, and possibly erases some of the coins r_A, leaving r'_A.
- Bob generates the second message as $m_2 = \mathsf{Bob}(\mathsf{crs}, g_B, m_1; r_B)$, and possibly erases r_B.
- Alice computes the output as $y = \mathsf{Alice}_2(\mathsf{crs}, x_A, r'_A, m_2)$.

Consider the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ that operate as follows:

1. The adversary $\mathcal{A}$ corrupts Bob.
2. The environment activates Alice with input x_A.
3. The adversary waits to receive the message m_1, corrupts Alice to learns her (partial) random coins r'_A, and sends $(\mathsf{crs}, m_1, r'_A)$ to the environment.
4. The environment, chooses a random function $g_B \leftarrow \{0,1\}^n$ and random coins $r_B \leftarrow \{0,1\}^*$, runs the code for Bob to generate the message $m_2 = \mathsf{Bob}(\mathsf{crs}, g_B, m_1; r_B)$, and checks whether $g_B(x_A) = \mathsf{Alice}_2(\mathsf{crs}, x_A, r'_A, m_2)$. If so, the environment outputs 1 (real) and otherwise outputs 0 (ideal).

By the assumed security of the protocol π_n, there exists a simulator $\mathcal{S}$, represented by a triplet of algorithms $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$, that can simulate this attack. Initially, the simulator generates the simulated crs by running an algorithm $(\mathsf{crs}, \mathsf{state}_1) \leftarrow \mathcal{S}_1(1^\kappa, 1^n)$. On a corruption request of Bob, the simulator corrupts the ideal (dummy) Bob, and sends crs to $\mathcal{A}$. Next, to simulate the first message $\mathcal{S}$ computes $(m_1, \mathsf{state}_2) \leftarrow \mathcal{S}_2(\mathsf{state}_1)$ and sends m_1 to $\mathcal{A}$. Finally, upon a corruption request of Alice, the simulator corrupts the ideal (dummy) Alice, learns her input x_A, and computes $r'_A \leftarrow \mathcal{S}_3(x_A, \mathsf{state}_2)$.

Given a random function $g : \{0,1\}^{\log n} \to \{0,1\}$, we will use the simulator $\mathcal{S}$ to construct a circuit $C^*(x)$ for computing the function g. We start by computing $(\mathsf{crs}, \mathsf{state}_1) \leftarrow \mathcal{S}_1(1^\kappa, 1^n)$ and $(m_1, \mathsf{state}_2) \leftarrow \mathcal{S}_2(\mathsf{state}_1)$ just like $\mathcal{S}$. Note that this part of the simulation is independent of the parties' inputs. Next, run the code of Bob on input g to compute $m_2 = \mathsf{Bob}(\mathsf{crs}, g, m_1; r_B)$ for a uniformly distributed $r_B \leftarrow \{0,1\}^*$. Finally, sample random coins $r \leftarrow \{0,1\}^*$ and hardwire the value $(\mathsf{crs}, \mathsf{state}_2, m_2, r)$ to C^*. On input x, the resulting circuit $C^*_{\mathsf{crs},\mathsf{state}_2,m_2,r}(x)$ first computes $r'_A = \mathcal{S}_3(x, \mathsf{state}_2; r)$ and later $y = \mathsf{Alice}_2(\mathsf{crs}, x, r'_A, m_2)$.

By the security of the protocol π_n, it holds that with overwhelming probability, it is possible to recover the string g by running $C^*_{\mathsf{crs},\mathsf{state}_2,m_2,r}(x)$ on every $x \in \{0,1\}^{\log n}$. However, the only component in $C^*_{\mathsf{crs},\mathsf{state}_2,m_2,r}$ that depends on g is m_2. Denote by G the random variable representing the function g, that takes a value uniformly at random in $\{0,1\}^n$, and by $\tilde{C}$ the random variable representing the circuit $C^*_{\mathsf{crs},\mathsf{state}_2,m_2,r}$ generated by the algorithm described above. Then, the entropy of G conditioned on $\tilde{C}$ is

$$H(G \mid \tilde{C}) = n - |m_2| > 0.$$

Combined together, we derive a contradiction. □

6 Adaptive Corruptions of All-But-One of the Parties

In this section, we prove an analog result in the adaptive setting to the result of Asharov et al. [5], who showed how to compute any function tolerating all-but-one corruptions using a two-round protocol in the threshold-PKI model assuming threshold FHE, which in turn can be instantiated using LWE. Our construction relies on *threshold equivocal FHE* (to be defined in Section 6.1.1) that allows simulating ciphertexts for honest parties and explaining them properly upon later corruptions.

We note that the simulation technique used in [5] (and similarly in [80]) does not translate to the adaptive setting. As observed in [5, 80], the threshold decryption protocol may leak some information about the shares of the secret key, and the simulator for the decryption protocol can be used to protect *exactly* one party. Since [5, 80] considered static corruptions, the set of corrupted parties was known ahead of time, and the simulator could choose one of the honest parties P_h as a special party for the simulation. The decryption protocol was simulated with respect to P_h, as if he is the only honest party. For this reason, proving security of *exactly* $n-1$ corruptions in [80] was considerably simpler than proving security of *up to* $n-1$ corruptions.[13]

The simulation strategy that was used in [5, 80] does not translate to the adaptive setting, since the party P_h that is chosen by the simulator may get corrupted after simulating the decryption protocol. The simulator cannot know in advance which party will be the last to remain honest. For this reason, we use a different simulation strategy, which allows the simulator to "correct" his choice of the party that is simulated as honest for the decryption protocol. Technically, this is done by having each party send shares of zero to each other party over a secure channel (that can be instantiated via NCE). These shares are used to hide the partial decryptions without changing their value. Since shares exchanged between pairs of honest parties remain hidden from the eyes of the adversary, the simulator has more freedom to replace the special party P_h upon corruption, by another honest party, even after simulating the decryption protocol.

6.1 Cryptographic Primitives used in the Protocol

Initially, we define equivocal FHE and show how to instantiate it assuming LWE. Next, we define threshold equivocal FHE.

6.1.1 Equivocal FHE

An equivocal FHE is an FHE scheme that is augmented with the capability to generate a public key in an "equivocal mode," allowing to explain any ciphertext as an encryption of any value.

Definition 6.1. *An equivocal fully homomorphic encryption (EFHE) scheme is a six-tuple of algorithms* ($\mathsf{EFHE.Gen}$, $\mathsf{EFHE.Enc}$, $\mathsf{EFHE.Eval}$, $\mathsf{EFHE.Dec}$, $\mathsf{EFHE.GenEquiv}$, $\mathsf{EFHE.Equiv}$) *satisfying the following properties:*

- ($\mathsf{EFHE.Gen}, \mathsf{EFHE.Enc}, \mathsf{EFHE.Eval}, \mathsf{EFHE.Dec}$) *is an FHE scheme.*
- $\mathsf{EFHE.GenEquiv}(1^\kappa, 1^d) \to (\mathsf{pk}, \mathsf{td})$*: on input the security parameter κ and a depth bound d, the equivocal key-generation algorithm outputs a public-key* pk *and a trapdoor* td*.*

[13] We note that the same problem arises also in the threshold FHE scheme for more general access structures [17, Def. 5.5], where the simulation is defined only for *maximal* invalid party sets.

- $\mathsf{EFHE.Equiv}(\mathsf{td},\mathsf{ct},m) \to r$: *on input a trapdoor* td, *a ciphertext* ct, *and a plaintext* m, *the equivocation algorithm outputs a value* r *in the randomness space.*

We require the following properties:

1. $(\mathsf{EFHE.Gen},\mathsf{EFHE.Enc},\mathsf{EFHE.Eval},\mathsf{EFHE.Dec})$ *is a correct, compact, and semantically secure FHE scheme (see Appendix A.2.6).*

2. ***Indistinguishability of equivocal keys.*** *For every depth bound* d, *the following distributions are computationally indistinguishable:*
   - *The honestly generated public key* $\{\mathsf{pk} \mid (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{EFHE.Gen}(1^\kappa,1^d)\}_\kappa$
   - *The equivocal public key* $\{\mathsf{pk} \mid (\mathsf{pk},\mathsf{td}) \leftarrow \mathsf{EFHE.GenEquiv}(1^\kappa,1^d)\}_\kappa$.

3. ***Indistinguishability of equivocated randomness.*** *For every depth bound* d *and for every message* $\mu \in \{0,1\}$, *the following distributions are computationally indistinguishable:*
   - *The public key, a ciphertext encrypting* μ, *the plaintext* μ, *and the random coins used in an honest encryption*
   $$\left\{(\mathsf{pk},\mathsf{ct},\mu,r) \mid (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{EFHE.Gen}(1^\kappa,1^d), \mathsf{ct} = \mathsf{EFHE.Enc}(\mathsf{pk},\mu;r)\right\}_\kappa .$$
   - *The equivocal public key, an encryption of zero, the plaintext* μ, *and the computed random coins generated by the equivocation algorithm*
   $$\left\{(\mathsf{pk},\mathsf{ct},\mu,\mathsf{EFHE.Equiv}(\mathsf{td},\mathsf{ct},\mu)) \mid (\mathsf{pk},\mathsf{td}) \leftarrow \mathsf{EFHE.GenEquiv}(1^\kappa,1^d), \mathsf{ct} \leftarrow \mathsf{EFHE.Enc}_{\mathsf{pk}}(0)\right\}_\kappa .$$

Constructing Equivocal FHE. We next show that equivocal FHE schemes exist assuming LWE. We will prove this statement using an extended, "dual-mode" version of homomorphic trapdoor functions from [59]. Recall that HTDF, were defined in [59] in a "meaningless" (or "equivocal") mode (see also Appendix A.2.3). That is, by computing $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa,1^d)$ it is possible to "commit" to $x \in \{0,1\}$ by computing $v = f_{\mathsf{pk},x}(u)$ for a random $u \in \mathcal{U}$. Equivalently, it is possible to sample a random element $v \leftarrow \mathcal{V}$ and "explain" it to every $x \in \{0,1\}$ by computing $u \leftarrow \mathsf{HTDF.Inv}_{\mathsf{sk},x}(v)$. To avoid confusion, we will denote in the remaining of this section the key-generation algorithm in this mode by $\mathsf{HTDF.GenEquiv}$. Note that using the equivocal mode, the commitment to x is *statistically hiding.*

Gorbunov et al. [59] also defined a "meaningful" (or "extractable") mode of homomorphic trapdoor functions. In this mode, the keys are generated via a different algorithm $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.GenExtract}(1^\kappa,1^d)$. The public key can again be used to "commit" to $x \in \{0,1\}$ by computing $v = f_{\mathsf{pk},x}(u)$ for a random $u \in \mathcal{U}$. However, the secret key is now used to extract the bit x from v, i.e., $x \leftarrow \mathsf{HTDF.Extract}_{\mathsf{sk}}(v)$. Note that using the extractable mode, the commitment to x is *statistically binding* and *computationally hiding.*

Gorbunov et al. [59] constructed a dual-mode HTDF scheme and proved that under the LWE assumption, the following properties are satisfied:

1. $(\mathsf{HTDF.GenEquiv}, f, \mathsf{HTDF.Inv}, \mathsf{HTDF.Eval}^{\mathsf{in}}, \mathsf{HTDF.Eval}^{\mathsf{out}})$ is an HTDF scheme according to Definition A.8.

2. For any depth-bound d and every $x \in \{0,1\}$ it holds that

$$\Pr\left[\ \mathsf{HTDF.Extract}_{\mathsf{sk}}(v) \neq x \ \middle|\ \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.GenExtract}(1^\kappa, 1^d)\\ v = f_{\mathsf{pk},x}(u) \text{ for } u \leftarrow \mathcal{U}\end{array}\right] \leq \mathsf{negl}(\kappa).$$

3. The public key in the equivocal mode is computationally indistinguishable from the public key in the extractable mode:

$$\left\{\mathsf{pk} \mid (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.GenEquiv}(1^\kappa, 1^d)\right\}_\kappa \overset{\mathrm{c}}{\equiv} \left\{\mathsf{pk} \mid (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.GenExtract}(1^\kappa, 1^d)\right\}_\kappa.$$

We can now proceed to construct an equivocal FHE from dual-mode HTDF. The extractable mode of the HTDF scheme will be used to define the "standard FHE," since extracting the bit x from v corresponds to decrypting a ciphertext (indeed, the extractable mode in [59] corresponds to the GSW FHE scheme [56]). The equivocal mode of the HTDF scheme will be used to define the equivocal mode of the FHE scheme, as it allows explaining a ciphertext to every bit. Formally:

- $\mathsf{EFHE.Gen}(1^\kappa, 1^d)$: run $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.GenExtract}(1^\kappa, 1^d)$ and return $(\mathsf{pk},\mathsf{sk})$.
- $\mathsf{EFHE.Enc}(\mathsf{pk}, x; r)$: parse r as an element $u \in \mathcal{U}$, compute $v = f_{\mathsf{pk},x}(u)$, and return v as the ciphertext.
- $\mathsf{EFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: parse each ciphertext ct_i as an element $v_i \in \mathcal{V}$, compute $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(C, v_1, \ldots, v_\ell)$, and return v^* as the evaluated ciphertext.
- $\mathsf{EFHE.Dec}(\mathsf{sk}, \mathsf{ct})$: parse ct as an element $v \in \mathcal{V}$, compute $x \leftarrow \mathsf{HTDF.Extract}_{\mathsf{sk}}(v)$, and return x as the plaintext.
- $\mathsf{EFHE.GenEquiv}(1^\kappa, 1^d)$: run $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{HTDF.GenEquiv}(1^\kappa, 1^d)$ and return $(\mathsf{pk},\mathsf{sk})$, i.e., sk acts as the trapdoor td.
- $\mathsf{EFHE.Equiv}(\mathsf{td}, \mathsf{ct}, x)$: parse ct as an element $v \in \mathcal{V}$ and td as sk, compute $u \leftarrow \mathsf{HTDF.Inv}_{\mathsf{sk},x}(v)$, and return u as the random coins.

The security of the equivocal FHE scheme follows immediately from the security of the dual-mode HTDF scheme. This proves the following lemma.

Lemma 6.2. *Assuming LWE there exist equivocal FHE schemes.*

Proof. As proven above, equivocal FHE schemes can be constructed from dual-mode HTDF schemes, and the security of the EFHE scheme follows immediately from the security of the dual-mode HTDF scheme. The lemma follows as dual-mode HTDF schemes exists under the LWE assumption [59]. For completeness, we describe the construction in Appendix C. □

6.1.2 Threshold Equivocal FHE

In a threshold FHE scheme, the key-generation and the decryption algorithms are in fact n-party protocols. We consider the simplest case of n-out-of-n threshold FHE and require a single-round decryption protocol (following [5, 60, 80, 46]). We note that threshold FHE for more general access structures are also known assuming LWE [17]. A threshold equivocal FHE scheme (TEFHE) is a threshold FHE scheme that admits an equivocal mode as in Section 6.1.1.

Definition 6.3 (TEFHE). *A threshold equivocal fully homomorphic encryption (TEFHE) is a seven-tuple of algorithms* ($\mathsf{TEFHE.Gen}$, $\mathsf{TEFHE.Enc}$, $\mathsf{TEFHE.Eval}$, $\mathsf{TEFHE.PartDec}$, $\mathsf{TEFHE.FinDec}$, $\mathsf{TEFHE.GenEquiv}$, $\mathsf{TEFHE.Equiv}$) *satisfying the following properties:*

- $\mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$*: on input the security parameter κ, a depth bound d, and the number of parties n, the key-generation algorithm outputs a public key* pk *and n secret key shares* $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$.

- $\mathsf{TEFHE.Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$*: on input a public key* pk *and a plaintext $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext* ct.

- $\mathsf{TEFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \to \mathsf{ct}$*: on input a public key* pk*, a circuit $C : \{0,1\}^\ell \to \{0,1\}$, and a tuple of ciphertexts* $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$*, the homomorphic-evaluation algorithm outputs a ciphertext* ct.

- $\mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct}) \to \mathsf{p}_i$*: on input $i \in [n]$, a secret key share* sk_i *and a ciphertext* ct*, the partial-decryption algorithm outputs a partial decryption* p_i.

- $\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}) \to \tilde{\mu}$*: on input a public key* pk *and a set* $\{\mathsf{p}_i\}_{i \in [n]}$*, the final-decryption algorithm outputs $\tilde{\mu} \in \{0, 1, \perp\}$.*

- $\mathsf{TEFHE.GenEquiv}(1^\kappa, 1^d) \to (\mathsf{pk}, \mathsf{td})$*: on input the security parameter κ and a depth bound d, the equivocal key-generation algorithm outputs a public-key* pk *and a trapdoor* td.

- $\mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}, m) \to r$*: on input a trapdoor* td*, a ciphertext* ct*, and a plaintext m, the equivocation algorithm outputs a value r in the randomness space.*

In the protocol, we will require some additional properties regarding the key-generation and threshold-decryption protocols.

Definition 6.4 (special TEFHE). *A special TEFHE is a TEFHE scheme satisfying the following properties:*

1. *On input 1^κ, 1^d, and 1^n, the key-generation algorithm* $\mathsf{TEFHE.Gen}$ *outputs* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ *where the public key* pk *defines a prime number q, and each secret key* sk_i *of the form* $\mathsf{sk}_i = (-\boldsymbol{s}_i, 1)$ *where $\boldsymbol{s}_i$ is uniformly distributed in $\mathbb{Z}_q^{n'-1}$ for some $n' = \mathrm{poly}(\kappa, d)$.*

2. *The partial-decryption algorithm* $\mathsf{p}_i \leftarrow \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$ *operates by computing* $\mathsf{p}_i = \langle \mathsf{ct}, \mathsf{sk}_i \rangle + e \mod q$.

3. *For every $v_1, \ldots, v_n \in \mathbb{Z}_q$, the final-decryption algorithm* $\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\})$ *satisfies the following* linearity *property*

$$\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1 + v_1, \ldots, \mathsf{p}_n + v_n\}) = \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}) + \sum_{i \in [n]} v_i.$$

We require the following properties from a special TEFHE scheme Π:

1. The FHE scheme that is defined by setting the decryption key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and the decryption algorithm is composed of executing $\mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$ for every $i \in [n]$ followed by $\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\})$ is a correct, compact, and secure equivocal FHE scheme for circuits of depth d (according to Definition 6.1, see also Appendix A.2.6).

2. Simulatability of partial decryption: there exists a PPT simulator $\mathsf{Sim}_{\text{TEFHE}}$ such that for integers n, d, and ℓ, and every circuit $C : \{0,1\}^\ell \to \{0,1\}$ of depth d, every $h \in [n]$, and every $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ in the support of $\mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$, the following distributions are statistically close:

$\mathsf{Expt}^{\mathsf{TEFHE\text{-}real}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$	$\mathsf{Expt}^{\mathsf{TEFHE\text{-}ideal}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$
Send $(h, \mathsf{pk}, \{\mathsf{sk}_j\}_{j\neq h})$ to $\mathcal{A}$ $\mathcal{A}$ sends $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ and $r_1, \ldots, r_\ell \in \{0,1\}^*$ $\forall j \in [\ell]$ compute $\mathsf{ct}_j = \mathsf{TEFHE.Enc}(\mathsf{pk}, \mu_j; r_j)$ Compute $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ Compute $\mathsf{p}_h \leftarrow \mathsf{TEFHE.PartDec}(h, \mathsf{sk}_h, \mathsf{ct})$ Send p_h to $\mathcal{A}$ Output whatever $\mathcal{A}$ outputs	Send $(h, \mathsf{pk}, \{\mathsf{sk}_j\}_{j\neq h})$ to $\mathcal{A}$ $\mathcal{A}$ sends $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ and $r_1, \ldots, r_\ell \in \{0,1\}^*$ $\forall j \in [\ell]$ compute $\mathsf{ct}_j = \mathsf{TEFHE.Enc}(\mathsf{pk}, \mu_j; r_j)$ Compute $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ Compute $\mu = C(\mu_1, \ldots, \mu_\ell)$ Compute $\mathsf{p}_h \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h, \mathsf{ct}, \mu, \{\mathsf{sk}_j\}_{j\neq h})$ Send p_h to $\mathcal{A}$ Output whatever $\mathcal{A}$ outputs

Lemma 6.5. *Assuming LWE there exist special TEFHE schemes.*

The lemma follows from Lemma 6.2 using standard techniques for constructing threshold FHE [5, 80]. For completeness, we describe the construction in Appendix C.

6.1.3 The Threshold-PKI Functionality

We define the protocol assuming a trusted pre-process phase where the keys of the TEFHE scheme are generated and distributed to the parties. We capture this assumption with the threshold-PKI ideal functionality (Figure 5).

Functionality $\mathcal{F}_{\mathsf{thresh\text{-}pki}}(\Pi, d)$

$\mathcal{F}_{\mathsf{thresh\text{-}pki}}$ proceeds as follows, interacting with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$, and parameterized by a TEFHE scheme Π and a depth bound d.

- Upon receiving a message $(\mathsf{init}, \mathsf{sid})$ from party P_i, do:
  1. If there is no value $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ recorded, then compute $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$, record it, and send (sid, P_i) to $\mathcal{S}$.
  2. Send $(\mathsf{sid}, P_i, \mathsf{pk})$ to $\mathcal{S}$ and a delayed output $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk}_i)$ to P_i.

