How to Construct a Leakage-Resilient (Stateless) Trusted Party

Daniel Genkin
UPenn and UMD
danielg3@seas.upenn.edu

Yuval Ishai
Technion and UCLA
yuvali@cs.technion.ac.il

Mor Weiss
Northeastern
m.weiss@northeastern.onmicrosoft.com

Abstract

Trusted parties and devices are commonly used in the real world to securely perform computations on secret inputs. However, their security can often be compromised by side-channel attacks in which the adversary obtains partial leakage on intermediate computation values. This gives rise to the following natural question: *To what extent can one protect the trusted party against leakage?*

Our goal is to design a hardware device T that allows $m \geq 1$ parties to securely evaluate a function $f(x_1, \ldots, x_m)$ of their inputs by feeding T with encoded inputs that are obtained using local secret randomness. Security should hold even in the presence of an active adversary that can corrupt a subset of parties and obtain restricted leakage on the internal computations in T.

We design hardware devices T in this setting both for zero-knowledge proofs and for general multi-party computations. Our constructions can unconditionally resist either AC^0 leakage or a strong form of "only computation leaks" (OCL) leakage that captures realistic side-channel attacks, providing different tradeoffs between efficiency and security.

Contents

1 Introduction

There is a long and successful line of work on protecting general computations against partial information leakage. Originating from the works on general secure multiparty computation (MPC) [?, ?, ?, ?], the question has been "scaled down" to the domain of protecting circuits against local probing attacks [?] and then extended to different types of global information leakage [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?].

Most of the works along this line consider the challenging goal of protecting computations against *continual leakage*. In a general instance of this problem, a desired ideal functionality is specified by a *stateful* circuit C, which maps the current input and state to the current output and the next state. The input and output are considered to be public whereas the state is secret. The goal is to securely realize the functionality C by a leakage-resilient randomized circuit $\hat{C}$. The circuit $\hat{C}$ is initialized with some randomized encoding $\hat{s}$ of an initial secret state s. The computation can then proceed in a virtually unlimited number of rounds, where in each round $\hat{C}$ receives an input, produces an output, and replaces the old encoding of the secret state by a fresh encoding of a new state.

The correctness goal is to ensure that $\hat{C}[\hat{s}]$ has the same input-output functionality as $C[s]$. The security goal is defined with respect to a class $\mathcal{L}$ of *leakage functions* ℓ, where each function ℓ returns some partial information on the values of the internal wires of $\hat{C}$. The adversary may adaptively choose a different function $\ell \in \mathcal{L}$ in each round. The security goal is to ensure that whatever the adversary learns by interacting with $\hat{C}[\hat{s}]$ and by additionally observing the leakage, it can simulate by interacting with $C[s]$ without obtaining any leakage.

While general solutions to the above problem are known for broad classes of leakage functions $\mathcal{L}$, they leave much to be desired. Some rely on leak-free hardware components [?, ?, ?, ?, ?]. Others make a heavy use of public-key cryptography [?, ?, ?, ?, ?] or even indistinguishability obfuscation [?]. Other issues include the need for internal fresh randomness in each round, big computational overhead that grows super-linearly with the amount of tolerable leakage, complex and subtle analysis, and poor concrete parameters. All of the above works suffer from at least some of these limitations.

In this work we take a step back, and study a simpler *stateless* variant of the problem, where both C and $\hat{C}$ are stateless circuits. The goal is to replace an ideal computation of $C(x)$ by a functionally equivalent but leakage-resilient computation $\hat{C}(\hat{x})$. Here x is a secret input which is randomly encoded into an encoded input $\hat{x}$ to protect it against leakage. Solutions for the above continuous leakage model can be easily specialized to the stateless model by considering a single round where the input is used as the initial secret state. This stateless variant of the problem has been considered before [?, ?, ?], but mainly as an intermediate step and not as an end goal.

Our work is motivated by the observation that this simpler setting, which is relevant to many real-world scenarios, does not only offer an opportunity to get around the limitations of previous solutions, but also poses new challenges that were not addressed before. For instance, can correctness be guaranteed even when the input encoding $\hat{x}$ is invalid, in the sense that the output corresponds to *some* valid input x? Can the solutions be extended to the case where the encoded inputs for $\hat{C}$ are contributed by several, mutually distrusting, parties? To further motivate these questions, we put them in the context of natural applications.

Protecting a trusted party. We consider the goal of protecting (stateless) trusted parties against leakage. Trusted Parties (TPs) are commonly used to perform computations that involve secret inputs. They are already widely deployed in payment terminals and access control readers, and will be even more so in future Trusted Platform Modules. TPs have several advantages over distributed protocols for secure multiparty computation (MPC) [?, ?, ?, ?]. First, they avoid the expensive interaction typically required by MPC protocols. Second, they are very light-weight and allow the computational complexity of the other (untrusted) parties to be independent of the complexity of the computation being performed. Finally, TPs may offer *unconditional* security against *computationally unbounded* adversaries.

An important special case which is a major focus of this work is that of a hardware implementation of zero-knowledge (ZK) proofs, a fundamental primitive for identification and a useful building block for cryptographic protocol design. Informally, a ZK hardware takes a statement and witness from a prover, and outputs the verified statement, or rej, to a verifier. While there are efficient ZK protocols without hardware (including non-interactive zero-knowledge protocols (NIZKs) [?, ?], or succinct non-interactive arguments of knowledge (SNARKs) [?]), such protocols do not (and cannot) have the last two features of TP-based solutions.

A primary concern when using trusted hardware are so-called "side-channel" attacks which allow the adversary to obtain leakage on the internal computations of the device (e.g., through measuring its running time [?], power consumption [?], or the electromagnetic radiation it emits [?]). Such attacks were shown to have devastating effects on security. As discussed above, a large body of works attempted to incorporate the information obtained through such leakage into the security model, and develop schemes that are provably secure in these models. More specifically, these works have focused on designing leakage-resilient circuit compilers (LRCCs) that, informally, compile any circuit C into its leakage-resilient version $\hat{C}$, where $\hat{C}$ withstands side-channel attacks in the sense that these reveal nothing about the (properly encoded) input $\hat{x}$. However, all of the schemes obtained in these works suffer from some of the limitations discussed above. In particular, none considers the questions of invalid encodings provided by malicious parties or combining encoded inputs that originate from mutually distrusting parties. These questions arise naturally in the context of ZK and in other contexts where TPs are used.

1.1 Our Contribution

Our main goal is to study the feasibility and efficiency of protecting TPs against general classes of leakage, without leak-free hardware or trusted setup. Eliminating the leak-free hardware unconditionally [?], or under computational assumptions [?, ?] has been a major research goal. However, in contrast to earlier works, we consider here the easier case of realizing a *stateless* TP in the presence of *one-time* leakage.

We model the TP as a leaky (but otherwise trusted) hardware device $\mathcal{T}$ that is used by $m \geq 1$ parties to execute a multiparty computation task. More specifically, in this setting each party locally encodes its input and feeds the encoded input into the device, that evaluates a boolean (or arithmetic) circuit on the encoded inputs, and returns the output. This computation should preserve the secrecy of the inputs, as well as the correctness of the output, in the presence of a computationally-unbounded adversary that corrupts a subset of the parties, and additionally obtains leakage on the internals of the device. (Notice that the secrecy requirement necessitates some encoding of the inputs, otherwise we cannot protect even against a probing attack on a single bit.)

We note that the stateless hardware should be reusable on an arbitrary number of different inputs. Thus, we cannot take previous leakage-secure computation protocols that employ correlated randomness (such as the ones from [?, ?]) and embed this randomness into the hardware. Indeed, we consider the internals of the hardware as being public, since any secret internal embedded values can be leaked over multiple invocations.

The model has several different variants, depending on whether the adversary is passive (i.e., only sees the inputs of corrupted parties and obtains leakage on the internals of the TP) or active (namely, it may also cause corrupted parties to provide the TP with ill-formed "encoded" inputs that may not correspond to any inputs for the original computation); whether there is a single party providing input to the TP (as in the ZK example described below) or multiple parties; whether the TP is deterministic or randomized (namely, has randomness gates that generate uniformly-random bits); and finally, whether the output of the TP is encoded or not (in the latter, one cannot protect the privacy of the output even when the adversary only obtains leakage on the internals of the TP *without* corrupting any parties, whereas in the former the outputs will remain private in this case). We focus on the variant with an active adversary, and a randomized TP with encoded outputs. We consider both the single-party and multi-party setting. In the ZK setting, we also construct deterministic TPs (at the expense of somewhat increasing the complexity of the prover and verifier).

The leakage model. We consider an extended version of the "only computation leaks" (OCL) model of Micali and Reyzin [**?**], also known as "OCL+" [**?**]. Informally, in this context, the wires of the circuit $\hat{C}$ are partitioned into a "left component" $\hat{C}_L$ and a "right component" $\hat{C}_R$. Leakage functions correspond to bounded-communication 2-party protocols between $\hat{C}_L, \hat{C}_R$, where the output of the leakage function is the transcript of the protocol when the views of $\hat{C}_L, \hat{C}_R$ consist of the internal values of the wires of these two "components". Following the terminology of Goyal et al. [**?**], we refer to this model as *bounded communication leakage (BCL)*. The model is formalized in the next definition.

Definition 1.1 (t-BCL [**?**]). *Let $t \in \mathbb{N}$ be a leakage bound parameter. We say that a deterministic 2-party protocol is* t-bounded *if its communication complexity is at most t. Given a t-bounded protocol* Π*, we define the* t-bounded-communication leakage (t-BCL) *function f_Π associated with* Π*, that given the views of the two parties, outputs the transcript of* Π*. The class $\mathcal{L}^t_{\mathrm{BCL}}$ consists of all t-BCL functions f_Π associated with t-bounded protocols* Π*, namely: $\mathcal{L}^t_{\mathrm{BCL}} = \{f_\Pi \ : \ \Pi \textit{ is } t - \textit{bounded}\}$.*

We say that a size-s circuit $\hat{C}$ is t-BCL resilient *if there exists a partition $\mathcal{P} = \{s_1, s_2\}$ of the wires of $\hat{C}$, such that the circuit resists any t-BCL function f_Π for a protocol* Π *that respects the partition $\mathcal{P}$.*

We note that BCL is broad enough to capture several realistic leakage attacks such as the sum of all circuit wires over the integers, as well as linear functions over the wires of the circuit. This captures several realistic attacks on hardware devices, where a single electromagnetic probe measures involuntary leakage which can be approximated by a linear function of the wires of the circuit.

1.2 Our Results

We construct TPs for both ZK proofs, and general MPC, which simultaneously achieve many of the desired features described above: they resist a wide class of leakage functions (BCL), without using any leak-free components, and are quite appealing from the perspective of asymptotic efficiency, since the complexity of the parties is *independent* of the size of the computation. Our constructions combine ideas and results from previous works on leakage-resilient circuits, with several new ideas, as discussed in Section **??**.

TPs for ZK. In the context of ZK, the hardware device enables the verification of NP-statements of the form "$(x, w) \in \mathcal{R}$" for an NP-relation $\mathcal{R}$. That is, the prover provides (x, w) as input to the device, which computes the function $f(x, w) = (x, \mathcal{R}(x, w))$. Since the device is leaky, the prover is unwilling to provide its secret witness w to the device "in the clear". Instead, the prover prepares in advance a "leak-free" encoding $\hat{w}$ of w, which it stores on a small isolated device (such as a smartcard or USB drive). It then provides $(x, \hat{w})$ as input to the leaky device (e.g., by plugging in his smartcard) which outputs the public verification outcome. We say that the hardware device is an $\mathcal{L}$-secure ZK circuit if it resists leakage from $\mathcal{L}$ with negligible error. We construct $\mathcal{L}^t_{\mathrm{BCL}}$-secure ZK circuits for NP:

Theorem 1.2 (Leakage-secure ZK circuit). *For any leakage bound $t \in \mathbb{N}$, statistical security parameter $\sigma \in \mathbb{N}$, and length parameter $n \in \mathbb{N}$, any NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ with verification circuit of size s, depth d, and n inputs has an $\mathcal{L}^t_{\mathrm{BCL}}$-secure ZK circuit $C_\mathcal{R}$ that outputs the outcome of verification, where $\mathcal{L}^t_{\mathrm{BCL}}$ is the family of all t-BCL functions. Moreover, to prove that $(x, w) \in \mathcal{R}$, the prover runs in time $\mathsf{poly}(t, \sigma, n, |w|)$, and $|C_\mathcal{R}| = \widetilde{O}(s + d(t + \sigma + n)) + \mathsf{poly}(t, \sigma, n)$.*

We also construct a variant of the ZK circuit that allows one to "trade" efficiency of the prover and verifier with the randomness used by the ZK circuit:

Theorem 1.3 (Deterministic leakage-secure ZK circuit). *For any leakage bound $t \in \mathbb{N}$, statistical security parameter $\sigma \in \mathbb{N}$, and length parameter $n \in \mathbb{N}$, any NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ with verification circuit of size s, depth d, and n inputs has a* deterministic *$\mathcal{L}^t_{\mathrm{BCL}}$-secure ZK circuit $C_\mathcal{R}$. Moreover, $|C_\mathcal{R}| = \widetilde{O}(s + d(t + \sigma + n)) + \mathsf{poly}(t, \sigma, n)$, to prove that $(x, w) \in \mathcal{R}$, the prover runs in time $\widetilde{O}(s + d(t + \sigma + n)) + \mathsf{poly}(t, \sigma, n, |w|)$, and the verifier runs in time $\mathsf{poly}(t, \sigma, n)$.*

General MPC. We consider hardware devices that allow the evaluation of general functions in both the single-party setting, and the multiparty setting with $m \geq 2$. More specifically, we construct *m-party LRCCs* that given a circuit C that takes inputs from m parties, output a circuit $\hat{C}$ that operates on encoded inputs and outputs. Informally, we say the m-party LRCC is $(\mathcal{L}, \epsilon)$-secure if the evaluation of $\hat{C}$ guarantees (except with probability ϵ) privacy of the honest parties' inputs, and correctness of the output, in the presence of an adversary that may actively corrupt a strict subset of parties, and obtain leakage from $\mathcal{L}$ on the internals of the device. We construct m-party LRCCs that are secure against t-BCL:

Theorem 1.4 (Leakage-secure m-party LRCC)**.** *For any leakage bound $t \in \mathbb{N}$, statistical security parameter $\sigma \in \mathbb{N}$, input and output length parameters $n, k \in \mathbb{N}$, and size and depth parameters $s, d \in \mathbb{N}$, any m-party function $f : (\{0,1\}^n)^m \to \{0,1\}^k$ computable by a circuit of size s and depth d has an m-party $\left(\mathcal{L}^t_{\mathrm{BCL}}, \epsilon\right)$-secure LRCC, where $\mathcal{L}^t_{\mathrm{BCL}}$ is the family of all t-BCL functions, and $\epsilon = \mathsf{negl}(\sigma)$. Moreover, the leakage-secure circuit has size $\widetilde{O}(s + d(t + \sigma \log m)) + m \cdot \mathsf{poly}(t, \sigma, \log m, k)$, its input encodings can be computed in time $\widetilde{O}(n) + \mathsf{poly}(t, \sigma, \log m, k)$, and its outputs can be decoded in time $\widetilde{O}(m \cdot k(t + \sigma \log m + k))$.*

1.3 Our Techniques

1.3.1 Leakage-Resilient Zero-Knowledge

Recall that the leaky ZK device allows a prover P to prove claims of the form "$(x, w) \in \mathcal{R}$" for some NP-relation $\mathcal{R}$. We model the device as a stateless boolean (or more generally, arithmetic) circuit C. Though C cannot be assumed to withstand leakage, using an LRCC it can be transformed into a leakage-resilient circuit $\hat{C}$. Informally, an LRCC is associated with a function class $\mathcal{L}$ (the *leakage class*), a (randomized) input encoding scheme E, and a (deterministic) output decoder $\mathsf{Dec}_{\mathsf{Out}}$. The LRCC compiles a circuit C into a (public) circuit $\hat{C}$ that emulates C over encoded inputs and outputs. $\hat{C}$ resists leakage from $\mathcal{L}$ in the sense that for any input z for C, and any $\ell \in \mathcal{L}$, the output of ℓ on the wire values of $\hat{C}$, when evaluated on $\mathsf{E}(z)$, can be efficiently simulated given only the description of C.

Our starting point in constructing leakage-resilient ZK hardware is the recent result of Goyal et al. [**?**], who use MPC protocols to protect computation against BCL leakage. More specifically, they design information-theoretically secure protocols in the OT-hybrid model that allow a user, aided by a pair of "honest-but-curious" servers, to compute a function of her input while preserving the privacy of the input and output even under BCL leakage on the internals of the servers. We observe that when these server programs are implemented as circuits (in particular, the OT calls are implemented by constant-sized sub-circuits), this construction gives an LRCC that resists BCL leakage.

In the context of designing leakage-resilient TPs, the main advantage of this construction over previous information-theoretically secure LRCCs that resist similar leakage classes [**?**, **?**, **?**] is that [**?**] *does not use any leak-free components*. More specifically, these LRCCs use the leak-free components (or leak-free preprocessing in [**?**]) to generate "masks", which are structured random bits that are used to mask the internal computations in $\hat{C}$, thus guaranteeing leakage-resilience.

These leak-free components could be eliminated if the parties include the masks as part of their input encoding. However, this raises three issues. First, in some constructions (e.g. [**?**, **?**, **?**]) the number of masks is proportional to the size of $\hat{C}$, so the running time of the parties would not be independent of the computation size (which defeats the purpose of delegating most of the computation to the TP). Second, in the multi-party setting, it is not clear how to combine the masks provided by different parties into a single set of masks to be used in $\hat{C}$, such that these masks are *unknown to each one of the parties*, which is crucial for the leakage-resilience property to hold. (We show in [**?**] how to do so for the LRCC of [**?**] which resists AC^0 leakage, but this construction has the efficiency shortcomings mentioned above.) Finally, even with a single party, these constructions totally break when the party provides "ill-formed" masks (namely, masks that do not have the required structure), since correctness is guaranteed *only when the masks have the required structure*. This is not only a theoretical concern, but rather an *actual* one. To see why, consider the ZK setting. If the prover provides the masks to the device then it *has a way* of choosing (ill-formed) masks that flip the output gate, thus causing the device to accept false NP statements. Alternative "solutions" also fail: the device cannot verify that the masks provided by the prover are well-formed, since the aforementioned constructions *crucially* rely on the fact that the leakage-resilience simulator can use ill-formed masks; and the verifier cannot provide the masks, since leakage-resilience relies on the leakage function not knowing the masks.

Though using the LRCC of [?] eliminates all these issues, it has one shortcoming: its leakage-resilience simulator is *inefficient*. In the context of ZK hardware, this gives *witness-indistinguishability*, namely the guarantee that a malicious verifier that can leak on the internals of the ZK hardware cannot distinguish between executions on the same statement x with different witnesses w, w'. This falls short of our desired security guarantee that leakage reveals *no* information about the witness. (In particular, notice that if a statement x has only one witness then witness-indistinguishability provides no security.) We note that this weaker security guarantee is inherent to the construction of [?].

To achieve efficient simulation, we leverage the fact that the construction of [?] operates over encodings that resist BCL leakage. We observe that one can obtain simulation-based security if the encodings at the output of $\hat{C}$ are decoded using a circuit $\hat{C}_{\mathsf{Dec}}$ that "tolerates" BCL leakage, in the sense that such leakage on its *entire* wire values can be simulated given only (related) BCL leakage on the inputs and outputs of the circuit [?]. Indeed, the simulator can evaluate $\hat{C}$ on an *arbitrary* (*non*-satisfying) "witness" (thus generating the entire wire values of $\hat{C}$, and in particular allowing the simulator to compute any leakage on them), and then simulate leakage on the internals of $\hat{C}_{\mathsf{Dec}}$ by computing (related) leakage on its inputs (namely, the outputs of $\hat{C}$) and output (which is $(x, 1)$). Since the outputs of $\hat{C}$ resist BCL leakage, this is indistinguishable from the leakage on the internal wires of $\hat{C}, \hat{C}_{\mathsf{Dec}}$ when $\hat{C}$ is evaluated on an actual witness. We note that the decoding circuit $\hat{C}_{\mathsf{Dec}}$ can be constructed using the LRCC of [?], which by a recent result of Bitansky et al. [?] is leakage-tolerant against BCL leakage.

Though this construction achieves efficient simulation, it is no longer sound. Indeed, soundness crucially relies on the fact that $\hat{C}_{\mathsf{Dec}}$ emulates C_{Dec} (which decodes the output of $\hat{C}$). Recall that in current LRCC constructions that offer information-theoretic security against wide leakage classes (e.g., [?, ?, ?]), the correctness of the computation crucially relies on the fact that the masks (which are provided as part of the input encoding) have the "correct" structure. Consequently, by providing $\hat{C}_{\mathsf{Dec}}$ with *ill-formed* masks, a malicious prover P^* can *arbitrarily* modify the functionality emulated by $\hat{C}_{\mathsf{Dec}}$, and in particular, may flip the output of $\hat{C}_{\mathsf{Dec}}$, causing the device to accept $x \notin L_{\mathcal{R}}$.[1] Recall that the device cannot verify that the masks are well-formed, since this would violate leakage-resilience.

To overcome this, we observe that when $\hat{C}_{\mathsf{Dec}}$ is generated using the LRCC of Dziembowski and Faust [?], the effect of ill-formed masks on the computation in $\hat{C}_{\mathsf{Dec}}$ is equivalent to adding a vector of fixed (but possibly different) field elements to the wires of C_{Dec}. Such attacks are called "additive attacks", and one can use *AMD circuits* [?, ?, ?] to protect against them. Informally, AMD circuits are randomized circuits that offer the best possible security under additive attacks, in the sense that the effect of every additive attack that may apply to all internal wires of the circuit can be simulated by an ideal attack that applies only to its inputs and outputs.

Thus, by replacing C_{Dec} with an AMD circuit C'_{Dec} before applying the LRCC, the effect of ill-formed encoded inputs is further restricted to an additive attack on the inputs and output of C_{Dec}. Finally, to protect the inputs and outputs of C'_{Dec} from additive attacks, we use the AMD code of [?]. (We note that encoding the inputs and outputs of C'_{Dec} using AMD codes is inherent to any AMD-based construction, otherwise a malicious prover P^* can use ill-formed encoded inputs to $\hat{C}'_{\mathsf{Dec}}$ to flip the output.) As we show in Section **??**, the resultant construction satisfies the properties of Theorem **??**. To obtain the *deterministic* circuit of Theorem **??**, we have the prover provide (as part of its input encoding) the randomness used by the $\widehat{C}$ component (which was generated using the LRCC of [?]), and the verifier

[1] We note that "ill-formed" encodings do not pose a problem for *stateful* circuits (intuitively, the compiled circuit can use the secret state to overcome the influence of ill-formed masks). However, we are interested in *stateless* circuits.

provides the randomness used by the AMD circuit in $\widehat{C}_{\mathsf{Dec}}$. (We note that the prover cannot provide this randomness, since the security of AMD circuits crucially relies on their randomness being *independent* of the additive attack. Therefore, if the prover provides the randomness for the AMD circuit, a malicious prover may correlate the randomness used by the AMD circuit with the additive attack, rendering the AMD circuit useless.)

1.3.2 General Leakage-Resilient Computation

Recall that the setting consists of $m \geq 1$ parties that utilize a leaky, but otherwise trusted, device to compute a joint function of their inputs; while protecting the privacy of the inputs, and the correctness of the output, against an active adversary that corrupts a subset of the parties, and may also obtain leakage on the internals of the device. More specifically, we construct *m-party LRCCs* that given a (boolean or arithmetic) circuit C with m inputs, output a circuit $\hat{C}$ that operates on encoded inputs and outputs. (Recall that encoded outputs are needed to guarantee privacy against adversaries that do not corrupt any parties.) As in other LRCCs, the circuit compiler is associated with an input encoder Enc, and an output decoder Dec (used to encode the inputs to, and the output of, $\hat{C}$, respectively).

The multiparty setting introduces an additional complication which did not arise in the ZK setting. Recall that the leakage-resilience property of $\hat{C}$ crucially relies on the fact that its internal computations are randomized using masks which are *unknown to the leakage function.* As already discussed in Section **??**, to avoid the need for leak-free hardware we let the participating parties provide these masks. Consequently, the adversary (who also chooses the leakage function) knows the identity of the masks provided by all corrupted parties. We note that this issue occurs *even in the passive setting*, in which parties are guaranteed to honestly encode their inputs. This raises the following question: *how can we preserve the leakage-resilience property when the leakage function "knows" a subset of the masks?*

Our solution is to first replace the circuit C with a circuit C' that computes an *m-out-of-m additive secret sharing* of the output of C. We then construct the leakage-resilient version $\hat{C}'$ of C' using the LRCC of [**?**], which outputs encodings of the secret shares which C' computes. Then, each encoding is refreshed in a leakage-resilient manner. (This is similar to using a leakage-resilient version of the decoder in the ZK setting of Section **??**.) More specifically, let C_{refresh} be a circuit that given an encoding of some value v outputs a fresh encoding of v. Similar to the construction of ZK circuits in Section **??**, we replace C_{refresh} with an AMD circuit C'_{refresh} that emulates C_{refresh} but operates on AMD encodings. Finally, we compile C'_{refresh} using the LRCC of [**?**] into a leakage-resilient circuit $\hat{C}'_{\mathsf{refresh}}$, which (as discussed in Section **??**) has the additional feature that ill-formed masks are detected. We use m copies of $\hat{C}'_{\mathsf{refresh}}$ to refresh the m secret shares, where the i'th copy is associated with the i'th party, who provides (as part of its input encoding) the masks needed for the computation of the i'th copy. Finally, the decoder Dec decodes the secret shares, and uses them to reconstruct the output.

Having the leakage-resilience circuit generate (encodings of) *secret-shares* of the output, instead of (an encoding of) the output itself guarantees leakage-resilience even when the adversary corrupts parties and learns the masks which they provide for the computation. At a very high level, this holds because even if the adversary learns (through the leakage, and knowledge of the masks) the *entire wire values* of the copies of $\hat{C}'_{\mathsf{refresh}}$ associated with corrupted parties, these only reveal information about the *secret shares* which these copies operate on. Therefore, the secrecy of the secret-sharing scheme guarantees that no information is revealed about the *actual* output, or inputs, of the computation. Thus, we obtain Theorem **??**. (The analysis is in fact much more complex, see Section **??** for the construction and its analysis.)

1.4 Open Problems

Our work leaves several interesting open problems for further research. One is that of making the TP deterministic, while minimizing the complexity of the parties. Currently, we can make the TP deterministic, but only at the expense of making the parties work as hard as the entire original computation. A natural approach is via derandomization of the LRCC of [?]. Another research direction is to obtain a better understanding of the leakage classes that can be handled in this model, and extend the results to the setting of continuous leakage with stateful circuits. Another question is that of improving the asymptotic and concrete efficiency of our constructions, by providing better underlying LRCCs, or better analysis of existing ones. These questions are interesting even in the simple setting of a single semi-honest party.

1.5 Related Work

Originating from [?], MPC techniques are commonly used as a defense against side-channel attacks (see [?, ?] and references therein). However, except for the works of [?, ?] (discussed below) these techniques either rely on cryptographic assumptions [?, ?], or on structured randomness which is generated by leak-free hardware, and is used to mask the internal computations [?, ?, ?, ?, ?]. To eliminate the leak-free hardware, the parties can provide the structured randomness as part of their input encoding. However, since the correctness of the computation crucially relies on the randomness having the "correct" structure, this allows corrupted parties to arbitrarily modify the functionality computed by the circuit, by providing randomness that does not have the required structure.

The only exception to the above are the works of [?, ?], that provide provable information-theoretic security guarantees (without relying on structured randomness) against probing attacks, and some natural types of "noisy" leakage, but fail to protect against other simple types of realistic attacks, such as the sum of a subset of wires over the integers. (For example, when an AND gate is implemented using the LRCC of [?], the sum of a subset of wires in the resultant circuit allows an adversary to distinguish between the case in which both inputs are 0, and the case in which one of them is 1.)

2 Preliminaries

Notation. Let $\mathbb{F}$ be a finite field, and Σ be a finite alphabet (i.e., a set of symbols). For a function f over Σ^n, we use $\mathsf{supp}(f)$ to denote the image of f, namely $\mathsf{supp}(f) = \{f(x) : x \in \Sigma^n\}$. For an NP-relation $\mathcal{R} = \mathcal{R}(x, w)$, we denote $L_{\mathcal{R}} = \{x : \exists w, (x, w) \in \mathcal{R}\}$. Vectors will be denoted by boldface letters (e.g., $\mathbf{a}$). If $\mathcal{D}$ is a distribution then $X \leftarrow \mathcal{D}$, or $X \in_R \mathcal{D}$, denotes sampling X according to the distribution $\mathcal{D}$. Given two distributions X, Y, $\mathsf{SD}(X, Y)$ denotes the statistical distance between X and Y. For a natural n, $\mathsf{negl}(n)$ denotes a function that is negligible in n. For a function family $\mathcal{L}$, we sometimes use the term "leakage family $\mathcal{L}$", or "leakage class $\mathcal{L}$". In the following, n usually denotes the input length, k usually denotes the output length, d, s denote depth and size, respectively (e.g., of circuits, as defined below), and m is used to denote the number of parties.

