Binary AMD Circuits from Secure Multiparty Computation

Daniel Genkin[1,2], Yuval Ishai[1,3], and Mor Weiss[1]

[1] Technion
{danielg3,yuvali,morw}@cs.technion.ac.il
[2] Tel Aviv University
[3] UCLA

Abstract. An *AMD circuit* over a finite field $\mathbb{F}$ is a randomized arithmetic circuit that offers the "best possible protection" against additive attacks. That is, the effect of every additive attack that may blindly add a (possibly different) element of $\mathbb{F}$ to every internal wire of the circuit can be simulated by an ideal attack that applies only to the inputs and outputs.

Genkin et al. (STOC 2014, Crypto 2015) introduced AMD circuits as a means for protecting MPC protocols against active attacks, and showed that every arithmetic circuit C over $\mathbb{F}$ can be transformed into an equivalent AMD circuit of size $O(|C|)$ with $O(1/|\mathbb{F}|)$ simulation error. However, for the case of the binary field $\mathbb{F} = \mathbb{F}_2$, their constructions relied on a tamper-proof output decoder and could only realize a weaker notion of security.

We obtain the first constructions of fully secure binary AMD circuits. Given a boolean circuit C and a statistical security parameter σ, we construct an equivalent binary AMD circuit C' of size $|C| \cdot \mathrm{polylog}(|C|, \sigma)$ (ignoring lower order additive terms) with $2^{-\sigma}$ simulation error. That is, the effect of toggling an arbitrary subset of wires can be simulated by toggling only input and output wires.

Our construction combines in a general way two types of "simple" honest-majority MPC protocols: protocols that only offer security against *passive* adversaries, and protocols that only offer *correctness* against active adversaries. As a corollary, we get a conceptually new technique for constructing active-secure two-party protocols in the OT-hybrid model, and reduce the open question of obtaining such protocols with constant computational overhead to a similar question in these simpler MPC models.

Keywords: Algebraic Manipulation Detection, AMD Circuits, Secure Multiparty Computation.

1 Introduction

In this paper we give the first construction of boolean circuits which are secure against attacks that can toggle an arbitrary subset of the wires, in the sense that every such attack is equivalent to attacking *only* the inputs and outputs of the circuit. We begin with a short overview of the problem and related background.

An *Algebraic Manipulation Detection* (AMD) code [3] over a finite field $\mathbb{F}$ is a randomized coding scheme that offers the best possible protection against *additive* attacks, namely attacks that can blindly add a fixed (but possibly different) element from $\mathbb{F}$ to every entry of the codeword. Since an attacker can destroy all information by adding a random field element to every symbol, the best one can hope for is to *detect* errors with high probability, rather than correct them.

An analogous goal of protecting *computations* against additive attacks was recently considered by Genkin et al. [11]. This goal is captured by the notion of an *AMD circuit*, a randomized arithmetic circuit which offers the best possible protection against additive attacks that may add a (possibly different) field element to every wire. Since the adversary can legitimately attack input and output wires, the best one can hope for is to limit the adversary to these inevitable attacks. That is, in an AMD circuit the effect of every additive attack that may apply to all internal wires in the circuit can be simulated by an ideal attack that applies only to the inputs and outputs. Combining such AMD circuits with a standard AMD code, one can also protect the inputs and outputs by employing small tamper-proof input encoder and output decoder.

The study of AMD circuits in [11] was motivated by the observation that in the simplest information-theoretic MPC protocols from the literature, that were only designed to offer protection against *passive* (i.e., semi-honest) adversaries, the effect of every *active* (malicious) adversary corresponds precisely to an additive attack on the circuit being evaluated. Thus, a useful paradigm for tackling the difficult goal of protecting against active attacks is to apply such a simple passive-secure protocol to an AMD-encoded computation. This paradigm seems quite promising even from a concrete efficiency perspective [13,10].

The main result of [11] is that every arithmetic circuit C over $\mathbb{F}$ can be transformed into an equivalent AMD circuit of size $O(|C|)$ with $O(1/|\mathbb{F}|)$ simulation error. This provides poor security guarantees over small fields, and in fact the construction used to achieve this can be completely broken when applied over the binary field $\mathbb{F} = \mathbb{F}_2$. (The natural approach of using an arithmetic circuit over a large extension field does not apply here, because the computation of field multiplications is also subject to attacks.) For the binary case, an alternative construction from [11] relies on the use of a tamper-proof output decoder and can only realize a weaker notion of security that allows for arbitrary correlations between the input and the event an attack is detected.