Figure 5: The threshold-PKI functionality

6.2 Semi-Malicious Security

Theorem 6.6. *Assume that special TEFHE exists, let $t < n$, and let $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ be an efficiently computable function of depth d. Then, Protocol $\pi_{\mathsf{allbutone}}$, defined in Figure 6, UC-realizes $\mathcal{F}^f_{\mathsf{sfe\text{-}abort}}$ in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{smt}})$-hybrid model, tolerating an adaptive, semi-malicious, PPT t-adversary, by a two-round protocol with communication complexity* $\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$.

Proof. We start by showing correctness of a non-aborting execution. In this case, for every $i \in [n]$ it holds that $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i)$ and $\sum_{j=1}^n \mathsf{s}_i^j = 0 \bmod q$; in addition, $\mathsf{ct} =$

$\mathsf{TEFHE.Eval}(\mathsf{pk}, C_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$. Next, each party computes $\mathsf{p}_i = \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$ and sends $\mathsf{m}_i = \mathsf{p}_i + \sum_{j=1}^n \mathsf{s}_j^i \mod q$. By the properties of special TEFHE, it holds that

$$\begin{aligned}
\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{m}_1, \ldots, \mathsf{m}_n\}) &= \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1 + \sum_{j=1}^n \mathsf{s}_j^1, \ldots, \mathsf{p}_n + \sum_{j=1}^n \mathsf{s}_j^n\}) \\
&= \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}) + \sum_{j=1}^n \mathsf{s}_j^1 + \ldots + \sum_{j=1}^n \mathsf{s}_j^n \\
&= \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}) + \sum_{j=1}^n \mathsf{s}_1^j + \ldots + \sum_{j=1}^n \mathsf{s}_n^j \\
&= \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}) + 0 + \ldots + 0 \mod q \\
&= C(x_1, \ldots, x_n).
\end{aligned}$$

Protocol $\pi_{\mathsf{allbutone}}$

- **Private Input:** Every party P_i, for $i \in [n]$, has private input $x_i \in \{0,1\}^{\ell_{\mathsf{in}}}$.
- **Common Input:** A special TEFHE scheme Π and a circuit C_f of depth d.
- **The Protocol:**

1. Upon receiving $(\mathsf{input}, \mathsf{sid}, x_i)$, party P_i proceeds as follows:
   (a) Invoke $\mathcal{F}_{\mathsf{thresh\text{-}pki}}(\Pi, d)$ with $(\mathsf{init}, \mathsf{sid})$ to receive $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk}_i)$. Let q be the prime associated with the public key pk (as per Definition 6.4).
   (b) Encrypt the input as $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i)$.
   (c) Sample uniformly distributed random values $\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n \leftarrow \mathbb{Z}_q$, conditioned on $\sum_{j=1}^n \mathsf{s}_i^j = 0 \mod q$.
   (d) Send $(\mathsf{sid}, \mathsf{ct}_i, \mathsf{s}_i^j)$ to P_j over a secure channel (via $\mathcal{F}_{\mathsf{smt}}$).
2. In case some party aborts, output $(\mathsf{output}, \mathsf{sid}, \bot)$ and halt. Otherwise, upon receiving $(\mathsf{sid}, \cdot)$ messages from all the parties, party P_i proceeds as follows:
   (a) Compute $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$.
   (b) Partially decrypt the result as $\mathsf{p}_i = \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$.
   (c) Set $\mathsf{m}_i = \mathsf{p}_i + \sum_{j=1}^n \mathsf{s}_j^i \mod q$ and send $(\mathsf{sid}, \mathsf{m}_i)$ to every party.
3. In case some party aborts, output $(\mathsf{output}, \mathsf{sid}, \bot)$ and halt. Otherwise, upon receiving $(\mathsf{sid}, \cdot)$ from all the parties, party P_i runs the final decrypt as $y = \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{m}_1, \ldots, \mathsf{m}_n\})$ and outputs $(\mathsf{output}, \mathsf{sid}, y)$.

Figure 6: Two-round MPC with semi-malicious security in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{smt}})$-hybrid model

We proceed to prove security of the protocol. Let $\mathcal{A}$ be an adaptive, semi-malicious adversary attacking $\pi_{\mathsf{allbutone}}$ in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{smt}})$-hybrid model. We will construct an ideal-process adversary $\mathcal{S}$, interacting with the ideal functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^f$ and with ideal (dummy) parties $\widetilde{P}_1, \ldots, \widetilde{P}_n$, such that no environment can distinguish between $\mathcal{S}$ and $\mathcal{A}$. The simulator $\mathcal{S}$ constructs virtual parties $P_1, \ldots, P_n$, and runs the adversary $\mathcal{A}$. Let $\mathsf{Sim}_{\text{TEFHE}}$ be the simulator that is guaranteed to exist for the TEFHE scheme Π (as per Definition 6.3).

Note that the internal state of a party P_i in the protocol consists of its input x_i, the message $(\mathsf{pk}, \mathsf{sk}_i)$ from $\mathcal{F}_{\mathsf{thresh\text{-}pki}}$, the random coins that include coins r_i^E for encrypting the input in the first round, coins r_i^D for partial decryption of the evaluated ciphertext for the second round, and the secret shares of zero $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n)$.[14] In addition, the internal state contains the messages P_i received in the first round, including the values $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ and $(\mathsf{s}_1^i, \ldots, \mathsf{s}_n^i)$, and in the second round $(\mathsf{m}_1, \ldots, \mathsf{m}_n)$.

Simulating the communication $\mathcal{Z}$: Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape.

Simulating the threshold-PKI functionality $\mathcal{F}_{\mathsf{thresh\text{-}pki}}$: The simulator $\mathcal{S}$ computes $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{TEFHE.GenEquiv}(1^\kappa, 1^d)$, for every $i \in [n]$ samples $\mathsf{sk}_i = (-\boldsymbol{s}_i, 1))$ with $\boldsymbol{s}_i \leftarrow \mathbb{Z}_q^{n'-1}$ (as per Definition 6.4), and gives $(\mathsf{pk}, \mathsf{sk}_i)$ to $\mathcal{A}$ for every corrupted party P_i.

Simulating the first round: For every honest party P_i, compute $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$, sample a secret sharing of zero $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n)$ (i.e., $\sum_j \mathsf{s}_i^j = 0 \mod q$), and send $(\mathsf{sid}, \mathsf{ct}_i, \mathsf{s}_i^j)$ to every corrupted P_j. Next, the simulator receives from $\mathcal{A}$ the message $(\mathsf{sid}, \mathsf{ct}_j, \mathsf{s}_j^i)$ on behalf of every corrupted party P_j and every honest party P_i, and reads from the special witness tape of the semi-malicious adversary $\mathcal{A}$ the input and random coins (x_i', r_i).

Sending input to the functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^f$: In case the adversary aborted in the first round, send $(\mathsf{abort}, \mathsf{sid})$ to the ideal functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^f$. Otherwise, for every corrupted party P_i, send the input value x_i' to the ideal functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^f$ and receive back the output y. The simulator does not release the output to the ideal (dummy) honest parties yet.

Simulating the second round: The simulator chooses an arbitrary honest party P_{h_1}. Let $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ be the ciphertexts in the first round of the simulation, $\mathcal{S}$ homomorphically evaluates the circuit as $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, computes for every honest party P_i (except for P_{h_1}) the partial decryption $\mathsf{p}_i = \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct}; r_i^D)$, for uniformly random r_i^D, and for P_{h_1} the simulator computes the simulated partial decryption $\mathsf{p}_{h_1} \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h_1, \mathsf{ct}, y, \{\mathsf{sk}_j\}_{j \neq h_1})$.

Next, for each honest party P_i (including P_{h_1}), let $(\mathsf{s}_1^i, \ldots, \mathsf{s}_n^i)$ be the shares sent to P_i from every corrupted party in the first round (by $\mathcal{A}$), and were sampled to be sent to P_i from every other honest party during the simulation (by $\mathcal{S}$). The simulator $\mathcal{S}$ sets $\mathsf{m}_i = \mathsf{p}_i + \sum_j \mathsf{s}_j^i \mod q$, gives $(\mathsf{sid}, \mathsf{m}_i)$ to $\mathcal{A}$, and receives messages from $\mathcal{A}$ on behalf of corrupted parties.

In case the adversary aborts in the second round, send $(\mathsf{abort}, \mathsf{sid})$ to the ideal functionality $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^f$. Otherwise, release the output values to the ideal honest parties.

Simulating corruption requests of party P_i: The simulator $\mathcal{S}$ corrupts the dummy party $\widetilde{P}_i$ and continues as follows, depending on the timing of the corruption:

- **Corruptions before the first round:** In this case $\widetilde{P}_i$ has not been activated with input yet. $\mathcal{S}$ sets the contents of P_i's random tape with uniformly random coins.

[14]The coin that determine $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n)$ can be, for example, an encoding of the first $n-1$ elements $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^{n-1})$ that fully determine s_i^n.

- **Corruptions after the first round and before the second round:** In this case $\widetilde{P}_i$ has been activated with input $(\mathsf{input}, \mathsf{sid}, x_i)$, but did not receive output yet. $\mathcal{S}$ sets the contents of P_i's input tape to $(\mathsf{input}, \mathsf{sid}, x_i)$ and the contents of the random tape to $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$ for the ciphertext ct_i that was used to simulate P_i's message, and to a uniformly random r_i^D (representing the random coins for the partial-decryption algorithm).

  Finally, the secret shares of zero $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n)$ have already been set for P_i during the simulation. Similarly, the messages received in the first round, i.e., $(\mathsf{s}_1^i, \ldots, \mathsf{s}_n^i)$ are set according to the simulation both for corrupted parties (as received from $\mathcal{A}$) and for honest parties (as simulated by $\mathcal{S}$).

- **Corruptions after the second round:** In this case $\widetilde{P}_i$ has received the output. $\mathcal{S}$ computes $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$ as in the previous case, but in order to set the random coins of the partial decryption, $\mathcal{S}$ proceeds as follows. Let h_k be the index of the recent party for which the decryption share was simulated (i.e., h_k with the largest k).

  - If $i \neq h_k$, i.e., the corrupted party is *not* the party that was used to simulate the threshold decryption, then set the decryption coins r_i^D as the coins used in the simulation of the second round. Set the shares of zero $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n)$ and the incoming messages $(\mathsf{s}_1^i, \ldots, \mathsf{s}_n^i)$ according to the values set in the simulation.
  - If $i = h_k$, then sample uniformly random $r_{h_k}^D$ and compute the partial decryption $\mathsf{p}_{h_k} = \mathsf{TEFHE.PartDec}(h_k, \mathsf{sk}_{h_k}, \mathsf{ct}; r_{h_k}^D)$. Next, choose an arbitrary honest party $P_{h_{k+1}}$ and simulate the partial decryption as $\mathsf{p}_{h_{k+1}} \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h_{k+1}, \mathsf{ct}, y, \{\mathsf{sk}_j\}_{j \neq h_{k+1}})$. Finally, the simulator needs to "adjust" the values for P_{h_k} and $P_{h_{k+1}}$. In particular, it must hold that:
    1. $\mathsf{m}_{h_k} = \mathsf{p}_{h_k} + \sum_j \mathsf{s}_j^{h_k} \mod q$.
    2. $\sum_j \mathsf{s}_{h_k}^j = 0 \mod q$.
    3. $\mathsf{m}_{h_{k+1}} = \mathsf{p}_{h_{k+1}} + \sum_j \mathsf{s}_j^{h_{k+1}} \mod q$.
    4. $\sum_j \mathsf{s}_{h_{k+1}}^j = 0 \mod q$.

    The free variables the simulator can use (since they have not been released to the adversary yet) are $\mathsf{s}_{h_{k+1}}^{h_{k+1}}$, $\mathsf{s}_{h_k}^{h_k}$, $\mathsf{s}_{h_k}^{h_{k+1}}$, and $\mathsf{s}_{h_{k+1}}^{h_k}$. The simulator proceeds to solve these 4 equations with respect to the 4 variables mentioned above.

We now turn to show that no environment can distinguish between the simulation of $\mathcal{S}$ with $\mathcal{F}_{\mathsf{sfe\text{-}abort}}^f$ and the execution of $\pi_{\mathsf{allbutone}}$ with $\mathcal{A}$ by defining a series of hybrid games. The output of each game is the output of the environment. Let $\mathcal{Z}$ be a PPT environment.

The game $\text{HYB}^1_{\pi_{\mathsf{allbutone}}, \mathcal{A}, \mathcal{Z}}$. In this game, we modify the real-model experiment $\text{REAL}_{\pi_{\mathsf{allbutone}}, \mathcal{A}, \mathcal{Z}}$ as follows. In the second round, an arbitrary honest party P_{h_1} (e.g., the honest party P_i with smallest index i) computes a simulated decryption share as $\mathsf{p}_{h_1} \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h_1, \mathsf{ct}, y, \{\mathsf{sk}_j\}_{j \neq h_1})$.

Upon a corruption request of some party P_i, let h_k be the index of the recent party for which the decryption share was simulated (i.e., h_k with the largest k). If $i = h_k$, then proceed to explain the state of P_{h_k} in the same way as done in the simulation, by computing $\mathsf{p}_{h_k} = \mathsf{TEFHE.PartDec}(h_k, \mathsf{sk}_{h_k}, \mathsf{ct}; r_{h_k}^D)$, choosing another honest party $P_{h_{k+1}}$ and simulating its partial decryption as $\mathsf{p}_{h_{k+1}} \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h_{k+1}, \mathsf{ct}, y, \{\mathsf{sk}_j\}_{j \neq h_{k+1}})$. Finally, adjust $\mathsf{s}_{h_{k+1}}^{h_{k+1}}$, $\mathsf{s}_{h_k}^{h_k}$, $\mathsf{s}_{h_k}^{h_{k+1}}$, and $\mathsf{s}_{h_{k+1}}^{h_k}$ by solving the four equations as done in the simulation.

Claim 6.7. $\text{REAL}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}} \overset{\text{s}}{\equiv} \text{HYB}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$.

Proof. This follows by the security of the threshold-decryption protocol of the TEFHE scheme. Consider an adversary $\mathcal{A}$, and environment $\mathcal{Z}$, and a distinguisher $\mathcal{D}$ that can distinguish between $\text{HYB}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ and $\text{REAL}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$. Consider a mental experiment where a challenger computes $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$ and emulates the honest parties running the protocol with $\mathcal{A}$ and $\mathcal{Z}$ (running using some random coins). Denote by $h \in [n]$ an index of a party that remained honest when $\mathcal{Z}$ produces output.

Now, consider an adversary $\mathcal{A}'$ for the *simulatability of partial decryption* of the TEFHE scheme. $\mathcal{A}'$ is running either with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}real}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$ or with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}ideal}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$; initially, $\mathcal{A}'$ receives $(h, \mathsf{pk}, \{\mathsf{sk}_j\}_{j\neq h})$. It uses the same coins as the challenger above to emulate the protocol toward $\mathcal{A}$ and $\mathcal{Z}$, with the difference that $\mathcal{A}'$ does not sample the threshold-PKI keys, but uses pk and $\{\mathsf{sk}_j\}_{j\neq h}$. After the first round, $\mathcal{A}'$ responds with the inputs values $\mu_1, \ldots, \mu_\ell$ and $r_1, \ldots, r_\ell$ used in the execution and receives $\tilde{\mathsf{p}}_h$. Next, instead of simulating the decryption share of P_h as $\mathsf{TEFHE.PartDec}(h, \mathsf{sk}_h, \mathsf{ct})$, it uses the value $\tilde{\mathsf{p}}_h$. Finally, $\mathcal{A}'$ invokes $\mathcal{D}$ on the output of $\mathcal{Z}$; if $\mathcal{D}$ returns HYB^1 than $\mathcal{A}'$ outputs ideal and if $\mathcal{D}$ returns REAL than $\mathcal{A}'$ outputs real.

First, note that when running with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}real}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$, it holds that $\tilde{\mathsf{p}}_h \leftarrow \mathsf{TEFHE.PartDec}(h, \mathsf{sk}_h, \mathsf{ct})$ and $\mathcal{A}'$ perfectly emulates the execution of $\mathcal{A}$ and $\mathcal{Z}$ in $\text{REAL}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$. On the other hand, when running with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}ideal}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$, it holds that $\tilde{\mathsf{p}}_h \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h, \mathsf{ct}, \mu, \{\mathsf{sk}_j\}_{j\neq h})$ and $\mathcal{A}'$ perfectly emulates the execution of $\mathcal{A}$ and $\mathcal{Z}$ in $\text{HYB}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$. The latter claim holds because simulated decryption shares of other corrupted parties remain perfectly hidden from $\mathcal{A}$ and $\mathcal{Z}$ due to the shares of zero, and are later replaced with genuine decryption shares. Therefore, the distinguishing probability of $\mathcal{A}'$ is the same as that of $\mathcal{D}$, and by the assumed security of the TEFHE scheme this distinguishing probability is negligible. □

The game $\text{HYB}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify the experiment $\text{HYB}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of computing $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$, the threshold-PKI functionality sets the secret-key's shares by sampling for every $i \in [n]$ the secret-key share $\boldsymbol{s}_i \leftarrow \mathbb{Z}_q^{n'-1}$ and $\mathsf{sk}_i = (-\boldsymbol{s}_i, 1)$ (where q is the prime specified in pk as per Definition 6.4), and gives $(\mathsf{pk}, \mathsf{sk}_i)$ to every party P_i.

Claim 6.8. $\text{HYB}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}} \equiv \text{HYB}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$.

Proof. By definition of special TEFHE (Definition 6.4), every secret-key share is of the form $\mathsf{sk}_i = (-\boldsymbol{s}_i, 1)$ where $-\boldsymbol{s}_i$ is identically distributed as uniform vectors in $\mathbb{Z}_q^{n'-1}$. Therefore, $\text{HYB}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ and $\text{HYB}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ are identically distributed. □

The game $\text{HYB}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify the experiment $\text{HYB}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of computing $(\mathsf{pk}, \mathsf{sk}'_1, \ldots, \mathsf{sk}'_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$ and re-sampling $\boldsymbol{s}_i \leftarrow \mathbb{Z}_q^{n'-1}$ and setting $\mathsf{sk}_i = (-\boldsymbol{s}_i, 1)$, the threshold-PKI functionality sets the public key as $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{TEFHE.GenEquiv}(1^\kappa, 1^d)$.

Claim 6.9. $\text{HYB}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\equiv} \text{HYB}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$.

Proof. Given an adversary $\mathcal{A}$, an environment $\mathcal{Z}$, and a distinguisher $\mathcal{D}$ that can distinguish between $\textsc{hyb}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ and $\textsc{hyb}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ we construct a distinguisher $\mathcal{D}'$ for the *indistinguishability of equivocal keys* property of the TEFHE scheme (see Definition 6.1). $\mathcal{D}'$ receives a public key pk as input and runs an execution of $\textsc{hyb}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ toward $\mathcal{A}$ and $\mathcal{Z}$ with the received public key pk. That is, $\mathcal{D}'$ chooses inputs $x_1, \ldots, x_n$ for all parties, gives pk as the public key of the threshold-PKI, returns random secret-key shares upon corruption requests, encrypts the correct input x_i for honest parties and sends shares of zero in the first round, and simulates the decryption shares for the second rounds and further corruption requests as explained in $\textsc{hyb}^1_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$. Next, $\mathcal{D}'$ invokes $\mathcal{D}$ on the output of $\mathcal{Z}$; if $\mathcal{D}$ outputs $\textsc{hyb}^2$ then $\mathcal{D}'$ outputs $\mathsf{non\text{-}equivocal\ pk}$ and if $\mathcal{D}$ outputs $\textsc{hyb}^3$ then $\mathcal{D}'$ outputs $\mathsf{equivocal\ pk}$.

If pk is computed using $\mathsf{TEFHE.Gen}$ the execution is identically distributed as $\textsc{hyb}^2_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$, whereas in case pk is computed using $\mathsf{TEFHE.GenEquiv}$ the execution is identically distributed as $\textsc{hyb}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$. Therefore, the claim follows by the security of the TEFHE scheme. □

The game $\textsc{hyb}^4_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify the experiment $\textsc{hyb}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of having each party encrypt its input as $\mathsf{ct}_i = \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i; r^E_i)$, each party P_i encrypts $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$ and computes $r^E_i \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$.

Claim 6.10. $\textsc{hyb}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{c}}{\equiv} \textsc{hyb}^4_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$.

Proof. Define a sequence of $n+1$ hybrids, where the α'th hybrid, denoted $\textsc{hyb}^{3,\alpha}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ for $\alpha \in [n+1]$, proceeds as $\textsc{hyb}^3_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$, for chosen inputs $x_1, \ldots, x_n$, with the following difference: When computing the encryption during the first round for party P_i with $i < \alpha$, compute $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$ and set the contents of the random tape to $r^E_i \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$; for $i \geq \alpha$, sample random r^E_i and compute $\mathsf{ct}_i = \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i; r^E_i)$.