Circuits. We consider boolean circuits C over the set $X = \{x_1, \cdots, x_n\}$ of variables. C is a directed acyclic graph whose vertices are called *gates* and whose edges are called *wires*. The wires of C are labeled with functions over X. Every gate in C of in-degree 0 has out-degree 1 and is either labeled by a variable from X and referred to as an *input gate*; or is labeled by a constant $\alpha \in \{0, 1\}$ and referred to as a const_α *gate*. Following [?], all other gates are labeled by one of the operations $\wedge, \vee, \neg, \oplus$, where $\wedge, \vee, \oplus$ vertices have fan-in 2 and fan-out 1; and $\neg$ has fan-in and fan-out 1. We write $C : \{0, 1\}^n \rightarrow \{0, 1\}^k$ to indicate that C is a boolean circuit with n inputs and k outputs. The *size* of a circuit C, denoted $|C|$, is the number of wires in C, together with input and output gates.

We also consider arithmetic circuits C over a finite field $\mathbb{F}$ and the set X. Similarly to the boolean case, C has input and constant gates, and all other gates are labeled by one of the following functions $+, -, \times$ which are the addition, subtraction, and multiplication operations of the field. We write $C : \mathbb{F}^n \rightarrow \mathbb{F}^k$ to indicate that C is an arithmetic circuit over $\mathbb{F}$ with n inputs and k outputs. Notice that boolean circuits can be viewed as arithmetic circuits over the binary field in a natural way. Therefore, we sometimes describe boolean circuits using the operations $+, -, \times$ instead of $\oplus, \neg, \wedge, \vee$.

Additive Attacks and Algebraic-Manipulation Detection (AMD) Circuits. Following the terminology of [?], an additive attack $\mathbf{A}$ affects the evaluation of a circuit C as follows. For every wire connecting gates a and b in C, a value specified by the attack $\mathbf{A}$ is added to the output of a and then the derived value is used for the computation of the gate b. Similarly, for every output gate, a value specified by $\mathbf{A}$ is added to the value of this output. Note that an additive attack on C is a fixed vector of (possibly different) field elements which is independent from the inputs and internal values of C. We denote the evaluation of C under additive attack $\mathbf{A}$ by $C^{\mathbf{A}}$.

At a high level, an additively-secure implementation of a function f is a circuit which evaluates f, and guarantees the "best" possible security against additive attacks, in the sense that any additive attack on it is equivalent (up to a small statistical distance) to an additive attack on the inputs and outputs of f. Formally,

Definition 2.1 (Additively-secure implementation [?])**.** *Let $\epsilon > 0$. A randomized circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ is an* ϵ-additively-secure implementation *of a function $f : \mathbb{F}^n \to \mathbb{F}^k$ if the following holds.*

- ***Completeness.*** *For every $x \in \mathbb{F}^n$, $\Pr\left[C\left(x\right) = f\left(x\right)\right] = 1$.*
- ***Additive-attack security.*** *For any additive attack $\mathbf{A}$ there exist $a^{\mathsf{In}} \in \mathbb{F}^n$, and a distribution $\mathcal{A}^{\mathsf{Out}}$ over $\mathbb{F}^k$, such that for every $\mathbf{x} \in \mathbb{F}^n$, $\mathsf{SD}(C^{\mathbf{A}}\left(\mathbf{x}\right), f\left(\mathbf{x} + \mathbf{a}^{\mathsf{in}}\right) + \mathcal{A}^{\mathsf{out}}) \leq \epsilon$.*

We also consider the notion of an additively-secure circuit compiler, which is a single PPT algorithm that compiles a given circuit C into its additively-secure implementation.

Definition 2.2 (Additively-secure circuit compiler)**.** *Let $n \in \mathbb{N}$ be an input length parameter, $k \in \mathbb{N}$ be an output length parameter, and $\epsilon\left(n\right) : \mathbb{N} \to \mathbb{R}^+$. Let Comp be a PPT algorithm that on input a circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, outputs a circuit $\hat{C}$. Comp is an* $\epsilon\left(n\right)$-additively-secure circuit compiler over $\mathbb{F}$ *if for every circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ that computes a function f_C, $\hat{C}$ is an $\epsilon\left(n\right)$-additively-secure implementation of f_C.*

We will need the following theorem.

Theorem 2.3 ([?])**.** *Let n be an input length parameter, and $\epsilon\left(n\right) : \mathbb{N} \to \mathbb{R}^+$ be a statistical error function. Then there exists an $\epsilon\left(n\right)$-additively-secure circuit compiler Comp over $\mathbb{F}_2$. Moreover, on input a depth-d boolean circuit $C : \{0,1\}^n \to \{0,1\}^k$, Comp outputs a circuit $\hat{C}$ such that $|\hat{C}| = |C| \cdot \mathsf{polylog}\left(|C|, \log\frac{1}{\epsilon(n)}\right) + \mathsf{poly}\left(n, k, d, \log\frac{1}{\epsilon(n)}\right)$. Furthermore, there exists a PPT algorithm Alg that on input C, $\epsilon\left(n\right)$, and an additive attack $\mathcal{A}$, outputs a vector $\mathbf{a}^{\mathsf{in}} \in \{0,1\}^n$, and a distribution $\mathcal{A}^{\mathsf{out}}$ over $\{0,1\}^k$, such that for any $\mathbf{x} \in \{0,1\}^n$ it holds that $\mathsf{SD}(\hat{C}^{\mathcal{A}}(\mathbf{x}), C(\mathbf{x} + \mathbf{a}^{\mathsf{in}}) + \mathcal{A}^{\mathsf{out}}) \leq \epsilon\left(n\right)$.*

Encoding schemes. An encoding scheme E over alphabet Σ is a pair $(\mathsf{Enc}, \mathsf{Dec})$ of algorithms, where the *encoding algorithm* Enc is a PPT algorithm that given a message $x \in \Sigma^n$ outputs an encoding $\hat{x} \in \Sigma^{\hat{n}}$ for some $\hat{n} = \hat{n}\left(n\right)$; and the *decoding algorithm* Dec is a deterministic algorithm, that given an $\hat{x}$ of length $\hat{n}$ in the image of Enc, outputs an $x \in \Sigma^n$. Moreover, $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(x\right)\right) = x\right] = 1$ for every $x \in \Sigma^n$. It would sometimes be convenient to explicitly describe the randomness used by Enc, in which case we think of Enc as a deterministic function $\mathsf{Enc}\left(x; r\right)$ of its input x, and random input r. Following [?], we say that a vector $\mathbf{v} \in \Sigma^{\hat{n}(n)}$ is *well-formed* if $\mathbf{v} \in \mathsf{Enc}\left(0^n\right)$.

Parameterized encoding schemes. We consider encoding schemes in which the encoding and decoding algorithms are given an additional input 1^t, which is used as a security parameter. Concretely, the encoding length depends also on t (and not only on n), i.e., $\hat{n} = \hat{n}\left(n, t\right)$, and for every t the resultant scheme is an encoding scheme (in particular, for every $x \in \Sigma^n$ *and every* $t \in \mathbb{N}$, $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(x, 1^t\right), 1^t\right) = x\right] = 1$). We call such schemes *parameterized.* For $n, t \in \mathbb{N}$, a vector $\mathbf{v} \in \Sigma^{\hat{n}(n,t)}$ is *well-formed* if $\mathbf{v} \in \mathsf{Enc}\left(0^n, 1^t\right)$.

Furthermore, we sometimes consider encoding schemes that take *a pair* of security parameters $1^t, 1^{t_{\mathsf{In}}}$. (t_{In} is used in cases when the encoding scheme employs an "internal" encoding scheme, and is used in the internal scheme.) In such cases, the encoding length depends on n, t, t_{In}, and the resultant scheme should be an encoding scheme for every $t, t_{\mathsf{In}} \in \mathbb{N}$. We will usually omit the term "parameterized", and use "encoding scheme" to describe both *parameterized and non-parameterized* encoding schemes.

Next, we define leakage-indistinguishable encoding schemes.

Definition 2.4 (Leakage-indistinguishability of functions and encodings, [?])**.** *Let D, D' be finite sets, $\mathcal{L}_D = \{\ell : D \to D'\}$ be a family of leakage functions, and $\epsilon > 0$. We say that two distributions X, Y over D are* $(\mathcal{L}_D, \epsilon)$-leakage-indistinguishable, *if for any function $\ell \in \mathcal{L}_D$, $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$. In case $\mathcal{L}_D$ consists of functions over a union of domains, we say that X, Y over D are $(\mathcal{L}_D, \epsilon)$-leakage-indistinguishable if $\mathsf{SD}(\ell(X), \ell(Y)) \leq \epsilon$ for every function $\ell \in \mathcal{L}$* with domain *D.*

Let $\mathcal{L}$ be a family of leakage functions. We say that a randomized function $f : \Sigma^n \to \Sigma^m$ is $(\mathcal{L}, \epsilon)$-leakage-indistinguishable *if for every $x, y \in \Sigma^n$, the distributions $f(x), f(y)$ are $(\mathcal{L}, \epsilon)$-leakage-indistinguishable. We say that an encoding scheme $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ is* $(\mathcal{L}, \epsilon)$-leakage-indistinguishable *if for every large enough $t \in \mathbb{N}$, $\mathsf{Enc}(\cdot, 1^t)$ is $(\mathcal{L}, \epsilon)$-leakage indistinguishable.*

AMD Encoding Schemes. Informally, an AMD encoding scheme is an encoding scheme which guarantees that additive attacks on codewords are detected by the decoder (except with small probability), where the decoder outputs (in addition to the decoded output) also a flag indicating whether an additive attack was detected. Formally,

Definition 2.5 (AMD encoding scheme, [?, ?])**.** *Let $\mathbb{F}$ be a finite field, $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, and $\epsilon(n, t) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$. An $(n, t, \epsilon(n, t))$*-algebraic manipulation detection (AMD) encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ over $\mathbb{F}$ *is an encoding scheme with the following guarantees.*

- ***Perfect completeness.*** *For every $\mathbf{x} \in \mathbb{F}^n$,* $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(\mathbf{x}, 1^t\right), 1^t\right) = (0, \mathbf{x})\right] = 1$.
- ***Additive soundness.*** *For every $0^{\hat{n}(n,t)} \neq \mathbf{a} \in \mathbb{F}^{\hat{n}(n,t)}$, and every $\mathbf{x} \in \mathbb{F}^n$,*

$$\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(\mathbf{x}, 1^t\right) + \mathbf{a}, 1^t\right) \notin \mathsf{ERR}\right] \leq \epsilon(n, t)$$

where $\mathsf{ERR} = (\mathbb{F} \setminus \{0\}) \times \mathbb{F}^n$*, and the probability is over the randomness of* Enc.

We will use the following theorem from the full version of [?].

Theorem 2.6 (AMD encoding scheme, [?])**.** *Let $\mathbb{F}$ be a finite field, and $n, t \in \mathbb{N}$. Then there exists an $\left(n, t, |\mathbb{F}|^{-t}\right)$-AMD encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ with encodings of length $\hat{n}(n, t) = O(n + t)$. Moreover, encoding and decoding of length-n inputs with parameter t can be performed by circuits of size $O(n + t)$.*

2.1 Leakage-Resilient Circuit Compilers (LRCCs)

In this section we define the notion of a leakage-resilient circuit compiler. This notion, and its variants defined in later sections, will be extensively used in this work.

Definition 2.7 (Circuit compiler with abort)**.** *We say that a triplet $(\mathsf{Comp}, \mathsf{E}, \mathsf{Dec}_{\mathsf{Out}})$ is a* circuit compiler with abort *if:*

- $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ *is an encoding scheme, where* Enc *on input $x \in \mathbb{F}^n$, and $1^t, 1^{t_{\mathsf{In}}}$, outputs a vector $\hat{x}$ of length $\hat{n}$ for some $\hat{n} = \hat{n}(n, t, t_{\mathsf{In}})$.*
- Comp *is a polynomial-time algorithm that given an arithmetic circuit C over $\mathbb{F}$, and 1^t, outputs an arithmetic circuit $\hat{C}$.*

- $\mathsf{Dec}_{\mathsf{Out}}$ *is a deterministic decoding algorithm associated with a length function* $\hat{n}_{\mathsf{Out}} : \mathbb{N} \to \mathbb{N}$ *that on input* $\hat{x} \in \mathbb{F}^{\hat{n}_{\mathsf{Out}}(n)}$ *outputs* $(f, x) \in \mathbb{F} \times \mathbb{F}^n$.

We require that $(\mathsf{Comp}, \mathsf{E}, \mathsf{Dec}_{\mathsf{Out}})$ *satisfy the following* correctness with abort *property: there exists a negligible function* $\epsilon(t) = \mathsf{negl}(t)$ *such that for any arithmetic circuit* C*, and input* x *for* C*,* $\Pr\left[\mathsf{Dec}_{\mathsf{Out}}\left(\hat{C}(\hat{x})\right) = (0, C(x))\right] \geq 1 - \epsilon(t)$*, where* $\hat{x} \leftarrow \mathsf{Enc}\left(x, 1^t, 1^{|C|}\right)$.

Informally, a circuit compiler is *leakage resilient* for a class $\mathcal{L}$ of functions if for every "not too large" circuit C, and every input x for C, the wire values of the compiled circuit $\hat{C}$, when evaluated on a random encoding $\hat{x}$ of x, can be simulated given only the description of C; and functions in $\mathcal{L}$ cannot distinguish between the actual and simulated wire values.

Notation 2.8. *For a Circuit* C*, a function* $\ell : \mathbb{F}^{|C|} \to \mathbb{F}^m$ *for some natural* m*, and an input* x *for* C*,* $[C, x]$ *denotes the wire values of* C *when evaluated on* x*, and* $\ell[C, x]$ *denotes the output of* ℓ *on* $[C, x]$.

Definition 2.9 (LRCC)**.** *Let* $t \in \mathbb{N}$ *be a security parameter, and* $\mathbb{F}$ *be a finite field. For a function class* $\mathcal{L}$*,* $\epsilon(t) : \mathbb{N} \to \mathbb{R}^+$*, and a size function* $\mathsf{S}(n) : \mathbb{N} \to \mathbb{N}$*, we say that* $(\mathsf{Comp}, \mathsf{E}, \mathsf{Dec}_{\mathsf{Out}})$ *is an* $(\mathcal{L}, \epsilon(t), \mathsf{S}(n))$-LRCC *if there exists a PPT algorithm* Sim *such that the following holds. For all sufficiently large* t*, every arithmetic circuit* C *over* $\mathbb{F}$ *of input length* n *and size at most* $\mathsf{S}(n)$*, every* $\ell \in \mathcal{L}$ *of input length* $|\hat{C}|$*, and every* $x \in \mathbb{F}^n$*, we have* $\mathsf{SD}\left(\ell\left[\mathsf{Sim}\left(C, 1^t\right)\right], \ell\left[\hat{C}, \hat{x}\right]\right) \leq \epsilon(t)$*, where* $\hat{x} \leftarrow \mathsf{Enc}\left(x, 1^t, 1^{|C|}\right)$.

If the above holds with an inefficient *simulator* Sim*, then we say that* $(\mathsf{Comp}, \mathsf{E})$ *is an* $(\mathcal{L}, \epsilon(t), \mathsf{S}(n))$-relaxed LRCC.

2.2 Gadget-Based Leakage-Resilient Circuit Compilers

In this section we describe gadget-based LRCCs [?, ?, ?], which are the basis of all our constructions. We choose to describe the operation of these compilers over a finite field $\mathbb{F}$, but the description naturally adjusts to the boolean case as well. At a high level, given a circuit C, a gadget-based LRCC replaces every wire in C with a bundle of wires, which carry an encoding of the wire value, and every gate with a sub-circuit that emulates the operation of the gate on encoded inputs. More specifically:

Gadgets. A bundle is a sequence of field elements, encoding a field element according to some encoding scheme E; and a gadget is a circuit which operates on bundles and emulates the operation of the corresponding gate in C. A gadget has both standard inputs, that represent the wires in the original circuit, and masking inputs (so-called "masks"), that are used to achieve privacy. More formally, a gadget emulates a specific boolean or arithmetic operation on the standard inputs, and outputs a bundle encoding the correct output. Every gadget G is associated with a set M_G of "well-formed" masking input bundles (e.g., in the LRCC of [?], M_G consists of sets of 0-encodings). For every standard input x, on input a bundle $\mathbf{x}$ encoding x, and *any* masking input bundles $\mathsf{m} \in M_G$, the output of the gadget G should be consistent with the operation on x. For example, if G computes multiplication, then for every standard input $x = (x_1, x_2)$, for every bundle encoding $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ of x according to E, and for every masking input bundles $\mathsf{m} \in M_G$, $G(\mathbf{x}, \mathsf{m})$ is a bundle encoding $x_1 \times x_2$ according to E. Because the encoding schemes we use have the property that the encoding function is onto its range, we may think of the masking input bundles m as encoding some set mask of values. The internal computations in the gadget will remain private as long as its masking input bundles are a uniformly random encoding of mask, *regardless* of the actual value of mask.

Gadget-based LRCCs. In our constructions, the compiled circuit $\hat{C}$ is obtained from a circuit C by replacing every wire with a bundle, and every gate with the corresponding gadget. Recall that the gadgets also have masking inputs (which in previous works [?, ?] were generated by leak-free hardware). These are provided as part of the encoded input of $\hat{C}$, in the following way. $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ uses an "inner" encoding scheme $\mathsf{E}^{\mathsf{In}} = (\mathsf{Enc}^{\mathsf{In}}, \mathsf{Dec}^{\mathsf{In}})$, where Enc uses $\mathsf{Enc}^{\mathsf{In}}$ to encode the inputs of C, concatenated with $0^{t_{\mathsf{In}}}$ for a "sufficiently large" t_{In} (these 0-encodings will be the masking inputs of the gadgets, that are used to achieve privacy); and Dec uses $\mathsf{Dec}^{\mathsf{In}}$ to decode its input, and discards the last t_{In} symbols.

3 LRCCs Used in this Work

In this section we review the various LRCC constructions used in this work.

3.1 The LRCC of [?]

We use a slight modification of the LRCC of Goyal et al. [?], which we describe in this section. Their construction uses *small-bias* encodings over $\mathbb{F}_2$, namely encodings for which linear distinguishers obtain only a small distinguishing advantage between encodings of 0 and 1. Formally:

Definition 3.1 (Small-bias encoding schemes). *Let $\epsilon \in (0,1)$, and $(\mathsf{Enc}, \mathsf{Dec})$ be an encoding scheme over $\mathbb{F}_2$ with encodings of length $\hat{n}$. We say that $(\mathsf{Enc}, \mathsf{Dec})$ is* ϵ-biased *if for every $x \in \mathbb{F}_2$, and every $\emptyset \neq S \subseteq [\hat{n}]$, $\left|\Pr\left[P_S\left(\mathsf{Enc}\left(x\right)\right) = 1\right] - \Pr\left[P_S\left(\mathsf{Enc}\left(x\right)\right) = 0\right]\right| \leq \epsilon$, where $P_S\left(z\right) = \oplus_{i \in S} z_i$, and the probability is over the randomness of* Enc.

At a high level, given a circuit C (which, without loss of generality, contains only NAND gates), its leakage-resilient version is constructed in three steps: first, C is compiled into a *parity resilient* circuit $C_{\oplus}$, which emulates the operation of C on small-bias encodings of its inputs, and resists leakage from the class of all parity function (namely, all functions that output the parity of a subset of wires). $C_{\oplus}$ is constructed using a single constant-size gadget $\mathcal{G}$ that operates over the small-bias encoding. Second, a GMW-style 2-party protocol π is constructed, which emulates $C_{\oplus}$ (gate-by-gate) on additive secret shares of the input, and outputs additive secret shares of the output. π uses an oracle to the functionality computed by the gadget $\mathcal{G}$. In the final step, each oracle call to $\mathcal{G}$ is replaced with a constant number of OT calls, and the resultant 2-party protocol is converted into a boolean circuit, in which the OT calls are implemented using a constant number of gates.[2] The resultant circuit C' operates on encoded inputs, and returns encoded outputs. More specifically, the encoding scheme first encodes each input bit using the small-bias encoding, then additively secret shares these encodings into two shares.

The reason we need to modify the compiler is the small-bias encoding it uses. The LRCC can use *any* small-bias encoding, and [?] construct a robust gadget $\mathcal{G}$, that can emulate *any* constant-sized boolean function, over inputs and outputs encoded according to *any* constant-sized small-bias encoding (the inputs and outputs may actually be encoded using different encoding schemes). However, the specific encoding used in [?] is insufficient for our needs. More specifically, we need an encoding scheme $\left(\mathsf{Enc} : \{0,1\} \times \{0,1\}^c \rightarrow \{0,1\}^{c'}, \mathsf{Dec} : \{0,1\}^{c'} \rightarrow \{0,1\}^2\right)$ (for some natural constants c, c')[3] satisfying the following two properties for some constant $\epsilon > 0$.

- **Property (1):** $(\mathsf{Enc}, \mathsf{Dec})$ is ϵ-biased, and $\left|\mathsf{supp}\left(\mathsf{Enc}\left(0; \cdot\right)\right)\right| = \left|\mathsf{supp}\left(\mathsf{Enc}\left(1; \cdot\right)\right)\right|$.
- **Property (2):** For every $\vec{0} \neq \mathbf{A} \in \{0,1\}^{c'}$, and every $b \in \{0,1\}$, $\Pr_{r \in_R \{0,1\}^c}\left[\mathsf{Enc}\left(b; r\right) \oplus \mathbf{A} \in \mathsf{supp}\left(\mathsf{Enc}\left(1 \oplus b; \cdot\right)\right)\right] \leq \epsilon$.

The first property is needed for the leakage-resilience property of the LRCC of [?]. The second property implies that with constant probability, additive attacks on encodings are "harmless", in the sense that they either do not change the encoded value, or result in an invalid encoding. The reason that the second property is needed will become clear in Section ??.

Since the encoding scheme used in [?] does not possess property (2), we replace it with an encoding that does.[4] As noted in [?], a probabilistic argument implies that for a large enough constant

[2] We note that the conversion from protocol to circuit is not explicitly described in [?].

[3] Dec returns a pair of bits of which one is a flag indicating whether decoding failed. This is necessary since for $c' > c+1$, not all possible inputs to Dec are valid encoding.

[4] To improve efficiency of our construction by a factor of 2, one could use the encoding of [?] (in which $c' = c + 1$) throughout the circuit, and only use our new encoding for the outputs of the circuit. However, to simplify the construction we choose to use the same encoding throughout the circuit.

c, and $c' = 2c$, most encoding schemes with a 1:1 Enc satisfy property (1). A similar argument shows that most encoding schemes posses property (2). Therefore, there exists an encoding scheme $\left(\mathsf{Enc}^{\oplus} : \{0,1\} \times \{0,1\}^c \rightarrow \{0,1\}^{2c}, \mathsf{Dec}^{\oplus} : \{0,1\}^{2c} \rightarrow \{0,1\}^2\right)$ with both properties. (Moreover, one can find an explicit description of this scheme, since c is constant.) Since $\mathcal{G}$ is a generic gadget, that can be used to emulate any function over any encoding, we can replace the encoding scheme of [**?**] with $\left(\mathsf{Enc}^{\oplus}, \mathsf{Dec}^{\oplus}\right)$.

We are now ready to define the encoding used by the LRCC of [**?**].

Construction 3.2. *The encoding scheme* $\left(\mathsf{Enc}^{\mathrm{GIMSS}}, \mathsf{Dec}^{\mathrm{GIMSS}}\right)$ *over* $\mathbb{F}_2$ *is defined as follows:*

- *for every* $x \in \mathbb{F}_2$, $\mathsf{Enc}^{\mathrm{GIMSS}}\left(x, 1^t\right)$*:*
  - *Generates* $x^1, \cdots, x^t \leftarrow \mathsf{Enc}^{\oplus}(x)$.
  - *Picks* $\boldsymbol{x}^L, \boldsymbol{x}^R \in \mathbb{F}_2^{2ct}$ *uniformly at random subject to the constraint that* $\boldsymbol{x}^L \oplus \boldsymbol{x}^R = \left(x^1, \cdots, x^t\right)$.
- $\mathsf{Dec}^{\mathrm{GIMSS}} : \mathbb{F}_2^{2ct} \times \mathbb{F}_2^{2ct} \rightarrow \mathbb{F}_2^2$*, on input* $\left(\boldsymbol{x}^L, \boldsymbol{x}^R\right)$ *operates as follows:*
  - *Computes* $\boldsymbol{x} = \boldsymbol{x}^L \oplus \boldsymbol{x}^R$*, and denotes* $\boldsymbol{x} = \left(x^1, \cdots, x^t\right)$*. (Intuitively,* $\boldsymbol{x}^L, \boldsymbol{x}^R$ *are interpreted as random secret shares of* $\boldsymbol{x}$*, and* $\boldsymbol{x}$ *consists of* t *copies of encodings, according to* $\mathsf{Enc}^{\oplus}$*, of a bit* b*.)*
  - *For every* $1 \leq i \leq t$*, let* $(f_i, x_i) = \mathsf{Dec}^{\oplus}\left(x^i\right)$*. (This step decodes each of the* t *copies of* b*.)*
  - *If there exist* $1 \leq i_1, i_2 \leq t$ *such that* $f_{i_1} \neq 0$*, or* $x_{i_1} \neq x_{i_2}$*, then sets* $f = 1$*. Otherwise, sets* $f = 0$*. (This step checks that all copies of* b *are consistent, and that no flag is set, otherwise the decoder sets a flag* f*.)*
  - *Outputs* $\left(f, x^1\right)$.

We will need the fact that every additive attack on encodings generated by Construction **??** is either "harmless" (in the sense that it does not change the encoded value), or causes a decoding failure. This is formalized in the next lemma.

Lemma 3.3. *Let* $t \in \mathbb{N}$ *be a security parameter. Then for every* $\vec{0} \neq \mathbf{A} \in \mathbb{F}_2^{4ct}$*, and for every* $x \in \mathbb{F}_2$*,*

$$\Pr\left[\mathsf{Dec}^{\mathrm{GIMSS}}\left(\mathsf{Enc}^{\mathrm{GIMSS}}\left(x, 1^t\right) + \mathbf{A}\right) \notin \{(0, x), \mathsf{ERR}\}\right] = \mathsf{negl}(t).$$

Proof. Let $\vec{0} \neq \mathbf{A} = \left(\mathbf{A}^L, \mathbf{A}^R\right) \in \mathbb{F}_2^{2ct} \times \mathbb{F}_2^{2ct}$, and let $\left(\mathbf{x}^L, \mathbf{x}^R\right) \leftarrow \mathsf{Enc}^{\mathrm{GIMSS}}\left(x, 1^t\right)$. Then on input $\left(\mathbf{y}^L, \mathbf{y}^R\right) = \left(\mathbf{x}^L, \mathbf{x}^R\right) + \left(\mathbf{A}^L, \mathbf{A}^R\right)$, the decoder $\mathsf{Dec}^{\mathrm{GIMSS}}$ first computes

$$\mathbf{x}' = \left(x^{1\prime}, \cdots, x^{t\prime}\right) = \mathbf{y}^L \oplus \mathbf{y}^R = \mathbf{x}^L \oplus \mathbf{x}^R \oplus \mathbf{A}^L \oplus \mathbf{A}^R$$

and then for every $1 \leq i \leq t$, computes $(f_i, x_i') \leftarrow \mathsf{Dec}^{\oplus}\left(x^i, 1^t\right)$. We consider two possible cases.

First, if $\mathbf{A}^L \oplus \mathbf{A}^R = \vec{0}$, then $\mathbf{x}' = \mathbf{x}^L \oplus \mathbf{x}^R$, namely the additive attack cancels out, and so the output of $\mathsf{Dec}^{\mathrm{GIMSS}}$ would be $(0, x)$ (with probability 1) by the correctness of the scheme.