The first hybrid is exactly an execution of $\textsc{hyb}^3$, whereas the $(n+1)$'th hybrid is an execution of $\textsc{hyb}^4$. By the security of the TEFHE scheme it holds that every two neighboring hybrids are computationally indistinguishable. Specifically, given an adversary $\mathcal{A}$, an environment $\mathcal{Z}$, and a distinguisher $\mathcal{D}$ that can distinguish between $\textsc{hyb}^{3,\alpha}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ and $\textsc{hyb}^{3,\alpha+1}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ (for some $\alpha \in [n]$) we construct a distinguisher $\mathcal{D}'$ for the *indistinguishability of equivocated randomness* property. Upon receiving $(\tilde{\mathsf{pk}}, \tilde{\mathsf{ct}}, \tilde{\mu}, \tilde{r})$, $\mathcal{D}'$ simulates an execution of $\textsc{hyb}^3$ where the input of party P_i is set to be $x_i = \tilde{\mu}$ with the following difference: For $1 \leq i < \alpha$ the ciphertext of an honest party P_i is computed as $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$ and the encryption coins are set to be $r^E_i \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$; for $\alpha < i \leq n$ the encryption coins r^E_i are uniformly sampled and the ciphertext is computed as $\mathsf{ct}_i = \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i; r^E_i)$; and for $i = \alpha$ the ciphertext is set to be $\mathsf{ct}_i = \tilde{\mathsf{ct}}$ and the encryption coins to $r^E_i = \tilde{r}$. Next, $\mathcal{D}'$ continues the simulation of $\textsc{hyb}^3$ and finally outputs whatever $\mathcal{D}$ outputs.

In case $(\tilde{\mathsf{pk}}, \tilde{\mathsf{ct}}, \tilde{\mu}, \tilde{r})$ is computed using $\mathsf{TEFHE.Gen}$ and $\mathsf{TEFHE.Enc}$ the execution is identically distributed as $\textsc{hyb}^{3,\alpha}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$ whereas in case $(\tilde{\mathsf{pk}}, \tilde{\mathsf{ct}}, \tilde{\mu}, \tilde{r})$ is computed using $\mathsf{TEFHE.GenEquiv}$ and $\mathsf{TEFHE.Equiv}$ the execution is identically distributed as $\textsc{hyb}^{3,\alpha+1}_{\pi_{\mathsf{allbutone}},\mathcal{A},\mathcal{Z}}$. Therefore, the claim follows by the security of the TEFHE scheme via a standard hybrid argument. □

The execution in $\textsc{hyb}^4$ is exactly like the simulation done by $\mathcal{S}$ in the ideal model; therefore, the theorem follows from Claims 6.7 to 6.10. □

6.3 Malicious Security

We now proceed to achieve security against malicious and adaptive adversaries. The underlying idea is to compile the semi-malicious protocol from Section 6.2 (Figure 6) to provide malicious security. Following the GMW paradigm, Asharov et al. [5, Thm. E.3] showed how to compile any semi-maliciously secure protocol into a maliciously secure protocol in the ZK-hybrid model that tolerates the same number of corruptions; using NIZK, their compiler does not increase the round complexity. The compiler in [5] is analyzed in the static-corruption setting. We start by proving an analogous result for the adaptive setting. We show how to compile a protocol tolerating an adaptive, semi-malicious adversary that is defined over a broadcast channel, into a protocol tolerating an adaptive, malicious adversary in the NIZK-hybrid model. The NIZK can be instantiated via the protocol in [62], or with shorter proofs via the construction in Section 3.

6.3.1 Malicious Security from Semi-Malicious Security

We consider a semi-malicious protocol in the correlated-randomness hybrid model (looking ahead, we will use the threshold-PKI model) represented by the next-message functions of the parties, i.e., $\mathsf{next\text{-}msg}_i^{\rho}(x_i, v_i, r_i, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1})$ represents the next message function of party P_i in round ρ taking the input value x_i, correlated randomness v_i, random coins r_i, and transcript of prior rounds $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}$, where $\boldsymbol{m}_j = (m_j^1, \ldots, m_j^n)$ are the messages broadcasted in round j. At the last round of the protocol, $\mathsf{next\text{-}msg}_i^R$ returns the output value. We define the NP-relation R_{nxtmsg} consisting of pairs (x, w) where the statement $x = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^i)$ and witness $w = (x_i, v_i, r_i)$ satisfy $m_\rho^i = \mathsf{next\text{-}msg}_i^{\rho}(x_i, v_i, r_i, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1})$.

Lemma 6.11 (semi-malicious to malicious security)**.** *Let f be an n-party function, let $t < n$, and let π_{sm} be an n-party protocol that UC-realizes $\mathcal{F}^f_{\mathsf{sfe\text{-}abort}}$ in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}^D_{\mathsf{corr\text{-}rand}})$-hybrid model (for some distribution D) tolerating an adaptive, semi-malicious t-adversary. Then, the protocol $\pi_{\mathsf{mal}} = \mathsf{comp}_{\mathsf{sm\text{-}to\text{-}mal}}(\pi_{\mathsf{sm}})$ (see Figure 7) UC-realizes $\mathcal{F}^f_{\mathsf{sfe\text{-}abort}}$ in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}^D_{\mathsf{corr\text{-}rand}}, \mathcal{F}^{R_{\mathsf{nxtmsg}}}_{\mathsf{nizk}})$-hybrid model tolerating an adaptive, malicious t-adversary.*

Proof. Let $\mathcal{Z}_{\mathsf{mal}}$ be an environment running with protocol π_{mal} and the dummy adversary $\mathcal{A}_{\mathsf{mal}}$. We will construct a semi-malicious environment $\mathcal{Z}_{\mathsf{sm}}$ running with protocol π_{sm} and a dummy adversary $\mathcal{A}_{\mathsf{sm}}$. The environment $\mathcal{Z}_{\mathsf{sm}}$ starts by invoking $\mathcal{Z}_{\mathsf{mal}}$ on its inputs and proceeds as follows.

- When $\mathcal{Z}_{\mathsf{mal}}$ sends $(\mathsf{input}, \mathsf{sid}, x_i)$ to some honest party P_i, the environment $\mathcal{Z}_{\mathsf{sm}}$ forwards the message to the relevant party, and whenever an honest party sends $(\mathsf{output}, \mathsf{sid}, y)$, the environment $\mathcal{Z}_{\mathsf{sm}}$ forwards the message to $\mathcal{Z}_{\mathsf{mal}}$.
- In every round ρ', the environment $\mathcal{Z}_{\mathsf{sm}}$ stores the message-vector $\boldsymbol{m}_{\rho'}$ that were broadcasted in the round ρ'. Once $\mathcal{Z}_{\mathsf{sm}}$ receives a message (sid, m_ρ^i) in round ρ from honest party P_i (via the adversary), it simulates $\mathcal{F}_{\mathsf{nizk}}$ by sending $(\mathsf{prove}, \mathsf{sid}, P_i, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^i))$ to $\mathcal{Z}_{\mathsf{mal}}$ and receiving back $(\mathsf{sid}, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^i), \Pi_\rho^i)$. Next, $\mathcal{Z}_{\mathsf{sm}}$ sends $(\mathsf{sid}, m_\rho^i, \Pi_\rho^i)$ to $\mathcal{Z}_{\mathsf{mal}}$.
- Once $\mathcal{Z}_{\mathsf{mal}}$ sends $(\mathsf{sid}, m_\rho^i, \Pi_\rho^i)$ on behalf of a corrupted party P_i, the environment $\mathcal{Z}_{\mathsf{sm}}$ sends $(\mathsf{verify}, P_j, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^i), \Pi_\rho^i)$ to $\mathcal{Z}_{\mathsf{mal}}$ (for some honest P_j), and waits for the answer $(\mathsf{witness}, (x_i, v_i, r_i))$. Next, $\mathcal{Z}_{\mathsf{sm}}$ verifies that $m_\rho^i = \mathsf{next\text{-}msg}_i^{\rho}(x_i, v_i, r_i, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1})$. If the witness satisfies the relation, then broadcast (sid, m_ρ^i) on behalf of P_i in the protocol π_{sm} and write the witness (x_i, v_i, r_i) on the special witness tape; otherwise, abort for P_i.

- Whenever $\mathcal{Z}_{\mathsf{mal}}$ asks to corrupt an honest party P_i, the environment $\mathcal{Z}_{\mathsf{sm}}$ corrupts P_i, learns his internal state, adds the calls to $\mathcal{F}_{\mathsf{nizk}}$ with the witness (x_i, v_i, r_i) and the proofs $\{\Pi_i^\rho\}$ that were used during the attack, and forwards the internal state to $\mathcal{Z}_{\mathsf{mal}}$.
- Once $\mathcal{Z}_{\mathsf{mal}}$ output some value, $\mathcal{Z}_{\mathsf{sm}}$ outputs the same value and halts.

Protocol $\mathsf{comp}_{\mathsf{sm\text{-}to\text{-}mal}}(\pi)$

- **Common Input:** An R-round, semi-malicious protocol π, represented by the next-message functions $\{\mathsf{next\text{-}msg}_i^\rho(\cdot)\}_{i\in[n],\rho\in[R]}$.
- **Private input:** Each party has a private input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$.
- **Hybrid model:** The parties have access to the NIZK functionality $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{nxtmsg}}}$, to the broadcast functionality $\mathcal{F}_{\mathsf{bc}}$, and to a correlated-randomness functionality $\mathcal{F}_{\mathsf{corr\text{-}rand}}^{D}$.
- **The Protocol:**

1. Upon receiving $(\mathsf{input}, \mathsf{sid}, x_i)$, party P_i invokes $\mathcal{F}_{\mathsf{corr\text{-}rand}}^{D}$ and gets back v_i.
2. For every round $\rho \in [R]$, party P_i proceeds as follows:
   (a) Compute $m_\rho^i = \mathsf{next\text{-}msg}(x_i, v_i, r_i, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1})$.
   (b) Send the message $(\mathsf{prove}, \mathsf{sid}, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^i), (x_i, v_i, r_i))$ to $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{nxtmsg}}}$ and receive back $(\mathsf{proof}, \mathsf{sid}, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^i), \Pi_\rho^i)$.
   (c) Broadcast the message $(\mathsf{sid}, m_\rho^i, \Pi_\rho^i)$.
   (d) When receiving $(\mathsf{sid}, m_\rho^j, \Pi_\rho^j)$ from P_j (via $\mathcal{F}_{\mathsf{bc}}$), send $(\mathsf{verify}, \mathsf{sid}, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^j), \Pi_\rho^j)$ to $\mathcal{F}_{\mathsf{nizk}}^{R_{\mathsf{nxtmsg}}}$ and receive back $(\mathsf{verification}, \mathsf{sid}, (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{\rho-1}, m_\rho^j), \Pi_\rho^j, b_\rho^j)$.
   (e) If $b_\rho^j = 0$ for some $j \in [n]$, proceed as if the party aborted (i.e., sent $\perp$).
   (f) After receiving the message m_ρ^j from P_j, for every $j \in [n[$, set $\boldsymbol{m}_\rho = (m_\rho^1, \ldots, m_\rho^n)$.
3. Compute $y = \mathsf{next\text{-}msg}(x_i, v_i, r_i, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_R)$ and output $(\mathsf{output}, \mathsf{sid}, y)$.

Figure 7: Compiling semi-malicious security to malicious security

By the construction above, it holds that

$$\text{REAL}_{\pi_{\mathsf{mal}},\mathcal{A}_{\mathsf{mal}},\mathcal{Z}_{\mathsf{mal}}} \equiv \text{REAL}_{\pi_{\mathsf{sm}},\mathcal{A}_{\mathsf{sm}},\mathcal{Z}_{\mathsf{sm}}}.$$

By the security of π_{sm}, there exists a simulator $\mathcal{S}_{\mathsf{sm}}$ such that

$$\text{IDEAL}_{\mathcal{F}_{\mathsf{sfe}}^f,\mathcal{S}_{\mathsf{sm}},\mathcal{Z}_{\mathsf{sm}}} \overset{\text{c}}{\equiv} \text{REAL}_{\pi_{\mathsf{sm}},\mathcal{A}_{\mathsf{sm}},\mathcal{Z}_{\mathsf{sm}}}.$$

We now turn to construct a simulator $\mathcal{S}_{\mathsf{mal}}$ using $\mathcal{S}_{\mathsf{sm}}$ and $\mathcal{Z}_{\mathsf{sm}}$. Whenever $\mathcal{S}_{\mathsf{mal}}$ receives a message from the environment $\mathcal{Z}_{\mathsf{mal}}$ it emulates $\mathcal{Z}_{\mathsf{sm}}$ receiving the message and communicating with $\mathcal{S}_{\mathsf{sm}}$. if $\mathcal{S}_{\mathsf{sm}}$ sends a message to $\mathcal{F}_{\mathsf{sfe}}^f$, then $\mathcal{S}_{\mathsf{mal}}$ sends the same message to the ideal functionality. If $\mathcal{S}_{\mathsf{mal}}$ receives a message from $\mathcal{F}_{\mathsf{sfe}}^f$, he forwards the message to $\mathcal{S}_{\mathsf{sm}}$. It follows by construction that

$$\text{IDEAL}_{\mathcal{F}_{\mathsf{sfe}}^f,\mathcal{S}_{\mathsf{sm}},\mathcal{Z}_{\mathsf{sm}}} \equiv \text{IDEAL}_{\mathcal{F}_{\mathsf{sfe}}^f,\mathcal{S}_{\mathsf{mal}},\mathcal{Z}_{\mathsf{mal}}}.$$

From the above we conclude that

$$\mathrm{REAL}_{\pi_{\mathsf{mal}},\mathcal{A}_{\mathsf{mal}},\mathcal{Z}_{\mathsf{mal}}} \stackrel{\mathrm{c}}{\equiv} \mathrm{IDEAL}_{\mathcal{F}^f_{\mathsf{sfe}},\mathcal{S}_{\mathsf{mal}},\mathcal{Z}_{\mathsf{mal}}}.$$

This concludes the proof of Lemma 6.11. □

6.3.2 Malicious Security with Sublinear Communication

In Section 6.3.1, it is important that the semi-malicious protocol will be defined purely over a broadcast channel; however, the protocol in Section 6.2 uses secure channels. To resolve this issue, the secret shares of zero that were sent over secure point-to-point channels should be encrypted and transmitted over the broadcast channel. As we consider adaptive corruptions, we need to use non-committing encryption (see Appendix A.2.5) instead of standard public-key encryption, and each non-committing public key should be used to encrypt n elements in $\mathbb{Z}_q$. We consider the distribution of the NCE public keys as part of the threshold-PKI functionality.[15] Alternatively, the public keys can be exchanged at the cost of an additional communication round.

Theorem 6.12 (Theorem 1.4, restated)**.** *Assume the existence of special TEFHE schemes and NCE schemes, let $t < n$, and let $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ be an efficiently computable function of depth d. Then, $\mathcal{F}^f_{\mathsf{sfe\text{-}abort}}$ can be UC-realized in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{nizk}})$-hybrid model, tolerating an adaptive, malicious, PPT t-adversary, by a two-round protocol with communication complexity* $\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$.

By instantiating the threshold-PKI functionality using an adaptively secure protocol, we get a protocol that is defined in the CRS-hybrid model. Using the recent protocol of Benhamouda et al. [13] we obtain the following corollary.

Corollary 6.13. *Assume the existence of two-round adaptively secure OT in the CRS model, and the additional assumptions in Theorem 6.12. Then, $\mathcal{F}^f_{\mathsf{sfe\text{-}abort}}$ can be UC-realized in the $(\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{nizk}})$-hybrid model, tolerating an adaptive, malicious PPT t-adversary, by a four-round protocol with communication complexity* $\mathrm{poly}(\ell_{\mathsf{in}}, \ell_{\mathsf{out}}, d, \kappa, n)$.

7 The Honest-Majority Setting

In this section, we show how to adjust the protocol from Section 6 that provides security with abort, into guaranteeing output delivery in the honest-majority setting. Gordon et al. [60] constructed a two-round protocol in the threshold-PKI model that guarantees output delivery assuming an honest majority in the static-corruption setting and has sublinear communication complexity. We apply some of the techniques from [60] on our adaptively secure protocol designed for the all-but-one setting, and achieve a matching result tolerating adaptive corruptions.

In the all-but-one case (Section 6) the decryption key was shared using additive secret sharing. As observed in [60], since the decryption of the GSW-based threshold FHE consists of linear operations, it is possible to use Shamir's secret sharing [91] instead. The problem with a naïve use of this idea is that when the partial decryptions are reconstructed, each decryption share is multiplied by the Lagrange coefficient, and thus also the smudging noise. This will result in blowing up the

[15]Note that unlike the TEFHE keys, the NCE keys can only be used an a priori bounded number of times.

noise and may end up with an incorrect decryption. Gordon et al. [60] overcame this problem by having each party secret share (using Shamir's scheme) its smudging noise in the first round of the protocol, and parties added shares of the smudging noise of non-aborting parties in a way that is compatible with the decryption algorithm.[16]

7.1 Cryptographic Primitives used in the Protocol

7.1.1 Threshold Secret Sharing Scheme

A key ingredient in the construction of the TEFHE scheme for the honest-majority setting is Shamir's $(t+1)$-out-of-n secret sharing [91]. Given integers $t < n$ and a prime $q > n$, Shamir's scheme consists of two algorithms.

- $(\mathsf{s}_1, \ldots, \mathsf{s}_n) \leftarrow \mathsf{Share}(s)$: to share a secret $s \in \mathbb{Z}_q$, choose a random polynomial $p(x)$ over $\mathbb{Z}_q$ of degree t conditioned on $p(0) = s$, and set the shares as $\mathsf{s}_i = p(i)$ for every $i \in [n]$.
- $s' = \mathsf{Recon}(\{\mathsf{s}_i\}_{i \in S})$: given shares $\{\mathsf{s}_i\}_{i \in S}$ for some subset $S \subseteq [n]$ of size $|S| \geq t+1$, compute $s' = \sum_{j \in S} L_j(0) \cdot \mathsf{s}_j$, where $L_j(x)$ is the Lagrange polynomial with respect to the set S

$$L_j(x) = \prod_{k \in S \setminus \{j\}} \frac{x-k}{j-k}.$$

Shamir's secret sharing perfectly hides the secret given any subset of at most t shares, and, as shown above, any subset of $t+1$ shares enables an efficient reconstruction of the secret. For the remaining of the section, we will consider $t = \lceil n/2 \rceil - 1$.

In the proof of protocol we will use an additional property of Shamir's scheme, which allows to generate "new" shares for a "new" secret in a way that is compatible to less than t "old" shares.

- $(\tilde{\mathsf{s}}_1, \ldots, \tilde{\mathsf{s}}_n) \leftarrow \mathsf{Reshare}(s, \{\mathsf{s}_i\}_{i \in S})$: To re-share a secret $s \in \mathbb{Z}_q$ given a set of shares $\{\mathsf{s}_i\}_{i \in S}$, where $|S| \leq t$, choose a random polynomial $p(x)$ over $\mathbb{Z}_q$ of degree t conditioned on $p(0) = s$ and $p(i) = \mathsf{s}_i$ for $i \in S$, and set the new shares as $\tilde{\mathsf{s}}_j = p(j)$ for every $j \in [n]$.

7.1.2 Honest-Majority Threshold Equivocal FHE

In Section 6, the threshold FHE was defined using an n-out-of-n secret sharing of the decryption key. We now adjust the definition of TEFHE to the honest-majority setting. The main adjustments are using an $n/2$-out-of-n Shamir secret sharing of the decryption key, and separating the smudging noise from the encryption and decryption algorithms so it is external to the scheme. We still require though the ability to simulate the smudging noise in a similar way to Definition 6.3.

Definition 7.1. *A special $n/2$-out-of-n TEFHE scheme is defined using the same syntax as in Definition 6.3, with the following adjustments:*

- *The key-generation algorithm $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$ outputs a public key pk that defines a prime q and a smudging-noise bound B_{smdg}. In addition, the algorithm outputs Shamir shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{Share}(\boldsymbol{t})$ of a value $\boldsymbol{t} = (-\boldsymbol{s}, 1)$, where $\boldsymbol{s} \leftarrow \mathbb{Z}_q^{n'-1}$ for some $n' = \mathrm{poly}(\kappa, d)$.*

[16] Recently, Boneh et al. [17] showed that this problem can be overcome in a different way, by using a special secret sharing scheme that ensures the Lagrange coefficients are binary values.

- *The partial-decryption algorithm* $\mathsf{p}_i \leftarrow \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$ *operates by computing* $\mathsf{p}_i = \langle \mathsf{ct}, \mathsf{sk}_i \rangle \mod q$.
- *The final-decryption algorithm* $\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_i\}_{i \in S})$*, for* $S \subseteq [n]$ *of size* $n/2$*, executes Shamir's reconstruction, i.e., outputs* $\mathsf{Recon}(\{\mathsf{p}_i\}_{i \in S})$.