Second, assume that $\mathbf{A}^L \oplus \mathbf{A}^R \neq \vec{0}$ and $\mathsf{Dec}^{\mathrm{GIMSS}}\left(\mathbf{x} \oplus \mathbf{A}, 1^t\right) \neq (0, x)$. We show that in this case $\mathsf{Dec}^{\mathrm{GIMSS}}$ outputs ERR except with negligible probability. Recall that $\mathsf{Enc}^{\oplus}$ has the property that for every $\vec{0} \neq \mathbf{A}'$, and every $z \in \mathbb{F}$, $\Pr\left[\mathsf{Enc}^{\oplus}(z) \oplus \mathbf{A}' \in \mathsf{supp}\left(\mathsf{Enc}^{\oplus}(\bar{z})\right)\right] \leq \epsilon$ for some constant $\epsilon \in (0,1)$, where the probability is over the randomness used by $\mathsf{Enc}^{\oplus}$ to generate the encoding. Consequently, for every $1 \leq i \leq t$, $\Pr\left[\mathsf{Dec}^{\oplus}\left(x^{i\prime}\right) = (0, \bar{x})\right] \leq \epsilon$. Since $\mathsf{Dec}^{\mathrm{GIMSS}}$ outputs $(0, \bar{x})$ only if all $x^{i\prime}$ decoded to $\bar{x}$, and each of these t copies was generated using fresh, independent randomness in $\mathsf{Enc}^{\oplus}$, this happens only with probability $\epsilon^t = \mathsf{negl}(t)$. ■

The final modification we need is in the gadget $\mathcal{G}$. Notice that unlike the semi-honest setting considered in [**?**], in our setting *the parties* provide the inputs to the leakage-resilient circuit, where a malicious party may provide inputs that are *not* properly encoded, and therefore do not correspond to *any* input for the original circuit. (We note that the inputs are the only encodings that may be invalid, since $\mathcal{G}$ is guaranteed to always output valid encodings.) To guarantee correctness of the computation even in this case, the encoded inputs should induce inputs to the original circuit. Therefore, we have $\mathcal{G}$ interpret invalid encodings as encoding the all-zeros string. More specifically, given encodings $\hat{x}, \hat{y}$, $\mathcal{G}$ operates as follows: decodes $\hat{x}, \hat{y}$ to obtain x, y, where if decoding failed then x, y are set to the all-zero strings; computes $z = \mathrm{NAND}(x, y)$; and outputs a fresh encoding of z.

Combining the aforementioned modifications, we have the following.

Construction 3.4 (LRCC, [**?**]). *Let* $c \in \mathbb{N}$ *and* $\epsilon \in (0,1)$ *be constants,* $t, t_{\mathsf{In}} \in \mathbb{N}$ *be security parameters, and* $n \in \mathbb{N}$ *be an input length parameter. Let* $\left(\mathsf{Enc}^{\oplus} : \mathbb{F}_2 \times \mathbb{F}_2^c \rightarrow \mathbb{F}_2^{2c}, \mathsf{Dec}^{\oplus} : \mathbb{F}_2^{2c} \rightarrow \mathbb{F}_2\right)$ *be an encoding scheme satisfying properties (1) and (2) described above. (We also use* $\mathsf{Enc}^{\oplus}, \mathsf{Dec}^{\oplus}$ *to denote the natural extension of encoding and decoding to bit strings, where every bit is encoded or decoded separately.) The relaxed LRCC with abort* $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{E}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ *is defined as follows.*

- *The input encoding scheme* $\mathsf{E}_{\mathsf{In}}^{\mathrm{GIMSS}} = \left(\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{In}}^{\mathrm{GIMSS}}\right)$ *is defined as follows:*
  - *for every* $x \in \mathbb{F}_2$, $\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}\left(x, 1^{t_{\mathsf{In}}}\right) = \left(\boldsymbol{x}^L, \boldsymbol{x}^R, \boldsymbol{r}\right)$ *where* $\boldsymbol{x}^L, \boldsymbol{x}^R$ *are a random additive secret sharing of* $\mathsf{Enc}^{\oplus}(x)$, *and* $r \in_R \mathbb{F}_2^{t_{\mathsf{In}}}$.
  - $\mathsf{Dec}_{\mathsf{In}}^{\mathrm{GIMSS}}\left(\left(\left(\boldsymbol{x}^L, \boldsymbol{x}^R\right), \boldsymbol{r}\right), 1^{t_{\mathsf{In}}}\right)$ *computes* $(f, x) = \mathsf{Dec}^{\oplus}\left(\boldsymbol{x}^L + \boldsymbol{x}^R\right)$, *and outputs* x.
- *The output decoding algorithm* $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}} : \mathbb{F}_2^{n \cdot t \cdot 2c} \times \mathbb{F}_2^{n \cdot t \cdot 2c} \rightarrow \mathbb{F}_2^{n+1}$, *on input* $\left(\boldsymbol{x}^L, \boldsymbol{x}^R\right) = \left(\left(\boldsymbol{x}_1^L, \cdots, \boldsymbol{x}_n^L\right), \left(\boldsymbol{x}_1^R, \cdots, \boldsymbol{x}_n^R\right)\right)$ *operates as follows:*
  - *For every* $1 \leq i \leq n$, *computes* $(f_i, x_i) = \mathsf{Dec}^{\mathrm{GIMSS}}\left(\left(\boldsymbol{x}_i^L, \boldsymbol{x}_i^R\right), 1^t\right)$ *(where* $\mathsf{Dec}^{\mathrm{GIMSS}}$ *is the decoder from Construction* **??***).*
  - *If there exist* $1 \leq i \leq n$ *such that* $f_i \neq 0$, *outputs* $(1, 0^n)$. *Otherwise, outputs* $(f, x_1, \cdots, x_n)$.
- *Let* $r \in \mathbb{N}$ *denote the number of random inputs used by each gadget* $\mathcal{G}$. *Then* $\mathsf{Comp}^{\mathrm{GIMSS}}$, *on input* 1^t *and a circuit* $C : \mathbb{F}^n \rightarrow \mathbb{F}^k$ *containing* s *NAND gates, outputs a circuit* $C^{\mathrm{GIMSS}} : \mathbb{F}_2^{4c \cdot n} \times \mathbb{F}_2^{r(s+t \cdot k)} \rightarrow \mathbb{F}_2^{4c \cdot k \cdot t}$ *generated as follows:*
  - *Let* $C' : \mathbb{F}_2^{2c \cdot n} \times \mathbb{F}_2^{r \cdot s} \rightarrow \mathbb{F}_2^{2c \cdot k}$ *denote the circuit in which every gate of* C *is replaced with the gadget* $\mathcal{G}$ *of [***?***] that emulates a NAND gate over encodings generated by* $\mathsf{Enc}^{\oplus}$. *The random inputs used by the gadgets in* C' *are taken from the second input to* C' *(each random input is used only once).*
  - *Let* $C'' : \mathbb{F}_2^{2c \cdot n} \times \mathbb{F}_2^{r(s+t \cdot k)} \rightarrow \mathbb{F}_2^{2c \cdot k \cdot t}$ *denote the circuit obtained from* C' *by adding after each output gadget of* C' *(namely each gadget whose output is an output of* C'*)* t *gadgets* $\mathcal{G}$ *emulating the identity function. As in* C', *the random inputs used by the gadgets in* C'' *are taken from the second input to* C''. *(This step encodes each output bit using the repetition code.)*[5]
  - *Let* π *denote a 2-party GMW-style protocol in the OT-hybrid model which emulates* C'' *gadget-by-gadget on inputs encoded according to* $\mathsf{Enc}^{\mathrm{GIMSS}}$ *(i.e., on additive shares of encodings according to* $\mathsf{Enc}^{\oplus}$*). Then* C^{GIMSS} *is the circuit obtained from* π *by implementing the programs of the parties as a circuit, where each OT call with inputs* $(x_0, x_1), b$ *is implemented using the following constant-sized circuit:* $\mathsf{OT}((x_0, x_1), b) = \left(x_0 \wedge \bar{b}\right) \oplus (x_1 \wedge b)$. *(The wires of this circuit are divided between the parties as follows: the input wires* x_0, x_1 *are assigned to the OT*

[5] This step, which we add to the LRCC of [**?**], is used to reduce the decoding error when the LRCC is used to construct leakage-secure ZK circuits in Section **??**. We note that this modification preserves the parity-resilience property since it is equivalent to duplicating each output of C t times before transforming it into C'.

sender; whereas the wires corresponding to $b, \bar{b}$, the outputs of the $\wedge$ gates, and the output of the $\oplus$ gate, are assigned to the OT receiver.[6]*)*

Goyal et al. [?] show that Construction **??** resists BCL (Definition **??**):

Theorem 3.5 (Implicit in [?])**.** *For every leakage-bound $t \in \mathbb{N}$, input and output lengths $n, k \in \mathbb{N}$, and size bound $s \in \mathbb{N}$, there exists an $\left(\mathcal{L}^t_{\mathrm{BCL}}, 2^{-t}, s\right)$-relaxed LRCC with abort, where $\mathcal{L}^t_{\mathrm{BCL}}$ is the family of all t-BCL functions. Moreover, on input a size-s, depth d circuit $C : \{0,1\}^n \to \{0,1\}^k$, the leakage-resilient circuit C^{GIMSS} has size $\widetilde{O}\left(s + td + t^2\right)$, the input encoder $\mathsf{Enc}^{\mathrm{GIMSS}}_{\mathsf{In}}$ can be implemented by a circuit of size $\widetilde{O}\left(n + t\right)$, and the output decoder $\mathsf{Dec}^{\mathrm{GIMSS}}_{\mathsf{Out}}$ can be implemented by a circuit of size $\widetilde{O}\left(t^2 + tk\right)$.*[7]

3.2 The Leakage-Tolerant Circuit-Compiler of [?]

In this section we describe the leakage-tolerant circuit-compiler (LTCC) obtained from [?] through the transformation of [?]. Informally, the LRCC of Dziembowski and Faust [?], denoted DF-LRCC, is a gadget-based LRCC which uses the inner-product encoding scheme that encodes a value x as a pair of vectors whose inner-product is x:

Definition 3.6 (Inner product encoding scheme)**.** *Let $\mathbb{F}$ be a finite field, and $n \in \mathbb{N}$ be an input length parameter. The* inner product encoding scheme $\mathsf{E}_{\mathsf{IP}} = \left(\mathsf{Enc}_{\mathsf{IP}}, \mathsf{Dec}_{\mathsf{IP}}\right)$ *over $\mathbb{F}$ is a parameterized encoding scheme defined as follows:*

- *For every input $x = (x_1, \cdots, x_n) \in \mathbb{F}^n$, and security parameter $t \in \mathbb{N}$, $\mathsf{Enc}_{\mathsf{IP}}\left(x, 1^t\right) = \left(\left(\boldsymbol{y}^L_1, \boldsymbol{y}^R_1\right), \cdots, \left(\boldsymbol{y}^L_n, \boldsymbol{y}^R_n\right)\right)$, where for every $1 \le i \le n$, $\boldsymbol{y}^L_i, \boldsymbol{y}^R_i$ are random in $\left(\mathbb{F} \setminus \{0\}\right)^t$ subject to the constraint that $\langle \boldsymbol{y}^L_i, \boldsymbol{y}^R_i \rangle = x_i$.*
- *For every $t \in \mathbb{N}$, and every $\left(\left(\boldsymbol{y}^L_1, \boldsymbol{y}^R_1\right), \cdots, \left(\boldsymbol{y}^L_n, \boldsymbol{y}^R_n\right)\right) \in \mathbb{F}^{2nt}$, $\mathsf{Dec}_{\mathsf{IP}}\left(\left(\boldsymbol{y}^L_1, \boldsymbol{y}^R_1\right), \cdots, \left(\boldsymbol{y}^L_n, \boldsymbol{y}^R_n\right)\right) = \left(\langle \boldsymbol{y}^L_1, \boldsymbol{y}^R_1 \rangle, \cdots, \langle \boldsymbol{y}^L_n, \boldsymbol{y}^R_n \rangle\right)$.*

More specifically, the DF-LRCC is an LRCC variant in which the compiled circuit takes un-encoded inputs, as well as masking inputs that are used in the gadgets. The construction uses 4 gadgets: a refresh gadget which emulates the identity function, and is used to generate fresh encodings of the wires; a *generalized-multiplication* gadget which emulates the function $f_c\left(x, y\right) = c - x \times y$, for a constant $c \in \mathbb{F}$; a *multiplication by a constant* gadget that emulates the function $f_c\left(x\right) = c \times x$, for a constant $c \in \mathbb{F}$; and an *addition by a constant* gadget that emulates the function $f_c\left(x\right) = c + x$, for a constant $c \in \mathbb{F}$. (The field operations $\times, +, -$ can be implemented using a constant number of these gadgets.) We will only need the following property of these gadgets: the effect of evaluating a gadget with ill-formed masking inputs is equivalent to an additive attack on the gate that the gadget emulates (this is formalized in Lemma **??**).

Construction 3.7 (Gadgets for an LRCC, [?])**.** *Let $\mathbb{F}$ be a finite field, and $\mathsf{E}_{\mathsf{IP}} = \left(\mathsf{Enc}_{\mathsf{IP}}, \mathsf{Dec}_{\mathsf{IP}}\right)$ denote the inner product encoding over $\mathbb{F}$ of Definition **??**. Each gadget consists of a* left component *C^L, and a* right component *C^R that are connected to each other. We use the term "X is sent from component Y to component Z" to denote that the value X computed in component Y is the input to some sub-computation performed in component Z.*

1. ***Refresh gadget:***[8] *inputs $\left(\boldsymbol{a}^L, \boldsymbol{a}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}(a, 1^{t^2})$ for $a \in \mathbb{F}$, and masking inputs $\left(\left(\boldsymbol{r}^{L,1}, \boldsymbol{r}^{L,2}\right), \left(\boldsymbol{r}^{R,1}, \boldsymbol{r}^{R,2}\right)\right) \in \mathsf{Enc}^{\mathsf{In}}_{\mathrm{DF}}(0, 1^{t^2})$; outputs $\left(\boldsymbol{a}^{L\prime}, \boldsymbol{a}^{R\prime}\right) \in \mathsf{Enc}_{\mathsf{IP}}(a, 1^{t^2})$.*

[6] Notice that this division of the wires preserves the leakage-resilience guarantee of [?]. Indeed, in [?] the view of the OT sender contains the input wires x_0, x_1, whereas the view of the OT receiver contains the input wire b and the output of the OT (i.e., the output of the $\oplus$ gate). Notice that $\bar{b}$ and the outputs of the $\wedge$ gates are computable from b and the OT output, so the view of the OT receiver contains exactly the same information in [?] and in our implementation of their protocol.

[7] The output decoder in the original construction of [?] has size $\widetilde{O}\left(t + k\right)$, the decoder of Construction **??** is larger due to the modified encoding we use, which replaces each encoded output bit with t copies.

[8] This refresh gadget is a simpler construction than the original gadget of [?], due to [?].

   (a) *C^L generates $\boldsymbol{b} \in \mathbb{F}^{t^2}$ such that $\boldsymbol{b}_i = \left(\boldsymbol{a}_i^L\right)^{-1} \times \boldsymbol{r}_i^{L,1}$ for every $1 \leq i \leq t^2$, and sends $\boldsymbol{b}$ to C^R.*

   (b) *C^R computes $\boldsymbol{c} \in \mathbb{F}^{t^2}$ such that $\boldsymbol{c}_i = \boldsymbol{b}_i \times \boldsymbol{r}_i^{R,1}$ for every $1 \leq i \leq t^2$.*

   (c) *C^R computes $\boldsymbol{a}^{R\prime} = \boldsymbol{a}^R + \boldsymbol{c}$.*

   (d) *C^R generates $\boldsymbol{d} \in \mathbb{F}^{t^2}$ such that $\boldsymbol{d}_i = \left(\boldsymbol{a}_i^{R\prime}\right)^{-1} \times \boldsymbol{r}_i^{R,2}$ for every $1 \leq i \leq t^2$, and sends $\boldsymbol{d}$ to C^L.*

   (e) *C^L computes $\boldsymbol{e} \in \mathbb{F}^{t^2}$ such that $\boldsymbol{e}_i = \boldsymbol{d}_i \times \boldsymbol{r}_i^{L,2}$ for every $1 \leq i \leq t^2$.*

   (f) *C^L computes $\boldsymbol{a}^{L\prime} = \boldsymbol{a}^L + \boldsymbol{e}$..*

2. ***Multiplication by constant gadget:*** *inputs constant $c \in \mathbb{F} \setminus \{0\}$, and $\left(\boldsymbol{a}^L, \boldsymbol{a}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(a, 1^t\right)$ for $a \in \mathbb{F}$; output $\left(\boldsymbol{b}^L, \boldsymbol{b}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(c \times a, 1^t\right)$.*

   (a) *C^L computes $\boldsymbol{b}_i^L = c \times \boldsymbol{a}_i^L$ for every $1 \leq i \leq t$.*

   (b) *C^R sets $\boldsymbol{b}^R = \boldsymbol{a}^R$.*

3. ***Addition by constant gadget:*** *inputs constant $c \in \mathbb{F}$, and $\left(\boldsymbol{a}^L, \boldsymbol{a}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(a, 1^t\right)$ for $a \in \mathbb{F}$; output $\left(\boldsymbol{b}^L, \boldsymbol{b}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(c + a, 1^t\right)$.*

   (a) *C^L sets $\boldsymbol{b}^L = \boldsymbol{a}^L$, and sends $\boldsymbol{a}_1^L$ to C^R.*

   (b) *C^R sets $\boldsymbol{b}^R = \boldsymbol{a}^R + \left(\left(\boldsymbol{a}_1^L\right)^{-1} \times c, 0, \cdots, 0\right)$.*

4. ***Generalized multiplication gadget:*** *inputs a constant $c \in \mathbb{F}$, $\left(\boldsymbol{a}^L, \boldsymbol{a}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(a, 1^t\right), \left(\boldsymbol{b}^L, \boldsymbol{b}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(b, 1^t\right)$ for $a, b \in \mathbb{F}$, and masking inputs $\left(\left(\boldsymbol{r}^{L,1}, \boldsymbol{r}^{L,2}\right), \left(\boldsymbol{r}^{R,1}, \boldsymbol{r}^{R,2}\right)\right) \in \mathsf{Enc}_{\mathrm{DF}}^{\mathsf{In}}\left(0, 1^t\right)$; output $\left(\boldsymbol{c}^L, \boldsymbol{c}^R\right) \in \mathsf{Enc}_{\mathsf{IP}}\left(c - a \times b, 1^t\right)$.*

   (a) *C^L generates a $t \times t$ Matrix $\boldsymbol{L} = \boldsymbol{a}^L \left(\boldsymbol{b}^L\right)^T = \left(a_i^L \times b_j^L\right)_{i,j \in [t]}$. We interpret L as a length-t^2 vector.*

   (b) *C^R enerates a $t \times t$ Matrix $\boldsymbol{R} = \boldsymbol{a}^R \left(\boldsymbol{b}^R\right)^T = \left(a_i^R \times b_j^R\right)_{i,j \in [t]}$. We interpret R as a length-t^2 vector.*

   (c) *C^L, C^R evaluate the Refresh gadget with inputs $\boldsymbol{L}, \boldsymbol{R}$, and masking inputs $\left(\left(\boldsymbol{r}^{L,1}, \boldsymbol{r}^{L,2}\right), \left(\boldsymbol{r}^{R,1}, \boldsymbol{r}^{R,2}\right)\right)$, to obtain $\boldsymbol{L}', \boldsymbol{R}'$ (which are length-t^2 vectors).*

   (d) *C^L sends $L_1', L_{t+1}', \cdots L_{t^2}'$ to C^R.*

   (e) *C^R computes $d = \langle \left(L_{t+1}', \cdots L_{t^2}'\right), \left(R_{t+1}', \cdots, R_{t^2}'\right) \rangle$.*

   (f) *C^R computes $\boldsymbol{c}^R = -\left(R_1', \cdots, R_t'\right) + \left(\left(L_1'\right)^{-1}(c - d), 0, \cdots, 0\right)$.*

   (g) *C^L computes $\boldsymbol{c}^L = \left(L_1', \cdots, L_t'\right)$.*

Remark 3.8 (Amplifying correctness)**.** *The execution in each gadget can fail (if the generated encodings are not valid inner-product encodings). However, [?] show that for $|\mathbb{F}| = \Omega(t)$, if each computation step is implemented using t copies of the corresponding gadget (and the output of the computation step is set to the output of the first gadget whose output is valid), then each computation step succeeds except with $\mathsf{negl}(t)$ probability. In what follows, we implicitly assume that each computation step is implemented using this amplification technique over t gadgets.*

As explained in Section **??**, we use a leakage-tolerant variant of the DF-LRCC. Roughly speaking, a leakage-tolerant circuit operates on un-encoded inputs and outputs (the input encoding function simply returns the inputs, concatenated with masking inputs), where any leakage on the computation can be simulated by related leakage *on the inputs and outputs alone.* (Leakage on the inputs and outputs is unavoidable since these are provided to the circuit "in the clear".) Formally,

Definition 3.9 (LTCC (for BCL)). *Let $t, \epsilon(t), \mathsf{S}(n)$ be as in Definition* **??**, *let $n, k \in \mathbb{N}$ be input and output length parameters (respectively), and let $\mathcal{L}_{\mathrm{BCL}}^{t}$ be the family of t-BCL functions. We say that a pair* $(\mathsf{Comp}, \mathsf{E})$ *is an* $\left(\mathcal{L}_{\mathrm{BCL}}^{t}, \epsilon(t), \mathsf{S}(n)\right)$-leakage-tolerant circuit-compiler (LTCC) *if* $\mathsf{Comp}, \mathsf{E}$ *have the syntax of Definition* **??**, *and satisfy the following properties for some negligible function $\epsilon(t) = \mathsf{negl}(t)$:*

- ***Correctness.*** *For any arithmetic circuit C, and input x for C, $\Pr\left[\hat{C}(\hat{x}) = C(x)\right] \geq 1 - \epsilon(t)$, where $\hat{x} \leftarrow \mathsf{Enc}\left(x, 1^{t}, 1^{|C|}\right)$.*

- ***(Oblivious) leakage-tolerance.*** *There exists a partition $\mathcal{P} = ((n_1, n_2), (k_1, k_2))$ of input and output lengths, and a PPT algorithm* Sim *such that the following holds for all sufficiently large $t \in \mathbb{N}$, all $n, k \in \mathbb{N}$, every arithmetic circuit $C : \mathbb{F}^{n} \rightarrow \mathbb{F}^{k}$ of size at most $\mathsf{S}(n)$, and every $\ell \in \mathcal{L}_{\mathrm{BCL}}^{t}$ of input length $|\hat{C}|$.* Sim *is given C, and outputs a* view translation circuit *$\mathcal{T} = (\mathcal{T}_1, \mathcal{T}_2)$ such that for every $(x_1, x_2) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2}$,*

$$\mathsf{SD}\left(\ell\left(\mathcal{T}_1\left(x_1, C(x_1, x_2)_1\right), \mathcal{T}_2\left(x_2, C(x_1, x_2)_2\right)\right), \ell\left[\hat{C}, (x_1, x_2)\right]\right) \leq \epsilon(t)$$

  where $C(x_1, x_2) = \left(C(x_1, x_2)_1, C(x_1, x_2)_2\right) \in \mathbb{F}^{k_1} \times \mathbb{F}^{k_2}$.

We use a recent result of Bitansky et al. [?], that show a general transformation from LRCCs with a strong simulation guarantee against OCL, to LTCCs. Recently, Dachman-Soled [?] observed that the DF-LRCC has this strong simulation property, namely the transformation can be applied directly to the DF-LRCC.[9] The final LTCC will use the following circuit $C^{\mathrm{LR-DF}}$:

Definition 3.10. *Let $t \in \mathbb{N}$ be a security parameter, and let $r = r(t)$ denote the maximal length of masking inputs used by a gadget of Construction* **??**. *For an arithmetic circuit $C : \mathbb{F}^{n} \rightarrow \mathbb{F}^{k}$ containing $+$ and $\times$ gates, defined the circuit $C^{\mathrm{LR-DF}} : \mathbb{F}^{n + r(t) \cdot (n + |C|)} \rightarrow \mathbb{F}^{k}$ as follows:*

- *The input $(x = (x_1, \cdots, x_n), \boldsymbol{m}) \in \mathbb{F}^{n} \times \left(\mathsf{supp}\left(\mathsf{Enc}_{\mathrm{DF}}^{\mathsf{In}}\left(0, 1^{t}\right)\right)\right)^{|C|+n}$ of $C^{\mathrm{LR-DF}}$ is interpreted as an input x for C, and a collection $\boldsymbol{m}$ of masking inputs for gadgets.*

- *Every gate of C is replaced with the corresponding gadget (as defined in Construction* **??**), *and gadgets corresponding to output gates are followed by decoding sub-circuits (computing the decoding algorithm* $\mathsf{Dec}_{\mathsf{IP}}$ *of the inner product encoding of Definition* **??**). *The masking inputs used in the gadgets are taken from $\boldsymbol{m}$ (every masking input in $\boldsymbol{m}$ is used at most once).*

- *Following each input gate x_i, an encoding sub-circuit (with some fixed, arbitrary randomness hard-wired into it) is added, computing the inner-product encoding of x_i.*

- *A refresh gadget is added following every encoding sub-circuit, where the masking inputs used in the gadgets are taken from $\boldsymbol{m}$.*

We now describe the LTCC of [?]. To simplify the notations and constructions, we define the LTCC only for circuits operating on pairs of inputs.

Construction 3.11 (Leakage-tolerant circuit compiler, [?] and [?]). *Let $t, t_{\mathsf{In}} \in \mathbb{N}$, and $n \in \mathbb{N}$ be an input length parameter. Let $\mathsf{S} : \mathbb{N}^{4} \rightarrow \mathbb{N}$ be a length function whose value is set below. The LTCC* $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ *is defined as follows:*

- $\mathsf{E}^{\mathrm{DF}} = \left(\mathsf{Enc}^{\mathrm{DF}}, \mathsf{Dec}^{\mathrm{DF}}\right)$, *where:*

[9] We note that though Bitansky et al. [?] construct leakage-tolerant circuits based on the DF-LRCC, since they are interested in obtaining UC-security against continuous leakage, they use a more complex variant of the LRCC. We prefer to use the DF-LRCC directly, since it suffices for our needs, and gives a much simpler construction.

  - *For every* $x \in \mathbb{F}^n$, $\mathsf{Enc}^{\mathrm{DF}}\left(x, 1^t, 1^{t_{\mathsf{In}}}\right) = \left(x, \left(\mathsf{Enc}^{\mathsf{In}}_{\mathrm{DF}}\left(0, 1^t\right)\right)^{2t_{\mathsf{In}}}\right)$, *where* $\left(\mathsf{Enc}^{\mathsf{In}}_{\mathrm{DF}}\left(0, 1^t\right)\right)^k$ *denotes* k *random and independent evaluations of* $\mathsf{Enc}^{\mathsf{In}}_{\mathrm{DF}}\left(0, 1^t\right)$.
  - $\mathsf{Dec}^{\mathrm{DF}}\left(\left(x, \boldsymbol{m}\right), 1^t, 1^{t_{\mathsf{In}}}\right) = x$.

- $\mathsf{Comp}^{\mathrm{DF}}$, *on input an arithmetic circuit* $C : \mathbb{F}^{n_L} \times \mathbb{F}^{n_R} \rightarrow \mathbb{F}^k$, *outputs the circuit* $C^{\mathrm{DF}} : \mathbb{F}^{2n_R + n_L + \mathsf{S}(t, n_L, n_R, |C|)} \rightarrow \mathbb{F}^k$ *constructed as follows:*
  - *Construct a circuit* $C_1 : \mathbb{F}^{n_R} \times \mathbb{F}^{n_R} \rightarrow \mathbb{F}^{n_R}$ *that evaluates the function* $f_1(x, y) = x + y$. *Denote* $s_1 = |C_1|$, *and let* C_1' *be the circuit obtained from* C_1 *by the transformation of Definition* **??**. *(Notice that if* y *is uniformly random then* C_1' *outputs a one-time pad encryption of* x*.)*
  - *Construct the circuit* $C_2 : \mathbb{F}^{n_L + n_R} \times \mathbb{F}^{n_R} \rightarrow \mathbb{F}^k$ *such that* $C_2((z, c), y) = C(c + y, z)$. *Denote* $s_2 = |C_2|$, *and let* C_2' *be the circuit obtained from* C_2 *by the transformation of Definition* **??**. *(Notice that if* c *is a one-time pad encryption of some value* x *with pad* y, *then* C_2' *emulates* C *on* x *and* z*.)*
  - *Let* $r = r(t)$ *denote the total length of masking inputs used by a gadget of Construction* **??**. *Then* $\mathsf{S} = \mathsf{S}(t, n_L, n_R, |C|) = r(t) \cdot (s_1 + s_2 + n_L + 4n_R)$. *(Notice that* S *is the number of masking inputs used in* C_1' *and* C_2'*.)*
  - $C^{\mathrm{DF}}(x, y, z) = C_2'(z, (C_1'(x, y)), y)$. *(Intuitively,* C^{DF} *first uses* C_1' *to encrypt* x *with pad* y, *and then evaluates* C_2' *on the encryption output by* C_1', z *and pad* y*.)*

We note that the correctness error of the LTCC of Theorem **??** might be abused by malicious parties (e.g., a malicious ZK prover in Section **??**, or malicious parties in Section **??**) to violate the correctness of the computation, which we overcome by checking whether a correctness error occurred, as described in the following remark.

Remark 3.12 (Dealing with gadget failures)**.** *We will actually use a modified version of Construction* **??**, *in which* C^{DF} *also computes an error flag, indicating whether the computation failed in one of its gadget (i.e., failed in* all t copies of the gadget*, see Remark* **??**). *More specifically, each of the two parties implementing the gadget computes* in the clear *a flag indicating whether its encoding of the output is a valid encoding (i.e., all entries are non-zero), and each party locally combines the flags it generated for all the gadgets. This additional computation is needed since malicious parties (e.g., a malicious prover in the leakage-secure ZK circuit of Construction* **??**) *may* not *choose the masking inputs at random, and might generate them in a "smart" way which will* always *cause gadgets to fail.*

We note that thought these flags are generated in the clear, they do not violate the leakage-tolerance property of Construction **??**. *The reason is these flags are generated locally (by each of the parties), and so could be generated by the leakage function from the simulated wire values which the LT simulator (of Definition* **??**) *generates. This observation gives a reduction from any t-BCL function on the modified circuit to a t-BCL function on the* original *circuit, and so when using Construction* **??** *as a building block, we will implicitly disregard these additional wires (remembering that any leakage on the modified circuit* with *the flags can be generated by related leakage on the original circuit). Finally, we note that in an honest execution the flag is set only with negligible probability (and so the fact that the flag is computed in the clear does not violate leakage-resilience).*

Remark 3.13. *To combine Construction* **??** *with Construction* **??**, *we assume that Construction* **??** *is implemented using a boolean circuit (implementing group operations via operations over* $\mathbb{F}_2$*) that operates over a standard basis.*

Dziembowski and Faust (Corollary 2 in the full version of [**?**]) show that the DF-LRCC resists OCL leakage, which by the result of [**?**] implies the existence of an LTCC against such leakage. Combined with Lemma **??** below (which shows a relation between OCL and BCL), we have the following:

Theorem 3.14 ([?] and [?], and Lemma **??**). *Let $t \in \mathbb{N}$ be a leakage bound, and $n, k \in \mathbb{N}$ be input and output length parameters. Then for every polynomial $p(t)$ there exist a finite field $\mathbb{F}$ of size $\Omega(t)$, and a negligible function $\epsilon(t) = \mathsf{negl}(t)$ for which there exists an $\left(\mathcal{L}^{\widetilde{t}}_{\mathrm{BCL}}, \epsilon(t), p(t)\right)$-LTCC, where $\widetilde{t} = 0.16t\log_2|\mathbb{F}|-1-\log_2|\mathbb{F}|$, and $\mathcal{L}^{T}_{\mathrm{BCL}}$ is the family of all $\widetilde{t}$-BCL functions.*

Theorem **??** relies on the next lemma which states that security against so-called "only computation leaks" (OCL) implies security against BCL. (One can also show that $2t$-BCL implies resilience against t-OCL.) Recall that in the context of OCL, the wires of the leakage-resilient circuit $\widehat{C}$ are divided according to some partition $\mathcal{P}$, into two "parts" $\widehat{C}_L, \widehat{C}_R$. The input encodings of $\widehat{C}$ are also divided into two parts, e.g., an encoding $\widehat{x}$ is divided into $\widehat{x}_L$ (which is the input of $\widehat{C}_L$) and $\widehat{x}_R$ (which constitutes the input to $\widehat{C}_R$) The adversary can (adaptively) pick functions $f^L_1, \cdots, f^L_{n_L}$, and $f^R_1, \cdots, f^R_{n_R}$ for some $n_L, n_R \in \mathbb{N}$, where the combined output lengths of $f^L_1, \cdots, f^L_{n_L}$ (and $f^R_1, \cdots, f^R_{n_R}$) is at most t. In the execution of $\widehat{C}$ on $\widehat{x}$, the adversary is given $f^L_i\left[\widehat{C}_L, \widehat{x}_L\right], 1 \leq i \leq n_L$ and $f^R_i\left[\widehat{C}_R, \widehat{x}_R\right], 1 \leq i \leq n_R$, and chooses the next leakage functions based on previous leakage. The output of the leakage is taken to be the combined outputs of all leakage functions $f^L_1, \cdots, f^L_{n_L}, f^R_1, \cdots, f^R_{n_R}$. We say that a circuit is $\left(\mathcal{L}^t_{\mathrm{OCL}}, \epsilon\right)$-leakage-resilient with relation to the partition $\mathcal{P} = \left(\widehat{C}_L, \widehat{C}_R\right)$, if the real-world output of the OCL functions can be efficiently simulated (given only the description of the circuit, and its outputs if $\widehat{C}$ computes the outputs in the clear), and the statistical distance between the actual and simulated wire values is at most ϵ. (We refer the reader to, e.g., [?] for a more formal definition of OCL.) We note that we allow the adversary to leak on the two components of the computation in an arbitrary order, a notion which is sometimes referred to as "OCL+".

Lemma 3.15 (OCL+-resilience implies BCL-resilience). *Let $\epsilon \in (0,1)$ be an error bound, $t \in \mathbb{N}$ be a leakage bound, and C be a boolean circuit. If C is $\left(\mathcal{L}^t_{\mathrm{OCL}}, \epsilon\right)$-leakage-resilient with relation to partition $\mathcal{P}$, then C is also $(\mathcal{L}, \epsilon)$-leakage-resilient for the family $\mathcal{L}$ of all t-BCL functions with relation to the same partition $\mathcal{P}$.*

Proof (sketch). Let ℓ be a t-BCL function that corresponds to a two party protocol Π, defined in relation to partition $\mathcal{P}$. Let $\mathsf{NextMsg}_L, \mathsf{NextMsg}_R$ be the next-message functions defining the messages the parties send, given their current view, and assume without loss of generality that the left party sends the first message in the protocol. Let $(\widehat{x}_L, \widehat{x}_R)$ be the input on which $\widehat{C}$ is evaluated, and denote $\mathcal{W}_L = \left[\widehat{C}_L, \widehat{x}_L\right]$, and $\mathcal{W}_R = \left[\widehat{C}_R, \widehat{x}_R\right]$.

To generate the transcript of Π, the adversary operates as follows. First, it picks $f^L_1(z) = \mathsf{NextMsg}_L(z)$. Then, given $f^L_1(\mathcal{W}_L)$, which is the first message that the left party sends in Π, it picks f^R_1 to be the function which $\mathsf{NextMsg}_R$ computes, conditioned on the event that $f^L_1(\mathcal{W}_L)$ was the first message which the right party received, and sends f^R_1, to be evaluated on $\mathcal{W}_R$. The adversary continues in this way until all messages of Π have been computed. Since Π is t-bounded, then in particular each of the two participating parties sends at most t bits, namely the leakage functions we have defined leak at most t bits on each of $\mathcal{W}_L, \mathcal{W}_R$. Therefore, the t-OCL resilience of C guarantees that the leakage can be efficiently simulated, up to statistical distance ϵ. ■

The following property of Construction **??** will be used to guarantee correctness of our constructions in the presence of malicious parties.

Lemma 3.16 (Ill-formed masking inputs correspond to additive attacks). *Let $\mathsf{S} : \mathbb{N}^4 \rightarrow \mathbb{N}$ be the length function from Definition **??**. Then Construction **??** has the following property. For every circuit $C : \mathbb{F}^{n_L} \times \mathbb{F}^{n_R} \rightarrow \mathbb{F}^k$, every security parameter $t \in \mathbb{N}$, and every $\mathsf{m} \in \mathbb{F}^{\mathsf{S}(t, n_L, n_R, |C|)}$, there exists an additive attack $\mathcal{A}_{\mathsf{m}}$ on C such that for every $x \in \mathbb{F}^{n_L+n_R}$, and every $\hat{x} = (x, \mathsf{m})$ it holds that $C^{\mathrm{DF}}(\hat{x}) = C^{\mathcal{A}_{\mathsf{m}}}(x)$. Moreover, there exists a PPT algorithm Alg such that $\mathsf{Alg}(\mathsf{m}) = \mathcal{A}_{\mathsf{m}}$.*

Proof. We analyze the effect of ill-formed masking inputs m in the gadgets of Construction **??**, and show that they correspond to applying an additive attack on the underlying gate.

- **Refresh gadget.** Denote $m = \langle \mathbf{r}^{L,1}, \mathbf{r}^{R,1} \rangle + \langle \mathbf{r}^{L,2}, \mathbf{r}^{R,2} \rangle$ (which, if the masking inputs are ill-formed, may not be 0). Then the output of the gadget encodes the value $\langle \mathbf{a}^{L\prime}, \mathbf{a}^{R\prime} \rangle$. We analyze this value. $\langle \mathbf{a}^{L\prime}, \mathbf{a}^{R\prime} \rangle = \sum_{i=1}^{t^2} a_i^{L\prime}, a_i^{r\prime}$ which, by the definition of $\mathbf{a}^{L\prime}, \mathbf{a}^{L\prime}$ is equal to

  $$\sum_{i=1}^{t^2} \left(a_i^L + e_i\right)\left(a_i^R + c_i\right) = \sum_{i=1}^{t^2} a_i^L a_i^R + \sum_{i=1}^{t^2} e_i \left(a_i^R + c_i\right) + \sum_{i=1}^{t^2} a_i^L c_i = a + \sum_{i=1}^{t^2} e_i a_i^{R\prime} + \sum_{i=1}^{t^2} a_i^L c_i$$

  which, by the definition of $\mathbf{c}, \mathbf{e}$, is equal to

  $$a + \sum_{i=1}^{t^2} \left(a_i^{R\prime}\right)^{-1} r_i^{R,2} r_i^{L,2} a_i^{R\prime} + \sum_{i=1}^{t^2} a_i^L \left(a_i^{L,1}\right)^{-1} r_i^{L,1} r_i^{R,1} = a + \langle \mathbf{r}^{L,1}, \mathbf{r}^{R,1} \rangle + \langle \mathbf{r}^{L,2}, \mathbf{r}^{R,2} \rangle = a + m$$

  Moreover, notice that m can be efficiently computed from $\mathbf{r}^{L,1}, \mathbf{r}^{R,1}, \mathbf{r}^{L,2}, \mathbf{r}^{R,2}$ by computing $\langle \mathbf{r}^{L,1}, \mathbf{r}^{R,1} \rangle + \langle \mathbf{r}^{L,2}, \mathbf{r}^{R,2} \rangle$.

- **Generalized multiplication gadget.** Denote $m = \langle \mathbf{r}^{L,1}, \mathbf{r}^{R,1} \rangle + \langle \mathbf{r}^{L,2}, \mathbf{r}^{R,2} \rangle$. The output of the gadget encodes the value $\langle \mathbf{c}^L, \mathbf{c}^R \rangle = \sum_{i=1}^{t} c_i^L c_i^R$ which, by the definition of $\mathbf{c}^L, \mathbf{c}^R$, is equal to

  $$L_1' \left(-R_1' + \left(L_1'\right)^{-1}(c - d)\right) + \sum_{i=2}^{t} L_i' \cdot \left(-R_i'\right) = c - \sum_{i=1}^{t} L_i' R_i' - d = c - \sum_{i=1}^{t^2} L_i' R_i' - m = c - a \times b - m$$

  where the second-but-last equality follows from the analysis of the refresh gadget.

- **Multiplication and addition by constant gadgets.** Notice that these gadget do not use any masking inputs, and so the computation in these gadgets is always correct (corresponds to computation under the all-zeros attack).

■

4 Leakage-Secure Zero-Knowledge

In this section we describe our leakage-secure zero-knowledge circuits. At a high level, an $\mathcal{L}$-secure ZK circuit for a family $\mathcal{L}$ of functions is a randomized algorithm Gen that given an error parameter ϵ, and an input length n, outputs a randomized *prover input encoder* Enc_P, and a circuit T. T takes an input from a *prover* P, and returns output to a *verifier* V, and is used by P to convince V that $x \in L_{\mathcal{R}}$. T guarantees soundness, and zero-knowledge even when V obtains leakage from $\mathcal{L}$ on the internals of T.

Definition 4.1 ($\mathcal{L}$-secure ZK circuit)**.** *Let $\mathcal{R} = \mathcal{R}(x, w)$ be an NP-relation, $\mathcal{L}$ be a family of functions, and $\epsilon > 0$ be an error parameter. We say that* Gen *is an* $\mathcal{L}$-secure zero-knowledge (ZK) circuit *if the following holds.*

- ***Syntax.*** Gen *is a deterministic algorithm that has input $\epsilon, 1^n$, runs in time* $\mathsf{poly}(n, \log(1/\epsilon))$*, and outputs* (Enc_P, T) *defined as follows.* Enc_P *is a randomized circuit that on input (x, w) such that $|x| = n$ (x is the* input*, and w is the* witness*) outputs the* prover input *y for T; and T is a randomized circuit that takes input y and returns $z \in \{0,1\}^{n+1}$.*

- ***Correctness.*** *For every $\epsilon > 0$, every $n \in \mathbb{N}$, and every $(x, w) \in \mathcal{R}$ such that $|x| = n$,* $\Pr\left[T\left(\mathsf{Enc}_P(x, w)\right) = (x, 1)\right] \geq 1 - \epsilon$*, where* $(\mathsf{Enc}_P, T) \leftarrow \mathsf{Gen}(\epsilon, 1^n)$*, and the probability is over the randomness used by* Enc_P, T.

- ***Soundness.*** *For every (possibly malicious, possibly unbounded) prover P^*, every $\epsilon > 0$, every $n \in \mathbb{N}$, and any $x \notin L_{\mathcal{R}}$ such that $|x| = n$, $\Pr\left[T\left(P^*(x)\right) = (x, 1)\right] \leq \epsilon$, where $(\mathsf{Enc}_P, T) \leftarrow \mathsf{Gen}\left(\epsilon, 1^n\right)$, and the probability is over the randomness used by P^*, T.*

- ***$\mathcal{L}$-Zero-knowledge.*** *For $(x, w) \in \mathcal{R}$ we define the following experiments.*
  - *For a (possibly malicious, possibly unbounded) verifier V^*, define the experiment $\mathsf{Real}_{V^*,\mathsf{Gen}}(x, w, \epsilon)$ where V^* has input x, ϵ, and chooses a leakage function $\ell \in \mathcal{L}$, and $\mathsf{Real}_{V^*,\mathsf{Gen}}(x, w, \epsilon) = \left(T\left(\mathsf{Enc}_P(x, w)\right), \ell\left[T, \mathsf{Enc}_P(x, w)\right]\right)$, where $(\mathsf{Enc}_P, T) \leftarrow \mathsf{Gen}\left(\epsilon, 1^n\right)$, and $[T, y]$ denotes the wires of T when evaluated on y.*
  - *For a* simulator *algorithm* Sim *that has input x, ϵ, and* one-time *oracle access to ℓ, the experiment $\mathsf{Ideal}_{\mathsf{Sim},\mathcal{R}}(x, w, \epsilon)$ is defined as follows: $\mathsf{Ideal}_{\mathsf{Sim},\mathcal{R}}(x, w, \epsilon) = \mathsf{Sim}^{\ell}(\epsilon, x)$, where $\mathsf{Sim}^{\ell}(\epsilon, x)$ is the output of* Sim*, given one-time oracle access to ℓ.*

  We say that Gen *has $\mathcal{L}$-zero-knowledge ($\mathcal{L}$-ZK) if for every (possibly malicious, possibly unbounded) verifier V^* there exists a simulator* Sim *such that for every $\epsilon > 0$, every $n \in \mathbb{N}$, and every $(x, w) \in \mathcal{R}$ such that $|x| = n$, $\mathsf{SD}\left(\mathsf{Real}_{V^*,\mathsf{Gen}}(x, w, \epsilon), \mathsf{Ideal}_{\mathsf{Sim},\mathcal{R}}(x, w, \epsilon)\right) \leq \epsilon$.*

4.1 The Leakage-Secure ZK Circuit

We now construct the leakage-secure ZK circuit by combining the LRCC $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{E}_{\mathsf{Inp}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ of Theorem **??** with the LTCC $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ of Theorem **??**.

At a high level, we compile the verification circuit $C_{\mathcal{R}}$ of an NP-relation $\mathcal{R}$ using $\mathsf{Comp}^{\mathrm{GIMSS}}$, where the prover provides the encoded input and witness for the compiled circuit $\hat{C}_{\mathcal{R}}$. $\hat{C}_{\mathcal{R}}$ has encoded outputs, and only guarantees that BCL leakage cannot distinguish between the executions on two different witnesses. To achieve full-fledged ZK, we use $\mathsf{Comp}^{\mathrm{DF}}$ to decode the outputs of $\hat{C}_{\mathcal{R}}$. Recall that circuits compiled with $\mathsf{Comp}^{\mathrm{DF}}$ have masking inputs, and moreover, their leakage-tolerance property crucially relies on the fact that the masks are *unknown to the leakage function.* Therefore, these masking inputs must be provided by the prover as part of the input encoding (which is generated using Enc_P). However, since the correctness of the computation is guaranteed *only when the masking inputs are well-formed*, a malicious prover P^* can violate soundness by providing ill-formed masking inputs (which were *not* drawn according to the "right" distribution), and thus modify the computed functionality, and potentially cause the circuit to accept $x \notin L_{\mathcal{R}}$. As discussed in Section **??**, the effect of ill-formed masking inputs corresponds to applying an additive attack on the original decoding circuit, so we can protect against such attacks by first replacing the decoding circuit with an AMD circuit.

Construction 4.2 (Leakage-secure ZK circuit)**.** *Let $n \in \mathbb{N}$ be an input length parameter, $t \in \mathbb{N}$ be a security parameter, and $c \in \mathbb{N}$ be a constant. Let $\mathcal{R} = \mathcal{R}(x, w)$ be an NP-relation, with verification circuit $C_{\mathcal{R}}$ of size $s = |C_{\mathcal{R}}|$. The leakage-secure ZK circuit uses the following building blocks (where any field operations are implemented via bit operations).*

- *The LRCC $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{E}_{\mathsf{In}}^{\mathrm{GIMSS}} = \left(\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{In}}^{\mathrm{GIMSS}}\right), \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ of Theorem **??** (Construction **??**), and its underlying small-bias encoding scheme $\left(\mathsf{Enc}^{\oplus} : \mathbb{F}_2 \times \mathbb{F}_2^c \rightarrow \mathbb{F}_2^{2c}, \mathsf{Dec}^{\oplus} : \mathbb{F}_2^{2c} \rightarrow \mathbb{F}_2^2\right)$.*

- *The LTCC $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ of Theorem **??** (Construction **??**) over a field $\mathbb{F} = \Omega(t)$, and its underlying encoding scheme $\mathsf{E}_{\mathrm{DF}}^{\mathsf{In}} = \left(\mathsf{Enc}_{\mathrm{DF}}^{\mathsf{In}}, \mathsf{Dec}_{\mathrm{DF}}^{\mathsf{In}}\right)$.*

- *The additively-secure circuit compiler $\mathsf{Comp}^{\mathsf{add}}$ of Theorem **??**.*

- *The AMD encoding scheme $\left(\mathsf{Enc}^{\mathsf{amd}}, \mathsf{Dec}^{\mathsf{amd}}\right)$ of Theorem **??**, with encodings of length $\hat{n}^{\mathsf{amd}}(n, t)$.*

On input $1^n, 1^t$, Gen *outputs (Enc_P, T) defined as follows.*

- *For every input* $x \in \{0,1\}^n$*, and witness* w*,* $\mathsf{Enc}_P(x,w) = \left(\mathsf{Enc}_{\mathrm{GIMSS}}\left((x,w),1^t\right), \mathsf{Enc}_{\mathrm{DF}}^{\mathsf{In}}\left(0^{s'},1^t\right)\right)$ *for a parameter* s' *whose value is set below.*

- *Let* n_w *be a bound on the maximal witness length for inputs of length* n*.* T *is obtained by concatenating the* decoding component T'' *to the* verification component C'' *(namely, applying* T'' *to the outputs of* C''*) which are defined next.*

  1. ***The verification component*** C''***.*** *Define* $C' : \mathbb{F}_2^{n+n_w} \to \mathbb{F}_2^{n+1}$ *as* $C'(x,w) = (x, C_{\mathcal{R}}(x,w))$*. Let* C'_2 *denote the circuit that emulates* C'*, but replaces each output bit with (the bit string representation of) the bit as an element of* $\mathbb{F}$*. Then* $C'' = \mathsf{Comp}^{\mathrm{GIMSS}}(C'_2)$*.*
  2. ***The decoding component.***
     - *Construct the circuit* $C^{\mathsf{amd}} : \mathbb{F}^{2c\cdot t\cdot(n+1)} \to \mathbb{F}^{\hat{n}^{\mathsf{amd}}(n+1,t)}$ *that operates as follows:*
       * *Decodes its input using* $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$ *to obtain the output* (f,x,z)*.*
       * *If* $f=1$*,* $x \notin \{0,1\}^n$*, or* $z \neq 1$*, then* C^{amd} *sets* $z'=0$*. Otherwise, it sets* $z'=1$*.*
       * *Generates* $\boldsymbol{e} \leftarrow \mathsf{Enc}^{\mathsf{amd}}\left((x,z'),1^t\right)$*, and outputs* $\boldsymbol{e}$*.*
     - *Generate* $\widehat{C}^{\mathsf{amd}} = \mathsf{Comp}^{\mathsf{add}}\left(C^{\mathsf{amd}}\right)$*.*
     - *Generate* $T' = \mathsf{Comp}^{\mathrm{DF}}\left(\widehat{C}^{\mathsf{amd}}\right)$*. Let* s' *denote the number of masking inputs used in* T'*.*
     - *Construct the circuit* T'' *that on input* y*, operates as follows:*
       * *Computes* $(f_L, f_R, \boldsymbol{e}) = T'(y)$*. (Recall that* f_L, f_R *are flags indicating whether a gadget of* T' *has failed.)*
       * *Computes* $(f,x,z) = \mathsf{Dec}^{\mathsf{amd}}\left(\boldsymbol{e},1^t\right)$*, where* $f,z \in \mathbb{F}$ *and* $x \in \mathbb{F}^n$*. If* $f = f_L = f_R = 0$*,* $x \in \{0,1\}^n$*, and* $z=1$ *then* T' *outputs* $(x,1)$*. Otherwise, it outputs* 0^{n+1}*.*

4.2 Proof of Theorem ??

In this section we prove Theorem **??**. We first prove, in the following series of lemmas, the properties of the ZK circuit of Construction **??**

Lemma 4.3 (Correctness of Construction **??**)**.** *If* $|\mathbb{F}| = \Omega(t)$*, then Construction* **??** *is correct.*

Proof. Let $\mathcal{R} = \mathcal{R}(x,w)$ be an NP-relation with verification circuit $C_{\mathcal{R}}$, and let $(x,w) \in \mathcal{R}$. Let $\epsilon > 0$ be an error parameter. We show that when Construction **??** is instantiated with a large enough security parameter t, then except with probability ϵ, $T(\mathsf{Enc}_P(x,w))$ outputs $(x,1)$.

Since T is obtained as the concatenation of C'' and T'', we analyze each of these ingredients separately. C'' is obtained by applying the LRCC of [**?**] (Construction **??**) to C'. Since (as proven in [**?**]) Construction **??** has perfect correctness, when $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$ is applied to the output z of C'', it outputs $(x,1)$ (with probability 1).

As for $T' = \mathsf{Comp}^{\mathrm{DF}}\left(\mathsf{Comp}^{\mathsf{add}}\left(C^{\mathsf{amd}}\right)\right)$, by the perfect correctness of the AMD encoding scheme, and the additively-secure circuit compiler, both C^{amd} and $\widehat{C}^{\mathsf{amd}}$ have perfect completeness. Therefore, if T' perfectly emulates $\mathsf{Comp}^{\mathsf{add}}\left(C^{\mathsf{amd}}\right)$ then when T' is applied to the output of C'' its output is in $\mathsf{supp}\left(\mathsf{Enc}^{\mathsf{amd}}\left((x,1),1^n\right)\right)$, so T'' outputs $(x,1)$ as we set out to prove. Since T' perfectly emulates $\mathsf{Comp}^{\mathsf{add}}\left(C^{\mathsf{amd}}\right)$ unless one of its gadgets fails, it suffices to prove that except with probability ϵ, no gadget in T' fails (i.e., has incorrect output). Let $\delta(t)$ be the negligible function that bounds the failure probability of the gadgets in T' (such a function exists; see Remark **??**). Let $s = |C_{\mathcal{R}}|$, and let s' denote the number of gadgets in T'. Then there exists a constant $c \in \mathbb{N}$ such that $s' = c \cdot s$ (this follows from Constructions **??** and **??**, and since each field operation can be implemented using a constant number of gadgets from Construction **??**). Let $t_0 \in \mathbb{N}$ be such that for every $t \geq t_0$, $s' \cdot \delta(t) \leq \epsilon$. We instantiate Construction **??** with security parameter t_0, in which case the output is correct except with probability ϵ. ■

Lemma 4.4 (Soundness of Construction **??**)**.** *If* $|\mathbb{F}| = \Omega(t)$ *then Construction* **??** *is sound.*

Proof. Let $x \notin L_{\mathcal{R}}$, and let $\epsilon > 0$. Let $\hat{x}, \hat{w}$ denote the encodings which P^* provides as input to C'', and assume these are valid encodings of some x', w'. (This assumption is without loss of generality, since invalid encodings are interpreted by the gadget $\mathcal{G}$ in C'' as encoding the all-zeros string.) Since C'' does not use any structured randomness, conditioned on the event that the inputs are valid encodings, C'' perfectly emulates C_2', and outputs a valid encoding of its output. Therefore, the output $\hat{y}$ of C'' encodes $y = (x', C_{\mathcal{R}}(x', w'))$ (where each bit is replaced with the bit-string representation of the corresponding field element in $\mathbb{F}$).

Let m' denote the masking inputs which P^* provided for T''. Notice first that if the masking inputs cause some gadget of T' to fail, then either f_L or f_R (computed in Step (**??**) of Construction **??**) are set, in which case the output is $(x', 0)$. Therefore, we can condition on the event that no gadget failed.

Lemma **??** guarantees that there exists an additive attack $\mathbf{A}^{\mathsf{m}'}$ on $\widehat{C}^{\mathsf{amd}}$[10] such that $T'(\hat{y}, \mathsf{m}') = \widehat{C}^{\mathsf{amd}, \mathbf{A}^{\mathsf{m}'}}(\hat{y})$.

Moreover, the additive-security of $\widehat{C}^{\mathsf{amd}}$ guarantees that there exists an ideal additive attack $\mathbf{a}^{\mathsf{in}}$ on the inputs of C^{amd}, and a distribution $\mathcal{A}^{\mathsf{Out}}$ over the outputs of C^{amd}, such that

$$\mathsf{SD}\left(\widehat{C}^{\mathsf{amd}, \mathbf{A}^{\mathsf{m}'}}(\hat{y}), C^{\mathsf{amd}}\left(\hat{y} + \mathbf{a}^{\mathsf{in}}\right) + \mathcal{A}^{\mathsf{Out}}\right) \leq \epsilon_{\mathsf{amd}}(t) = \mathsf{negl}(t).$$

Consequently, $\mathsf{SD}\left(T(P^*(x)), T''\left(C^{\mathsf{amd}}\left(\hat{y} + \mathbf{a}^{\mathsf{in}}\right) + \mathcal{A}^{\mathsf{Out}}\right)\right) \leq \epsilon_{\mathsf{amd}}(t)$. We show that $\Pr\left[T''\left(C^{\mathsf{amd}}\left(\hat{y} + \mathbf{a}^{\mathsf{in}}\right) + \mathcal{A}^{\mathsf{Out}}\right) = (x, 1)\right] \leq \epsilon'(t)$, by considering several possible cases. Choosing t to be large enough such that $\epsilon_{\mathsf{amd}}(t) + \epsilon'(t) < \epsilon$ proves the lemma. Fix some $\mathbf{a}^{\mathsf{out}} \leftarrow \mathcal{A}^{\mathsf{Out}}$.

1. $\mathbf{a}^{\mathsf{out}} \neq \mathbf{0}$. Since T'' uses $\mathsf{Dec}^{\mathsf{amd}}$ to decode the outputs of C^{amd} (which are AMD encodings), then the additive soundness of the AMD encoding guarantees that the decoding of each output symbol fails except with probability $|\mathbb{F}|^{-t}$, so decoding the outputs of C^{amd} fails except with probability $(n+1)|F|^{-t}$ (which, since $|\mathbb{F}| = \Omega(t)$ is $\mathsf{negl}(t)$ for every $t \geq n$). Therefore, we can condition on the event that $\mathbf{a}^{\mathsf{out}} = 0$.