We require the following properties from a special $n/2$-out-of-n TEFHE scheme:

1. Correctness: Consider the FHE scheme that is defined by setting the decryption key $\mathsf{sk} = \mathsf{Recon}(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$, and where the decryption algorithm consists of choosing an arbitrary set $S \subseteq [n]$, sampling for every $j \in S$ smudging noise $e_j \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$, and outputting $\langle \mathsf{sk}, \mathsf{ct} \rangle + \sum_{j \in S} e_j$. Then, this FHE scheme is a correct, compact, and secure equivocal FHE scheme for circuits of depth d (according to Definition 6.1).
2. Simulatability of partial decryption: there exists a PPT simulator $\mathsf{Sim}_{\text{TEFHE}}$ such that for integers n, d, and ℓ, and every circuit $C : \{0,1\}^\ell \rightarrow \{0,1\}$ of depth d, every $h \in [n]$, and every $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ in the support of $\mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^n)$, the following distributions are statistically close:

$\mathsf{Expt}^{\mathsf{TEFHE\text{-}hm\text{-}real}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$	$\mathsf{Expt}^{\mathsf{TEFHE\text{-}hm\text{-}ideal}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$
Send $(h, \mathsf{pk}, \{\mathsf{sk}_j\}_{j \neq h})$ to $\mathcal{A}$	Send $(h, \mathsf{pk}, \{\mathsf{sk}_j\}_{j \neq h})$ to $\mathcal{A}$
$\mathcal{A}$ sends $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ and $r_1, \ldots, r_\ell \in \{0,1\}^*$	$\mathcal{A}$ sends $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ and $r_1, \ldots, r_\ell \in \{0,1\}^*$
$\forall j \in [\ell]$ compute $\mathsf{ct}_j = \mathsf{TEFHE.Enc}(\mathsf{pk}, \mu_j; r_j)$	$\forall j \in [\ell]$ compute $\mathsf{ct}_j = \mathsf{TEFHE.Enc}(\mathsf{pk}, \mu_j; r_j)$
Compute $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$	Compute $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$
Compute $\mathsf{p}_h \leftarrow \mathsf{TEFHE.PartDec}(h, \mathsf{sk}_h, \mathsf{ct})$	Compute $\mu = C(\mu_1, \ldots, \mu_\ell)$
Sample $e_h \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$	Compute $\mathsf{p}_h \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h, \mathsf{ct}, \mu, \{\mathsf{sk}_j\}_{j \neq h})$
Send $\mathsf{p}_h + e_h$ to $\mathcal{A}$	Send p_h to $\mathcal{A}$
Output whatever $\mathcal{A}$ outputs	Output whatever $\mathcal{A}$ outputs

Lemma 7.2. *Assuming LWE there exist special* $n/2$*-out-of-*n *TEFHE schemes.*

Proof. The lemma follows from Lemma 6.2 by sharing the secret key sk using Shamir's scheme, defining the partial decryption as the inner product of the ciphertext and the share of the key, and the final decryption as the Shamir's reconstruction. Correctness follows by linearity of inner product. Simulatability of partial decryption follows identically as in Section 6.1.2. We note that the security of the scheme is not proven directly, but is combined with the security of the protocol below. □

7.2 Semi-Malicious Security

Theorem 7.3. *Assume the existence of special* $n/2$*-out-of-*n *TEFHE schemes, let* $t < n/2$*, and let* $f : (\{0,1\}^{\ell_{in}})^n \rightarrow \{0,1\}^{\ell_{out}}$ *be an efficiently computable function of depth* d*. Then, Protocol* π_{hm}*, defined in Figure 8, UC-realizes* $\mathcal{F}^f_{\mathsf{sfe\text{-}god}}$ *in the* $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{smt}})$*-hybrid model, tolerating an adaptive, semi-malicious, PPT* t*-adversary, by a two-round protocol with communication complexity* $\mathrm{poly}(\ell_{in}, \ell_{out}, d, \kappa, n)$.

Protocol π_{hm}

- **Private Input:** Every party P_i, for $i \in [n]$, has private input $x_i \in \{0,1\}^{\ell_{\mathsf{in}}}$.
- **Common Input:** A special $n/2$-out-of-n TEFHE scheme Π and a circuit C_f of depth d.
- **The Protocol:**

1. Upon receiving $(\mathsf{input}, \mathsf{sid}, x_i)$, party P_i proceeds as follows:
   (a) Invoke $\mathcal{F}_{\mathsf{thresh\text{-}pki}}(\Pi, d)$ with $(\mathsf{init}, \mathsf{sid})$ to receive $(\mathsf{sid}, \mathsf{pk}, \mathsf{sk}_i)$. Let q be the prime associated with pk and let B_{smdg} the associated smudging-noise bound (as per Definition 7.1).
   (b) Encrypt the input as $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i)$.
   (c) Sample smudging noise $e_i \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$.
   (d) Secret share e_i using Shamir's secret sharing as $(\mathsf{s}_{i,1}, \ldots, \mathsf{s}_{i,n}) \leftarrow \mathsf{Share}(e_i)$.
   (e) Send $(\mathsf{sid}, \mathsf{ct}_i, \mathsf{s}_{i,j})$ to P_j over a secure channel (via $\mathcal{F}_{\mathsf{smt}}$).
2. Denote by S_1 the set of parties who sent an $(\mathsf{sid}, \cdot)$ message, party P_i proceeds as follows:
   (a) Let C_{f,S_1} be the circuit C_f where for every $j \notin S_1$ a default input value is hard-wired.
   (b) Compute $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C_{f,S_1}, \{\mathsf{ct}_j\}_{j \in S_1})$.
   (c) Partially decrypt the result as $\mathsf{p}_i = \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$.
   (d) Set $\mathsf{m}_i = \mathsf{p}_i + \sum_{j \in S_1} \mathsf{s}_{j,i}$ and send $(\mathsf{sid}, \mathsf{m}_i)$ to every party.
3. Denote by S_2 the set of parties who sent an $(\mathsf{sid}, \cdot)$ message. Party P_i computes the final decryption as $y = \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{m}_j\}_{j \in S_2})$ and outputs $(\mathsf{output}, \mathsf{sid}, y)$.

Figure 8: MPC with g.o.d. and semi-malicious security in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{smt}})$-hybrid model

Proof. We start by showing correctness. Denote by S_1 the set of non-aborting parties in round 1, then for every $i \in S_1$ it holds that $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i)$ and $(\mathsf{s}_{i,1}, \ldots, \mathsf{s}_{i,n})$ are Shamir shares of e_i; in addition, $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C_{f,S_1}, \{\mathsf{ct}_j\}_{j \in S_1})$. Next, each party computes $\mathsf{p}_i = \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$ and sends $\mathsf{m}_i = \mathsf{p}_i + \sum_{j \in S_1} \mathsf{s}_{j,i} \mod q$. Denote by S_2 the set of non-aborting parties in round 1. By linearity of inner product, and since $|S_2| \geq n/2$ (due to the honest-majority assumption), it holds that:

$$\begin{aligned}
\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{m}_i\}_{i \in S_2}) &= \mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_i + \sum_{j \in S_1} \mathsf{s}_{j,i}\}_{i \in S_2}) \\
&= \sum_{i \in S_2} L_i(0) \cdot (\mathsf{p}_i + \sum_{j \in S_1} \mathsf{s}_{j,i}) \\
&= \sum_{i \in S_2} L_i(0) \cdot (\langle \mathsf{ct}, \mathsf{sk}_i \rangle + \sum_{j \in S_1} \mathsf{s}_{j,i}) \\
&= \langle \mathsf{ct}, \sum_{i \in S_2} L_i(0) \cdot \mathsf{sk}_i \rangle + \sum_{j \in S_1} \sum_{i \in S_2} L_i(0) \cdot \mathsf{s}_{j,i} \\
&= \langle \mathsf{ct}, \mathsf{sk} \rangle + \sum_{j \in S_1} e_j.
\end{aligned}$$

By *correctness* of the special $n/2$-out-of-n TEFHE scheme, the decryption is correct and outputs $C_{f,S_1}(\{x_i\}_{i \in S_1})$

We proceed to prove security of the protocol. Let $\mathcal{A}$ be an adaptive, semi-malicious adversary attacking π_{hm} in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{smt}})$-hybrid model. We will construct an ideal-process adversary $\mathcal{S}$, interacting with the ideal functionality $\mathcal{F}^{f}_{\mathsf{sfe\text{-}god}}$ and with ideal (dummy) parties $\widetilde{P}_1, \ldots, \widetilde{P}_n$, such that no environment can distinguish between $\mathcal{S}$ and $\mathcal{A}$. The simulator $\mathcal{S}$ constructs virtual parties $P_1, \ldots, P_n$, and runs the adversary $\mathcal{A}$. Let $\mathsf{Sim}_{\text{TEFHE}}$ be the simulator that is guaranteed to exist for the TEFHE scheme Π.

Note that the internal state of a party P_i in the protocol consists of its input x_i, the message $(\mathsf{pk}, \mathsf{sk}_i)$ from $\mathcal{F}_{\mathsf{thresh\text{-}pki}}$, the random coins that include coins r_i^E for encrypting the input in the first round, coins r_i^{sm} for sampling the smudging noise e_i, and its secret sharing $(\mathsf{s}_{i,1}, \ldots, \mathsf{s}_{i,n})$ of e_i.[17] In addition, the internal state contains the messages P_i received in the first round, including the values $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ and $(\mathsf{s}_{1,i}, \ldots, \mathsf{s}_{n,i})$, and in the second round $(\mathsf{m}_1, \ldots, \mathsf{m}_n)$.

Simulating the communication $\mathcal{Z}$: Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape.

Simulating the threshold-PKI functionality $\mathcal{F}_{\mathsf{thresh\text{-}pki}}$: The simulator $\mathcal{S}$ computes $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{TEFHE.GenEquiv}(1^{\kappa}, 1^{d})$, samples $\boldsymbol{s} \leftarrow \mathbb{Z}_q^{n'-1}$ (as per Definition 7.1), sets $\mathsf{sk} = \boldsymbol{t} = (-\boldsymbol{s}, 1)$, computes Shamir shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{Share}(\mathsf{sk})$, and gives $(\mathsf{pk}, \mathsf{sk}_i)$ to $\mathcal{A}$ for every corrupted party P_i.

Simulating the first round: For every honest party P_i, compute $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$, sample $e_i \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$, share it as $(\mathsf{s}_{i,1}^{(0)}, \ldots, \mathsf{s}_{i,n}^{(0)}) \leftarrow \mathsf{Share}(e_i)$, and send $(\mathsf{sid}, \mathsf{ct}_i, \mathsf{s}_{i,j}^{(0)})$ to every corrupted P_j. Finally, the simulator receives from $\mathcal{A}$ the message $(\mathsf{sid}, \mathsf{ct}_j, \mathsf{s}_{j,i}^{(0)})$ on behalf of every corrupted party P_j and every honest party P_i, and reads from the special witness tape of the semi-malicious adversary $\mathcal{A}$ the input and random coins (x_i', r_i).

Sending input to the functionality $\mathcal{F}^{f}_{\mathsf{sfe\text{-}god}}$: On behalf of every aborting corrupted party, set x_i' to be the default input value (e.g., zero), and for every non-aborting corrupted party P_i set the input value x_i' as read from the witness tape. Send the input values $\{x_i'\}$ to the ideal functionality $\mathcal{F}^{f}_{\mathsf{sfe\text{-}god}}$ and receive back the output y.

Simulating the second round: Let $\mathcal{I}$ be the set of currently corrupted parties, and let $\{\mathsf{ct}_j\}_{j \in S_1}$ be the ciphertexts of non-aborting parties in the first round of the simulation. $\mathcal{S}$ homomorphically evaluates the circuit as $\mathsf{ct} = \mathsf{TEFHE.Eval}(\mathsf{pk}, C_{f,S_1}, \{\mathsf{ct}_j\}_{j \in S_1})$ and chooses an arbitrary honest party P_{h_1}. For every honest party P_i with $i \neq h_1$, $\mathcal{S}$ computes the partial decryption $\mathsf{p}_i = \mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct})$, and for h_1, $\mathcal{S}$ computes the simulated partial decryption $\tilde{\mathsf{p}}_{h_1} \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h_1, \mathsf{ct}, y, \{\mathsf{sk}_j\}_{j \neq h_1})$ (recall that the simulated $\tilde{\mathsf{p}}_{h_1}$ includes the smudging noise); $\mathcal{S}$ re-shares the smudging noise e_{h_1} to shares of zero as $(\mathsf{s}_{h_1,1}^{(1)}, \ldots, \mathsf{s}_{h_1,n}^{(1)}) \leftarrow \mathsf{Reshare}(0, \{\mathsf{s}_{h_1,j}^{(0)}\}_{j \in \mathcal{I}})$. For every $i \neq h_1$ and j set $\mathsf{s}_{i,j}^{(1)} = \mathsf{s}_{i,j}^{(0)}$ (i.e., the shares sent to honest parties from corrupted parties in the first round (by $\mathcal{A}$), and were sampled (by $\mathcal{S}$) to be sent by honest parties, except for P_{h_1}).

[17] Without loss of generality, we can assume that $(\mathsf{s}_i^1, \ldots, \mathsf{s}_i^n)$ fully determine the coins used to sample e_i and to share it.

Finally, the simulator $\mathcal{S}$ sets $\mathsf{m}_i = \mathsf{p}_i + \sum_j \mathsf{s}_{j,i}^{(1)} \mod q$, gives $(\mathsf{sid}, \mathsf{m}_i)$ to $\mathcal{A}$, and receives messages from $\mathcal{A}$ on behalf of corrupted parties.

Simulating corruption requests of party P_i: The simulator $\mathcal{S}$ corrupts the dummy party $\widetilde{P}_i$ and continues as follows, depending on the timing of the corruption:

- **Corruptions before the first round:** In this case $\widetilde{P}_i$ has not been activated with input yet. $\mathcal{S}$ sets the contents of P_i's random tape with uniformly random coins.
- **Corruptions after the first round and before the second round:** In this case $\widetilde{P}_i$ has been activated with input $(\mathsf{input}, \mathsf{sid}, x_i)$, but did not receive output yet. $\mathcal{S}$ sets the contents of P_i's input tape to $(\mathsf{input}, \mathsf{sid}, x_i)$ and the contents of the random tape to $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$ for the ciphertext ct_i that was used to simulate P_i's message.

  Finally, the secret shares of the smudging noise $(\mathsf{s}_{i,1}^{(0)}, \ldots, \mathsf{s}_{i,n}^{(0)})$ have already been set for P_i during the simulation. Similarly, the messages received in the first round, i.e., $(\mathsf{s}_{1,i}^{(0)}, \ldots, \mathsf{s}_{n,i}^{(0)})$ are set according to the simulation both for corrupted parties (as received from $\mathcal{A}$) and for honest parties (as simulated by $\mathcal{S}$).
- **Corruptions after the second round:** In this case $\widetilde{P}_i$ has received the output. $\mathcal{S}$ computes $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$ as in the previous case.

  Let h_k be the index of the recent honest party that was "chosen" during the simulation; in particular, all of the secret shares are denoted as $\mathsf{s}_{i,j}^{(k)}$.

  - If $i \neq h_k$, i.e., the corrupted party is *not* the "chosen" honest party, then set the shares of the smudging noise $(\mathsf{s}_{i,1}^{(k)}, \ldots, \mathsf{s}_{i,n}^{(k)})$, and the incoming messages $(\mathsf{s}_{1,i}^{(k)}, \ldots, \mathsf{s}_{n,i}^{(k)})$ according to the values set in the simulation.
  - If $i = h_k$, then the simulator needs to adjust the secret shares to be compatible with the view of the adversary. Let $\mathcal{I}$ be the set of currently corrupted parties. To "fix" the shares sent by P_{h_k}, the simulator computes $\mathsf{p}_{h_k} = \mathsf{TEFHE.PartDec}(h_k, \mathsf{sk}_{h_k}, \mathsf{ct})$, samples a new smudging noise $e'_{h_k} \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ and re-shares it as $(\mathsf{s}_{h_k,1}^{(k+1)}, \ldots, \mathsf{s}_{h_k,n}^{(k+1)}) \leftarrow \mathsf{Reshare}(e'_{h_k}, \{\mathsf{s}_{h_k,j}^{(k)}\}_{j \in \mathcal{I}})$.
    Next, to "fix" the incoming shares, the simulator chooses another honest party $P_{h_{k+1}}$, computes $\tilde{\mathsf{p}}_{h_{k+1}} \leftarrow \mathsf{Sim}_{\text{TEFHE}}(h_{k+1}, \mathsf{ct}, y, \{\mathsf{sk}_j\}_{j \neq h_{k+1}})$, and "adjusts" the shares of $P_{h_{k+1}}$ to be shares of zero. In particular, it must hold that:
    1. $\mathsf{m}_{h_k} = \mathsf{p}_{h_k} + \sum_j \mathsf{s}_j^{h_k} \mod q$.
    2. $\sum_j \mathsf{s}_{h_k}^j = e_{h_k} \mod q$.
    3. $\mathsf{m}_{h_{k+1}} = \tilde{\mathsf{p}}_{h_{k+1}} + \sum_j \mathsf{s}_j^{h_{k+1}} \mod q$.
    4. $\sum_j \mathsf{s}_{h_{k+1}}^j = 0 \mod q$.

    The simulator can use the free variables (that have not been released to the adversary yet) $\mathsf{s}_{h_{k+1}}^{h_{k+1}}$, $\mathsf{s}_{h_k}^{h_k}$, $\mathsf{s}_{h_k}^{h_{k+1}}$, and $\mathsf{s}_{h_{k+1}}^{h_k}$. The simulator proceeds to solve these 4 equations with respect to the 4 variables mentioned above.

We now turn to show that no environment can distinguish between the simulation of $\mathcal{S}$ with $\mathcal{F}_{\mathsf{sfe\text{-}god}}^f$ and the execution of π_{hm} with $\mathcal{A}$ by defining a series of hybrid games. The output of each game is the output of the environment. Let $\mathcal{Z}$ be a PPT environment.

The game $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify the real-model experiment $\textsc{real}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ as follows. Denote the shares sent from party P_i to party P_j by $\mathsf{s}^{(0)}_{i,j}$. After the first round, once the output value y is already determined by the inputs, an arbitrary honest party P_{h_1} (e.g., the honest party P_i with smallest index i) replaces the smudging noise e_{h_1} used for the first round (and the shares $(\mathsf{s}^{(0)}_{h_1,1},\ldots,\mathsf{s}^{(0)}_{h_1,n}) \leftarrow \mathsf{Share}(e_{h_1})$) as follows:

Let $\mathcal{I}$ be the set of currently corrupted parties. First, the simulated decryption share is computed as $\tilde{\mathsf{p}}_{h_1} \leftarrow \mathsf{Sim}_{\textsc{tefhe}}(h_1,\mathsf{ct},y,\{\mathsf{sk}_j\}_{j\neq h_1})$ (recall that the simulated $\tilde{\mathsf{p}}_{h_1}$ includes the simulated smudging noise). Next, set the shares of P_{h_1} to be shares of zero as $(\mathsf{s}^{(1)}_{h_1,1},\ldots,\mathsf{s}^{(1)}_{h_1,n}) \leftarrow \mathsf{Reshare}(0,\{\mathsf{s}^{(0)}_{h_1,j}\}_{j\in\mathcal{I}})$. For all other i,j, denote the shares $\mathsf{s}^{(1)}_{i,j} = \mathsf{s}^{(0)}_{i,j}$. The parties compute the second-round messages using the "new" shares $\mathsf{s}^{(1)}_{i,j}$.

Upon a corruption request of party P_i, let h_k be the index of the recent honest party that was "chosen." In particular, all of the secret shares are denoted as $\mathsf{s}^{(k)}_{i,j}$. If $i \neq h_k$, then the state of P_{h_k} is revealed to the adversary without any change. If $i = h_k$, then proceed to explain the state of P_{h_k} in the same way as done in the simulation, by adjusting the secret shares to be compatible with the view of the adversary. Let $\mathcal{I}$ be the set of currently corrupted parties.

- To "fix" the state of P_{h_k}, compute $\mathsf{p}_{h_k} = \mathsf{TEFHE.PartDec}(h_k,\mathsf{sk}_{h_k},\mathsf{ct})$, sample a new smudging noise $e'_{h_k} \leftarrow [-B_{\mathsf{smdg}},B_{\mathsf{smdg}}]$, and re-share it as $(\mathsf{s}^{(k+1)}_{h_k,1},\ldots,\mathsf{s}^{(k+1)}_{h_k,n}) \leftarrow \mathsf{Reshare}(e'_{h_k},\{\mathsf{s}^{(k)}_{h_k,j}\}_{j\in\mathcal{I}})$.
- To "fix" the incoming shares, choose another honest party $P_{h_{k+1}}$, compute $\tilde{\mathsf{p}}_{h_{k+1}} \leftarrow \mathsf{Sim}_{\textsc{tefhe}}(h_{k+1},\mathsf{ct},y,\{\mathsf{sk}_j\}_{j\neq h_{k+1}})$, and adjust $\mathsf{s}^{(k+1)}_{h_{k+1},h_{k+1}}$, $\mathsf{s}^{(k+1)}_{h_k,h_k}$, $\mathsf{s}^{(k+1)}_{h_k,h_{k+1}}$, and $\mathsf{s}^{(k+1)}_{h_{k+1},h_k}$ by solving the four equations as done in the simulation.
- For all other values i,j set $\mathsf{s}^{(k+1)}_{i,j} = \mathsf{s}^{(k)}_{i,j}$.

Claim 7.4. $\textsc{real}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{s}}{\equiv} \textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$.

Proof. This follows by the security of the threshold-decryption protocol of the TEFHE scheme and by the honest-majority assumption. Consider an adversary $\mathcal{A}$, and environment $\mathcal{Z}$, and a distinguisher $\mathcal{D}$ that can distinguish between $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ and $\textsc{real}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. Consider a mental experiment where a challenger computes $(\mathsf{pk},\mathsf{sk}_1,\ldots,\mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa,1^d,1^n)$ and emulates the honest parties running the protocol with $\mathcal{A}$ and $\mathcal{Z}$ (running using some random coins). Denote by $h \in [n]$ an index of a party that remained honest when $\mathcal{Z}$ produces output.

Now, consider an adversary $\mathcal{A}'$ for the *simulatability of partial decryption* of the TEFHE scheme (Definition 7.1). $\mathcal{A}'$ is running either with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}hm\text{-}real}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$ or with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}hm\text{-}ideal}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$; initially, $\mathcal{A}'$ receives $(h,\mathsf{pk},\{\mathsf{sk}_j\}_{j\neq h})$. It uses the same coins as the challenger above to emulate the protocol toward $\mathcal{A}$ and $\mathcal{Z}$, with the difference that $\mathcal{A}'$ does not sample the threshold-PKI keys, but uses pk and $\{\mathsf{sk}_j\}_{j\neq h}$. After the first round, $\mathcal{A}'$ responds with the inputs values $\mu_1,\ldots,\mu_\ell$ and $r_1,\ldots,r_\ell$ used in the execution and receives $\tilde{\mathsf{p}}_h$. Next, instead of simulating the decryption share of P_h as $\mathsf{TEFHE.PartDec}(h,\mathsf{sk}_h,\mathsf{ct})$ and sampling and sharing the smudging noise $e_h \leftarrow [-B_{\mathsf{smdg}},B_{\mathsf{smdg}}]$, it sets the decryption share to the value $\tilde{\mathsf{p}}_h$, and prepares shares of zero on behalf of P_h. Finally, $\mathcal{A}'$ invokes $\mathcal{D}$ on the output of $\mathcal{Z}$; if $\mathcal{D}$ returns $\textsc{hyb}^1$ than $\mathcal{A}'$ outputs ideal and if $\mathcal{D}$ returns $\textsc{real}$ than $\mathcal{A}'$ outputs real.

First, note that when running with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}hm\text{-}real}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$, it holds that $\tilde{\mathsf{p}}_h \leftarrow \mathsf{TEFHE.PartDec}(h,\mathsf{sk}_h,\mathsf{ct}) + e_h$ with $e_h \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ and $\mathcal{A}'$ perfectly emulates the execution of $\mathcal{A}$ and $\mathcal{Z}$ in $\textsc{real}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. On the other hand, when running with $\mathsf{Expt}^{\mathsf{TEFHE\text{-}hm\text{-}ideal}}_{\Pi,\mathcal{A},\mathsf{pk},\{\mathsf{sk}_i\}_{i\in[n]},h,C}(\kappa)$, it holds that $\tilde{\mathsf{p}}_h \leftarrow \mathsf{Sim}_{\textsc{tefhe}}(h,\mathsf{ct},\mu,\{\mathsf{sk}_j\}_{j\neq h})$ and $\mathcal{A}'$ perfectly emulates the execution of $\mathcal{A}$ and $\mathcal{Z}$ in $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. The latter claim holds because simulated decryption shares of other corrupted parties remain perfectly hidden from $\mathcal{A}$ and $\mathcal{Z}$, and are later replaced with genuine decryption shares. Therefore, the distinguishing probability of $\mathcal{A}'$ is the same as that of $\mathcal{D}$, and by the assumed security of the TEFHE scheme this distinguishing probability is negligible. □

The game $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. In this game, we modify the experiment $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of computing $(\mathsf{pk},\mathsf{sk}_1,\ldots,\mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa,1^d,1^n)$, the threshold-PKI functionality sets the secret-key's shares by sampling for every $i\in[n]$ the secret-key share $\boldsymbol{s} \leftarrow \mathbb{Z}_q^{n'-1}$ (where q is the prime specified in pk as per Definition 6.4) and $\mathsf{sk} = \boldsymbol{t} = (-\boldsymbol{s},1)$, computes $(\mathsf{sk}_1,\ldots,\mathsf{sk}_n) \leftarrow \mathsf{Share}(\mathsf{sk})$, and gives $(\mathsf{pk},\mathsf{sk}_i)$ to every party P_i.

Claim 7.5. $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}} \equiv \textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$.

Proof. By definition of special $n/2$-out-of-n TEFHE scheme (Definition 7.1), the secret key if of the form $\mathsf{sk} = \boldsymbol{t} = (-\boldsymbol{s},1)$ where $\boldsymbol{s} \leftarrow \mathbb{Z}_q^{n'-1}$ is uniform distributed, and $(\mathsf{sk}_1,\ldots,\mathsf{sk}_n)$ are Shamir shares of sk. Therefore, $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ and $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ are identically distributed. □

The game $\textsc{hyb}^3_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. In this game, we modify the experiment $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of computing $(\mathsf{pk},\mathsf{sk}_1,\ldots,\mathsf{sk}_n) \leftarrow \mathsf{TEFHE.Gen}(1^\kappa,1^d,1^n)$ and re-sampling the key shares, the threshold-PKI functionality sets the keys as $(\mathsf{pk},\mathsf{td}) \leftarrow \mathsf{TEFHE.GenEquiv}(1^\kappa,1^d)$ and samples the secret key as before.

Claim 7.6. $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{c}}{\equiv} \textsc{hyb}^3_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$.

Proof. Given an adversary $\mathcal{A}$, an environment $\mathcal{Z}$, and a distinguisher $\mathcal{D}$ that can distinguish between $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ and $\textsc{hyb}^3_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ we construct a distinguisher $\mathcal{D}'$ for the *indistinguishability of equivocal keys* property of the TEFHE scheme (see Definition 6.1). $\mathcal{D}'$ receives a public key pk as input and runs an execution of $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ toward $\mathcal{A}$ and $\mathcal{Z}$ with the received public key pk. That is, $\mathcal{D}'$ chooses inputs $x_1,\ldots,x_n$ for all parties, gives pk as the public key of the threshold-PKI, samples a random secret-key and computes corresponding Shamir shares, encrypts the correct input x_i for honest parties and sends shares of the smudging noise in the first round, and simulates the decryption shares for the second rounds and further corruption requests as explained in $\textsc{hyb}^1_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. Next, $\mathcal{D}'$ invokes $\mathcal{D}$ on the output of $\mathcal{Z}$; if $\mathcal{D}$ outputs $\textsc{hyb}^2$ then $\mathcal{D}'$ outputs $\mathsf{non\text{-}equivocal\ pk}$ and if $\mathcal{D}$ outputs $\textsc{hyb}^3$ then $\mathcal{D}'$ outputs $\mathsf{equivocal\ pk}$.

In case pk is computed using $\mathsf{TEFHE.Gen}$ the execution is identically distributed as $\textsc{hyb}^2_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ whereas in case pk is computed using $\mathsf{TEFHE.GenEquiv}$ the execution is identically distributed as $\textsc{hyb}^3_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. Therefore, the claim follows by the security of the TEFHE scheme. □

The game $\textsc{hyb}^4_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. In this game, we modify the experiment $\textsc{hyb}^3_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ as follows. Instead of having each party encrypt its input as $\mathsf{ct}_i = \mathsf{TEFHE.Enc}(\mathsf{pk},x_i;r_i^E)$, each party P_i encrypts $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk},0^{\ell_{\mathsf{in}}})$ and computes $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td},\mathsf{ct}_i,x_i)$.

Claim 7.7. $\textsc{hyb}^{3}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \textsc{hyb}^{4}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$.

Proof. Define a sequence of $n+1$ hybrids, where the α'th hybrid, denoted $\textsc{hyb}^{3,\alpha}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ for $\alpha \in [n+1]$, proceeds as $\textsc{hyb}^{3}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$, for chosen inputs $x_1, \ldots, x_n$, with the following difference: When computing the encryption during the first round for party P_i with $i < \alpha$, compute $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$ and set the contents of the random tape to $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$; for $i \geq \alpha$, sample random r_i^E and compute $\mathsf{ct}_i = \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i; r_i^E)$.

The first hybrid is exactly an execution of $\textsc{hyb}^3$, whereas the $(n+1)$'th hybrid is an execution of $\textsc{hyb}^4$. By the security of the TEFHE scheme it holds that every two neighboring hybrids are computationally indistinguishable. Specifically, given an adversary $\mathcal{A}$, an environment $\mathcal{Z}$, and a distinguisher $\mathcal{D}$ that can distinguish between $\textsc{hyb}^{3,\alpha}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ and $\textsc{hyb}^{3,\alpha+1}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ (for some $\alpha \in [n]$) we construct a distinguisher $\mathcal{D}'$ for the *indistinguishability of equivocated randomness* property. Upon receiving $(\tilde{\mathsf{pk}}, \tilde{\mathsf{ct}}, \tilde{\mu}, \tilde{r})$, $\mathcal{D}'$ simulates an execution of $\textsc{hyb}^3$ where the input of party P_i is set to be $x_i = \tilde{\mu}$ with the following difference: For $1 \leq i < \alpha$ the ciphertext of an honest party P_i is computed as $\mathsf{ct}_i \leftarrow \mathsf{TEFHE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{in}}})$ and the encryption coins are set to be $r_i^E \leftarrow \mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}_i, x_i)$; for $\alpha < i \leq n$ the encryption coins r_i^E are uniformly sampled and the ciphertext is computed as $\mathsf{ct}_i = \mathsf{TEFHE.Enc}(\mathsf{pk}, x_i; r_i^E)$; and for $i = \alpha$ the ciphertext is set to be $\mathsf{ct}_i = \tilde{\mathsf{ct}}$ and the encryption coins to $r_i^E = \tilde{r}$. Next, $\mathcal{D}'$ continues the simulation of $\textsc{hyb}^3$ and finally outputs whatever $\mathcal{D}$ outputs.

In case $(\tilde{\mathsf{pk}}, \tilde{\mathsf{ct}}, \tilde{\mu}, \tilde{r})$ is computed using $\mathsf{TEFHE.Gen}$ and $\mathsf{TEFHE.Enc}$ the execution is identically distributed as $\textsc{hyb}^{3,\alpha}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$ whereas in case $(\tilde{\mathsf{pk}}, \tilde{\mathsf{ct}}, \tilde{\mu}, \tilde{r})$ is computed using $\mathsf{TEFHE.GenEquiv}$ and $\mathsf{TEFHE.Equiv}$ the execution is identically distributed as $\textsc{hyb}^{3,\alpha+1}_{\pi_{\mathsf{hm}},\mathcal{A},\mathcal{Z}}$. Therefore, the claim follows by the security of the TEFHE scheme via a standard hybrid argument. □

The execution in $\textsc{hyb}^4$ is exactly like the simulation done by $\mathcal{S}$ in the ideal model; therefore, the theorem follows from Claims 7.4 to 7.7. □

7.3 Malicious Security

We proceed to prove security against malicious and adaptive adversaries in an analog way to Section 6.3, using the compiler from semi-malicious security to malicious security. Note that the output of the compiled protocol is the output of the underlying, semi-maliciously secure protocol, where malicious parties who cheat in their proofs are considered as aborting parties. Thus, if the underlying protocol guarantees output delivery, so does the compiled protocol. As in the all-but-one case, we require all communication to be sent over a broadcast channel; hence, the secret shares of the smudging noise should be encrypted using non-committing encryption (see Appendix A.2.5). We consider the distribution of the NCE public keys as part of the threshold-PKI functionality.

Theorem 7.8 (Theorem 1.5, restated)**.** *Assume the existence of special $n/2$-out-of-n TEFHE schemes and of NCE schemes, let $t < n/2$, and let $f : (\{0,1\}^{\ell_{in}})^n \rightarrow \{0,1\}^{\ell_{out}}$ be an efficiently computable function of depth d. Then, $\mathcal{F}^{f}_{\mathsf{sfe\text{-}god}}$ can be UC-realized in the $(\mathcal{F}_{\mathsf{thresh\text{-}pki}}, \mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{nizk}})$-hybrid model, tolerating an adaptive, malicious PPT t-adversary, by a two-round protocol with communication complexity* $\mathrm{poly}(\ell_{in}, \ell_{out}, d, \kappa, n)$.

By instantiating the threshold-PKI functionality using the adaptively secure protocol of Damgård and Ishai [42], we get a protocol that is defined in the plain model.

Corollary 7.9. *Consider the same assumptions as in Theorem 7.8. Then, $\mathcal{F}^{f}_{\mathsf{sfe\text{-}god}}$ can be UC-realized in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{nizk}})$-hybrid model, tolerating an adaptive, malicious PPT t-adversary, by a constant-round protocol with communication complexity* $\mathrm{poly}(\ell_{in}, \ell_{out}, d, \kappa, n)$.

Acknowledgments

We thank the anonymous reviewers for many helpful comments.

Bibliography

[1] M. Ajtai. Generating hard instances of the short basis problem. In *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–9, 1999.

[2] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science STACS*, pages 75–86, 2009.

[3] P. Ananth, A. R. Choudhuri, A. Goel, and A. Jain. Round-optimal secure multiparty computation with honest majority. In *Advances in Cryptology – CRYPTO 2018, part II*, pages 395–424, 2018.

[4] P. Ananth, S. Badrinarayanan, A. Jain, N. Manohar, and A. Sahai. From FE combiners to secure MPC and back. In *Proceedings of the 17th Theory of Cryptography Conference, TCC 2019, part I*, pages 199–228, 2019.

[5] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology – EUROCRYPT 2012*, pages 483–501, 2012.

[6] S. Badrinarayanan, A. Jain, N. Manohar, and A. Sahai. Secure MPC: laziness leads to GOD. In *Advances in Cryptology – ASIACRYPT 2020, part III*, pages 120–150, 2020.

[7] B. Barak and A. Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 543–552, 2005.

[8] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 186–195, 2004.

[9] D. Beaver. Plug and play encryption. In *Advances in Cryptology – CRYPTO '97*, pages 75–89, 1997.

[10] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology – EUROCRYPT '92*, pages 307–323, 1992.

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[12] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, pages 169–188, 2011.

[13] F. Benhamouda, H. Lin, A. Polychroniadou, and M. Venkitasubramaniam. Two-round adaptively secure multiparty computation from standard assumptions. In *Proceedings of the 16th Theory of Cryptography Conference, TCC 2018, part I*, pages 175–205, 2018.

[14] N. Bitansky, S. Garg, H. Lin, R. Pass, and S. Telang. Succinct randomized encodings and their applications. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 439–448, 2015.

[15] N. Bitansky, R. Canetti, S. Garg, J. Holmgren, A. Jain, H. Lin, R. Pass, S. Telang, and V. Vaikuntanathan. Indistinguishability obfuscation for RAM programs and succinct randomized encodings. *SIAM Journal on Computing*, 47(3):1123–1210, 2018.

[16] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT 2004*, pages 223–238, 2004.

[17] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Advances in Cryptology – CRYPTO 2018, part I*, pages 565–596, 2018.

[18] E. Boyle, N. Gilboa, and Y. Ishai. Breaking the circuit size barrier for secure computation under DDH. In *Advances in Cryptology – CRYPTO 2016, part I*, pages 509–539, 2016.

[19] E. Boyle, N. Gilboa, and Y. Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *Advances in Cryptology – EUROCRYPT 2017, part II*, pages 163–193, 2017.

[20] E. Boyle, R. Cohen, D. Data, and P. Hubáček. Must the communication graph of MPC protocols be an expander? In *Advances in Cryptology – CRYPTO 2018, part III*, pages 243–272, 2018.

[21] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13 (1):143–202, 2000.

[22] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[23] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648, 1996.

[24] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2002.

[25] R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, and T. Malkin. Adaptive versus non-adaptive security of multi-party protocols. *Journal of Cryptology*, 17(3):153–207, 2004.

[26] R. Canetti, R. Pass, and A. Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *Proceedings of the 48th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 249–259, 2007.

[27] R. Canetti, S. Goldwasser, and O. Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part II*, pages 557–585, 2015.

[28] R. Canetti, J. Holmgren, A. Jain, and V. Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 429–437, 2015.

[29] R. Canetti, O. Poburinnaya, and M. Venkitasubramaniam. Better two-round adaptive multi-party computation. In *Proceedings of the 20th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 396–427, 2017.

[30] R. Canetti, O. Poburinnaya, and M. Venkitasubramaniam. Equivocating yao: constant-round adaptively secure multiparty computation in the plain model. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 497–509, 2017.

[31] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[32] C. Cho, N. Döttling, S. Garg, D. Gupta, P. Miao, and A. Polychroniadou. Laconic oblivious transfer and its applications. In *Advances in Cryptology – CRYPTO 2017, part II*, pages 33–65, 2017.

[33] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Improved non-committing encryption with applications to adaptively secure protocols. In *Advances in Cryptology – ASIACRYPT 2009*, pages 287–302, 2009.

[34] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

[35] R. Cohen. Asynchronous secure multiparty computation in constant time. In *Proceedings of the 19th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 183–207, 2016.

[36] R. Cohen and Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, 2017.

[37] R. Cohen and C. Peikert. On adaptively secure multiparty computation with a short CRS. In *Proceedings of the 10th Conference on Security and Cryptography for Networks (SCN)*, pages 129–146, 2016.

[38] R. Cohen, J. A. Garay, and V. Zikas. Completeness theorems for adaptively secure broadcast, 2021. https://eprint.iacr.org/2021/775.

[39] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology – EUROCRYPT '99*, pages 311–326, 1999.

[40] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In *Advances in Cryptology – ASIACRYPT 2013, part I*, pages 316–336, 2013.

[41] D. Dachman-Soled, J. Katz, and V. Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part II*, pages 586–613, 2015.

[42] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology – CRYPTO 2005*, pages 378–394, 2005.

[43] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Advances in Cryptology – CRYPTO 2000*, pages 432–450, 2000.

[44] I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology – CRYPTO 2003*, pages 247–264, 2003.

[45] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO 2012*, pages 643–662, 2012.

[46] I. Damgård, A. Polychroniadou, and V. Rao. Adaptively secure multi-party computation from LWE (via equivocal FHE). In *Proceedings of the 19th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 208–233, 2016.

[47] C. Ganesh, Y. Kondi, A. Patra, and P. Sarkar. Efficient adaptively secure zero-knowledge from garbled circuits. In *Proceedings of the 21th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 499–529, 2018.

[48] J. A. Garay, D. Wichs, and H. Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *Advances in Cryptology – CRYPTO 2009*, pages 505–523, 2009.

[49] J. A. Garay, Y. Ishai, R. Ostrovsky, and V. Zikas. The price of low communication in secure multi-party computation. In *Advances in Cryptology – CRYPTO 2017, part I*, pages 420–446, 2017.

[50] S. Garg and A. Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part II*, pages 614–637, 2015.

[51] S. Garg and A. Sahai. Adaptively secure multi-party computation with dishonest majority. In *Advances in Cryptology – CRYPTO 2012*, pages 105–123, 2012.

[52] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-round secure multiparty computation. In *Advances in Cryptology – CRYPTO 2002*, pages 178–193, 2002.

[53] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178, 2009.

[54] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 99–108, 2011.

[55] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 197–206, 2008.

[56] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology – CRYPTO 2013, part I*, pages 75–92, 2013.

[57] C. Gentry, J. Groth, Y. Ishai, C. Peikert, A. Sahai, and A. D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, 28(4):820–843, 2015.

[58] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[59] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 469–477, 2015.

[60] S. D. Gordon, F. Liu, and E. Shi. Constant-round MPC with fairness and guarantee of output delivery. In *Advances in Cryptology – CRYPTO 2015, part II*, pages 63–82, 2015.

[61] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology – ASIACRYPT 2010*, pages 321–340, 2010.

[62] J. Groth, R. Ostrovsky, and A. Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM*, 59(3):11:1–11:35, 2012.

[63] C. Hazay and A. Patra. Efficient one-sided adaptively secure computation. *Journal of Cryptology*, 30 (1):321–371, 2017.

[64] C. Hazay and M. Venkitasubramaniam. On the power of secure two-party computation. In *Advances in Cryptology – CRYPTO 2016, part II*, pages 397–429, 2016.

[65] C. Hazay and M. Venkitasubramaniam. Composable adaptive secure protocols without setup under polytime assumptions. In *Proceedings of the 14th Theory of Cryptography Conference, TCC 2016-B, part I*, pages 400–432, 2016.

[66] C. Hazay, Y. Lindell, and A. Patra. Adaptively secure computation with partial erasures. In *Proceedings of the 34th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 291–300, 2015.

[67] C. Hazay, A. Polychroniadou, and M. Venkitasubramaniam. Constant round adaptively secure protocols in the tamper-proof hardware model. In *Proceedings of the 20th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 428–460, 2017.

[68] B. Hemenway, R. Ostrovsky, and A. Rosen. Non-committing encryption from Φ-hiding. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part I*, pages 591–608, 2015.

[69] B. Hemenway, R. Ostrovsky, S. Richelson, and A. Rosen. Adaptive security with quasi-optimal rate. In *Proceedings of the 13th Theory of Cryptography Conference, TCC 2016-A, part I*, pages 525–541, 2016.

[70] M. Hirt and V. Zikas. Adaptively secure broadcast. In *Advances in Cryptology – EUROCRYPT 2010*, pages 466–485, 2010.

[71] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology – CRYPTO 2008*, pages 572–591, 2008.

[72] Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *Proceedings of the 6th Theory of Cryptography Conference, TCC 2009*, pages 294–314, 2009.

[73] Y. Ishai, O. Pandey, and A. Sahai. Public-coin differing-inputs obfuscation and its applications. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part II*, pages 668–697, 2015.

[74] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In *Advances in Cryptology – CRYPTO 2004*, pages 335–354, 2004.

[75] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. In *Proceedings of the 10th Theory of Cryptography Conference, TCC 2013*, pages 477–498, 2013.

[76] J. Katz, A. Thiruvengadam, and H. Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In *Proceedings of the 16th International Conference on the Theory and Practice of Public-Key Cryptography (PKC)*, pages 14–31, 2013.