2. $\mathbf{a}^{\mathsf{in}} \neq \mathbf{0}$. Notice that $\hat{y}$, as the output of C'', is a valid encoding, and recall that it consists of $n+1$ collections $\hat{y}_1, \cdots, \hat{y}_{n+1}$, where each collection consists of t encodings $\hat{y}_{i,1}, \cdots, \hat{y}_{i,t}$ of the same value y_i. We assume that $\hat{y} + \mathbf{a}^{\mathsf{in}}$ *does not* encode y, since the case that it encodes y is covered by the case that $\mathbf{a}^{\mathsf{in}} = 0$. Since $\hat{y} + \mathbf{a}^{\mathsf{in}}$ does not encode y, there are two possible cases.

   - $\hat{y} + \mathbf{a}^{\mathsf{in}}$ is an invalid encoding. In this case, the decoding inside C^{amd} fails except with probability $(n+1)|\mathbb{F}|^{-t}$. Conditioned on the event that the decoding failed, C^{amd} outputs an AMD encoding of 0, in which case T'' outputs 0^{n+1}. (Indeed, since we have conditioned on the event that $\mathbf{a}^{\mathsf{out}} = 0$, the decoding in T'' returns $z = 0$.)
   - $\hat{y} + \mathbf{a}^{\mathsf{in}}$ is a valid encoding of $y' \neq y$, and assume $y_i' \neq y_i$. In particular, $\mathbf{a}^{\mathsf{in}}$ succeeded in flipping the encoded value in each of the encodings $\hat{y}_{i,1}, \cdots, \hat{y}_{i,t}$ (since during decoding, these values are compared). Since the encodings scheme $\mathsf{E}^{\oplus}$ has ϵ'-additive soundness, for each $1 \leq j \leq t$ this happens only with probability ϵ', and so the probability that this occurs for *all* $1 \leq j \leq t$ copies is at most $(\epsilon')^t = \mathsf{negl}(t)$.

3. $\mathbf{a}^{\mathsf{in}} = \mathbf{0}$. Recall that $y = (x', C_{\mathcal{R}}(x', w'))$ is the value encoded by $\hat{y}$. We consider two possible cases.

   - First, if $x' \neq x$ then since we have conditioned on the event that $\mathbf{a}^{\mathsf{out}} = 0$, then the AMD decoding in T succeeds, and returns x'. Therefore, the output in any case is not $(x, 1)$ (with probability 1).

[10] If m' consists of well-formed vectors then the corresponding additive attack on $\widehat{C}^{\mathsf{amd}}$ is the all-zero string.

- If $x' = x$ then $C_{\mathcal{R}}(x', w') = 0$. Since we have conditioned on the event that $\mathbf{a}^{\mathsf{out}} = 0$, the AMD decoding in T returns $z = 0$, so T outputs 0^n (with probability 1).

■

The following notation will be useful for the formulate the zero-knowledge property of Construction **??**.

Notation 4.5. *For a finite field $\mathbb{F}$, a parameter $k \in \mathbb{N}$, and a family $\mathcal{L}$ of leakage functions, denote by $\mathcal{L}_f^k$ the family of all functions ℓ_f^k such that $\ell_f^k(y_1, \cdots, y_N) = \ell(y_1, \cdots, y_N, f(y_{N-k+1}, \cdots, y_N))$. (That is, the inputs to ℓ are the inputs to ℓ_f^k, and the output of f on the last k inputs of ℓ_f^k.)*

Recall that $\mathcal{L}_{\mathrm{BCL}}^t$ denotes the family of all t-BCL functions (as defined in Definition **??**).

Lemma 4.6 (Zero-knowledge of Construction **??**)**.** *Let $t \in \mathbb{N}$ be a leakage bound, $\mathcal{R}$ be an NP-relation with verification circuit $C_{\mathcal{R}}$ of input length n and size $|C_{\mathcal{R}}| = s$, and $\epsilon > 0$. Let $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$ be the output-decoder of Construction **??**, and let $\widehat{C}^{\mathsf{amd}}$ denote the circuit constructed from $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$ in Step (**??**) of Construction **??**. Let s' denote the combined sizes of the circuits C_1', C_2' of Construction **??**, when they are constructed for the circuit $\widehat{C}^{\mathsf{amd}}$. If:*

- *Construction **??** is an $\left(\mathcal{L}_{\mathrm{BCL}}^t, \epsilon, s'\right)$-LTCC with a simulator $\mathsf{Sim}^{\mathrm{DF}}$ that outputs a view-translation circuit $\mathcal{T}$, and*
- *Construction **??** is an $\left(\mathcal{L}_{\mathrm{BCL}}^t, \epsilon, s\right)$-relaxed LRCC with abort.*

*Then Construction **??** is $\mathcal{L}$-leakage resilient, with statistical distance $2\epsilon + \mathsf{negl}(t)$.*

Before proving the lemma, we describe the high-level idea of the simulation. The simulator Sim *honestly* evaluates C'' on an *arbitrary* input, to obtain an encoding $\hat{y}$ of the output. Then, Sim uses the simulator $\mathsf{Sim}^{\mathrm{DF}}$ of Construction **??** (whose existence is guaranteed from the leakage-tolerance property of the compiler) to construct a view-translator circuit $(\mathcal{T}_1, \mathcal{T}_2)$ which, given the inputs and outputs of T'', generates a simulated wire assignment to its wires. Applying T'' to $\hat{y}$, and combining this with the wire values of C'', gives simulated wire values for T''.

Proof. The assumptions of the lemma guarantee that when the compilers of Constructions **??** and **??** are applied to the verification circuit $C_{\mathcal{R}}$, as they are in the construction of T, the resultant circuits C'', T'' are leakage-resilient, and leakage-tolerant, respectively. Let V^* be a malicious (possibly unbounded) verifier, then the simulator Sim, on input (ϵ, x) performs the following:

1. **Obtaining the leakage function.** Invokes V^* on input x, to obtain a leakage function ℓ.
2. **Generating the view-translator for T''.** Runs the simulator $\mathsf{Sim}^{\mathrm{DF}}$ to obtain view-translator circuits $(\mathcal{T}_1, \mathcal{T}_2)$.
3. **Simulating the wire values of C''.** Encodes the all-zeros string using Enc_P, and evaluates C'' on the encoding. Let $\mathcal{W}_{\mathcal{R}}$ denote the wires values of C'' in this evaluation. Let $\mathcal{W}_{\mathsf{Out}}$ denote the restriction of $\mathcal{W}_{\mathcal{R}}$ to the outputs of C''.
4. **Simulating the wire values of T''.** Generates a fresh encoding $y \leftarrow \mathsf{Enc}^{\mathsf{amd}}\left((x, 1), 1^t\right)$ and uses $(\mathcal{T}_1, \mathcal{T}_2)$, with inputs $(\mathcal{W}_{\mathsf{Out}}, y)$ to obtain simulate wire values $\widehat{\mathcal{W}}_{\mathsf{Dec}}$ of T'. Constructs the wire values $\mathcal{W}_{\mathsf{Dec}}$ by honestly computing $\mathsf{Dec}^{\mathsf{amd}}\left(y, 1^t\right)$ and concatenating these wires values to $\widehat{\mathcal{W}}_{\mathsf{Dec}}$.
5. Outputs $((x, 1), \ell(\mathcal{W}_{\mathcal{R}}, \mathcal{W}_{\mathsf{Dec}}))$.

We now claim that for every $\epsilon > 0$, every $n \in \mathbb{N}$, every $(x, w) \in \mathcal{R}$ such that $|x| = n$, and every $\ell \in \mathcal{L}$ it holds that

$$\mathsf{SD}\left(\left(T\left(\mathsf{Enc}_P\left(x, w\right)\right), \ell\left[T, \mathsf{Enc}_P\left(x, w\right)\right]\right), \mathsf{Sim}\left(x, \epsilon\right)\right) \leq 2\epsilon. \tag{1}$$

Notice that the output of T' is $\mathsf{negl}\left(t\right)$-statistically close in both worlds, since $x \in L_{\mathcal{R}}$ and so by correctness, the output of T is $(x, 1)$ except with probability $\mathsf{negl}\left(t\right)$, so we can condition on the event that $(x, 1)$ it the output in both worlds (this can only increase the statistical distance by an additive $\mathsf{negl}\left(t\right)$ factor), in which case y is identically distributed to the output of T', so we can condition both distributions on the event that y was the output of T' in both worlds.

We show that when Construction **??** is applied with a large enough t, both distributions in Eq. (**??**) are ϵ- statistically-close to the hybrid distribution $\mathcal{H}$ defined as follows:

- Encode $(\hat{x}, \hat{w}) \leftarrow \mathsf{Enc}_P\left(x, w\right)$, and evaluates C'' on $(\hat{x}, \hat{w})$. Let $\mathcal{W}'_{\mathcal{R}}$ denote the wire values of C'' during this evaluation, and $\mathcal{W}'_{\mathsf{Out}}$ denote its restriction to the outputs of C''.
- Compute a fresh encoding $y \leftarrow \mathsf{Enc}^{\mathsf{amd}}\left(\left(x, 1\right), 1^t\right)$, and apply $(\mathcal{T}_1, \mathcal{T}_2)$ to $(\mathcal{W}'_{\mathsf{Out}}, y)$ to obtain simulated wire values $\widehat{\mathcal{W}'_{\mathsf{Dec}}}$ of T', and generate the wire values $\mathcal{W}'_{\mathsf{Dec}}$ by honestly computing $\mathsf{Dec}^{\mathsf{amd}}\left(y, 1^t\right)$ and concatenating these wires values to $\widehat{\mathcal{W}'_{\mathsf{Dec}}}$.
- $\mathcal{H} = \left(y, \ell\left(\mathcal{W}'_{\mathcal{R}}, \mathcal{W}'_{\mathsf{Dec}}\right)\right)$.

$\mathsf{SD}\left(\left(y, \ell\left[T, \left(\mathsf{Enc}_P\left(x, w\right)\right)\right]\right), \mathcal{H}\right) \leq \epsilon$, because the only difference between both distributions is the wire values of T' (in particular, we can condition both distributions on some possible value for the wires of C''), and so the leakage-tolerance of Construction **??** guarantees that the statistical distance is at most ϵ.

Second, we claim that $\mathsf{SD}\left(\mathcal{H}, \mathsf{Sim}\left(x, \epsilon\right)\right) \leq \epsilon$. Since we have conditioned on the event that the output of T' is y, it suffices to show that $\mathsf{SD}\left(\ell\left(\mathcal{W}'_{\mathcal{R}}, \mathcal{W}'_{\mathsf{Dec}}\right), \ell\left(\mathcal{W}_{\mathcal{R}}, \mathcal{W}_{\mathsf{Dec}}\right)\right) \leq \epsilon$. Notice that $\ell\left(\mathcal{W}'_{\mathcal{R}}, \mathcal{W}'_{\mathsf{Dec}}\right) = \ell\left(\mathcal{W}'_{\mathcal{R}}, \left(\mathcal{T}_1, \mathcal{T}_2\right)\left(\mathcal{W}'_{\mathsf{Out}}, y\right)\right)$, and $\ell\left(\mathcal{W}_{\mathcal{R}}, \mathcal{W}_{\mathsf{Dec}}\right) = \ell\left(\mathcal{W}_{\mathcal{R}}, \left(\mathcal{T}_1, \mathcal{T}_2\right)\left(\mathcal{W}_{\mathsf{Out}}, y\right)\right)$. Since we have conditioned on the value of y, it can be fixed into $\mathcal{T}_1, \mathcal{T}_2$, and let ℓ' be the function that on input $\widetilde{\mathcal{W}_{\mathcal{R}}}$ first computes $\widetilde{\mathcal{W}'} = \left(\mathcal{T}_1, \mathcal{T}_2\right)\left(\widetilde{\mathcal{W}_{\mathsf{Dec}}}\right)$, where $\widetilde{\mathcal{W}_{\mathsf{Dec}}}$ are the last $4tc\left(n+1\right)$ wires in $\widetilde{\mathcal{W}_{\mathcal{R}}}$ (i.e., the wires that correspond to the output of C''), and then computes $\ell\left(\widetilde{\mathcal{W}_{\mathcal{R}}}, \widetilde{\mathcal{W}'}\right)$. Then $\ell' \in \mathcal{L}^t_{\mathrm{BCL}}$, $\ell'\left(\mathcal{W}'_{\mathcal{R}}\right) = \ell\left(\mathcal{W}'_{\mathcal{R}}, \left(\mathcal{T}_1, \mathcal{T}_2\right)\left(\mathcal{W}'_{\mathsf{Out}}, y\right)\right)$, and $\ell'\left(\mathcal{W}_{\mathcal{R}}\right) = \ell\left(\mathcal{W}_{\mathcal{R}}, \left(\mathcal{T}_1, \mathcal{T}_2\right)\left(\mathcal{W}_{\mathsf{Out}}, y\right)\right)$. Therefore, by the relaxed leakage-resilience property of Construction **??**, the statistical distance is at most ϵ. ■

We are now ready to prove Theorem **??**.

Proof (of Theorem **??**). We show that Construction **??** has the required properties. Let $\mathsf{Dec}^{\mathrm{GIMSS}}_{\mathsf{Out}}$ be the output-decoder of Construction **??**, and let $\widehat{C}^{\mathsf{amd}}$ denote the circuit constructed from $\mathsf{Dec}^{\mathrm{GIMSS}}_{\mathsf{Out}}$ in Step (**??**) of Construction **??**. Let s' denote the combined sizes of the circuits C_1, C_2 of Construction **??**, when they are constructed for the circuit $\widehat{C}^{\mathsf{amd}}$. Since $\left|C^{\mathsf{amd}}\right| = \mathsf{poly}\left(|x|\right)$, and the blowup in C_1, C_2 is polynomial in $|x|$, there exists a polynomial $p\left(n\right)$ such that for all n, $p\left(n\right) \geq s'\left(n\right)$.

For a parameter t, let $f\left(t\right)$ denote the minimal size of $\mathbb{F}$ for which Lemmas **??**, **??**, and **??**, and Theorem **??**, hold, and notice that $f\left(t\right) = O\left(t\right)$. Let $t' \in \mathbb{N}$ be such that for every $t'' \geq t'$, $t'' \leq 0.16t'' \log_2 f\left(t''\right) - 1 - \log_2 f\left(t''\right)$. We instantiate Construction **??** with security parameter $\hat{t} = \max\left\{t', t, \sigma, n\right\}$, where $n = |x|$ is the input length. Then correctness and soundness follow directly from Lemma **??** and Lemma **??**, respectively. As for zero-knowledge, Theorem **??** guarantees that there exists a negligible function $\epsilon^{\mathrm{DF}}\left(\hat{t}\right) = \mathsf{negl}\left(\hat{t}\right)$, such that Construction **??** is an $\left(\mathcal{L}^t_{\mathrm{BCL}}, \epsilon^{\mathrm{DF}}\left(\hat{t}\right), p\left(\hat{t}\right)\right)$-LTCC with simulator $\mathsf{Sim}^{\mathrm{DF}}$that outputs view-translation circuits $(\mathcal{T}_1, \mathcal{T}_2)$. Moreover, Theorem **??** guarantees that Construction **??** is an $\left(\mathcal{L}^t_{\mathrm{BCL}}, 2^{-\hat{t}}, s\right)$-relaxed LRCC with abort. Therefore, by Lemma **??**, Construction **??** is $\mathcal{L}$-leakage-resilient with statistical distance $2\epsilon' + \mathsf{negl}\left(\hat{t}\right)$, where $\epsilon' = \max\left\{\epsilon^{\mathrm{DF}}\left(\hat{t}\right), 2^{-\hat{t}}\right\} = \mathsf{negl}\left(\sigma\right)$.

Regarding the complexity of the construction, given a size-s, depth-d circuit $C_{\mathcal{R}}$ with length-n inputs, $|C_2'| = s+n$, and consequently $|C''| = \widetilde{O}\left(s+n+d\hat{t}+\hat{t}^2\right)$ (by Theorem **??**). Moreover, since $\left|\mathsf{Dec}_{\mathsf{out}}^{\mathrm{GIMSS}}\right| = \widetilde{O}\left(\hat{t}^2+\hat{t}n\right)$, and has depth $O\left(\log\hat{t}\right)$ (by the definition of $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$, see Construction **??**) and $\left|\mathsf{Enc}^{\mathsf{amd}}\right| = O\left(n+\hat{t}\right)$ (by Theorem **??**, when instantiated with security parameter $\hat{t}$), then $\left|C^{\mathsf{amd}}\right| = \widetilde{O}\left(\hat{t}^2+\hat{t}n\right)$, and so $\left|\widehat{C}^{\mathsf{amd}}\right| = \left(\hat{t}^2+\hat{t}n\right)\cdot\mathsf{poly}\log\left(\hat{t}n\right) + \mathsf{poly}\left(n,\log\hat{t},\hat{t}\right) = \mathsf{poly}\left(\hat{t},n\right)$ (by Theorem **??**, when instantiated with error parameter $2^{-\hat{t}}$). Since the LTCC of Theorem **??** has polynomial blowup, $|T'| = \mathsf{poly}\left(\hat{t},n\right) = \mathsf{poly}\left(t,\sigma,n\right)$, and consequently $|T''| = \mathsf{poly}\left(t,\sigma,n\right)$. Therefore, the overall size of the leakage-secure ZK circuit is $\widetilde{O}\left(s+d\cdot\max\{t,\sigma,n\}\right) + \mathsf{poly}\left(t,\sigma,n\right)$. Finally, the prover only: (1) encodes its input and witness using $\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}$, which (by Theorem **??**) takes time $\widetilde{O}\left(n+|w|+\hat{t}\right) = \widetilde{O}\left(n+|w|+t+\sigma\right)$; and (2) generate the masking inputs for T' (there are $O\left(|T'|\right)$ such inputs, and each masking input can be generated in polynomial time (in its length)), which takes time $\mathsf{poly}\left(t,\sigma,n\right)$. Therefore, the prover runs in time $\mathsf{poly}\left(t,\sigma,n,|w|\right)$. ■

The proof of Theorem **??** is similar to that of Theorem **??**. Here we only sketch the main differences.
Proof (Of Theorem **??** (sketch))**.** The leakage-secure ZK circuit is obtained by modifying Construction **??** to take the randomness needed for C'' from the prover, and the randomness needed for $\widehat{C}^{\mathsf{amd}}$ from the verifier. (Notice the randomness used by $\widehat{C}^{\mathsf{amd}}$ includes the randomness used to generate the AMD encoding $\mathbf{e}$ in C^{amd}, as well as the randomness used by the additively-secure implementation.) First, notice that letting the prover chose the randomness of C'' does not violate soundness, notice that Construction **??** has perfect correctness, and so the output of C'' will be exactly the output of C_2', *regardless of the randomness chosen by the prover.* Second, to see why letting the verifier choose the randomness for $\widehat{C}^{\mathsf{amd}}$ preserves the zero-knowledge property, notice that we can condition both the real and ideal worlds on the randomness selected by the verifier for the execution. Hard-wiring this randomness into $\widehat{C}^{\mathsf{amd}}$ gives (the same) deterministic circuit in both worlds, and so we can repeat the proof of Lemma **??** for this new circuit (since now the randomness provided by the verifier constitutes part of the description of the circuit $\widehat{C}^{\mathsf{amd}}$, which is in any case public in the context of leakage-tolerance and leakage-resilience).

Regarding the prover and verifier runtime, since C'' uses $O\left(|C''|\right)$ random bits, the prover runtime is $\mathsf{poly}\left(t,\sigma,n,|w|\right)+\widetilde{O}\left(s+d\cdot\max\{t,\sigma,n\}+\left(\max\{t,\sigma,n\}\right)^2\right) = \mathsf{poly}\left(t,\sigma,n,|w|\right)+\widetilde{O}\left(s+d\cdot\max\{t,\sigma,n\}\right)$. Regarding the verifier runtime, since $\left|\widehat{C}^{\mathsf{amd}}\right| = \mathsf{poly}\left(\hat{t},n\right) = \mathsf{poly}\left(t,\sigma,n\right)$, and it uses a polynomial amount of randomness, the verifier runs in time $\mathsf{poly}\left(t,\sigma,n\right)$. ■

5 Multiparty LRCCs: Definition

In this section we define the notion of multiparty LRCCs, a generalization of leakage-secure ZK circuits to evaluation of general functions with $m \geq 1$ parties. We first formalize the notion of secure computation with a single piece of trusted (but leaky) hardware device, where security with abort holds in the presence of adversaries that corrupt a subset of parties, and obtain leakage (from a pre-defined leakage class) on the internals of the device. This raises the following points.

1. The output should include a flag signaling whether there was an abort.

2. Leakage on the wires of the device should reveal nothing about the internal computations, or the inputs of the honest parties, other than what can be computed from the output. This necessitates randomized computation.

3. The inputs should be encoded, otherwise leakage on the input wires may reveal information that cannot be computed from the outputs. This should be contrasted with the ZK setting, in which x is assumed to be public, and so when all parties are honest the output is $(x, 1)$ and can therefore be computed in the clear.

To guarantee that an adversary that only obtains leakage on the internals of the device (but does not corrupt any parties) learns nothing about the inputs or internal computations, the outputs must be encoded. Therefore, the device, which is implemented as a circuit, is associated with an input encoding algorithm Enc, and an output decoding algorithm Dec. The above discussion is formalized in the next definition.

Definition 5.1 (Secure function implementation)**.** *Let $m \in \mathbb{N}$, $f : (\{0,1\}^n)^m \to \{0,1\}^k$ be an m-argument function, $\mathcal{L}$ be a family of leakage functions, and $\epsilon > 0$. We say that $(\mathsf{Enc}, C, \mathsf{Dec})$ is an m-party $(\mathcal{L}, \epsilon)$-secure implementation of f if it satisfies the following requirements.*

- ***Syntax:***
  - $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^{\hat{n}}$ *is a randomized function, called* the input encoder*.*
  - $C : \left(\{0,1\}^{\hat{n}}\right)^m \to \{0,1\}^{\hat{k}}$ *is a randomized circuit.*
  - $\mathsf{Dec} : \{0,1\}^{\hat{k}} \to \{0,1\}^{k+1}$ *is a deterministic function called* the output decoder*.*
- ***Correctness.*** *For every $x_1, \cdots, x_m \in \{0,1\}^n$,*

$$\Pr\left[\mathsf{Dec}\left(C\left(\mathsf{Enc}\left(x_1\right), \cdots, \mathsf{Enc}\left(x_m\right)\right)\right) = \left(0, f\left(x_1, \cdots, x_m\right)\right)\right] \geq 1 - \epsilon.$$

- ***Security.*** *For every adversary $\mathcal{A}$ there exists a simulator Sim such that for every input $(x_1, \cdots, x_m) \in (\{0,1\}^n)^m$, and every leakage function $\ell \in \mathcal{L}$, $\mathsf{SD}\left(\mathsf{Real}, \mathsf{Ideal}\right) \leq \epsilon$, where $\mathsf{Real}, \mathsf{Ideal}$ are defined as follows.*

  Real*:*

  - *$\mathcal{A}$ picks a set $\mathsf{B} \subset [m]$ of corrupted parties, and (possibly ill-formed) encoded inputs $x'_i \in \{0,1\}^{\hat{n}}$ for every $i \in \mathsf{B}$.*
  - *For every uncorrupted party $j \notin \mathsf{B}$, let $x'_j = \mathsf{Enc}\left(x_j\right)$.*
  - *If $\mathsf{B} \neq \emptyset$ then $z = \left(C\left(x'_1, \cdots, x'_m\right), \mathsf{Dec}\left(C\left(x'_1, \cdots, x'_m\right)\right)\right)$, otherwise z is empty. (Intuitively, z represents the information $\mathcal{A}$ has about the output of C. If $\mathsf{B} = \emptyset$ then $\mathcal{A}$ learns nothing.)*
  - $\mathsf{Real} = \left(\mathsf{B}, \{x'_i\}_{i \in \mathsf{B}}, \ell\left[C, \left(x'_1, \cdots, x'_m\right)\right], z\right)$.

  Ideal*:*

  - *Sim picks a set $\mathsf{B} \subset [m]$ of corrupted parties and receives their inputs $\{x_i\}_{i \in \mathsf{B}}$. Sim then chooses effective inputs $w_i \in \{0,1\}^n$ for every $i \in \mathsf{B}$, and if $\mathsf{B} \neq \emptyset$ obtains $f\left(w_1 \cdots, w_m\right)$, where $w_j = x_j$ for every $j \notin \mathsf{B}$.*
  - *Sim chooses $b \in \{0,1\}$. (Intuitively, b indicates whether to abort the computation.)*
  - *If $\mathsf{B} \neq \emptyset$ and $b = 0$, set $y = \left(0, f\left(w_1, \cdots, w_m\right)\right)$, if $\mathsf{B} \neq \emptyset$ and $b = 1$, set $y = \left(1, 0^k\right)$, and if $\mathsf{B} = \emptyset$ then y is empty.*
  - *Let $\left(W, \{x'_i\}_{i \in \mathsf{B}}\right)$ denote the output of Sim, where W contains a bit for each wire of C, and $x'_i \in \{0,1\}^{\hat{n}}$ for every $i \in \mathsf{B}$. Denote the restriction of W to the output wires by W_{Out}.*
  - *If $\mathsf{B} \neq \emptyset$, let $z = \left(W_{\mathsf{Out}}, y\right)$. Otherwise, z is empty.*
  - $\mathsf{Ideal} = \left(\mathsf{B}, \{x'_i\}_{i \in \mathsf{B}}, \ell\left(W\right), z\right)$.

We say that $(\mathsf{Enc}, C, \mathsf{Dec})$ is a passive-secure implementation *of f if the security property holds with the following modifications: (1) $\mathcal{A}$ does not choose $x'_i, i \in \mathsf{B}$, and instead, $x'_i \leftarrow \mathsf{Enc}\left(x_i\right)$ for every $i \in \mathsf{B}$; and (2) Sim always chooses $b = 0$.*

We now define an m-party LRCC which, informally, is an asymptotic version of Definitions **??**.

Definition 5.2 (m-party circuit)**.** *Let $m \in \mathbb{N}$. We say that a boolean circuit C is an* m-party circuit *if its input can be partitioned into m equal-length strings, i.e., $C : (\{0,1\}^n)^m \to \{0,1\}^k$ for some $n, k \in \mathbb{N}$.*

Definition 5.3 (Multiparty LRCCs and passive-secure multiparty LRCCs)**.** *Let $m \in \mathbb{N}$, $\mathcal{L}$ be a family of leakage functions, $\mathsf{S}(n)$ be a size function, and $\epsilon(n) : \mathbb{N} \to \mathbb{R}^+$. Let Comp be a PPT algorithm that on input m, and an m-party circuit $C : (\{0,1\}^n)^m \to \{0,1\}^k$, outputs a circuit $\hat{C}$.*

We say that $(\mathsf{Enc}, \mathsf{Comp}, \mathsf{Dec})$ *is an m-party* $(\mathcal{L}, \epsilon(n), \mathsf{S}(n))$-leakage-resilient circuit compiler (m-party LRCC, or multiparty LRCC) *if there exists a PPT simulator* Sim *such that for all sufficiently large n's, and every m-party circuit $C : (\{0,1\}^n)^m \to \{0,1\}^k$ of size at most $\mathsf{S}(n)$ that computes a function f_C, $\left(\mathsf{Enc}, \hat{C}, \mathsf{Dec}\right)$ is an $(\mathcal{L}, \epsilon(n))$-secure implementation of f_C, where the security property holds with simulator* Sim *that is given the description of C, and has black-box access to the adversary. We say that* $(\mathsf{Enc}, \mathsf{Comp}, \mathsf{Dec})$ *is a* passively-secure m-party $(\mathcal{L}, \epsilon(n), \mathsf{S}(n))$-LRCC *if $\left(\mathsf{Enc}, \hat{C}, \mathsf{Dec}\right)$ is an $(\mathcal{L}, \epsilon(n))$-passively-secure implementation of f_C, where security holds with simulator* Sim.

Remark 5.4. *Definitions **??**- **??** naturally extend to the arithmetic setting in which C is an arithmetic circuit over a finite field $\mathbb{F}$. When discussing the arithmetic setting, we explicitly state the field over which we are working (e.g., we use "multiparty LRCC over $\mathbb{F}$" to denote that the multiparty LRCC is in the arithmetic setting with field $\mathbb{F}$).*

6 A Passive-Secure Multiparty LRCC

In this section we construct a passive-secure multiparty LRCC (this construction is somewhat more efficient than the (fully-secure) multiparty LRCC which will be described in Section **??**, and its analysis will be a warm-up for the analysis of the multiparty LRCC). As described in Section **??**, the high-level idea of the leakage-resilient circuit $\hat{C}$ for a given circuit C is as follows. First, we use the LRCC of [**?**] to generate a leakage-resilient version of the circuit C^{share} that emulates C but outputs a secret-sharing of the outputs. Then, we refresh each secret-share using a circuit $\widehat{C}_{\mathsf{Dec}}$, generated from a refreshing circuit C_{Dec} using the LTCC of [**?**]. More specifically, we use multiple copies of $\widehat{C}_{\mathsf{Dec}}$, where the i'th copy refreshes the i'th secret share, and takes its masking inputs from the i'th party. (We note that there is no need to protect against invalid input encodings using AMD circuits, since in the passive setting all input encodings are valid.) This intuition is formalized in the following construction.