[77] Y. Lindell. Adaptively secure two-party computation with erasures. In *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA)*, pages 117–132, 2009.

[78] Y. Lindell and H. Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *Journal of Cryptology*, 24(4):761–799, 2011.

[79] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, 2012.

[80] P. Mukherjee and D. Wichs. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology – EUROCRYPT 2016, part II*, pages 735–763, 2016.

[81] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 427–437, 1990.

[82] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO 2002*, pages 111–126, 2002.

[83] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology – CRYPTO 2012*, pages 681–700, 2012.

[84] A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In *Advances in Cryptology – CRYPTO 2011*, pages 525–542, 2011.

[85] A. Patra and D. Ravi. On the exact round complexity of secure three-party computation. In *Advances in Cryptology – CRYPTO 2018, part II*, pages 425–458, 2018.

[86] W. Quach, H. Wee, and D. Wichs. Laconic function evaluation and applications. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 859–870, 2018.

[87] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

[88] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 84–93, 2005.

[89] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[90] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 2014.

[91] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[92] M. Venkitasubramaniam. On adaptively secure protocols. In *Proceedings of the 9th Conference on Security and Cryptography for Networks (SCN)*, pages 455–475, 2014.

[93] A. C. Yao. How to generate and exchange secrets (extended abstract). In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.

A Preliminaries (Cont'd)

In Appendix A.1, we define the LWE and adaptive LWE assumptions. In Appendix A.2, we formally define the cryptographic primitives used in the paper.

A.1 Cryptographic Assumptions

A.1.1 Learning With Errors

The decisional learning with errors (LWE) problem, introduced by Regev [88], is defined as follows.

Definition A.1 (decision LWE)**.** *Let $n = n(\kappa)$ and $q = q(\kappa)$ be integer parameters and $\chi = \chi(\kappa)$ be a distribution over $\mathbb{Z}$. The learning with errors (LWE) assumption $\mathsf{LWE}_{n,q,\chi}$ states that for all polynomials $m = \mathrm{poly}(\kappa)$ the following distributions are computationally indistinguishable:*

$$(\boldsymbol{A}, \boldsymbol{s}^T\boldsymbol{A} + \boldsymbol{e}) \overset{\mathrm{c}}{\equiv} (\boldsymbol{A}, \boldsymbol{u}),$$

where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{e} \leftarrow \chi^m$, and $\boldsymbol{u} \leftarrow \mathbb{Z}_q^m$.

We rely on LWE security with the following range of parameters. We assume that for any polynomial $p = p(\kappa) = \mathrm{poly}(\kappa)$ there exists some polynomial $n = n(\kappa) = \mathrm{poly}(\kappa)$, some $q = q(\kappa) = 2^{\mathrm{poly}(\kappa)}$, and some $B = B(\kappa)$-bounded distribution $\chi = \chi(\kappa)$ such that $q/B \geq 2^p$ and the $\mathsf{LWE}_{n,q,\chi}$ assumption holds. Throughout the paper, the LWE assumption without further specification refers to the above parameters. The sub-exponentially secure LWE assumption further assumes that $\mathsf{LWE}_{n,q,\chi}$ with the above parameters is sub-exponentially secure, meaning that there exists some $\epsilon > 0$ such that the distinguishing advantage of any polynomial-time distinguisher is $2^{-\kappa^\epsilon}$.

A.1.2 Adaptive Learning With Errors

Quach et al. [86] used the following natural variant of the LWE problem, denoted *adaptive LWE.*

Definition A.2 (decision ALWE)**.** *We define the decision adaptive LWE assumption $\mathsf{ALWE}_{n,k,q,\chi}$ with parameter $n, k, q \in \mathbb{Z}$ and a distribution χ over $\mathbb{Z}$ which are all parametrized by the security parameter κ. Let $m = n \cdot \lceil \log q \rceil$. We let $\boldsymbol{G} \in \mathbb{Z}^{n\times m}$ be the gadget matrix (as defined in [79], see also Appendix C.1). For any polynomial $m' = m'(\kappa)$, we consider the following two games $\textsc{game}^\beta$ for $\beta \in \{0,1\}$, between a challenger and an adversary $\mathcal{A}$.*

- *The Challenger picks k random matrices $A_i \leftarrow \mathbb{Z}_q^{n\times m}$ for $i \in [k]$, and sends them to $\mathcal{A}$.*
- *$\mathcal{A}$ adaptively picks $x_1, \ldots, x_k \in \{0,1\}$, and sends it to the Challenger.*
- *The Challenger samples $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and computes for all $i \in [k]$*

$$\begin{cases} \boldsymbol{b}_i = \boldsymbol{s}^T(\boldsymbol{A}_i - x_i \cdot \boldsymbol{G}) + \boldsymbol{e}_i \text{ where } \boldsymbol{e}_i \leftarrow \chi^m, & \text{if } \beta = 0. \\ \boldsymbol{b}_i \leftarrow \mathbb{Z}_q^m, & \text{if } \beta = 1. \end{cases}$$

  The Challenger also picks $\boldsymbol{A}_{k+1} \leftarrow \mathbb{Z}_q^{n\times m'}$ and computes

$$\begin{cases} \boldsymbol{b}_{k+1} = \boldsymbol{s}^T\boldsymbol{A}_{k+1} + \boldsymbol{e}_{k+1} \text{ where } \boldsymbol{e}_{k+1} \leftarrow \chi^{m'}, & \text{if } \beta = 0. \\ \boldsymbol{b}_{k+1} \leftarrow \mathbb{Z}_q^{m'}, & \text{if } \beta = 1. \end{cases}$$

  The challenger sends $\boldsymbol{A}_{k+1}$ and $\{\boldsymbol{b}_i\}_{i\in[k+1]}$ to the adversary.

The $\mathsf{ALWE}_{n,k,q,\chi}$ assumption states that for all polynomial $m = m(\kappa)$, the games $\textsc{game}^0$ and $\textsc{game}^1$ are computationally indistinguishable.

A.2 Cryptographic Primitives

A.2.1 Laconic Function Evaluation

Quach et al. [86] introduced the notion of laconic function evaluation (LFE), and constructed an LFE scheme for all circuits under the adaptive LWE assumption. At a high level, LFE allows to "compress" a function into a short digest that enables encrypting the input to the function as a short ciphertext. The size of the digest and the ciphertext (and therefore the computational cost of the encryption algorithm) only depend on the depth of the circuit representing the function, and are independent of the circuit size.

Quach et al. [86] considered another variant of LFE that is *function-hiding*, meaning that the digest hides all partial information about the underlying function. Following [86], we assume that the circuit class $\mathcal{C}$ associates every circuit $C \in \mathcal{C}$ with some circuit parameters $C.\mathsf{params}$ that remain unhidden. Unless specified otherwise, we will consider $\mathcal{C}$ to be the class of all circuits with $C.\mathsf{params} = (1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, 1^d)$ consisting of the input size ℓ_{in}, output size ℓ_{out}, and the depth d of the circuit.

Definition A.3 (LFE)**.** *A* laconic function evaluation *(LFE) scheme for a class of circuits $\mathcal{C}$ consists of four algorithm* $\Pi = (\mathsf{LFE.crsGen}, \mathsf{LFE.Compress}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$*.*

- $\mathsf{LFE.crsGen}(1^\kappa, \mathsf{params}) \to \mathsf{crs}$*: given an input the security parameter and circuit parameters* params *the CRS generation algorithm outputs a uniformly random common random string* crs*.*
- $\mathsf{LFE.Compress}(\mathsf{crs}, C) \to \mathsf{digest}_C$*: given as input the common random string* crs *and a circuit* $C \in \mathcal{C}$*, the compression algorithm outputs a digest* digest_C*.*
- $\mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_C, x) \to \mathsf{ct}$*: given as input the common random string* crs*, a digest* digest_C*, and a message x, the encryption algorithm outputs a ciphertext* ct*.*
- $\mathsf{LFE.Dec}(\mathsf{crs}, C, r, \mathsf{ct}) \to y$*: given as input the common random string* crs*, a circuit $C \in \mathcal{C}$, the compression random coins r, and a ciphertext* ct*, the deterministic decryption algorithm outputs a message y.*

We require the following properties from an LFE scheme Π:

Correctness. For every security parameter κ, parameters params, and circuit $C \in \mathcal{C}$ with $C.\mathsf{params} = \mathsf{params}$ it holds that:

$$\Pr\left[y = C(x) \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params}) \\ \mathsf{digest}_C = \mathsf{LFE.Compress}(\mathsf{crs}, C; r) \\ \mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_C, x) \\ y = \mathsf{LFE.Dec}(\mathsf{crs}, C, r, \mathsf{ct}) \end{array} \right] = 1.$$

Security. There exists a PPT simulator $\mathsf{Sim}_{\mathrm{LFE}}$ for the scheme Π such that for every stateful PPT adversary $\mathcal{A}$, it holds that

$$\left| \Pr\left[\mathsf{Expt}^{\mathsf{LFE\text{-}real}}_{\Pi,\mathcal{A}}(\kappa) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}_{\Pi,\mathcal{A}}(\kappa) = 1\right] \right| \le \mathsf{negl}(\kappa),$$

for the experiments $\mathsf{Expt}^{\mathsf{LFE\text{-}real}}$ and $\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}$ defined below:

$\mathsf{Expt}^{\mathsf{LFE\text{-}real}}_{\Pi,\mathcal{A}}(\kappa)$	$\mathsf{Expt}^{\mathsf{LFE\text{-}ideal}}_{\Pi,\mathcal{A}}(\kappa)$
$\mathsf{params} \leftarrow \mathcal{A}(1^\kappa)$	$\mathsf{params} \leftarrow \mathcal{A}(1^\kappa)$
$\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$	$\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$
$x^*, C, r \leftarrow \mathcal{A}(\mathsf{crs})$	$x^*, C, r \leftarrow \mathcal{A}(\mathsf{crs})$
s.t. $C \in \mathcal{C}$ and $C.\mathsf{params} = \mathsf{params}$	s.t. $C \in \mathcal{C}$ and $C.\mathsf{params} = \mathsf{params}$
$\mathsf{digest}_C = \mathsf{LFE.Compress}(\mathsf{crs}, C; r)$	$\mathsf{digest}_C = \mathsf{LFE.Compress}(\mathsf{crs}, C; r)$
$\mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{crs}, \mathsf{digest}_C, x^*)$	$\mathsf{ct} \leftarrow \mathsf{Sim}_{\text{LFE}}(\mathsf{crs}, C, r, \mathsf{digest}_C, C(x^*))$
Output $\mathcal{A}(\mathsf{ct})$	Output $\mathcal{A}(\mathsf{ct})$

(Statistical) Function-Hiding. An LFE scheme Π is function hiding if there exists a PPT simulator Sim_{FH} for the scheme Π such that for all stateful PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathsf{Expt}^{\mathsf{FH\text{-}real}}_{\Pi,\mathcal{A}}(\kappa) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{FH\text{-}ideal}}_{\Pi,\mathcal{A}}(\kappa) = 1\right]\right| \leq \mathsf{negl}(\kappa),$$

for the experiments $\mathsf{Expt}^{\mathsf{FH\text{-}real}}$ and $\mathsf{Expt}^{\mathsf{FH\text{-}ideal}}$ defined below:

$\mathsf{Expt}^{\mathsf{FH\text{-}real}}_{\Pi,\mathcal{A}}(\kappa)$	$\mathsf{Expt}^{\mathsf{FH\text{-}ideal}}_{\Pi,\mathcal{A}}(\kappa)$
$\mathsf{params} \leftarrow \mathcal{A}(1^\kappa)$	$\mathsf{params} \leftarrow \mathcal{A}(1^\kappa)$
$\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$	$\mathsf{crs} \leftarrow \mathsf{LFE.crsGen}(1^\kappa, \mathsf{params})$
$C \leftarrow \mathcal{A}(\mathsf{crs})$	$C \leftarrow \mathcal{A}(\mathsf{crs})$
s.t. $C \in \mathcal{C}$ and $C.\mathsf{params} = \mathsf{params}$	s.t. $C \in \mathcal{C}$ and $C.\mathsf{params} = \mathsf{params}$
$\mathsf{digest}_C \leftarrow \mathsf{LFE.Compress}(\mathsf{crs}, C)$	$\mathsf{digest}_C \leftarrow \mathsf{Sim}_{\text{FH}}(\mathsf{crs}, C.\mathsf{params})$
Output $\mathcal{A}(\mathsf{digest}_C)$	Output $\mathcal{A}(\mathsf{digest}_C)$

Compactness. Consider the class $\mathcal{C}$ of all circuits with $C.\mathsf{params} = (1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}, 1^d)$ consisting of the input size ℓ_{in}, the output length ℓ_{out}, and the depth d of the circuit. We say the LFE scheme is compact if

- The CRS is of size $\text{poly}(\kappa, \ell_{\mathsf{in}}, d)$. The digest is of size $\text{poly}(\kappa)$.
- The running time of the encryption algorithm and the size of its output (the ciphertext) are $\tilde{O}(\ell_{\mathsf{out}}) \cdot \text{poly}(\kappa, \ell_{\mathsf{in}}, d)$.
- The running time of the compression and the decryption algorithms is $\tilde{O}(|C|) \cdot \text{poly}(\kappa, \ell_{\mathsf{in}}, d)$.

Theorem A.4 ([86]). *Assuming sub-exponential hardness of LWE there exists a function-hiding LFE scheme that is compact for circuits C with $C.\mathsf{params} = (1^{\ell_{in}}, 1^{\ell_{out}}, 1^d)$ of depth d, input size ℓ_{in}, and output size ℓ_{out} such that the CRS is a uniform random string. Further, if the function-hiding property is not required, the compression algorithm* $\mathsf{LFE.Compress}$ *can be made deterministic.*

A.2.2 Explainability Compiler

The notion of an explainability compiler was introduced by Dachman-Soled et al. [41] in the context of adaptively secure MPC as an extension of the technique of Sahai and Waters [90] for constructing sender-deniable encryption. At a high-level, the compiler can take any randomized algorithm Alg and produce an algorithm $\widetilde{\mathsf{Alg}}$ with the same functionality and has roughly the same size, along with an $\mathsf{Explain}$ algorithm. For any input/output pair (x, y), the $\mathsf{Explain}$ algorithm can produce coins r such that $y = \widetilde{\mathsf{Alg}}(x; r)$. A similar notion was also used by Canetti et al. [27].

Definition A.5 (selective explainability compiler)**.** *A PPT algorithm* Comp *is an explainability compiler with selective security for a circuit class* $\mathcal{C}$ *if for every efficient, randomized circuit* $\mathsf{Alg} \in \mathcal{C}$*, the following hold:*

- ***Polynomial slowdown.*** *There is a polynomial* $p(\cdot)$ *such that, for any* $(\widetilde{\mathsf{Alg}}, \mathsf{Explain})$ *output by* $\mathsf{Comp}(1^\kappa, \mathsf{Alg})$ *it holds that* $|\widetilde{\mathsf{Alg}}| \leq p(\kappa) \cdot |\mathsf{Alg}|$*.*
- ***Statistical functional equivalence.*** *With overwhelming probability over the choice of* $(\widetilde{\mathsf{Alg}}, \cdot)$ *as output by* $\mathsf{Comp}(1^\kappa, \mathsf{Alg})$*, the distribution of* $\widetilde{\mathsf{Alg}}(x)$ *is statistically close to the distribution of* $\mathsf{Alg}(x)$ *for every input* x*.*
- ***Explainability.*** *For every stateful PPT adversary* $\mathcal{A}$ *it holds that*

  $$\left|\Pr\left[\mathsf{Expt}^{\mathsf{Explain\text{-}Static}}_{\mathsf{Comp},\mathsf{Alg},\mathcal{A}}(\kappa) = 1\right]\right| \leq 1/2 + \mathsf{negl}(\kappa),$$

  for the experiment $\mathsf{Expt}^{\mathsf{Explain\text{-}Static}}$ *defined below:*

$\mathsf{Expt}^{\mathsf{Explain\text{-}Static}}_{\mathsf{Comp},\mathsf{Alg},\mathcal{A}}(\kappa)$
$x^* \leftarrow \mathcal{A}(1^\kappa)$
$(\widetilde{\mathsf{Alg}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{Alg})$
Sample $r_0 \leftarrow \{0,1\}^*$
Compute $y^* = \widetilde{\mathsf{Alg}}(x^*; r_0)$
Compute $r_1 \leftarrow \mathsf{Explain}(x^*, y^*)$
Sample $b \leftarrow \{0,1\}$
Compute $b' \leftarrow \mathcal{A}(\widetilde{\mathsf{Alg}}, y^*, r_b)$
Output 1 *if and only if* $b' = b$

Theorem A.6 ([41])**.** *Assuming the existence of an indistinguishable obfuscator for* $\mathsf{P/poly}$ *and of one-way functions, there exists an explainability compiler with selective security for* $\mathsf{P/poly}$*.*

The definition and construction in [41] achieve selective security in the sense that the adversary must choose the challenge input x^* *before* learning the compiled algorithm $\widetilde{\mathsf{Alg}}$. In some of our constructions, it will be simpler to use explainability compilers with adaptive security, where the adversary can choose the challenge input *after* seeing $\widetilde{\mathsf{Alg}}$. We denote the adaptive game as $\mathsf{Expt}^{\mathsf{Explain\text{-}Adapt}}_{\mathsf{Comp},\mathsf{Alg},\mathcal{A}}(\kappa)$. Adaptive security follows from selective security via complexity leveraging [16] by assuming sub-exponential security of the cryptographic primitives.

$\mathsf{Expt}^{\mathsf{Explain\text{-}Adapt}}_{\mathsf{Comp},\mathsf{Alg},\mathcal{A}}(\kappa)$
$(\widetilde{\mathsf{Alg}}, \mathsf{Explain}) \leftarrow \mathsf{Comp}(1^\kappa, \mathsf{Alg})$
$x^* \leftarrow \mathcal{A}(1^\kappa, \widetilde{\mathsf{Alg}})$
Sample $r_0 \leftarrow \{0,1\}^*$
Compute $y^* = \widetilde{\mathsf{Alg}}(x^*; r_0)$
Compute $r_1 \leftarrow \mathsf{Explain}(x^*, y^*)$
Sample $b \leftarrow \{0,1\}$
Compute $b' \leftarrow \mathcal{A}(y^*, r_b)$
Output 1 if and only if $b' = b$

Corollary A.7. *Assuming the existence of an indistinguishable obfuscator for* $\mathsf{P/poly}$ *and of one-way functions, both with sub-exponential security, there exists an explainability compiler with adaptive security for* $\mathsf{P/poly}$*.*

A.2.3 Homomorphic Trapdoor Functions

Homomorphic trapdoor functions (HTDFs) were introduced by [59] as a unification of homomorphic encryption and homomorphic signatures.

Definition A.8 (HTDF [59])**.** *A homomorphic trapdoor function (HTDF) consists of the following five polynomial-time algorithms* $(\mathsf{HTDF.Gen}, f, \mathsf{HTDF.Inv}, \mathsf{HTDF.Eval}^{\mathsf{in}}, \mathsf{HTDF.Eval}^{\mathsf{out}})$ *with the following syntax:*

- $\mathsf{HTDF.Gen}(1^\kappa, 1^d) \to (\mathsf{pk}, \mathsf{sk})$*: given an input the security parameter and the depth-bound, the key-generation procedure outputs a public key* pk *and a secret key* sk*. The security parameter defines the* index space $\mathcal{X}$*, the* input space $\mathcal{U}$*, the* output space $\mathcal{V}$ *and some efficiently sampleable input distribution* $D_{\mathcal{U}}$ *over* $\mathcal{U}$*. We require that membership in the sets* $\mathcal{U}, \mathcal{V}, \mathcal{X}$ *can be efficiently tested and that one can efficiently sample uniformly at random from* $\mathcal{V}$*.*

- $f_{\mathsf{pk},x} : \mathcal{U} \to \mathcal{V}$*: the algorithm* f *is parametrized by a public key* pk *and an index* $x \in \mathcal{X}$*.*

- $\mathsf{HTDF.Inv}_{\mathsf{sk},x} : \mathcal{V} \to \mathcal{U}$*: the algorithm* $\mathsf{HTDF.Inv}$ *is parametrized by a secret key* pk *and an index* $x \in \mathcal{X}$*.*

- $u^* = \mathsf{HTDF.Eval}^{\mathsf{in}}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ *and* $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(g, v_1, \ldots, v_\ell)$ *are deterministic input/output homomorphic-evaluation algorithms. The algorithms take as input some function* $g : \mathcal{X}^\ell \to \mathcal{X}$ *and values* $x_i \in \mathcal{X}, u_i \in \mathcal{U}, v_i \in \mathcal{V}$*. The outputs are* $u^* \in \mathcal{U}$ *and* $v^* \in \mathcal{V}$*.*

We require the following properties from an HTDF scheme:

- ***Correctness.*** *Let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d)$*, let* $x_1, \ldots, x_\ell \in \mathcal{X}$*, let* $g : \mathcal{X}^\ell \to \mathcal{X}$ *of depth at most* d*, and let* $y := g(x_1, \ldots, x_\ell)$*. Let* $u_1, \ldots, u_\ell \in \mathcal{U}$ *and set* $v_i := f_{pk,x_i}(u_i)$ *for* $i \in [\ell]$*. Let* $u^* = \mathsf{HTDF.Eval}^{\mathsf{in}}(g, (x_1, u_1), \ldots, (x_\ell, u_\ell))$ *and let* $v^* = \mathsf{HTDF.Eval}^{\mathsf{out}}(g, v_1, \ldots, v_\ell)$*. Then, we require that* $u^* \in \mathcal{U}$ *and* $f_{pk,y}(u^*) = v^*$*.*

- ***Distributional equivalence of inversion.*** *The following distributions are statistically close:*

$$\left\{(\mathsf{pk}, \mathsf{sk}, x, u, v) \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d),\ x \in \mathcal{X}, u \leftarrow \mathcal{U}, v = f_{\mathsf{pk},x}(u)\right\}_\kappa$$
$$\overset{s}{\equiv}$$
$$\left\{(\mathsf{pk}, \mathsf{sk}, x, u', v') \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d),\ x \in \mathcal{X}, v' \leftarrow \mathcal{V}, u' = \mathsf{HTDF.Inv}_{\mathsf{sk},x}(v')\right\}_\kappa,$$

  where $x \in \mathcal{X}$ *is an arbitrary random variable that depends on* (pk, sk)*.*

- ***Claw-free security.*** *For every PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{c} f_{\mathsf{pk},x}(u) = f_{\mathsf{pk},x'}(u') \\ u, u' \in \mathcal{U}, \quad x, x' \in \mathcal{X}, \quad x \neq x' \end{array} \middle| \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HTDF.Gen}(1^\kappa, 1^d) \\ (x, x', u, u') \leftarrow \mathcal{A}(1^\kappa, \mathsf{pk}) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

Theorem A.9 ([59])**.** *Under the LWE assumption, there exists an HTDF scheme.*

A.2.4 Strong One-Time Signatures

Definition A.10 (strong one-time signatures)**.** *A strong one-time signatures scheme consists of three polynomial-time algorithms* $(\mathsf{Sig.Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *with the following syntax:*

- $\mathsf{Sig.Gen}(1^\kappa) \to (\mathsf{vk}, \mathsf{sigk})$*: given an input the security parameter, the key-generation procedure outputs a public verification key* vk *and a secret signing key* sigk*.*

- $\mathsf{Sign}_{\mathsf{sigk}}(m) \to \sigma$*: given an input a signing key* sigk *and a message* m*, the signing algorithm outputs a signature* σ*.*

- $b = \mathsf{Vrfy}_{\mathsf{vk}}(\sigma, m)$*: given an input a verification key* vk*, a signature* σ*, and a message* m*, the verification algorithm outputs a bit* $b \in \{0,1\}$*.*

We require the following properties from the signature scheme:

- ***(Perfect) correctness.*** *For every message* $m \in \{0,1\}^*$*, it holds that*

$$\Pr\left[\mathsf{Vrfy}_{\mathsf{vk}}(\sigma, m) = 1 \mid (\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.Gen}(1^\kappa), \sigma \leftarrow \mathsf{Sign}_{\mathsf{sigk}}(m)\right] = 1.$$

- ***Strong existential unforgeability under one-time chosen message attack.*** *For every PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{c|c} & (\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.Gen}(1^\kappa) \\ (m', \sigma') \neq (m, \sigma) & m \leftarrow \mathcal{A}(\mathsf{vk}) \\ \mathsf{Vrfy}_{\mathsf{vk}}(\sigma', m') = 1 & \sigma \leftarrow \mathsf{Sign}_{\mathsf{sigk}}(m) \\ & (m', \sigma') \leftarrow \mathcal{A}(\sigma) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

We will use a strong one-time signature scheme that has fixed-length signatures, i.e., where there is a polynomial upper bound $\ell_{SIG}(\kappa)$ on the length of the signatures. Fixed-length strong one-time signatures can be constructed from one-way functions (from universal one-way hash functions [81] and Lamport signatures used in combination with Merkle trees [89]). The existence of fully homomorphic trapdoor functions therefore implies the existence of fixed-length strong one-time signatures.

A.2.5 Non-Committing Encryption

A non-committing encryption scheme [23, 9, 43] is a public-key encryption scheme with the capability to efficiently simulate a public key and a ciphertext that can be explained as an encryption of any message.

Definition A.11 (NCE)**.** *A non-non-committing (bit) encryption scheme consists of four algorithms* $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec}, \mathsf{NC.Sim})$ *such that the following properties hold:*

- *The triplet* $(\mathsf{NC.Gen}, \mathsf{NC.Enc}, \mathsf{NC.Dec})$ *forms a public-key encryption scheme.*

- $\mathsf{NC.Sim}$ *is a simulation algorithm that on input* 1^κ*, outputs* $(\mathsf{pk}, \mathsf{ct}, \rho_G^0, \rho_E^0, \rho_G^1, \rho_E^1)$*, such that for any* $\mu \in \{0,1\}$ *the following distributions are computationally indistinguishable:*

– *the joint view of an honest sender and an honest receiver in a normal encryption of* μ

$$\{(\mathsf{pk}, \mathsf{ct}, r_G, r_E) \mid (\mathsf{sk}, \mathsf{pk}) = \mathsf{NC.Gen}(1^\kappa; r_G), \mathsf{ct} = \mathsf{NC.Enc}(\mathsf{pk}, \mu; r_E)\},$$

– *the simulated view of an encryption of* μ

$$\left\{(\mathsf{pk}, \mathsf{ct}, \rho_G^\mu, \rho_E^\mu) \mid (\mathsf{pk}, \mathsf{ct}, \rho_G^0, \rho_E^0, \rho_G^1, \rho_E^1) \leftarrow \mathsf{NC.Sim}(1^\kappa)\right\}.$$

NCE schemes exist under wide range of assumptions, including LWE [33, 84, 68, 69].

A.2.6 Fully Homomorphic Encryption

Definition A.12. *A (leveled) fully homomorphic encryption scheme (FHE) consists of* 4 *PPT algorithms:*

- $\mathsf{FHE.Gen}(1^\kappa, 1^d) \rightarrow (\mathsf{pk}, \mathsf{sk})$*: on input the security parameter* κ *and a depth bound* d*, the key generation algorithm outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{FHE.Enc}(\mathsf{pk}, \mu) \rightarrow \mathsf{ct}$*: on input a public key* pk *and a plaintext* $\mu \in \{0,1\}$*, the encryption algorithm outputs a ciphertext* ct*.*
- $\mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \rightarrow \mathsf{ct}$*: on input a public key* pk*, a circuit* $C : \{0,1\}^\ell \rightarrow \{0,1\}$*, and a tuple of ciphertexts* $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$*, the homomorphic-evaluation algorithm outputs a ciphertext* ct*.*
- $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}) \rightarrow \tilde{\mu}$*: on input a secret key* sk *and a ciphertext* ct*, the decryption algorithm outputs* $\tilde{\mu} \in \{0,1\}$*.*

We require the FHE scheme to be *correct*, meaning that when initialized with depth bound d the scheme correctly evaluates all circuits of depth at most d, and *compact*, meaning that the size of the decryption circuit (and of the evaluated ciphertext) is independent of d.

Definition A.13. *Let* $\Pi = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ *be an FHE scheme.*

- Π *is correct if for every depth bound* $d \in \mathbb{N}^+$*, every circuit* $C : \{0,1\}^\ell \rightarrow \{0,1\}$ *of depth at most* d *and every series of inputs* $\mu_1, \ldots, \mu_\ell \in \{0,1\}$ *it holds that*

$$\Pr\left[\mathsf{FHE.Dec}\left(\mathsf{sk}, \mathsf{FHE.Eval}\left(\mathsf{pk}, C, \mathsf{FHE.Enc}(\mathsf{pk}, \mu_1), \ldots, \mathsf{FHE.Enc}(\mathsf{pk}, \mu_\ell)\right)\right) \neq C\left(\mu_1, \ldots, \mu_\ell\right)\right] \\ \leq \mathsf{negl}(\kappa).$$

- Π *is compact if there exists a polynomial* $s(\cdot)$ *such that for every* κ*, every depth bound* d*, every circuit* $C : \{0,1\}^\ell \rightarrow \{0,1\}$ *of depth at most* d*, and every* $\mu_1, \ldots, \mu_\ell \in \{0,1\}$*, the following holds. For* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(1^\kappa, 1^d)$*, ciphertexts* $\mathsf{ct}_j \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, \mu_j)$ *for* $j \in [\ell]$*, and* $\mathsf{ct} \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$*, we have that* $|\mathsf{ct}| \leq s(\kappa)$*.*
- Π *is semantically secure if for every* κ *and every depth bound* d*, it holds that for every PPT adversary* $\mathcal{A}$ *the following experiment* $\mathsf{Expt}^{FHE}_{\mathcal{A},\Pi}(1^\kappa, 1^d)$ *outputs* 1 *with negligible probability.*

  $\mathsf{Expt}^{FHE}_{\mathcal{A},\Pi}(1^\kappa, 1^d)$

1. *On input the security parameter* 1^κ *and depth bound* 1^d*, the challenger generates* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.Gen}(1^\kappa, 1^d)$*, chooses a random* $b \overset{R}{\leftarrow} \{0,1\}$*, and computes the ciphertext* $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, b)$*. Next, the challenger hands* $(\mathsf{pk}, \mathsf{ct})$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs* b'*. The experiments outputs* 1 *if* $b = b'$*.*

By abuse of notation, we consider in the paper encryption strings rather than bits. This should be interpreted as encrypting the string bit by bit.

B The UC Framework

In this section, we describe the UC framework, for more details see [22].

B.1 The Real Model

An execution of a protocol π in the real model consists of n PPT *interactive Turing machines* (ITMs) $P_1, \ldots, P_n$ representing the parties, along with two additional ITMs: an *adversary* $\mathcal{A}$, describing the behavior of the corrupted parties and an *environment* $\mathcal{Z}$, representing the external network environment in which the protocol operates. The environment gives inputs to the honest parties, receives their outputs, and can communicate with the adversary at any point during the execution. The adversary controls the operations of the corrupted parties.

In more details, each ITM is initialized with the security parameter κ and random coins, where the environment receives an additional auxiliary input. The protocol proceeds by a sequence of *activations*, where the environment is activated first and at each point a single ITM is active. When the environment is activated it can read the output tapes of all honest parties and of the adversary, and it can activate one of the parties or the adversary by writing on its input tape. Once a party is activated it can perform a local computation, write on its output tape or send messages to other parties by writing on its outgoing communication tapes. After the party completes its operations the control is returned to the environment. Once the adversary is activated it can send messages on behalf of the corrupted parties or send a message to the environment by writing on its output tape. In addition, $\mathcal{A}$ controls the communication between the parties, and so it can read the contents of the messages on outgoing tapes of honest parties and write messages on their incoming tapes. We assume that only messages that were sent in the past by some party can be delivered, and each message can be delivered at most once.[18] $\mathcal{A}$ can also corrupt an honest party, gain access to all its tapes and control all its actions. Whenever a party is corrupted the environment is notified. If $\mathcal{A}$ wrote on the incoming tape of an honest party, this party is activated next, otherwise the environment is activated. The protocol completes once $\mathcal{Z}$ outputs a single bit.

A *semi-honest* adversary always instructs the corrupted parties to follow the protocol. A *malicious* adversary may instruct the corrupted parties to deviate from the protocol arbitrarily. In this work we also consider *semi-malicious* adversaries [5], that instruct the corrupted parties to follow the protocol but can choose arbitrary random coins for them. Formally, the adversary has a special *witness tape*. In each round of the protocol, whenever the adversary produces a new protocol message m on behalf of some party P_k, it must also write to its special witness tape some

[18] We assume that all the communication is authenticated yet visible to the adversary; formally, we work in the $\mathcal{F}_{\mathsf{auth}}$-hybrid model.

pair (x, r) of input x and randomness r that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r. Note that the witnesses given in different rounds need not be consistent. Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). Lastly, the adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

Let $\textsc{real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z, \boldsymbol{r})$ denote $\mathcal{Z}$'s output on input z and security parameter κ, after interacting with adversary $\mathcal{A}$ and parties $P_1, \ldots, P_n$ running protocol π with random tapes $\boldsymbol{r} = (r_1, \ldots, r_n, r_{\mathcal{A}}, r_{\mathcal{Z}})$ as described above. Let $\textsc{real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ denote the random variable $\textsc{real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z, \boldsymbol{r})$, when the vector $\boldsymbol{r}$ is uniformly chosen.

B.2 The Ideal Model

A computation in the ideal model consists of n *dummy* parties $P_1, \ldots, P_n$, an *ideal-process adversary* (simulator) $\mathcal{S}$, an *environment* $\mathcal{Z}$, and an *ideal functionality* $\mathcal{F}$. As in the real model, the environment gives inputs to the honest (dummy) parties, receives their outputs, and can communicate with the ideal-process adversary at any point during the execution. The dummy parties act as channels between the environment and the ideal functionality, meaning that they send the inputs received from $\mathcal{Z}$ to $\mathcal{F}$ and vice versa. The ideal functionality $\mathcal{F}$ defines the desired behavior of the computation. $\mathcal{F}$ receives the inputs from the dummy parties, executes the desired computation and sends the output to the parties. The ideal-process adversary does not see the communication between the parties and the ideal functionality; however, $\mathcal{S}$ can communicate with $\mathcal{F}$.

Hiding the communication between the ideal functionality and the parties from the adversary may be too restrictive; it is often desired to provide the adversary the power to determine *when* a party will receive the message. We say that the ideal functionality $\mathcal{F}$ sends a *delayed output* v to a party P if $\mathcal{F}$ first sends to the adversary a message that it is ready to generate an output to P. In case the output is public $\mathcal{F}$ sends v to the adversary. When the adversary replies to the message, $\mathcal{F}$ outputs the value v to P.[19]

Let $\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z, \boldsymbol{r})$ denote $\mathcal{Z}$'s output on input z and security parameter κ, after interacting with ideal-process adversary $\mathcal{S}$ and dummy parties $P_1, \ldots, P_n$ that interact with ideal functionality $\mathcal{F}$ with random tapes $\boldsymbol{r} = (r_{\mathcal{S}}, r_{\mathcal{Z}})$ as described above. Let $\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z)$ denote the random variable $\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa, z, \boldsymbol{r})$, when the vector $\boldsymbol{r}$ is uniformly chosen.

Definition B.1. *We say that a protocol π UC-realizes an ideal functionality $\mathcal{F}$ in the presence of adaptive malicious (resp., semi-malicious) adversaries, if for any* PPT *adaptive malicious (resp., semi-malicious) adversary $\mathcal{A}$ and any* PPT *environment $\mathcal{Z}$, there exists a* PPT *ideal-process adversary $\mathcal{S}$ such that the following two distribution ensembles are computationally indistinguishable*

$$\left\{\textsc{real}_{\pi,\mathcal{A},\mathcal{Z}}\left(\kappa, z\right)\right\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*} \overset{\text{c}}{\equiv} \left\{\textsc{ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}\left(\kappa, z\right)\right\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}.$$

[19]The ideal-process adversary may never release messages from the ideal functionality to the dummy parties and so termination of the computation is not guaranteed. In order to rule out trivial protocols that never produce output, we follow [24] and consider *non-trivial protocols* that have the following property: if the real-model adversary delivers all messages and does not corrupt any parties, then the ideal-process adversary also delivers all messages and does not corrupt any parties. We note that using techniques from [75] guaranteed termination can be enforced.

B.3 The Hybrid Model

The $\mathcal{F}$*-hybrid model* is a combination of the real and ideal models, it extends the real model with an ideal functionality $\mathcal{F}$. The parties communicate with each other in exactly the same way as in the real model described above; however, they can interact with $\mathcal{F}$ as in the ideal model. An important property of the UC framework is that the ideal functionality $\mathcal{F}$ in a $\mathcal{F}$-hybrid model can be replaced with a protocol that UC-realizes $\mathcal{F}$.

Let the global output $\text{HYBRID}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, z)$ denote $\mathcal{Z}$'s output on input z and security parameter κ, after interacting in a $\mathcal{F}$-hybrid model with adversary $\mathcal{A}$ and parties $P_1, \ldots, P_n$ with uniformly distributed random tapes $\boldsymbol{r} = (r_1, \ldots, r_n, r_{\mathcal{A}}, r_{\mathcal{Z}})$ running protocol π.

Theorem B.2 (Canetti [22]). *Let $\mathcal{F}$ be an ideal functionality and let ρ be a protocol that UC-realizes $\mathcal{F}$ in the presence of adaptive malicious (resp., semi-malicious) adversaries, and let π be a protocol that UC-realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model in the presence of adaptive malicious (resp., semi-malicious) adversaries. Then for any* PPT *adaptive malicious (resp., semi-malicious) real-model adversary $\mathcal{A}$ and any* PPT *environment $\mathcal{Z}$, there exists a* PPT *adaptive malicious (resp., semi-malicious) adversary $\mathcal{S}$ in the $\mathcal{F}$-hybrid model such that*

$$\left\{\text{REAL}_{\pi^{\rho},\mathcal{A},\mathcal{Z}}\left(\kappa, z\right)\right\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*} \overset{\text{c}}{\equiv} \left\{\text{HYBRID}^{\mathcal{F}}_{\pi,\mathcal{S},\mathcal{Z}}\left(\kappa, z\right)\right\}_{\kappa\in\mathbb{N},z\in\{0,1\}^*}.$$

B.4 Some Ideal Functionalities

We next describe several ideal functionalities that are used throughout the paper.

B.4.1 Common Reference String

The *common reference string* functionality samples a string from some pre-determined distribution and provides the string to all the parties. The CRS functionality is described in Figure 9.

Functionality $\mathcal{F}^{D}_{\mathsf{crs}}$

$\mathcal{F}^{D}_{\mathsf{crs}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$, and parametrized by a distribution D.

- Upon receiving a message $(\mathsf{init}, \mathsf{sid})$ from party P_i, do:
    1. If there is no value $(\mathsf{sid}, \mathsf{crs})$ recorded, then sample $\mathsf{crs} \leftarrow D$ and record it.
    2. Send $(\mathsf{sid}, \mathsf{crs})$ as a public delayed output to P_i.

Figure 9: The common reference string functionality

In case the distribution D is the uniform distribution, we refer to the functionality as a *uniform reference string* functionality.

B.4.2 Secure Message Transmission

The *secure message transmission (SMT)* functionality models a secure and private channel between two parties. The sender can send a message to the receiver such that the adversary learns only a specified leakage of the message, e.g., its length. If the sender is corrupted before the message was delivered to the receiver, the adversary is allowed to change the message. The secure message transmission functionality is described in Figure 10.

Functionality $\mathcal{F}_{\mathsf{smt}}$

$\mathcal{F}_{\mathsf{smt}}$ proceeds as follows, running with a sender T, a receiver R and an adversary $\mathcal{S}$.

- Upon receiving a message $(\mathsf{send}, T, R, \mathsf{sid}, m)$ from T, send the message $(\mathsf{sent}, T, R, \mathsf{sid}, |m|)$ to $\mathcal{S}$, generate a private delayed output $(\mathsf{sent}, T, \mathsf{sid}, m)$ to R and halt.
- Upon receiving a message $(\mathsf{corrupt}, \mathsf{sid}, P)$ from $\mathcal{S}$, where $P \in \{T, R\}$, disclose m to $\mathcal{S}$. Next, if $\mathcal{S}$ provides a value m' and $P = T$, and no output has been written yet to R, then output $(\mathsf{sent}, T, \mathsf{sid}, m')$ to R and halt.

Figure 10: The secure message transmission functionality

B.4.3 Secure Function Evaluation

Secure function evaluation (SFE) is a multiparty primitive where a set of n parties wish to compute a (possibly randomized) function $f\colon (\{0,1\}^*)^n \times \{0,1\}^* \to (\{0,1\}^*)^n$, where $f = (f_1, \ldots, f_n)$. That is, for a vector of inputs $\boldsymbol{x} = (x_1, \ldots, x_n) \in (\{0,1\}^*)^n$ and random coins $r \in_R \{0,1\}^*$, the output-vector is $(f_1(\boldsymbol{x}; r), \ldots, f_n(\boldsymbol{x}; r))$. The output for the i'th party (with input x_i) is defined to be $f_i(\boldsymbol{x}; r)$. The secure function evaluation functionality, $\mathcal{F}^f_{\mathsf{sfe}}$, is presented in Figure 11.

Functionality $\mathcal{F}^f_{\mathsf{sfe}}$

$\mathcal{F}^f_{\mathsf{sfe}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$, and parametrized by an n-party function $f\colon (\{0,1\}^*)^n \times \{0,1\}^* \to (\{0,1\}^*)^n$. For every P_i initialize an input value $x_i = \bot$ and an output value $y_i = \bot$.

- Upon receiving a message $(\mathsf{input}, \mathsf{sid}, v)$ from some party P_i, set $x_i = v$ and send a message $(\mathsf{input}, \mathsf{sid}, P_i, |v|)$ to $\mathcal{S}$.
- Upon receiving a message $(\mathsf{output}, \mathsf{sid})$ from some party P_i, do:
  1. If $x_j = \bot$ for some honest P_j, ignore the message.
  2. Otherwise, if $y_1, \ldots, y_n$ have not been set yet, then choose $r \in_R \{0,1\}^*$ and compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n; r)$.
  3. Generate a delayed output $(\mathsf{output}, \mathsf{sid}, y_i)$ to P_i and send $(\mathsf{output}, \mathsf{sid}, P_i)$ to $\mathcal{S}$.