Construction 6.1 (Passive-secure multiparty LRCC)**.** *Let $m \in \mathbb{N}$ denote the number of parties, $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, and $k \in \mathbb{N}$ be an output length parameter. The m-party passive-secure LRCC uses the following building blocks:*

- *The LRCC $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{E}_{\mathsf{In}}^{\mathrm{GIMSS}} = \left(\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{In}}^{\mathrm{GIMSS}}\right), \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ of Theorem **??** (Construction **??**), and its underlying small-bias encoding scheme $\left(\mathsf{Enc}^{\oplus} : \mathbb{F}_2 \times \mathbb{F}_2^c \to \mathbb{F}_2^{2c}, \mathsf{Dec}^{\oplus} : \mathbb{F}_2^{2c} \to \mathbb{F}_2\right)$. Let $\hat{n}_{\mathsf{In}}(n,t)$ ($\hat{n}(n,t)$) denote the length of encodings which $\mathsf{Enc}_{\mathrm{GIMSS}}^{\mathsf{In}}$ ($\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$) outputs (takes as input).*
- *The LTCC $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ of Theorem **??** (Construction **??**) over a field $\mathbb{F} = \Omega(t)$, and its underlying encoding scheme $\mathsf{E}_{\mathrm{DF}}^{\mathsf{In}} = \left(\mathsf{Enc}_{\mathrm{DF}}^{\mathsf{In}}, \mathsf{Dec}_{\mathrm{DF}}^{\mathsf{In}}\right)$ that outputs encodings of length $\hat{n}^{\mathrm{DF}}(n,t)$.*

The m-party passive-secure LRCC $(\mathsf{Enc}, \mathsf{Comp}, \mathsf{Dec})$ *is defined as follows.*

- *For every $n, t, t_{\mathsf{In}} \in \mathbb{N}$ and every $x \in \mathbb{F}^n$, $\mathsf{Enc}\left(x, 1^t, 1^{t_{\mathsf{In}}}\right) = \left(\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}\left(x, 1^t, 1^{t_{\mathsf{In}}}\right), \mathsf{Enc}_{\mathsf{In}}^{\mathrm{DF}}\left(0^{t_{\mathsf{In}}}, 1^t\right)\right)$.*
- *For every $y = \left(y^1, \cdots, y^m\right) \in \left(F^{\hat{n}_{\mathsf{In}}(k+1,t)}\right)^m$, $\mathsf{Dec}\left(y, 1^t\right)$ computes $\left(f_i, z^i\right) = \mathsf{Dec}_{\mathrm{GIMSS}}^{\mathsf{Out}}\left(y^i, 1^t\right)$, and outputs $\left(0, \sum_{i=1}^m z^i\right)$.*

- Comp *on input $m \in \mathbb{N}$, and an m-party circuit $C : (\mathbb{F}^n)^m \to \mathbb{F}^k$:*
  1. *Constructs the circuit $C^{\mathsf{share}} : (\mathbb{F}^n)^m \to \mathbb{F}^{mk}$ that operates as follows:*
     - *Evaluates C on inputs $x_1, \cdots, x_m$ to obtain the output $y = C(x_1, \cdots, x_m)$.*
     - *Generates $y_1, \cdots, y_{m-1} \in_R \mathbb{F}^k$, and sets $y_m = y \oplus \sum_{i=1}^{m-1} y_i$. ($y_1, \cdots, y_m$ are random additive secret shares of y.)*
     - *For every $1 \leq i \leq m$, generates y'_i by replacing each bit of y_i with (the bit string representation of) the bit as an element of $\mathbb{F}$.*
     - *Outputs $(y'_1, \cdots, y'_m)$.*
  2. *Computes $C' = \mathsf{Comp}^{\mathrm{GIMSS}}\left(C^{\mathsf{share}}\right)$.*
  3. *Constructs the circuit $C^{\mathsf{Dec}} : \mathbb{F}^{\hat{n}(k,t)} \to \mathbb{F}^{\hat{n}_{\mathsf{In}}(k+1,t)}$ that operates as follows:*
     - *Decodes its input using $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$ to obtain (f, z).*
     - *Generates a random encoding $\mathsf{Enc}_{\mathrm{GIMSS}}^{\mathsf{In}}\left((f,z), 1^t\right)$, and outputs it.*
  4. *Generate $C'' = \mathsf{Comp}^{\mathrm{DF}}\left(C^{\mathsf{Dec}}\right)$.*
  5. *Outputs the circuit $\hat{C}$ obtained by concatenating a copy of C'' to each of the m outputs of C'. (We note that the i'th copy of C'' takes its masking inputs from the encoding of the i'th input to $\hat{C}$.)*

The next theorem states that Construction **??** is a passive-secure multiparty LRCC.

Theorem 6.2. *Let $n, k \in \mathbb{N}$ be input and output length parameters, $\boldsymbol{S}(n) : \mathbb{N} \to \mathbb{N}$ be a size function, $\epsilon(n) : \mathbb{N} \to (0,1)$ be an error function, $t \in \mathbb{N}$ be a leakage bound, and $\mathsf{m} \in \mathbb{N}$ denote the number of parties. Let $\mathcal{L}$ denote the family of all t-BCL functions. If:*

- $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ *is an $(\mathcal{L}, \epsilon, \boldsymbol{S}(n) + 2m)$-relaxed LRCC with abort, where $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}$ can be evaluated using circuits of size s^{GIMSS}, and*
- $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ *is an $\left(\mathcal{L}, \epsilon, 2s^{\mathrm{GIMSS}}\right)$-LTCC.*

*Then Construction **??** is a passively-secure m-party $(\mathcal{L}, (2m+1)\epsilon, \boldsymbol{S}(n))$-LRCC.*

Proof. First, it follows directly from the construction that the compiler has the required syntax. Second, notice that for every m-party circuit C, the circuit C^{share} obtained in Step (**??**) of Construction **??** satisfies $\left|C^{\mathsf{share}}\right| \leq |C| + 2m$, and the circuit C^{Dec} obtained in Step (**??**) of Construction **??** satisfies $\left|C^{\mathsf{Dec}}\right| \leq 2s^{\mathrm{GIMSS}}$. Therefore, if $|C| \leq \boldsymbol{S}(n)$, then the leakage-resilience of $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ guarantees that the circuit C' obtained in Step (**??**) is $(\mathcal{L}, \epsilon)$-leakage-resilient, and the leakage-tolerance of $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ guarantees that the circuit C'' obtained in Step (**??**) is $(\mathcal{L}, \epsilon)$-leakage-tolerant. (This will be needed to argue security.)

Correctness. The perfect correctness of $\mathsf{Comp}^{\mathrm{GIMSS}}$ guarantees that C' perfectly emulates C^{share}, i.e., C' outputs an encoding (which can be decoded using $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$) of the output of C^{share}, which in turn is (by the definition of C^{share}) an additive secret sharing of the output of C. Moreover, the $1-\epsilon$ correctness of $\mathsf{Comp}^{\mathrm{DF}}$ guarantees that except with probability ϵ, each copy of C'' perfectly emulates C^{Dec} when its inputs are well formed. Using the union bound, except with probability $m\epsilon$ *all* copies of C'' perfectly emulate C^{Dec}. Therefore, conditioned on this event the output of $\widehat{C}$ on well-formed input encodings of $(x_1, \cdots, x_m)$ is an additive secret sharing of $C(x_1, \cdots, x_m)$. Consequently, the output is $C(x_1, \cdots, x_m)$ except with probability at most $m\epsilon$.

Security. We describe a simulator Sim that simulates the wire values of $\widehat{C}$. Sim uses the adversary $\mathcal{A}$ as a black-box to determine the set $\mathsf{B} = \{i_1, \cdots, i_r\}$ of corrupted parties (Sim chooses to corrupt the same set of parties).

We first consider the case that $\mathsf{B} = \emptyset$. In this case, $\mathsf{Real} = \left(\mathsf{B}, \ell\left[\widehat{C}, (\widehat{x_1}, \cdots, \widehat{x_m})\right]\right)$, where for every $1 \leq i \leq m$, $\widehat{x_i} \leftarrow \mathsf{Enc}\left(x_i, 1^t, 1^{t_{\mathsf{In}}}\right)$. The simulator operates as follows.

1. **Obtaining the leakage function.** Invokes $\mathcal{A}$ on input 1^n to obtain a leakage function ℓ.

2. **Simulating the wire values of C'.** Uses Enc to generate m encodings of 0^n, and evaluates C' on these encodings. Let $\widetilde{\mathcal{W}}$ denote the wires values of C' in this evaluation. Let $\widetilde{\mathcal{W}}^1_{\mathsf{Out}},\cdots,\widetilde{\mathcal{W}}^m_{\mathsf{Out}}$ denote the restriction of $\widetilde{\mathcal{W}}$ to the m outputs of C'. (Intuitively, the simulator emulates the evaluation of C' when the inputs of all parties are the all-0 strings.)

3. **Simulating the wire values of C''.**
   - Runs the simulator $\mathsf{Sim}^{\mathrm{DF}}$ to obtain view-translator circuits $(\mathcal{T}_1,\mathcal{T}_2)$.
   - Generates m random encodings $y^1,\cdots,y^m \leftarrow \mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}\left(0^{k+1},1^t,1^{t_{In}}\right)$. (Intuitively, $y^1,\cdots,y^m$ simulate the output encodings of C''.)
   - For every $1\leq i\leq m$, Sim uses $(\mathcal{T}_1,\mathcal{T}_2)$, with inputs $\left(\widetilde{\mathcal{W}}^i_{\mathsf{Out}},y^i\right)$ to obtain simulated wire values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ for the i'th copy of C''.
   - Sets $\widehat{\mathcal{W}}_{\mathsf{Dec}}=\left(\widetilde{\mathcal{W}}^1_{\mathsf{Dec}},\cdots,\widetilde{\mathcal{W}}^m_{\mathsf{Dec}}\right)$.

4. Outputs $\left(\widetilde{\mathcal{W}},\widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)$.

Let $\mathsf{Ideal}=\left(\mathsf{B},\ell\left(\widetilde{\mathcal{W}},\widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)\right)$. We now claim that for every $\epsilon>0$, every $n\in\mathbb{N}$, every $(x_1,\cdots,x_m)\in(\{0,1\}^n)^m$, and every $\ell\in\mathcal{L}$ it holds that $\mathsf{SD}(\mathsf{Real},\mathsf{Ideal})\leq(2m+1)\,\epsilon$. Since B is identically distributed in both distributions, it suffices to bound the statistical distance conditioned on B. Let $\mathsf{Real}'=\ell\left[\widehat{C},(\widehat{x_1},\cdots,\widehat{x_m})\right],\mathsf{Ideal}'=\ell\left(\widetilde{\mathcal{W}},\widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)$. We show that for a large enough t, $\mathsf{Real}',\mathsf{Ideal}'$ are both statistically close to the hybrid distribution $\mathcal{H}$ defined as follows:

- For every $1\leq i\leq m$, encode $\widehat{x_i}\leftarrow\mathsf{Enc}\left(x_i,1^t,1^{|C|}\right)$, and evaluates C' on $(\widehat{x_1},\cdots,\widehat{x_m})$. Let $\mathcal{W}'$ denote the wire values of C' during this evaluation, and $\mathcal{W}'_{\mathsf{Out}}$ denote its restriction to the outputs of C'.

- interpret $\mathcal{W}'_{\mathsf{Out}}$ as m encodings $\mathcal{W}^{1\prime}_{\mathsf{Out}},\cdots,\mathcal{W}^{m\prime}_{\mathsf{Out}}$. For every $1\leq i\leq m$, decode $y^i=\mathsf{Dec}^{\mathsf{Out}}_{\mathrm{GIMSS}}\left(\mathcal{W}^{i\prime}_{\mathsf{Out}},1^t\right)$, and compute a fresh encoding $y^{i\prime}\leftarrow\mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}\left(y^i,1^t\right)$.

- For every $1\leq i\leq m$, apply $(\mathcal{T}_1,\mathcal{T}_2)$ to $\left(\mathcal{W}^{i\prime}_{\mathsf{Out}},y^{i\prime}\right)$ to obtain simulated wire values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C''.

- Set $\mathcal{W}'_{\mathsf{Dec}}=\left(\mathcal{W}^{1\prime}_{\mathsf{Dec}},\cdots,\mathcal{W}^{m\prime}_{\mathsf{Dec}}\right)$.

- $\mathcal{H}=\ell\left(\mathcal{W}',\mathcal{W}'_{\mathsf{Dec}}\right)$.

$\mathsf{SD}\left(\mathsf{Real}',\mathcal{H}\right)\leq m\epsilon$. Indeed, the only difference between these distributions is the wire values of the m copies of C'' (in particular, we can condition both distributions on some possible value for the wires of C'), and so we can bound the statistical distance using the leakage-tolerance of Construction **??** and a hybrid argument over the copies of C''. More specifically, we define a sequence $\mathcal{H}^0,\mathcal{H}^1,\cdots,\mathcal{H}^m$ of hybrids which are generated identically to $\mathcal{H}$, except that in $\mathcal{H}^i$, the wires of the first i copies of C'' are honestly generated, and the wires of the copies $i+1,\cdots,m$ of C'' are generated using the translation circuits $(\mathcal{T}_1,\mathcal{T}_2)$. Then $\mathcal{H}^0=\mathcal{H},\mathcal{H}^m=\mathsf{Real}'$, and so if $\mathsf{SD}\left(\mathsf{Real}',\mathcal{H}\right)>m\epsilon$ then there exists an $1\leq i\leq m$ such that $\mathsf{SD}\left(\mathcal{H}^i,\mathcal{H}^{i-1}\right)>\epsilon$. Let $\mathcal{W}_{-i}$ denote all wires except for the internal wires of the i'th copy of C'' (in particular, $\mathcal{W}_{-i}$ also includes the inputs and outputs of the i'th copy). Using an averaging argument, we can fix all wires in $\mathcal{W}_{-i}$ while preserving the statistical distance. Let ℓ' be the leakage function (with $\mathcal{W}_{-i}$ hard-wired into it) that on input wire values $\mathcal{W}_i$ of the i'th copy of C'', applies ℓ to $(\mathcal{W}_{-i},\mathcal{W}_i)$. Then $\ell'\in\mathcal{L}$. Let $\mathcal{W}^i_i,\mathcal{W}^{i-1}_i$ denote the wire values of C'' in $\mathcal{H}^i,\mathcal{H}^{i-1}$, respectively, and let y^i,z^i denote the input and output (respectively) of the i'th copy of C'' (notice that these are the same in both hybrids, since we have conditioned on $\mathcal{W}_{-i}$). Then $\ell'\left(\mathcal{W}_i^{i-1}\right) = \ell'\left((\mathcal{T}_1,\mathcal{T}_2)\left(y^i,z^i\right)\right)$. Moreover, by the negation assumption

$$\mathsf{SD}\left(\ell\left(\mathcal{W}_i^i\right),\ell'\left((\mathcal{T}_1,\mathcal{T}_2)\left(y^i,z^i\right)\right)\right) = \mathsf{SD}\left(\ell\left(\mathcal{W}_i^i\right),\ell'\left(\mathcal{W}_i^{i-1}\right)\right) > \epsilon$$

which contradicts the $\mathcal{L}$-leakage-tolerance of Construction **??**.

Second, we claim that $\mathsf{SD}\left(\mathcal{H},\mathsf{Ideal}'\right) \leq (m+1)\,\epsilon$. Recall that

$$\mathcal{H} = \ell\left(\mathcal{W}',(\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}_{\mathsf{Out}}^{1\prime},y^{1\prime}\right),\cdots,(\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}_{\mathsf{Out}}^{m\prime},y^{m\prime}\right)\right)$$

and

$$\mathsf{Ideal}' = \ell\left(\widetilde{\mathcal{W}},(\mathcal{T}_1,\mathcal{T}_2)\left(\widetilde{\mathcal{W}}_{\mathsf{Out}}^{1},y^{1}\right),\cdots,(\mathcal{T}_1,\mathcal{T}_2)\left(\widetilde{\mathcal{W}}_{\mathsf{Out}}^{m},y^{m}\right)\right).$$

We define an additional hybrid distribution $\mathcal{H}' = \left(\widetilde{\mathcal{W}},(\mathcal{T}_1,\mathcal{T}_2)\left(\widetilde{\mathcal{W}}_{\mathsf{Out}}^{1},y^{1\prime}\right),\cdots,(\mathcal{T}_1,\mathcal{T}_2)\left(\widetilde{\mathcal{W}}_{\mathsf{Out}}^{m},y^{m\prime}\right)\right)$ (That is, $\mathcal{H}'$ is identical to Ideal', *except that to simulate the internal wires of the m copies of C'', the outputs are sampled according to the distribution over $\left(y^{1\prime},\cdots,y^{m\prime}\right)$.*) Then:

- $\mathsf{SD}\left(\mathcal{H}',\mathsf{Ideal}'\right) \leq m\epsilon$. Indeed, we can condition both distributions on the value of $\widetilde{\mathcal{W}}$, and let ℓ' be the leakage function (with $\widetilde{\mathcal{W}}$ hard-wired into it) that on input $\left(z^1,\cdots,z^m\right)$, outputs $\ell\left((\mathcal{T}_1,\mathcal{T}_2)\left(\widetilde{\mathcal{W}}_{\mathsf{Out}}^{1},z^1\right),\cdots,(\mathcal{T}_1,\mathcal{T}_2)\left(\widetilde{\mathcal{W}}_{\mathsf{Out}}^{m},z^m\right)\right)$. Then $\ell' \in \mathcal{L}$, $\mathcal{H}' \equiv \ell'\left(y^{1\prime},\cdots,y^{m\prime}\right)$, whereas $\mathsf{Ideal}' \equiv \ell'\left(y^1,\cdots,y^m\right)$. Therefore, $\mathsf{SD}\left(\mathcal{H}',\mathsf{Ideal}'\right) \leq m\epsilon$ by the leakage-resilience of $\mathsf{Enc}_{\mathrm{GIMSS}}^{\mathsf{In}}$ and a union bound. (The $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that the input encoding is $(\mathcal{L},\epsilon)$-leakage-indistinguishable, since leakage functions may choose to leak only on the inputs of the compiled circuit.)

- $\mathsf{SD}\left(\mathcal{H}',\mathcal{H}\right) \leq \epsilon$. Indeed, since the outputs $y^{1\prime},\cdots,y^{m\prime}$ of the m copies of C'' are identically distributed in both distributions, we can condition both distributions on these values. Let ℓ' be the leakage function (with $\left(y^{1\prime},\cdots,y^{m\prime}\right)$ hard-wired into it) that on input wire value $\mathcal{W}''$ for C', extracts the outputs $\left(\mathcal{W}_{\mathsf{Out}}^{1''},\cdots,\mathcal{W}_{\mathsf{Out}}^{m''}\right)$ of C', generates, $\mathcal{W}_{\mathsf{Dec}}^{i''} = (\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}_{\mathsf{Out}}^{i''},y^{i\prime}\right)$ for every $1 \leq i \leq m$, and outputs $\ell\left(\mathcal{W}'',\mathcal{W}_{\mathsf{Dec}}^{1''},\cdots,\mathcal{W}_{\mathsf{Dec}}^{m''}\right)$. Then $\ell' \in \mathcal{L}$, and notice that $\ell'\left(\mathcal{W}'\right) = \mathcal{H}$, whereas $\ell'\left(\widetilde{\mathcal{W}}\right) = \mathcal{H}'$. Since $\mathcal{W}',\widetilde{\mathcal{W}}$ are both generated by evaluating C' on different inputs, the $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that $\mathsf{SD}\left(\ell'\left(\mathcal{W}'\right),\ell'\left(\widetilde{\mathcal{W}}\right)\right) = \mathsf{SD}\left(\mathcal{H},\mathcal{H}'\right) \leq \epsilon$.

Next, we consider the case that $\mathsf{B} \neq \emptyset$. The difference from the first case is that now Real includes the output of $\widehat{C}$, the decoded output of Dec, and the inputs chosen by the adversary for the corrupted parties; and Sim receives the outcome of the computation, and is required to simulate the outputs of $\widehat{C}$, and the inputs chosen by the adversary. The simulator operates as follows. (Notice that the only differences from the case that $\mathsf{B} = \emptyset$ are: (1) in Step (**??**), when for the parties in B, Sim generates encodings of their actual inputs; and (2) in Step (**??**), when Sim uses the actual output of C to generate the output of C''.)

1. **Obtaining B and the leakage function.** Invokes $\mathcal{A}$ on input 1^n to obtain the set B of corrupted parties, and a leakage function ℓ.

2. **Obtaining inputs (of corrupted parties) and output.** Chooses to corrupt the set B, and obtains $\{x_i\}_{i\in\mathsf{B}}$ (Which it also forwards to $\mathcal{A}$), and $y = C\left(x_1,\cdots,x_m\right)$.

3. **Simulating the wire values of C'.** For every $i \in \mathsf{B}$, computes $\widehat{x_i} \leftarrow \mathsf{Enc}\left(x_i,1^t,1^{|C|}\right)$, and for every $i \notin \mathsf{B}$, generates $\widehat{x_i} \leftarrow \mathsf{Enc}\left(0^n,1^t,1^{|C|}\right)$. Sim then evaluates C' on $\left(\widehat{x_1},\cdots,\widehat{x_m}\right)$, and let $\widetilde{\mathcal{W}}$ denote the wires values of C' in this evaluation. Let $\widetilde{\mathcal{W}}_{\mathsf{Out}}^{1},\cdots,\widetilde{\mathcal{W}}_{\mathsf{Out}}^{m}$ denote the restriction of $\widetilde{\mathcal{W}}$ to the m outputs of C'. (Intuitively, the simulator emulates the evaluation of C' when the inputs of all honest parties are the all-0 strings.)

4. **Simulating the wire values of C''.**
   - Runs the simulator $\mathsf{Sim}^{\mathrm{DF}}$ to obtain view-translator circuits $(\mathcal{T}_1, \mathcal{T}_2)$.
   - For every $i \in \mathsf{B}$, sets $y^i = \mathsf{Dec}^{\mathrm{GIMSS}}_{\mathsf{Out}}\left(\widetilde{\mathcal{W}}^i_{\mathsf{Out}}, 1^t, 1^{|C|}\right)$.
   - Picks $y^i, i \notin \mathsf{B}$ uniformly at random subject to the constraint that $y = \oplus_{i=1}^m y^i$, and generates a random encoding $\widetilde{y}^i \leftarrow \mathsf{Enc}_{\mathrm{GIMSS}}\left(y^i, 1^t, 1^{|C|}\right)$.
   - For every $i \notin \mathsf{B}$, uses $(\mathcal{T}_1, \mathcal{T}_2)$, with inputs $\left(\widetilde{\mathcal{W}}^i_{\mathsf{Out}}, \widetilde{y}^i\right)$ to obtain simulated wire values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ for the i'th copy of C''.
   - For every $i \in \mathsf{B}$, honestly evaluates C'' with input $\widetilde{\mathcal{W}}^i_{\mathsf{Out}}$, and masks as defined in $\widehat{x}_i$, to obtain the wires values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ of the i'th copy of C'', and its output $\widetilde{y}^i$.
   - Sets $\widetilde{\mathcal{W}}_{\mathsf{Dec}} = \left(\widetilde{\mathcal{W}}^1_{\mathsf{Dec}}, \cdots, \widetilde{\mathcal{W}}^m_{\mathsf{Dec}}\right)$.
5. Outputs $\left(\{\widetilde{x}_i\}_{i\in\mathsf{B}}, \widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)$.

Let

$$\mathsf{Ideal} = \left(\mathsf{B}, \{\widehat{x}_i\}_{i\in\mathsf{B}}, \ell\left(\widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right), \left(\left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right), y\right)\right)$$

and

$$\mathsf{Real} = \left(\mathsf{B}, \{x'_i\}_{i\in\mathsf{B}}, \ell\left(\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}\right), \left(\left(\widehat{y}^1, \cdots, \widehat{y}^m\right), y\right)\right)$$

where $\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}$ are the wire values of C', and the m copies of C'' (respectively) in the real world execution on the inputs $\{x'_i\}_{i\in\mathsf{B}}$ chosen by the adversary (and honest encodings $\widehat{x}_i$ of x_i for every $i \notin \mathsf{B}$), and $\left(\widehat{y}^1, \cdots, \widehat{y}^m\right)$ are the outputs of the m copies of C''.

We claim that for every $n \in \mathbb{N}$, every $(x_1, \cdots, x_m) \in (\{0,1\}^n)^m$, and every $\ell \in \mathcal{L}$ it holds that $\mathsf{SD}\left(\mathsf{Real}, \mathsf{Ideal}\right) \leq (2m-1)\,\epsilon$. Since B is identically distributed in both distributions, and so are the output y and the inputs $\{\widehat{x}_i\}_{i\in\mathsf{B}}, \{x'_i\}_{i\in\mathsf{B}}$ (in both cases these are honestly-generated encodings of $\{x_i\}_{i\in\mathsf{B}}$), it suffices to prove indistinguishability conditioned on these values. Let

$$\mathsf{Ideal}' = \left(\ell\left(\widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right), \left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right)\right)$$

and

$$\mathsf{Real}' = \left(\ell\left(\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}\right), \left(\widehat{y}^1, \cdots, \widehat{y}^m\right)\right).$$

We show that for a large enough t, $\mathsf{Real}', \mathsf{Ideal}'$ are both statistically close to the hybrid distribution $\mathcal{H}$ defined as follows:

- For every $i \notin \mathsf{B}$, encode $\widehat{x}'_i \leftarrow \mathsf{Enc}\left(x_i, 1^t, 1^{|C|}\right)$. For every $i \in \mathsf{B}$, set $\widehat{x}'_i = \widehat{x}_i$ ($\{\widehat{x}_i\}_{i\in\mathsf{B}}$ are the input encodings both $\mathsf{Real}, \mathsf{Ideal}$ were conditioned on), and evaluates C' on $(\widehat{x}'_1, \cdots, \widehat{x}'_m)$. Let $\mathcal{W}'$ denote the wire values of C' during this evaluation, and $\mathcal{W}'_{\mathsf{Out}}$ denote its restriction to the outputs of C'. Interpret $\mathcal{W}'_{\mathsf{Out}}$ as m encodings $\mathcal{W}^{1\prime}_{\mathsf{Out}}, \cdots, \mathcal{W}^{m\prime}_{\mathsf{Out}}$.
- For every $i \notin \mathsf{B}$, decode $y^i = \mathsf{Dec}^{\mathsf{Out}}_{\mathrm{GIMSS}}\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, 1^t\right)$, compute a fresh encoding $y^{i\prime} \leftarrow \mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}\left(y^i, 1^t\right)$, and apply $(\mathcal{T}_1, \mathcal{T}_2)$ to $\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, y^{i\prime}\right)$ to obtain simulated wire values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C''.
- For every $i \in \mathsf{B}$, honestly evaluate C'' with input $\mathcal{W}^{i\prime}_{\mathsf{Out}}$, and masks as defined in $\widehat{x}'_i$, to obtain the wires values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C'', and its output $y^{i\prime}$.
- Set $\mathcal{W}'_{\mathsf{Dec}} = \left(\mathcal{W}^{1\prime}_{\mathsf{Dec}}, \cdots, \mathcal{W}^{m\prime}_{\mathsf{Dec}}\right)$.
- $\mathcal{H} = \left(\ell\left(\mathcal{W}', \mathcal{W}'_{\mathsf{Dec}}\right), \left(y'_1, \cdots, y'_m\right)\right)$.

We first claim that $\mathsf{SD}\left(\mathsf{Real}',\mathcal{H}\right) \le (m-|\mathsf{B}|)\,\epsilon \le (m-1)\,\epsilon$. Indeed, the only difference between these distributions is the wire values of the $m-|\mathsf{B}|$ copies of C'' that correspond to the output shares of the honest parties. Similar to the case that $B=\emptyset$, we can bound the statistical distance using a hybrid argument, moving from $\mathcal{H}$ to Real' by changing the wire values of one of these copies (i.e., a copy of C'' that decodes the share of an honest party) at a time, from simulated to actual wire values.