Figure 11: The secure function evaluation functionality

Note that UC protocols do not provide *guaranteed termination*, since the adversary has full control over the communication. Therefore, it is standard to claim about security in situations where the environment provides sufficiently many activations to the parties, and the adversary delivers all the messages (see [24] for further discussion). We would like to define UC analogs to *security with abort* and *guaranteed output delivery* that are normally defined in the stand-alone model (see

[36]). One way is to use the synchronous model of UC [75], where guaranteed termination can be achieved independently of the adversary. Another way is to slightly adjust the $\mathcal{F}_{\mathsf{sfe}}$ functionality, as discussed below.

Note that the $\mathcal{F}_{\mathsf{sfe}}$ functionality in some sense guarantee the output delivery, since although the adversary has the power to "hang" the computation, he cannot force an honest party to output an incorrect result. Stated differently, if the protocol terminates, i.e., the adversary provides inputs and deliver the output then the protocol satisfies guaranteed output delivery. For clarity, we denote this functionality by $\mathcal{F}_{\mathsf{sfe\text{-}god}}$. In the no-honest-majority setting, the adversary has an extra capability, as he can force all parties to output $\perp$ even when the protocol terminates. To capture this capability, we denote by $\mathcal{F}_{\mathsf{sfe\text{-}abort}}$ the $\mathcal{F}_{\mathsf{sfe}}$ functionality that allows the adversary to send a special $(\mathsf{abort}, \mathsf{sid})$ message at any time. In case some honest party has already received the output value, the functionality ignores this message. Otherwise, the functionality sets the output of all honest parties to $\perp$.

B.4.4 Broadcast

The *broadcast* functionality enables a sender to reliably deliver a message to all other parties. If the sender is corrupted before the message was delivered to the receivers, the adversary is allowed to change the message. We model this functionality as a special case of the SFE functionality for the function $f(x, \lambda, \ldots, \lambda) = (x, \ldots, x)$ (where λ denotes the empty string).

C Constructing TEFHE From LWE

In this section, we present an explicit construction of threshold equivocal FHE scheme. we start by describing in Appendix C.1 the GSW scheme, followed by its equivocal variant in Appendix C.2. In Appendix C.3, we present the threshold equivocal FHE. Unlike the rest of the paper, in this section we use n to denote the dimension of the lattice, and N to denote the number of parties.

C.1 GSW Fully Homomorphic Encryption

We now describe the GSW [56] fully homomorphic encryption scheme. We use in the construction the public gadget matrix as defined by Micciancio and Peikert [79], which is a matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ with some special structure, such that given a matrix $\mathbf{V} \in \mathbb{Z}_q^{n\times m}$ everyone can compute a "short" matrix $\mathbf{G}^{-1}(\mathbf{V}) \in \mathbb{Z}_q^{m\times m}$ satisfying $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{V}) = \mathbf{V}$. The GSW scheme is defined as follows:

- $\mathsf{params} \leftarrow \mathsf{GSW.Setup}(1^\kappa, 1^d)$: Choose a lattice dimension parameter $n = n(\kappa, d)$, a B_χ-bounded error distribution $\chi = \chi(\kappa, d)$, and a modulus q of size $q = B_\chi 2^{\omega(d\kappa \log \kappa)}$ such that $\mathsf{LWE}_{n-1,q,\chi,B_\chi}$ holds. Choose $m = n\log(q) + \omega(\log \kappa)$. Finally, choose a random matrix $\mathbf{B} \in \mathbb{Z}_q^{n-1\times m}$. Output $\mathsf{params} := (q, n, m, \chi, B_\chi, \mathbf{B})$. We stress that all the other algorithms implicitly get params as input even if we usually do not write this explicitly.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{GSW.Keygen}(\mathsf{params})$: We separately describe two sub-algorithms to generate secret-key and public-key, respectively:

  - $\mathsf{sk} \leftarrow \mathsf{GSW.SKGen}(\mathsf{params})$: Sample uniformly at random $\boldsymbol{s} \leftarrow \mathbb{Z}_q^{n-1}$ and output $\mathsf{sk} = \boldsymbol{t} = (-\boldsymbol{s}, 1) \in \mathbb{Z}_q^n$.

– $\mathsf{pk} \leftarrow \mathsf{GSW.PKGen}(\mathsf{params}, \mathsf{sk})$: Let $\mathsf{sk} = \boldsymbol{t} = (-\boldsymbol{s}, 1)$, sample $\boldsymbol{e} \leftarrow \chi^m$, set $\boldsymbol{b} := \boldsymbol{s} \cdot \mathbf{B} + \boldsymbol{e} \in \mathbb{Z}_q^m$ and output $\mathsf{pk} = \mathbf{A}$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is defined as $\mathbf{A} = \begin{bmatrix} \mathbf{B} \\ \boldsymbol{b} \end{bmatrix}$.

- $\mathsf{ct} \leftarrow \mathsf{GSW.Enc}(\mathsf{pk}, \mu)$: Choose a short random matrix as the randomness $\mathbf{R} \leftarrow \{0,1\}^{m\times m}$. Then, output the encryption of the message $\mu \in \{0,1\}$ as $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$, defined as

$$\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}.$$

- $\mu' = \mathsf{GSW.Dec}(\mathsf{sk}, \mathsf{ct})$: We decompose the decryption algorithm into two parts:

  – $v = \mathsf{GSW.Dec}^1(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{sk} = \boldsymbol{t}$ and $\mathsf{ct} = \mathbf{C}$. Consider the public vector $\boldsymbol{w} = (0, \ldots, 0, \lceil q/2 \rceil) \in \mathbb{Z}_q^n$ and output $v = \boldsymbol{t} \cdot \mathbf{C} \cdot \mathbf{G}^{-1}(\boldsymbol{w}^T) \in \mathbb{Z}_q$.

  – $\mu' = \mathsf{GSW.Dec}^2(v)$: Output $\mu' = \left| \left\lfloor \frac{v}{q/2} \right\rceil \right|$.

- $\mathsf{GSW.Eval}$: We define the homomorphic evaluation by defining addition and multiplication. Given ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{n\times m}$, define:

  – $\mathsf{GSW.Add}(\mathbf{C}_1, \mathbf{C}_2)$: Output $\mathbf{C}^{(+)} = \mathbf{C}_1 + \mathbf{C}_2 \in \mathbb{Z}_q^{n\times m}$.

  – $\mathsf{GSW.Mult}(\mathbf{C}_1, \mathbf{C}_2)$: Output $\mathbf{C}^{(\times)} = \mathbf{C}_1 \cdot \mathbf{G}^{-1}(\mathbf{C}_2) \in \mathbb{Z}_q^{n\times m}$.

Theorem C.1 ([56])**.** *The scheme as defined above is a secure FHE scheme under the* $\mathsf{LWE}_{n-1,q,\chi,B_\chi}$ *assumption.*

Proof (sketch). Semantic security of the scheme is proved in two steps. First, the public key $\mathbf{A}$ is replaced with a uniformly random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$; this step is secure under the LWE assumption. Second, the ciphertext $\mathbf{C} = \mathbf{AR} + \mu\mathbf{G}$ is replaced with a uniformly random matrix $\mathbf{C} \leftarrow \mathbb{Z}_q^{n\times m}$; this step is secure due to the leftover hash lemma. We refer the reader to [56] for further details.

To prove correctness, we start by defining noisy ciphertexts.

Definition C.2. *A β-noisy ciphertext of some message μ under secret key $\mathsf{sk} = \boldsymbol{t} \in \mathbb{Z}_q^n$ is a matrix $\boldsymbol{C} \in \mathbb{Z}_q^{n\times m}$ such that $\boldsymbol{t}\boldsymbol{C} = \mu\boldsymbol{t}\boldsymbol{G} + \boldsymbol{e}$ for some $\boldsymbol{e}$ satisfying $\|\boldsymbol{e}\|_\infty \leq \beta$.*

We proceed to analyze the noise behavior of ciphertexts under encryption, evaluation, and decryption operation.

- **Encryption.** Consider a public key $\mathbf{A}$ and a secret key $\boldsymbol{t}$ generated by $\mathsf{GSW.Keygen}(1^\kappa, 1^d)$; it holds that $\boldsymbol{t}\mathbf{A} = \boldsymbol{e}$ with $\|\boldsymbol{e}\|_\infty \leq B_\chi$. Therefore, a ciphertext $\mathbf{C} = \mathbf{AR} + \mu\mathbf{G} \leftarrow \mathsf{GSW.Enc}(\mathsf{pk}, \mu)$ satisfies $\boldsymbol{t}\mathbf{C} = \boldsymbol{e}\mathbf{R} + \mu\boldsymbol{t}\mathbf{G}$ with $\|\boldsymbol{e}\mathbf{R}\|_\infty \leq mB_\chi$. That is, $\mathbf{C}$ is a mB_χ-noisy ciphertext of μ under secret key $\boldsymbol{t}$. We denote $\beta_{\mathsf{init}} = mB_\chi$.

- **Addition.** Let $\mathbf{C}_1$ (resp. $\mathbf{C}_2$) be a β_1-noisy (resp. β_2-noisy) ciphertext of μ_1 (resp. μ_2) under secret key $\boldsymbol{t}$, i.e., $\boldsymbol{t}\mathbf{C}_i = \mu_i\boldsymbol{t}\mathbf{G} + \boldsymbol{e}_i$ with $\|\boldsymbol{e}_i\|_\infty \leq \beta_i$ for $i \in \{1,2\}$. Then, for $\mathbf{C}^{(+)} = \mathbf{C}_1 + \mathbf{C}_2$ it holds that $\boldsymbol{t}\mathbf{C}^{(+)} = (\mu_1 + \mu_2)\boldsymbol{t}\mathbf{G} + \boldsymbol{e}_1 + \boldsymbol{e}_2$, i.e., $\mathbf{C}^{(+)}$ is a $(\beta_1 + \beta_2)$-noisy ciphertext of $\mu_1 + \mu_2$ under $\boldsymbol{t}$.

- **Multiplication.** Let $\mathbf{C}_1$ and $\mathbf{C}_2$ as above. Then, for $\mathbf{C}^{(\times)} = \mathbf{C}_1\mathbf{G}^{-1}(\mathbf{C}_2)$ it holds that $\boldsymbol{t}\mathbf{C}^{(\times)} = \mu_1\mu_2\boldsymbol{t}\mathbf{G} + \boldsymbol{e}$ where $\boldsymbol{e} = \boldsymbol{e}_1\mathbf{G}^{-1}(\mathbf{C}_2) + \mu_1\boldsymbol{e}_2$. Therefore, $\|\boldsymbol{e}\|_\infty \le (m\beta_1 + \beta_2)$, i.e., $\mathbf{C}^{(\times)}$ is a $(m\beta_1 + \beta_2)$-noisy ciphertext of $\mu_1\mu_2$ under $\boldsymbol{t}$. The same calculation holds for NAND gates.

- **Decryption.** Let $\mathbf{C}$ be a β-noisy ciphertext of μ under $\boldsymbol{t}$, i.e., $\boldsymbol{t}\mathbf{C} = \mu\boldsymbol{t}\mathbf{G} + \boldsymbol{e}$ with $\|\boldsymbol{e}\|_\infty \le \beta$. Then, $\boldsymbol{t}\mathbf{C}\mathbf{G}^{-1}(\boldsymbol{w}^T) = \mu\lceil q/2\rfloor + e'$ with $e' = \langle \boldsymbol{e}, \mathbf{G}^{-1}(\boldsymbol{w}^T)\rangle$. it holds that $|e'| \le m\beta$. The decryption will be correct as long as $|e'| \le q/4$, i.e., as long as $\beta \le q/(4m)$. We denote $\beta_{\mathsf{max}} = q/(4m)$.

Consider a homomorphic evaluation of Boolean circuit of depth d consisting of NAND gates. The inputs are encrypted as β_{init}-noisy ciphertexts, and each level multiplies the noise by a factor of at most $(m+1)$. Therefore, the final output is β_{final}-noisy ciphertexts, where $\beta_{\mathsf{final}} = (m+1)^d\beta_{\mathsf{init}}$. To ensure correctness of decryption, we require that $\beta_{\mathsf{final}} \le \beta_{\mathsf{max}}$ meaning $B_\chi 4m^2(m+1)^d < q$ which is satisfied by the choice of parameters for the scheme. □

C.2 GSW/GVW Equivocal Fully Homomorphic Encryption

We now describe the construction of the additional algorithms for the equivocal FHE scheme based on the GSW FHE scheme, following [59].

Lemma C.3 ([1, 55, 2, 79])**.** *There exist efficient algorithms* $\mathsf{TrapGen}$, SamPre, *and* Sam *such that the following holds. Given integers $n \ge 1$ and $q \ge 2$, there exists some $m^* = m^*(n,q) = O(n\log q)$ and $\beta_{\mathsf{sam}} = \beta_{\mathsf{sam}}(n,q) = O(n\sqrt{\log q})$ such that for all $m \ge m^*$ and all k (polynomial in n) it holds that:*

1. *$\boldsymbol{U} \leftarrow \mathsf{Sam}(1^m, 1^k, q)$ samples a matrix $\boldsymbol{U} \in \mathbb{Z}_q^{m\times k}$ which satisfies $\|\boldsymbol{U}\|_\infty \le \beta_{\mathsf{sam}}$ (with probability 1).*

2. *We have the statistical indistinguishability requirements:*

$$\boldsymbol{A} \overset{\mathrm{s}}{\equiv} \boldsymbol{A}' \qquad \text{and} \qquad (\boldsymbol{A}, \mathsf{td}, \boldsymbol{U}, \boldsymbol{V}) \overset{\mathrm{s}}{\equiv} (\boldsymbol{A}, \mathsf{td}, \boldsymbol{U}', \boldsymbol{V}'),$$

   where $\boldsymbol{A}$ is sampled as $(\boldsymbol{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and $\boldsymbol{A}' \leftarrow \mathbb{Z}_q^{n\times m}$ is uniformly random. Likewise, $\boldsymbol{U} \leftarrow \mathsf{Sam}(1^m, 1^k, q)$, $\boldsymbol{V} = \boldsymbol{A}\boldsymbol{U}$, $\boldsymbol{V}' \leftarrow \mathbb{Z}_q^{n\times k}$ is uniformly random, and $\boldsymbol{U}' \leftarrow \mathsf{SamPre}(\boldsymbol{A}, \boldsymbol{V}', \mathsf{td})$. The statistical distance is negligible in n. Moreover, we guarantee that any $\boldsymbol{U}' \in \mathsf{SamPre}(\boldsymbol{A}, V', \mathsf{td})$ always satisfies $\boldsymbol{A}\boldsymbol{U}' = \boldsymbol{V}'$ and $\|\boldsymbol{U}'\|_\infty \le \beta_{\mathsf{sam}}$.

We proceed to define the additional algorithms for the equivocal FHE:

- $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{GSW.GenEquiv}(\mathsf{params})$: Select $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ and set $\mathsf{pk} = \mathbf{A}$.

- $\mathsf{GSW.Equiv}(\mathsf{td}, \mathsf{ct}, \mu)$: Let $\mathsf{ct} = \mathbf{C}$ and output $\mathsf{SamPre}(\mathbf{A}, \mathbf{C} - \mu\mathbf{G}, \mathsf{td})$.

C.3 Threshold Equivocal Fully Homomorphic Encryption

We proceed by adjusting the EFHE scheme to support threshold key-generation and decryption.

- $\mathsf{TEFHE.Gen}(1^\kappa, 1^d, 1^N) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$: Compute $\mathsf{params} \leftarrow \mathsf{GSW.Setup}(1^\kappa, 1^d)$, set $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{GSW.Keygen}(\mathsf{params})$, and let $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ be a linear secret sharing of sk, i.e., $\sum_{i\in[N]} \mathsf{sk}_i = \mathsf{sk} \mod q$.

- $\mathsf{TEFHE.Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$: Output $\mathsf{GSW.Enc}(\mathsf{pk}, \mu)$.
- $\mathsf{TEFHE.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \to \mathsf{ct}$: Output $\mathsf{GSW.Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$.
- $\mathsf{TEFHE.PartDec}(i, \mathsf{sk}_i, \mathsf{ct}) \to \mathsf{p}_i$: Output partial decryption $\mathsf{p}_i = \mathsf{GSW.Dec}^1(\mathsf{sk}_i, \mathsf{ct}) + e_i$, where $e_i \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ is some random "smudging noise," where $B_{\mathsf{smdg}} = 2^{d\kappa \log \kappa} B_\chi$.
- $\mathsf{TEFHE.FinDec}(\mathsf{pk}, \{\mathsf{p}_1, \dots, \mathsf{p}_N\}) \to \tilde{\mu}$: Output $\mathsf{GSW.Dec}^2(\sum_{i=1}^N \mathsf{p}_i)$.
- $\mathsf{TEFHE.GenEquiv}(1^\kappa, 1^d) \to (\mathsf{pk}, \mathsf{td})$: Output $\mathsf{GSW.GenEquiv}(\mathsf{params})$.
- $\mathsf{TEFHE.Equiv}(\mathsf{td}, \mathsf{ct}, \mu) \to r$: Output $\mathsf{GSW.Equiv}(\mathsf{td}, \mathsf{ct}, \mu)$.

We will use the following "smudging lemma" [5] to prove correctness of the threshold scheme.

Lemma C.4 ([5])**.** *Let $B_1 = B_1(\kappa)$ and $B_2 = B_2(\kappa)$ be positive integers, and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow [-B_2, B_2]$ be chosen uniformly at random. Then, the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\kappa)$.*

To prove correctness, we need to show that given a ciphertext matrix $\mathbf{C}$, it holds that by computing $\mathsf{p}_i = \mathsf{GSW.Dec}^1(\mathsf{sk}_i, \mathsf{ct}) + e_i$ for $e_i \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ followed by $\mathsf{GSW.Dec}^2(\sum_{i=1}^N \mathsf{p}_i)$, we get the same result as computing $v = \mathsf{GSW.Dec}^1(\mathsf{sk}, \mathbf{C})$ followed by $\mathsf{GSW.Dec}^2(v)$. Note that by defining $\mathbf{C}_1 = \mathbf{C}\mathbf{G}^{-1}(\boldsymbol{w}^T) \in \mathbb{Z}_q^n$, it holds by linearity of inner product that:

$$\begin{aligned}
\sum_{i=1}^N \mathsf{p}_i &= \sum_{i=1}^N (\mathsf{GSW.Dec}^1(\boldsymbol{t}_i, \mathbf{C}) + e_i) = \sum_{i=1}^N (\boldsymbol{t}_i \mathbf{C}\mathbf{G}^{-1}(\boldsymbol{w}^T) + e_i) = \sum_{i=1}^N (\boldsymbol{t}_i \mathbf{C}_1 + e_i) \\
&= \sum_{i=1}^N (\langle \boldsymbol{t}_i, \mathbf{C}_1 \rangle + e_i) = \langle \sum_{i=1}^N \boldsymbol{t}_i, \mathbf{C}_1 \rangle + \sum_{i=1}^N e_i = \langle \boldsymbol{t}, \mathbf{C}_1 \rangle + \sum_{i=1}^N e_i \\
&= \mathsf{GSW.Dec}^1(\boldsymbol{t}, \mathbf{C}) + \sum_{i=1}^N e_i.
\end{aligned}$$

To prove simulatability, we construct a simulator $\mathsf{Sim}_{\text{TEFHE}}$ that receives as input the ciphertext ct describing a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$, a plaintext $\mu \in \{0, 1\}$, and the secret key of all parties but the i'th $\{\mathsf{sk}_j\}_{j \neq i}$, where each sk_j is of the form $\boldsymbol{t}_j = (\boldsymbol{s}_j, 1) \in \mathbb{Z}_q^n$. The simulator starts by setting $\mathbf{C}_1 = \mathbf{C}\mathbf{G}^{-1}(\boldsymbol{w}^T)$ for $\boldsymbol{w} = (0, \dots, 0, \lceil q/2 \rceil) \in \mathbb{Z}_q^n$, and computing $\gamma_j = \langle \boldsymbol{t}_j, \mathbf{C}_1 \rangle$ for every $j \neq i$. Next, sample smudging noise $e_i^{\mathsf{sm}} \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ and set

$$\mathsf{p}'_i = \mu \lceil q/2 \rceil + e_i^{\mathsf{sm}} - \sum_{j \neq i} \gamma_j.$$

To prove the statistical indistinguishability, note that by the same calculation as used to argue correctness, we know that $\sum_{j \in [N]} \gamma_j = \mu \lceil q/2 \rceil + e'$ with $|e'| \leq \beta_{\mathsf{final}} m N = 2^{O(d \log \kappa)} B_\chi$. Denote that real partial decryption is $\mathsf{p}_i = \gamma_i + e_i^{\mathsf{sm}}$, then it holds that

$$\mathsf{p}_i = \mu \lceil q/2 \rceil + e_i^{\mathsf{sm}} + e' - \sum_{j \neq i} \gamma_j.$$

The difference between the real value p_i and the simulated value p'_i is the noise e' of norm $|e'| = 2^{O(d \log \kappa)} B_\chi$. By Lemma C.4, the distributions e_i^{sm} and $e_i^{\mathsf{sm}} + e'$ are statistically close since $e_i^{\mathsf{sm}} \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ for $B_{\mathsf{smdg}} = 2^{O(d\kappa \log \kappa)} B_\chi$, hence $B_{\mathsf{smdg}}/|e'| \geq 2^\kappa$.