Second, we claim that $\mathsf{SD}\left(\mathcal{H},\mathsf{Ideal}'\right) \le m\epsilon$. We define an additional hybrid distribution $\mathcal{H}'$ which is generated similar to $\mathcal{H}$, except that for $i \notin \mathsf{B}$, the internal wires $\widetilde{\mathcal{W}}^{i'}_{\mathsf{Dec}}$ of the i'th copy of C'' are generated as $(\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}^{i'}_{\mathsf{Out}},\widetilde{y}^i\right)$, where the $\widetilde{y}^i$'s for $i \notin \mathsf{B}$ are generated as in Ideal (i.e., as encodings of y^i's that are random subject to the constraint that together with $y^i, i \in \mathsf{B}$ they sum to y). Then:

- $\mathsf{SD}\left(\mathcal{H}',\mathcal{H}\right) \le (m-1)\,\epsilon$. Indeed, we can condition both distributions on the values of $\mathcal{W}'$, $\left\{\mathcal{W}^{i'}_{\mathsf{Dec}}\right\}_{i\in\mathsf{B}}$, and $\left\{y^{i'}\right\}_{i\in\mathsf{B}}$. Let ℓ' be the leakage function (with $\mathcal{W}, \left\{\mathcal{W}^{i'}_{\mathsf{Dec}}\right\}_{i\in\mathsf{B}}$, and $\left\{y^{i'}\right\}_{i\in\mathsf{B}}$ hard-wired into it) which on input $\left\{z^i\right\}_{i\notin\mathsf{B}}$, operates as follows. For every $i \notin \mathsf{B}$, it computes $\mathcal{W}^{i'}_{\mathsf{Dec}} = (\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}^{i'}_{\mathsf{Out}}, z^i\right)$, and outputs $\ell\left(\mathcal{W}',\mathcal{W}^{1'}_{\mathsf{Dec}},\cdots,\mathcal{W}^{m'}_{\mathsf{Dec}}\right)$. Then $\ell' \in \mathcal{L}$, $\mathcal{H} \equiv \ell'\left(y^{1'},\cdots,y^{m'}\right)$, whereas $\mathcal{H}' \equiv \ell'\left(\widetilde{y}^1,\cdots,\widetilde{y}^m\right)$. Therefore, $\mathsf{SD}\left(\mathcal{H}',\mathcal{H}\right) \le (m-|B|)\,\epsilon \le (m-1)\,\epsilon$ by the leakage-resilience of $\mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}$ and the union bound. (The $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that the input encoding is $(\mathcal{L},\epsilon)$-leakage-indistinguishable, since leakage functions may choose to leak only on the inputs of the compiled circuit.)

- $\mathsf{SD}\left(\mathcal{H}',\mathsf{Ideal}\right) \le \epsilon$. Notice first that $\left\{\widetilde{\mathcal{W}}^{i}_{\mathsf{Out}}\right\}_{i\in\mathsf{B}} \equiv \left\{\mathcal{W}^{i'}_{\mathsf{Out}}\right\}_{i\in\mathsf{B}}$ since C^{share} outputs a random additive secret sharing of C's output (and $\mathsf{B} \neq [m]$, so the shares of $i \in \mathsf{B}$ are uniformly random in both distributions), and C' outputs random encodings of the outputs of C^{share}. Consequently, the outputs $\widetilde{y}^1,\cdots,\widetilde{y}^m$ of the m copies of C'' are identically distributed in both distributions: for $i \notin \mathsf{B}$ this is by the choice of $\widetilde{y}^i$; whereas for $i \in \mathsf{B}$ this is because in both distributions they are generated in the same way from the inputs of the copies of C'' that correspond to $i \in \mathsf{B}$. Moreover, since for every $i \notin \mathsf{B}$, $\mathcal{W}^{i'}_{\mathsf{Dec}}$ and $\widetilde{\mathcal{W}}^{i}_{\mathsf{Dec}}$ are obtained by honestly evaluating the i'th copy of C'', and the same masks are used in both (the masks are provided as part of the encoding $\hat{x}_i$, which we have fixed), then these wires are also identical in both distributions, and we can further condition both distributions on the value of these wires. Let ℓ' be the leakage function (with $\left(\widetilde{y}^1,\cdots,\widetilde{y}^m\right)$ and $\left\{\widetilde{\mathcal{W}}^{i}_{\mathsf{Dec}}\right\}_{i\in\mathsf{B}}$ hard-wired into it) that on input wire value $\mathcal{W}''$ for C' operates as follows. First, it extracts the outputs $\left(\mathcal{W}^{1''}_{\mathsf{Out}},\cdots,\mathcal{W}^{m''}_{\mathsf{Out}}\right)$ of C'. Then, for every $i \notin \mathsf{B}$ it generates $\mathcal{W}^{i''}_{\mathsf{Dec}} = (\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}^{i''}_{\mathsf{Out}},\widetilde{y}^i\right)$. Finally, it outputs $\ell\left(\mathcal{W}'',\left\{\widetilde{\mathcal{W}}^{i}_{\mathsf{Dec}}\right\}_{i\in\mathsf{B}},\left\{\mathcal{W}^{i''}_{\mathsf{Dec}}\right\}_{i\notin\mathsf{B}}\right)$. Then $\ell' \in \mathcal{L}$, and notice that $\ell'\left(\mathcal{W}'\right) = \mathcal{H}'$, whereas $\ell'\left(\widetilde{\mathcal{W}}\right) = \mathsf{Ideal}$. Since $\mathcal{W}',\widetilde{\mathcal{W}}$ are both generated by evaluating C' on different inputs, the $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that $\mathsf{SD}\left(\mathcal{H}',\mathsf{Ideal}\right) = \mathsf{SD}\left(\ell'\left(\mathcal{W}'\right),\ell'\left(\widetilde{\mathcal{W}}\right)\right) \le \epsilon$.

Second, we claim that $\mathsf{SD}\left(\mathcal{H},\mathsf{Ideal}'\right) \le \epsilon$. Indeed, since the output $\left(\widehat{y^1},\cdots,\widehat{y^m}\right)$ of C'' is identically distributed in both distributions, we can condition both distributions on this value. Let ℓ' be the leakage function (with $\left(\widehat{y^1},\cdots,\widehat{y^m}\right)$ hard-wired into it) that on input wire value $\mathcal{W}''$ for C', extracts the outputs $\mathcal{W}''_{\mathsf{Out}}$ of C', generates $\mathcal{W}''_{\mathsf{Dec}} = (\mathcal{T}_1,\mathcal{T}_2)\left(\mathcal{W}''_{\mathsf{Out}},\left(\widehat{y^1},\cdots,\widehat{y^m}\right)\right)$, and outputs $\ell\left(\mathcal{W}'',\mathcal{W}''_{\mathsf{Dec}}\right)$. Then $\ell' \in \mathcal{L}$, and notice that $\ell'\left(\widetilde{\mathcal{W}}\right) = \ell\left(\widetilde{\mathcal{W}},\widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)$, whereas $\ell'\left(\mathcal{W}'\right) = \ell\left(\mathcal{W}',\mathcal{W}'_{\mathsf{Dec}}\right)$. Since $\mathcal{W}',\widetilde{\mathcal{W}}$ are both generated by evaluating C' on different inputs, the $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that $\mathsf{SD}\left(\ell'\left(\mathcal{W}'\right),\ell'\left(\widetilde{\mathcal{W}}\right)\right) = \mathsf{SD}\left(\mathcal{H},\mathsf{Ideal}\right) \le \epsilon$. ■

7 A Multiparty LRCC

In this section we construct a multiparty LRCC that withstands active adversaries. The high-level idea of the construction is as follows. Given an m-party protocol C, we first replace it with a circuit C^{share} that emulates C but outputs a secret-sharing of the outputs, then compile C^{share} using the LRCC of [?]. We then refresh each of the shares using a circuit C_{Dec}. However, to guarantee leakage-resilience, and correctness of the computation in the presence of actively-corrupted parties, we first replace the circuit C_{Dec} with its additively-secure version C'_{Dec}, then compile C'_{Dec} using the LTCC of [?] to obtain a leakage-tolerant circuit $\widehat{C}'_{\mathsf{Dec}}$. We use m copies of $\widehat{C}'_{\mathsf{Dec}}$, where the i'th copy refreshes the i'th secret share, using masking inputs provided by the i'th party. Each party provides, as its input encoding to the device, both a leakage-resilient encoding of its input, and the masking inputs needed for the computation in $\widehat{C}_{\mathsf{Dec}}$. (We note that the only difference from the construction of a passive-secure MPCC is that C_{Dec} is replaced with an AMD circuit.) The output decoder decodes each of the secret shares, and reconstructs the output from the shares (unless it detects that one of the parties provided ill-formed masking inputs, in which case the computation aborts). This is formalized in the next construction.

Construction 7.1 (Multiparty LRCC)**.** *Let $m \in \mathbb{N}$ denote the number of parties, $t \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, $k \in \mathbb{N}$ be an output length parameter, and $c \in \mathbb{N}$ be a constant. The m-party LRCC uses the following building blocks:*

- *The LRCC $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{E}_{\mathsf{In}}^{\mathrm{GIMSS}} = \left(\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{In}}^{\mathrm{GIMSS}}\right), \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ of Theorem **??** (Construction **??**), where the outputs of the leakage-resilient circuit are encoded by the encoding scheme $\left(\mathsf{Enc}_{\mathrm{GIMSS}} : \mathbb{F}_2 \to \mathbb{F}_2^{4ct}, \mathsf{Dec}_{\mathrm{GIMSS}} : \mathbb{F}_2^{4ct} \to \mathbb{F}_2^2\right)$.*
- *The LTCC $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ of Theorem **??** (Construction **??**) over a field $\mathbb{F} = \Omega(t)$, and its underlying encoding scheme $\mathsf{E}_{\mathrm{DF}}^{\mathsf{In}} = \left(\mathsf{Enc}_{\mathrm{DF}}^{\mathsf{In}}, \mathsf{Dec}_{\mathrm{DF}}^{\mathsf{In}}\right)$ that outputs encodings of length $\hat{n}^{\mathrm{DF}}(n,t)$.*
- *The additively-secure circuit compiler $\mathsf{Comp}^{\mathsf{add}}$ of Theorem **??**.*

*The m-*party LRCC $(\mathsf{Enc}, \mathsf{Comp}, \mathsf{Dec})$ *is defined as follows.*

- *For every $n, t, t_{\mathsf{In}} \in \mathbb{N}$ and every $x \in \mathbb{F}^n$, $\mathsf{Enc}\left(x, 1^t, 1^{t_{\mathsf{In}}}\right) = \left(\mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}\left(x, 1^t, 1^{t_{\mathsf{In}}}\right), \mathsf{Enc}_{\mathsf{In}}^{\mathrm{DF}}\left(0^{t_{\mathsf{In}}}, 1^t\right)\right)$.*
- *For every $y = \left(\left(f_L^1, f_R^1, y^1\right), \cdots, \left(f_L^m, f_R^m, y^m\right)\right) \in \left(F^{2+2tc(k+1)}\right)^m$, $\mathsf{Dec}\left(y, 1^t\right)$ computes $\left(f_i, z^i\right) = \mathsf{Dec}_{\mathrm{GIMSS}}^{\mathsf{Out}}\left(y^i, 1^t\right)$. If $f_L^i = f_R^i = \mathsf{f}_i^0 = 0$ for all $1 \leq i \leq m$ then Dec outputs $\left(0, \sum_{i=1}^m z^i\right)$, otherwise it outputs $\left(1, 0^k\right)$. (Intuitively, each triplet $\left(f_L^i, f_R^i, y^i\right)$ consists of a pair of flags output by the LTCC, indicating whether the computation in one of its gadgets failed; and an encoding of a flag, concatenated with an additive secret share of the output.)*
- Comp *on input $m \in \mathbb{N}$, and an m-party circuit $C : (\mathbb{F}^n)^m \to \mathbb{F}^k$:*
  1. *Constructs the circuit $C^{\mathsf{share}} : (\mathbb{F}^n)^m \to \mathbb{F}^{mk}$ that operates as follows:*
     - *Evaluates C on inputs $x_1, \cdots, x_m$ to obtain the output $y = C(x_1, \cdots, x_m)$.*
     - *Generates $y_1, \cdots, y_{m-1} \in_R \mathbb{F}^k$, and sets $y_m = y \oplus \sum_{i=1}^{m-1} y_i$. ($y_1, \cdots, y_m$ are random additive secret shares of y.)*
     - *For every $1 \leq i \leq m$, generates y'_i by replacing each bit of y_i with (the bit string representation of) the bit as an element of $\mathbb{F}$.*
     - *Outputs $(y'_1, \cdots, y'_m)$.*
  2. *Computes $C' = \mathsf{Comp}^{\mathrm{GIMSS}}\left(C^{\mathsf{share}}\right)$.*
  3. *Construct the circuit $C^{\mathsf{Dec}} : \mathbb{F}^{4ct(k+1)} \to \mathbb{F}^{4ct(k+1)}$ that operates as follows:*
     - *Decodes its input using $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$ to obtain a flag $f \in \mathbb{F}_2$ and output $z \in \mathbb{F}^k$.*

- *If $f = 1$, sets $z' = 0^k$, otherwise $z' = z$.*
- *Generates $\mathbf{e} \leftarrow \mathsf{Enc}_{\mathrm{GIMSS}}\left(\left(f, z'\right), 1^t\right)$, and outputs $\mathbf{e}$.*

4. *Generate $\widehat{C}^{\mathsf{amd}} = \mathsf{Comp}^{\mathsf{add}}\left(C^{\mathsf{Dec}}\right)$.*
5. *Generate $C'' = \mathsf{Comp}^{\mathrm{DF}}\left(\widehat{C}^{\mathsf{amd}}\right)$.*
6. *Outputs the circuit $\hat{C}$ obtained by concatenating a copy of C'' to each of the m outputs of C'. (We note that the i'th copy of C'' takes its masking inputs from the encoding of the i'th input to $\hat{C}$.)*

The next theorem states that Construction **??** is a multiparty LRCC.

Theorem 7.2 (Multiparty LRCC)**.** *Let $n, k \in \mathbb{N}$ be input and output length parameters, $\boldsymbol{S}(n) : \mathbb{N} \rightarrow \mathbb{N}$ be a size function, $\epsilon(n), \epsilon'(n) : \mathbb{N} \rightarrow (0,1)$ be error functions, $t \in \mathbb{N}$ be a leakage bound, let $c \in \mathbb{N}$ be a constant, and let $\mathsf{m} \in \mathbb{N}$ denote the number of parties. Let $\mathcal{L}$ denote the family of all t-BCL functions. If:*

- *$\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ is an $(\mathcal{L}, \epsilon, \boldsymbol{S}(n) + 2m)$-relaxed LRCC with abort, where for security parameter t, $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathrm{GIMSS}}$ can be evaluated using circuits of size $s^{\mathrm{GIMSS}}(t)$,*
- *$\mathsf{Comp}^{\mathsf{add}}$ is an $\epsilon'(n)$-additively-secure circuit compiler over $\mathbb{F}$, where there exist: (1) $B : \mathbb{N} \rightarrow \mathbb{N}$ such that for any circuit C, $\mathsf{Comp}^{\mathsf{add}}(C)$ has size at most $B(|C|)$; and (2) a PPT algorithm Alg' that given an additive attack $\mathcal{A}$ outputs the ideal attack $\left(\mathbf{a}^{\mathsf{in}}, \mathcal{A}^{\mathsf{Out}}\right)$ (whose existence follows from the additive-attack security property of Definition **??**), and*
- *$\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ is an $\left(\mathcal{L}, \epsilon, B\left(2s^{\mathrm{GIMSS}}(t) + ck\right)\right)$-LTCC.*

*Then Construction **??** is an m-party $(\mathcal{L}, (2m+1)\epsilon(n) + \epsilon'(n) + \mathsf{negl}(t), \boldsymbol{S}(n))$-LRCC.*

*Moreover, if on input a circuit of size s, $\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{Comp}^{\mathrm{DF}}$ output circuits of size $\widehat{s}^{\mathrm{GIMSS}}(s)$, and $s^{\mathrm{DF}}(s)$, respectively, then on input a circuit C of size s, the compiler of Construction **??** outputs a circuit $\widehat{C}$ of size $\widehat{s}^{\mathrm{GIMSS}}(s+2m) + s^{\mathrm{DF}}\left(B\left(2s^{\mathrm{GIMSS}}(t) + ck\right)\right)$.*

Proof. We show that Construction **??** has the required properties. It follows directly from the construction that the compiler has the required syntax.

Complexity. Let C be a circuit to be compiled, and denote $s = |C|$. Then the circuit C^{share} constructed in Step (**??**) has size $s + 2m$. By the assumptions of the theorem, the circuit C' Constructed in Step (**??**) has size $\hat{s}^{\mathrm{GIMSS}}(s+2m)$. Moreover, since $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathrm{GIMSS}}$ have size $s^{\mathrm{GIMSS}}(t)$, then there exists a constant $c \in \mathbb{N}$ such that the circuit C^{Dec} constructed in Step (**??**) has size $2s^{\mathrm{GIMSS}}(t) + ck$. Consequently, the circuit $\widehat{C}^{\mathsf{amd}}$ from Step (**??**) has size $B\left(2s^{\mathrm{GIMSS}}(t) + ck\right)$, and so the circuit C'' from Step (**??**) has size $s^{\mathrm{DF}}\left(B\left(2s^{\mathrm{GIMSS}}(t) + ck\right)\right)$. Therefore, the combined size of the compiled circuit $\widehat{C}$ is $\hat{s}^{\mathrm{GIMSS}}(s+2m) + s^{\mathrm{DF}}\left(B\left(2s^{\mathrm{GIMSS}}(t) + ck\right)\right)$.

In particular, we have shown that for every m-party circuit C, $\left|C^{\mathsf{share}}\right| \leq |C| + 2m$, and $\left|\widehat{C}^{\mathsf{amd}}\right| \leq B\left(2s^{\mathrm{GIMSS}}(t) + ck\right)$. Therefore, if $|C| \leq \boldsymbol{S}(n)$, then the leakage-resilience of $\left(\mathsf{Comp}^{\mathrm{GIMSS}}, \mathsf{Enc}_{\mathsf{In}}^{\mathrm{GIMSS}}, \mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}\right)$ guarantees that C' is $(\mathcal{L}, \epsilon)$-leakage-resilient, and the leakage-tolerance of $\left(\mathsf{Comp}^{\mathrm{DF}}, \mathsf{E}^{\mathrm{DF}}\right)$ guarantees that C'' is $(\mathcal{L}, \epsilon)$-leakage-tolerant. (This will be needed to argue security.)

Correctness. The perfect correctness of $\mathsf{Comp}^{\mathrm{GIMSS}}$ guarantees that C' perfectly emulates C^{share}, i.e., C' outputs an encoding (which can be decoded using $\mathsf{Dec}_{\mathsf{Out}}^{\mathrm{GIMSS}}$) of the output of C^{share}, which in turn is (by the definition of C^{share}) an additive secret sharing of the output of C. Moreover, the $1 - \epsilon$ correctness of $\mathsf{Comp}^{\mathrm{DF}}$ guarantees that except with probability ϵ, each copy C'' perfectly emulates $\widehat{C}^{\mathsf{amd}}$ when its inputs are well formed (which corresponds to the case in which there is no additive attack on $\widehat{C}^{\mathsf{amd}}$). Using the union bound, except with probability $m\epsilon$ all copies of C'' perfectly emulate $\widehat{C}^{\mathsf{amd}}$, and conditioned on this event, the perfect correctness of $\mathsf{Comp}^{\mathsf{amd}}$ guarantees that C'' perfectly emulates C^{Dec}. Therefore, when $\widehat{C}$ is evaluated on a well-formed encoding of $(x_1, \cdots, x_m)$, conditioned on the event that non of the copies of C'' failed, its output is identically distributed to valid encodings, according to $\mathsf{Enc}_{\text{GIMSS}}$, of an additive secret sharing of $C(x_1, \cdots, x_m)$. Finally, the perfect correctness of the encoding scheme $(\mathsf{Enc}_{\text{GIMSS}}, \mathsf{Dec}_{\text{GIMSS}})$ guarantees that all the decodings in Dec succeed, and so the output is $C(x_1, \cdots, x_m)$. In summary, the output is $C(x_1, \cdots, x_m)$ except with probability $m\epsilon$.

Security. We describe a simulator Sim that simulates the wire values of $\widehat{C}$. Sim uses the adversary $\mathcal{A}$ as a black-box to determine the set $\mathsf{B} = \{i_1, \cdots, i_r\}$ of corrupted parties (Sim chooses to corrupt the same set of parties).

We first consider the case that $\mathsf{B} = \emptyset$. The proof of this case follows identically to the case $\mathsf{B} = \emptyset$ in the proof of Theorem **??**, except that $\mathsf{Enc}_{\text{GIMSS}}$ (instead of $\mathsf{Enc}_{\text{GIMSS}}^{\mathsf{Enc}}$) is used to generate the outputs of the copies of C'', and we use the fact that the $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** implies that encodings generated by $\mathsf{Enc}_{\text{GIMSS}}$ are $\mathcal{L}$-leakage-indistinguishable (since it is used to generate the output encodings of the leakage-resilient circuit, and leakage functions can choose to leak only on the outputs).

Next, we consider the case that $\mathsf{B} \neq \emptyset$. There are two main differences from the case that $\mathsf{B} = \emptyset$: (1) Real now includes the output of $\widehat{C}$, the decoded output of Dec, and the inputs chosen by the adversary for the corrupted parties, and Sim receives the outcome of the computation, and is required to simulate the outputs of $\widehat{C}$, and the inputs chosen by the adversary; and (2) the adversary may provide ill-formed encodings as its input to the computation.

The simulator will first invoke $\mathcal{A}$ on input 1^n to obtain the set B of corrupted parties, and a leakage function ℓ. Sim then chooses to corrupt the set B of parties, and receives their inputs $\{x_i\}_{i \in \mathsf{B}}$, which it also provides to $\mathcal{A}$. It then receives from $\mathcal{A}$ effective inputs $\{\widetilde{w}_i'\}_{i \in \mathsf{B}}$ to be used for the computation. For every $i \in \mathsf{B}$, Sim interprets $\widetilde{w}_i' = \left(\widetilde{x}_i', \mathsf{mask}_i'\right)$, where $\widetilde{x}_i'$ is the encoded input of the i'th party to C', and mask_i' are the masks it provides for the i'th copy of C''. (Notice that this interpretation is consistent with the way $\hat{C}$ interprets its input encoding.) Recall that if $\widetilde{x}_i'$ is not a valid encoding according to $\mathsf{Enc}_{\text{GIMSS}}^{\mathsf{In}}$ then it is interpreted in $\widehat{C}$ as encoding 0^n. Therefore, we can assume without loss of generality that every $\widetilde{x}_i'$ is a valid encoding of some $x_i' \in \mathbb{F}_2^n$ (since for invalid encodings $\widetilde{x}_i'$, we set $x_i' = 0^n$). We consider two possible cases, according to whether the masks mask_i' are well-formed or not.

First, consider the case that for every $i \in \mathsf{B}$, mask_i' consists of well-formed masks (i.e., inner-product encodings of 0). In this case, Sim chooses $\{x_i'\}_{i \in \mathsf{B}}$ as the effective inputs of the parties in B. The simulator then receives $y = C(x_1', \cdots, x_n')$, where for every $i \notin \mathsf{B}$, $x_i' = x_i$. The simulator uses $\{\widetilde{w}_i\}_{i \in \mathsf{B}}$ as the inputs the adversary would have used in the real world. Notice that in this case, all masks used in the copies of C'' are well-formed, so no additive attack is launched on the copies of $\widehat{C}^{\mathsf{amd}}$, and consequently they emulate the computation in C^{Dec}. The proof now continues similarly to the case $\mathsf{B} \neq \emptyset$ in the proof of Theorem **??**. The only differences are that Sim:

- Uses $\widetilde{w}_i'$ as the inputs of the parties in B (instead of generating them as valid encodings of x_i).
- Uses $\mathsf{Enc}_{\text{GIMSS}}$ (instead of $\mathsf{Enc}_{\text{GIMSS}}^{\mathsf{In}}$) to generate the encodings at the outputs of the copies of C''.
- Sets the value of b (which indicates whether to abort the computation or not) based on whether any of the copies of C'' corresponding to parties in B set a flag. More specifically, if for some $i \in \mathsf{B}$ the i'th copy of C'' outputs $f_L^i \neq 0$ or $f_R^i \neq 0$ then Sim chooses $b = 1$ (indicating to abort the computation), otherwise it chooses $b = 0$.

The analysis of the simulator now follows similarly to the proof of Theorem **??**. The difference (other than the aforementioned modifications in the simulation) is that *active* parties may influence the computation in the copies of C'' by choosing masking inputs that would cause a gadget to fail. (We note that since the masking inputs used in all the copies of C'' are well-formed, this is the only reason the decoding in Dec might fail, and cause the output $\left(1, 0^k\right)$.) However, this would only influence the copy of C'' corresponding to that malicious party, and the probability that the copy fails happens with the same probability in both the real world and the simulation (since the distribution over the inputs and masking inputs of the copy is the same in both worlds). (We note that for copies of C'' that correspond to $i \notin \mathsf{B}$, the gadgets might fail only with negligible probability, since in a real-world execution this happens only with negligible probability, and the simulated and actual leakage on C'' is statistically close by the leakage-tolerance of the LTCC.)

Finally, consider the case that not all mask'_i are well-formed, i.e., there for at least one $i \in \mathsf{B}$, mask'_i is *not* an inner-product encoding of the all-zeros string. Let $\mathcal{I} = \left\{i \in \mathsf{B} \ : \ \mathsf{mask}'_i \text{ is ill formed}\right\}$. Then Lemma **??** guarantees that for every $i_0 \in \mathcal{I}$, there exists an additive attack $\mathbf{A}_{i_0}$ such that evaluating the i_0'th copy of C'' with masks mask'_{i_0} is equivalent to evaluating the underlying circuit $\widehat{C}^{\mathsf{amd}}$ under the additive attack $\mathbf{A}_{i_0}$. Moreover, this attack can be efficiently extracted from mask'_{i_0}. Then the additive-attack security of $\widehat{C}^{\mathsf{amd}}$ guarantees that there exists an ideal additive attack $\mathbf{a}^{\mathsf{in}}_{i_0}$ on the inputs of C^{Dec}, and a distribution $\mathcal{A}_{i_0}$ over additive attacks on the outputs of C^{Dec}, such that for any input z of C^{Dec}, $\mathsf{SD}\left(\widehat{C}^{\mathsf{amd},\mathbf{A}_{i_0}}(z), C^{\mathsf{Dec}}\left(z + \mathbf{a}^{\mathsf{in}}_{i_0}\right) + \mathcal{A}_{i_0}\right) \leq \epsilon'(n)$, and these ideal attacks can be efficiently computed from $\mathbf{A}_{i_0}$. We consider two possible cases.

First, assume that $\mathbf{a}^{\mathsf{in}}_{i_0} \neq \vec{0}$ for some $i_0 \in \mathcal{I}$. In this case, Sim chooses $b = 1$ (i.e., chooses to abort the computation). (Notice that Sim can determine whether $\mathbf{a}^{\mathsf{in}}_{i_0} \neq \vec{0}$, since $\mathbf{a}^{\mathsf{in}}_{i_0}$ can be computed efficiently from $\mathbf{a}^{\mathsf{in}}_{i_0}$, which can be computed efficiently from $\mathbf{A}_{i_0}$.) The simulation in the case continues in the following way:

1. **Simulating the wire values of C'.** For every $i \in \mathsf{B}$, Sim sets $\widetilde{x}_i = \widetilde{x}'_i$, and for every $i \notin \mathsf{B}$, Sim generates $\widetilde{x}_i \leftarrow \mathsf{Enc}_{\text{GIMSS}}\left(0^n, 1^t\right)$. Sim then evaluates C' on $(\widetilde{x_1}, \cdots, \widetilde{x_m})$, and let $\widetilde{\mathcal{W}}$ denote the wires values of C' in this evaluation. Let $\widetilde{\mathcal{W}}^1_{\mathsf{Out}}, \cdots, \widetilde{\mathcal{W}}^m_{\mathsf{Out}}$ denote the restriction of $\widetilde{\mathcal{W}}$ to the m outputs of C'. (Intuitively, the simulator emulates the evaluation of C' on the effective inputs chosen by the adversary, using the all-0 string as the input for the honest parties.)

2. **Simulating the wire values of C''.**
   - Runs the simulator Sim^{DF} to obtain view-translator circuits $(\mathcal{T}_1, \mathcal{T}_2)$.
   - For every $i \in \mathsf{B}$, honestly evaluates C'' with input $\widetilde{\mathcal{W}}^i_{\mathsf{Out}}$, and masks mask_i, to obtain the wires values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ of the i'th copy of C'', and its output $\widetilde{y}^i$.
   - For every $i \notin \mathsf{B}$, picks $y^i \in_R \mathbb{F}_2^k$ uniformly at random, and generates a random encoding $\widetilde{y}^i \leftarrow \mathsf{Enc}_{\text{GIMSS}}\left(y^i, 1^t, 1^{|C|}\right)$. Then, Sim uses $(\mathcal{T}_1, \mathcal{T}_2)$, with inputs $\left(\widetilde{\mathcal{W}}^i_{\mathsf{Out}}, \widetilde{y}^i\right)$ to obtain simulated wire values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ for the i'th copy of C''.
   - Sets $\widetilde{\mathcal{W}}_{\mathsf{Dec}} = \left(\widetilde{\mathcal{W}}^1_{\mathsf{Dec}}, \cdots, \widetilde{\mathcal{W}}^m_{\mathsf{Dec}}\right)$.

3. Outputs $\left(\{\widetilde{w}'_i\}_{i\in\mathsf{B}}, \widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)$.

Let
$$\mathsf{Ideal} = \left(\mathsf{B}, \{\widetilde{w}'_i\}_{i\in\mathsf{B}}, \ell\left(\widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right), \left(\left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right), \left(1, 0^k\right)\right)\right)$$
and
$$\mathsf{Real} = \left(\mathsf{B}, \{\widetilde{w}'_i\}_{i\in\mathsf{B}}, \ell\left(\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}\right), \left(\left(y^{1\prime}, \cdots, y^{m\prime}\right), y'\right)\right)$$
where $\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}$ are the wire values of C', and the m copies of C'' (respectively) in the real world execution on the inputs $\{\widetilde{w}'_i\}_{i\in\mathsf{B}}$ chosen by the adversary (and honest encodings $\widehat{x}_i$ of x_i for every $i \notin \mathsf{B}$), $\left(y^{1\prime}, \cdots, y^{m\prime}\right)$ are the outputs of the m copies of C'', and y' is the output of the decoder Dec.

We claim that for every $n \in \mathbb{N}$, every $(x_1, \cdots, x_m) \in (\{0,1\}^n)^m$, and every $\ell \in \mathcal{L}$ it holds that $\mathsf{SD}\left(\mathsf{Real}, \mathsf{Ideal}\right) \leq (m - |\mathsf{B}| + 2)\,\epsilon + \epsilon' + \mathsf{negl}(t) \leq (m+2)\,\epsilon + \epsilon' + \mathsf{negl}(t)$. Since $\mathsf{B}, \{\widetilde{w}'_i\}_{i\in\mathsf{B}}$ are identically distributed in both distributions (in both cases these were chosen by the adversary), it suffices to prove indistinguishability conditioned on these values. Moreover, since $\mathbf{a}^{\mathsf{in}}_{i_0} \neq \vec{0}$ for some $i_0 \in \mathcal{I}$, then except with probability ϵ', the evaluation of the i_0'th copy of C'' with mask'_{i_0} is equivalent to evaluating C^{Dec} under the additive attack $\mathbf{a}^{\mathsf{in}}_{i_0}$ on its inputs. Lemma **??** guarantees that this attack is detected by the decoder $\mathsf{Dec}^{\mathsf{Out}}_{\mathrm{GIMSS}}$ except with $\mathsf{negl}\left(t\right)$ probability, so the i_0'th copy of C'' will set a flag, which (by the definition of Dec) will cause the output to be $\left(0, 1^k\right)$. Therefore, we can further condition both $\mathsf{Real}, \mathsf{Ideal}$ on the event that an additive attack was detected, and the output of Dec is $\left(0, 1^k\right)$ (this will only increase the statistical distance by at most $\epsilon' + \mathsf{negl}\left(t\right)$).

Let

$$\mathsf{Ideal}' = \left(\ell\left(\widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right), \left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right)\right)$$

and

$$\mathsf{Real}' = \left(\ell\left(\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}\right), \left(y^{1\prime}, \cdots, y^{m\prime}\right)\right).$$

We show that for a large enough t, $\mathsf{Real}', \mathsf{Ideal}'$ are both statistically close to the hybrid distribution $\mathcal{H}$ defined as follows:

- For every $i \notin \mathsf{B}$, encode $\widehat{x}'_i \leftarrow \mathsf{Enc}\left(x_i, 1^t, 1^{|C|}\right)$. For every $i \in \mathsf{B}$, set $\widehat{x}'_i = \widetilde{x}'_i$ ($\{\widetilde{x}'_i\}_{i \in \mathsf{B}}$ are the input encodings both $\mathsf{Real}, \mathsf{Ideal}$ were conditioned on), and evaluate C' on $(\widehat{x}'_1, \cdots, \widehat{x}'_m)$. Let $\mathcal{W}'$ denote the wire values of C' during this evaluation, and $\mathcal{W}'_{\mathsf{Out}}$ denote its restriction to the outputs of C'. Interpret $\mathcal{W}'_{\mathsf{Out}}$ as m encodings $\mathcal{W}^{1\prime}_{\mathsf{Out}}, \cdots, \mathcal{W}^{m\prime}_{\mathsf{Out}}$.

- For every $i \notin \mathsf{B}$, decode $y^i = \mathsf{Dec}^{\mathsf{Out}}_{\mathrm{GIMSS}}\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, 1^t\right)$, compute a fresh encoding $y^{i\prime} \leftarrow \mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}\left(y^i, 1^t\right)$, and apply $(\mathcal{T}_1, \mathcal{T}_2)$ to $\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, y^{i\prime}\right)$ to obtain simulated wire values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C''.

- For every $i \in \mathsf{B}$, honestly evaluate C'' with input $\mathcal{W}^{i\prime}_{\mathsf{Out}}$, and masks mask'_i, to obtain the wires values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C'', and its output $y^{i\prime}$.

- Set $\mathcal{W}'_{\mathsf{Dec}} = \left(\mathcal{W}^{1\prime}_{\mathsf{Dec}}, \cdots, \mathcal{W}^{m\prime}_{\mathsf{Dec}}\right)$.

- $\mathcal{H} = \left(\ell\left(\mathcal{W}', \mathcal{W}'_{\mathsf{Dec}}\right), \left(y^{1\prime}, \cdots, y^{m\prime}\right)\right)$.

We first claim that $\mathsf{SD}\left(\mathsf{Real}', \mathcal{H}\right) \leq \left(m - |\mathsf{B}|\right)\epsilon$. Indeed, the only difference between these distributions is the wire values of the $m - |\mathsf{B}|$ copies of C'' that correspond to the output shares of the honest parties. Similar to the case that $B = \emptyset$, we can bound the statistical distance using a hybrid argument, moving from $\mathcal{H}$ to Real' by changing the wire values of one of these copies (i.e., a copy of C'' that decodes the share of an honest party) at a time, from simulated to actual wire values.

Second, we claim that $\mathsf{SD}\left(\mathcal{H}, \mathsf{Ideal}'\right) \leq 2\epsilon$. We define an additional hybrid distribution $\mathcal{H}'$ which is generated similar to $\mathcal{H}$, except that for $i \notin \mathsf{B}$, the internal wires $\widetilde{\mathcal{W}}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C'' are generated as $(\mathcal{T}_1, \mathcal{T}_2)\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, \widetilde{y}^i\right)$, where the $\widetilde{y}^i$'s for $i \notin \mathsf{B}$ are generated as in Ideal (i.e., encode uniformly random values). Then $\mathsf{SD}\left(\mathcal{H}', \mathsf{Ideal}\right) \leq \epsilon$ by similar arguments to the ones used to prove the case $\mathsf{B} \neq \emptyset$ in the proof of Theorem **??**. Similar arguments show also that $\mathsf{SD}\left(\mathcal{H}', \mathcal{H}\right) \leq \epsilon$, where we also use the fact that at least one of the copies of C'' caught the additive attack (since we have conditioned on this case), and therefore outputted 0^k. Consequently, the output of $\widehat{C}$ contains at most $k-1$ of the k additive secret shares of the output, and so all these shares (and in particular, the shares corresponding to honest parties) are random in $\mathcal{H}$ (i.e., identically distributed to how they were chosen in $\mathcal{H}'$).

Now, consider the case that $\mathbf{a}^{\mathsf{in}}_{i_0} = \vec{0}$ for all $i_0 \in \mathcal{I}$. In this case, Sim choses $\{\widetilde{x}'_i\}_{i \in \mathsf{B}}$ as the inputs of the corrupted parties, and receives the output $y = f\left(x'_1, \cdots, x'_m\right)$, where for every $i \notin \mathsf{B}$, $x'_i = x_i$. Then, Sim operates as follows:

1. **Simulating the wire values of C'.** For every $i \in \mathsf{B}$, Sim sets $\widetilde{x}_i = \widetilde{x}'_i$, and for every $i \notin \mathsf{B}$, generates $\widetilde{x}_i \leftarrow \mathsf{Enc}_{\mathrm{GIMSS}}\left(0^n, 1^t\right)$. Sim then evaluates C' on $(\widetilde{x_1}, \cdots, \widetilde{x_m})$, and let $\widetilde{\mathcal{W}}$ denote the wires values of C' in this evaluation. Let $\widetilde{\mathcal{W}}^1_{\mathsf{Out}}, \cdots, \widetilde{\mathcal{W}}^m_{\mathsf{Out}}$ denote the restriction of $\widetilde{\mathcal{W}}$ to the m

outputs of C'. (Intuitively, the simulator emulates the evaluation of C' on the effective inputs chosen by the adversary, using the all-0 string as the input for the honest parties.)

2. **Simulating the wire values of C''.**
   - Runs the simulator $\mathsf{Sim}^{\mathrm{DF}}$ to obtain view-translator circuits $(\mathcal{T}_1, \mathcal{T}_2)$.
   - For every $i \in \mathsf{B}$, honestly evaluates C'' with input $\widetilde{\mathcal{W}}^i_{\mathsf{Out}}$, and masks mask_i, to obtain the wires values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ of the i'th copy of C'', and its output $\widetilde{y}^i$. Let $y^i = \mathsf{Dec}_{\mathrm{GIMSS}}\left(\widetilde{\mathcal{W}}^i_{\mathsf{Out}}, 1^t\right)$.
   - Picks $y^i, i \notin \mathsf{B}$ uniformly at random subject to the constraint that $y = \oplus_{i=1}^m y^i$, and generates a random encoding $\widetilde{y}^i \leftarrow \mathsf{Enc}_{\mathrm{GIMSS}}\left(y^i, 1^t, 1^{|C|}\right)$. Then, Sim uses $(\mathcal{T}_1, \mathcal{T}_2)$, with inputs $\left(\widetilde{\mathcal{W}}^i_{\mathsf{Out}}, \widetilde{y}^i\right)$ to obtain simulated wire values $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ for the i'th copy of C''.
   - Sets $\widetilde{\mathcal{W}}_{\mathsf{Dec}} = \left(\widetilde{\mathcal{W}}^1_{\mathsf{Dec}}, \cdots, \widetilde{\mathcal{W}}^m_{\mathsf{Dec}}\right)$.
3. Outputs $\left(\{\widetilde{w}'_i\}_{i\in\mathsf{B}}, \widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right)$.
4. **Deciding whether to abort.** For every $i \in \mathsf{B}$, checks whether a gadget failed in the i'th copy of C'', and if so chooses $b = 1$. Otherwise, computes $\mathsf{Dec}\left(\left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right), 1^t\right)$, and if decoding failed (i.e., Dec set a flag) then Sim picks $b = 1$. Otherwise, it sets $b = 0$.

Let

$$\mathsf{Ideal} = \left(\mathsf{B}, \{\widetilde{w}'_i\}_{i\in\mathsf{B}}, \ell\left(\widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right), \left(\left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right), z\right)\right)$$

where $z = \left(1, 0^k\right)$ if Sim chose $b = 1$, otherwise $z = (0, y)$, and

$$\mathsf{Real} = \left(\mathsf{B}, \{\widetilde{w}'_i\}_{i\in\mathsf{B}}, \ell\left(\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}\right), \left(\left(y^{1\prime}, \cdots, y^{m\prime}\right), y'\right)\right)$$

where $\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}$ are the wire values of C', and the m copies of C'' (respectively) in the real world execution on the inputs $\{\widetilde{w}'_i\}_{i\in\mathsf{B}}$ chosen by the adversary (and honest encodings $\widehat{x}_i$ of x_i for every $i \notin \mathsf{B}$), $\left(y^{1\prime}, \cdots, y^{m\prime}\right)$ are the outputs of the m copies of C'', and y' is the output of the decoder Dec.

We claim that for every $n \in \mathbb{N}$, every $(x_1, \cdots, x_m) \in (\{0,1\}^n)^m$, and every $\ell \in \mathcal{L}$ it holds that $\mathsf{SD}(\mathsf{Real}, \mathsf{Ideal}) \leq (m - |\mathsf{B}| + 2)\,\epsilon + \epsilon' + \mathsf{negl}(t) \leq (m+2)\,\epsilon + \epsilon' + \mathsf{negl}(t)$. Since $\mathsf{B}, \{\widetilde{w}'_i\}_{i\in\mathsf{B}}$ are identically distributed in both distributions (in both cases these were chosen by the adversary), it suffices to prove indistinguishability conditioned on these values.

Let

$$\mathsf{Ideal}' = \left(\ell\left(\widetilde{\mathcal{W}}, \widetilde{\mathcal{W}}_{\mathsf{Dec}}\right), \left(\left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right), z\right)\right)$$

and

$$\mathsf{Real}' = \left(\ell\left(\mathcal{W}, \mathcal{W}_{\mathsf{Dec}}\right), \left(\left(y^{1\prime}, \cdots, y^{m\prime}\right), y'\right)\right).$$

We show that for a large enough t, $\mathsf{Real}', \mathsf{Ideal}'$ are both statistically close to the hybrid distribution $\mathcal{H}$ defined as follows:

- For every $i \notin \mathsf{B}$, encode $\widehat{x}'_i \leftarrow \mathsf{Enc}\left(x_i, 1^t, 1^{|C|}\right)$. For every $i \in \mathsf{B}$, set $\widehat{x}'_i = \widetilde{x}'_i$ ($\{\widetilde{x}'_i\}_{i\in\mathsf{B}}$ are the input encodings both $\mathsf{Real}, \mathsf{Ideal}$ were conditioned on), and evaluate C' on $(\widehat{x}'_1, \cdots, \widehat{x}'_m)$. Let $\mathcal{W}'$ denote the wire values of C' during this evaluation, and $\mathcal{W}'_{\mathsf{Out}}$ denote its restriction to the outputs of C'. Interpret $\mathcal{W}'_{\mathsf{Out}}$ as m encodings $\mathcal{W}^{1\prime}_{\mathsf{Out}}, \cdots, \mathcal{W}^{m\prime}_{\mathsf{Out}}$.
- For every $i \notin \mathsf{B}$, decode $y^i = \mathsf{Dec}^{\mathsf{Out}}_{\mathrm{GIMSS}}\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, 1^t\right)$, compute a fresh encoding $y^{i\prime\prime} \leftarrow \mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}\left(y^i, 1^t\right)$, and apply $(\mathcal{T}_1, \mathcal{T}_2)$ to $\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, y^{i\prime\prime}\right)$ to obtain simulated wire values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C''.

- For every $i \in \mathsf{B}$, honestly evaluate C'' with input $\mathcal{W}^{i\prime}_{\mathsf{Out}}$, and masks mask'_i, to obtain the wires values $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C'', and its output $y^{i\prime}$.
- Set $\mathcal{W}'_{\mathsf{Dec}} = \left(\mathcal{W}^{1\prime}_{\mathsf{Dec}}, \cdots, \mathcal{W}^{m\prime}_{\mathsf{Dec}}\right)$.
- Compute $y'' = \mathsf{Dec}\left(\left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right), 1^t\right)$.
- $\mathcal{H} = \left(\ell\left(\mathcal{W}', \mathcal{W}'_{\mathsf{Dec}}\right), \left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right), y''\right)$.

We first claim that $\mathsf{SD}\left(\mathsf{Real}', \mathcal{H}\right) \leq (m - |\mathsf{B}|)\,\epsilon$. Indeed, the only difference between these distributions is the wire values of the $m - |\mathsf{B}|$ copies of C'' that correspond to the output shares of the honest parties. Similar to the case that $B = \emptyset$, we can bound the statistical distance using a hybrid argument, moving from $\mathcal{H}$ to Real' by changing the wire values of one of these copies (i.e., a copy of C'' that decodes the share of an honest party) at a time, from simulated to actual wire values.

Second, we claim that $\mathsf{SD}\left(\mathcal{H}, \mathsf{Ideal}'\right) \leq (m+1)\,\epsilon$. We define an additional hybrid distribution $\mathcal{H}'$ which is generated similar to $\mathcal{H}$, except that for $i \notin \mathsf{B}$, the internal wires $\mathcal{W}^{i\prime}_{\mathsf{Dec}}$ of the i'th copy of C'' are generated as $(\mathcal{T}_1, \mathcal{T}_2)\left(\mathcal{W}^{i\prime}_{\mathsf{Out}}, \widetilde{y}^i\right)$, where the $\widetilde{y}^i$'s for $i \notin \mathsf{B}$ are generated as in Ideal (i.e., are uniformly random), and $\bar{y}''$ is generated as the output of Dec on $\left\{\widetilde{y}^i\right\}_{i \in \mathsf{B}}, \left\{y^{i\prime\prime}\right\}_{i \notin \mathsf{B}}$. Then:

- $\mathsf{SD}\left(\mathcal{H}', \mathcal{H}\right) \leq \epsilon m$. Indeed, we can condition both distributions on the value of $\mathcal{W}'$, and let ℓ' be the leakage function (with $\mathcal{W}'$ hard-wired into it) that on input $\left(z^1, \cdots, z^m\right)$, outputs $\ell\left((\mathcal{T}_1, \mathcal{T}_2)\left(\mathcal{W}^{1\prime}_{\mathsf{Out}}, z^1\right), \cdots, (\mathcal{T}_1, \mathcal{T}_2)\left(\mathcal{W}^{m\prime}_{\mathsf{Out}}, z^m\right)\right)$. Then $\ell' \in \mathcal{L}$, $\mathcal{H} \equiv \left(\ell'\left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right), \left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right), y''\right)$, whereas $\mathcal{H}' \equiv \left(\ell'\left(\bar{y}^1, \cdots, \bar{y}^m\right), \left(\bar{y}^1, \cdots, \bar{y}^m\right), \bar{y}''\right)$, where for $i \in \mathsf{B}$, $\bar{y}^i = \widetilde{y}^i$, and for $i \notin \mathsf{B}$, $\bar{y}^i = y^{i\prime\prime}$. We claim first that $\bar{y}'', y''$ are identically distributed. In both distributions, these are obtained as the output of Dec. Since $\left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right), \left(\bar{y}^1, \cdots, \bar{y}^m\right)$ are additive secret shares of y, then if decoding succeeds, Dec outputs $(0, y)$ in both cases. Moreover, the probability that decoding fails is identical in both distributions (in which case Dec outputs $\left(1, 0^k\right)$), since it depends on the outputs of the copies of C'' that correspond to $i \in \mathsf{B}$, and these are identically generated in both distributions. Second, $\left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right), \left(\bar{y}^1, \cdots, \bar{y}^m\right)$ are identically distributed: $\left\{y^{i\prime\prime}\right\}_{i \in \mathsf{B}} \equiv \left\{\bar{y}^i\right\}_{i \in \mathsf{B}}$ since these were generated in the same way, and $\left\{y^{i\prime\prime}\right\}_{i \in \mathsf{B}}, \left\{\bar{y}^i\right\}_{i \in \mathsf{B}}$ are both random shares subject to the constraint that together with the shares of $i \in \mathsf{B}$, they sum to y. Consequently, we can condition both distributions on the outputs of the copies of C'', and the output of Dec. This implies that $\mathsf{SD}\left(\mathcal{H}', \mathcal{H}\right) = \mathsf{SD}\left(\ell'\left(\bar{y}^1, \cdots, \bar{y}^m\right), \ell'\left(y^{1\prime\prime}, \cdots, y^{m\prime\prime}\right)\right) \leq \epsilon m$ by the leakage-resilience of $\mathsf{Enc}^{\mathsf{In}}_{\mathrm{GIMSS}}$, and using the union bound. (The $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that the input encoding is $(\mathcal{L}, \epsilon)$-leakage-indistinguishable, since leakage functions may choose to leak only on the inputs of the compiled circuit.)

- $\mathsf{SD}\left(\mathcal{H}', \mathsf{Ideal}'\right) \leq \epsilon$. Notice first that the outputs $\widetilde{y}^1, \cdots, \widetilde{y}^m$ of the m copies of C'' are identically distributed in both distributions, and consequently so is the output of Dec. Therefore, we can condition both distributions on these values. Moreover, the inputs to the copies of C'' that correspond to $i \in \mathsf{B}$ are also identically distributed in both distributions (these are uniformly random values) and consequently for $i \in \mathsf{B}$, the internal wires $\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}$ of the i'th copy of C'' are also identically distributed (since they were generated in the same way). Therefore, we can further condition on these values.

  Let ℓ' be the leakage function (with $\left(\widetilde{y}^1, \cdots, \widetilde{y}^m\right)$, and $\left\{\widetilde{\mathcal{W}}^i_{\mathsf{Dec}}\right\}_{i \in \mathsf{B}}$ hard-wired into it) that on input wire value $\mathcal{W}''$ for C', extracts the outputs $\left\{\mathcal{W}^{i\prime\prime}_{\mathsf{Out}}\right\}_{i \notin \mathsf{B}}$ of C' that correspond to $i \notin \mathsf{B}$, generates $\mathcal{W}^{i\prime\prime}_{\mathsf{Dec}} = (\mathcal{T}_1, \mathcal{T}_2)\left(\mathcal{W}^{i\prime\prime}_{\mathsf{Out}}, \widetilde{y}^i\right)$ for every $i \notin \mathsf{B}$, and outputs $\ell\left(\mathcal{W}'', \mathcal{W}^{1''}_{\mathsf{Dec}}, \cdots, \mathcal{W}^{m''}_{\mathsf{Dec}}\right)$. Then $\ell' \in \mathcal{L}$, and notice that $\ell'\left(\mathcal{W}'\right) = \mathcal{H}'$ (under our conditioning), whereas $\ell'\left(\widetilde{\mathcal{W}}\right) = \mathsf{Ideal}'$ (under our conditioning). Since $\mathcal{W}', \widetilde{\mathcal{W}}$ are both generated by evaluating C' on different inputs, the $\mathcal{L}$-leakage-resilience of the LRCC of Theorem **??** guarantees that $\mathsf{SD}\left(\ell'\left(\mathcal{W}'\right), \ell'\left(\widetilde{\mathcal{W}}\right)\right) = \mathsf{SD}\left(\mathcal{H}', \mathsf{Ideal}'\right) \leq \epsilon$.

■

We are now ready to prove Theorem **??**.

Proof (Of Theorem **??**). We show that Construction **??** has the required properties. Let $C : (\{0,1\}^n)^m \to \{0,1\}^k$ be a circuit of size s. Then the circuit C^{share} constructed in Step (**??**) of Construction **??** has size $s + 2m$. Moreover, for parameter t, the circuit C^{Dec} constructed in Step (**??**) has size $s^{\mathsf{Dec}} = \widetilde{O}(tk)$, and depth $O(\log t)$ (this follows from the definition of $\mathsf{Dec}^{\mathrm{GIMSS}}_{\mathsf{Out}}$ and $\mathsf{Enc}_{\mathrm{GIMSS}}$). Therefore, $\widehat{C}^{\mathsf{amd}}$ has size $\mathsf{poly}(t,k)$ and error $\mathsf{negl}(t)$ (this follows from Theorem **??**). Therefore, there exists a polynomial $p(t)$ such that for all t, $p(t) \geq \left|\widehat{C}^{\mathsf{amd}}\right|$ (here, we fix k and increase t).

For a parameter t, let $f(t)$ denote the minimal size of $\mathbb{F}$ for which Theorem **??** holds with relation to $p(t)$, and notice that $f(t) = O(t)$. Let $t' \in \mathbb{N}$ be such that for every $t'' \geq t'$, $t'' \leq 0.16t'' \log_2 f(t'') - 1 - \log_2 f(t'')$. We instantiate Construction **??** with security parameter $\hat{t} = \max\{t', t, \sigma \cdot \log m, k\}$ for the LTCC of Theorem **??**, and security parameter $\tilde{t} = \max\{t, \sigma \log m\}$ for the LRCC of Theorem **??**. Then correctness follows directly from Theorem **??**. As for soundness, Theorem **??** guarantees that there exists a negligible function $\epsilon^{\mathrm{DF}}(\hat{t}) = \mathsf{negl}(\hat{t})$, such that Construction **??** is an $\left(\mathcal{L}^t_{\mathrm{BCL}}, \epsilon^{\mathrm{DF}}(\hat{t}), p(\hat{t})\right)$-LTCC with simulator $\mathsf{Sim}^{\mathrm{DF}}$ that outputs view-translation circuits $(\mathcal{T}_1, \mathcal{T}_2)$, where $p(\hat{t}) \geq \left|\widehat{C}^{\mathsf{amd}}\right|$ by the choice of $\hat{t}$. Moreover, Theorem **??** guarantees that when instantiated with security parameter $\tilde{t}$, Construction **??** is an $\left(\mathcal{L}^t_{\mathrm{BCL}}, 2^{-\tilde{t}}, s + 2m\right)$-relaxed LRCC with abort. Therefore, by Theorem **??**, the circuit output by Construction **??** is $\left(\mathcal{L}^t_{\mathrm{BCL}}, \epsilon' + \mathsf{negl}(\hat{t})\right)$-leakage-resilient, where $\epsilon' = (2m+1)\max\left\{\epsilon^{\mathrm{DF}}(\hat{t}), 2^{-\tilde{t}}\right\} = \mathsf{negl}(\sigma)$.

Regarding the complexity of the construction, if C has size s and depth d, then (as noted above) $\left|C^{\mathsf{share}}\right| = s + 2m$, and consequently $|C''| = \widetilde{O}\left(s + 2m + d\tilde{t} + \tilde{t}^2\right)$ (by Theorem **??**). Moreover, $\left|\widehat{C}^{\mathsf{amd}}\right| \leq \mathsf{poly}\left(\hat{t}, k\right)$, and since the compiler of Theorem **??** causes a polynomial blowup, each copy of C'' has size $\mathsf{poly}\left(\hat{t}, k\right)$, so over all the size of the leakage-resilient circuit $\widehat{C}$ obtained from C is $\widetilde{O}\left(s + 2m + d\tilde{t} + \tilde{t}^2\right) + m \cdot \mathsf{poly}\left(\hat{t}, k\right) = \widetilde{O}\left(s + m + d \cdot \max\{t, \sigma \log m\}\right) + m \cdot \mathsf{poly}(t, \sigma, \log m, k) = \widetilde{O}\left(s + d\left(t + \sigma \log m\right)\right) + m \cdot \mathsf{poly}(t, \sigma, \log m, k)$.

As for the input encoding, each party: (1) encodes its input to C'' by running $\mathsf{Enc}^{\mathrm{GIMSS}}_{\mathsf{In}}$, which (by Theorem **??**) takes time $\widetilde{O}\left(n + \tilde{t}\right) = \widetilde{O}\left(n + \max\{t, \sigma \log m\}\right) = \widetilde{O}\left(n + t + \sigma \log m\right)$; and (2) generate the masking inputs for its copy of C'' (there are $O(|C''|)$ such inputs, and each masking input can be generated in polynomial time (in its length)), which takes time $\mathsf{poly}\left(\hat{t}, k\right) = \mathsf{poly}(t, \sigma, \log m, k)$. Therefore, the inputs can be encoded in time prover runs in time $\widetilde{O}(n) + \mathsf{poly}(t, \sigma, \log m, k)$.

As for the output decoding, Dec evaluates $\mathsf{Dec}^{\mathrm{GIMSS}}_{\mathsf{Out}}$ m times, where each evaluation takes time $\widetilde{O}\left(\hat{t}^2 + \hat{t}k\right) = \widetilde{O}\left(k \cdot \max\{t, \sigma \log m, k\}\right) = \widetilde{O}\left(k\left(t + \sigma \log m + k\right)\right)$, then compares the $O(m)$ flags to 0, which takes $O(m)$ time, and finally sums the m secret shares, which takes $O(mk)$ time (since each secret share has length k). Overall, decoding takes $\widetilde{O}\left(m \cdot k\left(t + \sigma \log m + k\right)\right)$ time. ■

Acknowledgments

This work was supported in part by the 2017-2018 Rothschild Postdoctoral Fellowship; by the Warren Center for Network and Data Sciences; by the financial assistance award 70NANB15H328 from the U.S. Department of Commerce, National Institute of Standards and Technology; and by the Defense Advanced Research Project Agency (DARPA) under Contract #FA8650-16-C-7622. The second author was supported in part by NSF-BSFgrant2015782, BSFgrant2012366, ISFgrant1709/14,ERCgrant 742754,DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSFgrants1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipmentgrantfrom Intel, and an Okawa Foundation ResearchGrant. This material is based upon work supported by the DARPA through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the DoD, the NSF, or the U.S. Government. The third author was supported in part by NSF grants CNS-1314722, CNS-1413964, and The Eric and Wendy Schmidt Postdoctoral Grant for Women in Mathematical and Computing Sciences.