The goal of this work is to remedy this state of affairs and provide fully secure AMD circuits over small fields, with a primary focus on the binary case. Binary AMD circuits can be viewed as standard (randomized) boolean circuits (over the full basis) that are subject to arbitrary *toggling* attacks: the adversary may choose to toggle the values of an arbitrary subset of the wires. This seems quite natural even from a pure fault tolerance perspective and can be viewed as a strict generalization of the classical "random noise" fault model considered by von Neumann [20], Dobrushin and Ortyukov [7], and Pippenger [18]. Such a toggling attack model may not be too far from some real-life scenarios like faults introduced by faulty hardware or cosmic radiation.

In the context of applications to MPC, the binary case is important because it enables us to apply the AMD circuits methodology also to natural protocols that are cast in the *OT-hybrid model.* These include the simple passive-secure version of the GMW protocol [12]. In contrast, the MPC applications in [11] for the case of dishonest majority could only apply to arithmetic extensions of the GMW protocol that employ an arithmetic extension of OT denoted by OLE.[4] Replacing OLE by OT is particularly attractive in light of efficient OT extension techniques [14,17] that do not apply to OLE.

We obtain the first constructions of fully secure binary AMD circuits. Given a boolean circuit C and a statistical security parameter σ, we construct an equivalent binary AMD circuit $\widehat{\mathsf{C}}$ of size $|C|\cdot\mathrm{polylog}(|C|,\sigma)$ (ignoring lower order additive terms) with $2^{-\sigma}$ simulation error. That is, the effect of toggling an arbitrary subset of wires can be simulated by toggling only input and output wires.

Our construction combines in a general way two types of "simple" honest-majority MPC protocols: protocols that only offer security against *passive* adversaries, and protocols that only offer *correctness* against active adversaries. It proceeds according to the following steps. First, we use the correct-only MPC protocol to convert a relatively simple AMD circuit that provides only *constant correctness* (i.e., any "potentially harmful" attack is detected with some positive probability) into one that offers full correctness (i.e., attacks are detected except with $2^{-\sigma}$ probability). However, this notion of correctness is not enough, mainly because it does not rule out correlations between the input and the event an attack is detected. We eliminate such correlations generically by distributing the computation using a passive-secure MPC protocol. The analysis of this step crucially relies on a recent lemma due to Bogdanov et al. [1] that uses the degree of approximating the OR function by real-valued polynomials to upper bound its best-case advantage in distinguishing between two distributions that are t-wise indistinguishable.

As a byproduct, we get a conceptually different technique for constructing active-secure two-party protocols in the OT-hybrid model from these simpler building blocks. This technique is appealing because in a sense it counters the common wisdom that "security" is more than a combination of "correctness" and "secrecy." Indeed, our construction shows a *general* way to obtain full security (for MPC protocols in the OT-hybrid model) by only combining one MPC protocol that guarantees *correctness* and another that only guarantees *secrecy*, namely security in the presence of passive attacks. Moreover, the "correct-only MPC" component can be instantiated by a trivial protocol in which each party performs the entire computation locally. (To get the asymptotic efficiency mentioned above, we need to apply more sophisticated correct-only MPC protocols that offer better efficiency.) This can be compared with the IPS compiler [16], which also provides a general way of obtaining active-secure protocols in the

[4] An Oblivious Linear-function Evaluation (OLE) over a field $\mathbb{F}$ takes a field element $x \in \mathbb{F}$ from Receiver and a pair $(a,b) \in \mathbb{F}^2$ from Sender and delivers $ax+b$ to Receiver. In the case of binary fields, OLE can be realized via a single call to standard (bit-)OT.

OT-hybrid model, but requires an honest-majority MPC protocol that provides *active security* (which is strictly stronger than relying on active correctness and passive security).

In addition to its conceptual appeal, our new methodology also sheds new light on an intriguing open question about the complexity of secure computation [15]: Are there active-secure two-party protocols that achieve *constant computational overhead*? In other words, does the asymptotic multiplicative cost of protecting against active adversaries have to grow with the level of security? This question is open even when allowing a trusted source of correlated randomness, and in particular it is open in the OT-hybrid model. The best known protocols [6] have a polylogarithmic overhead in the security parameter (a result that we can match using binary AMD circuits). Our work reduces this question to the same open question in arguably simpler models. Indeed, while our construction involves some additional ad-hoc components (on top of the two types of MPC protocols discussed above) the additional cost they incur depends only on the input and output sizes, and not on the size of the computation. Furthermore, our construction also employs AMD codes to encode the entire protocol transcript, but these can be implemented with constant computational overhead (see Claim 18 and Corollary 1 in Section 6).

1.1 Our Results and Techniques

We now provide more details about our results, and the underlying techniques (summarized in Figure 1 below). We begin by defining the notion of *additive correctness*, which allows the evaluation of a function $f : \mathbb{F}^n \to \mathbb{F}^k$ in the presence of an additive attack[5] on the circuit computing f.

Definition 1 (Additive correctness; cf. full version of [11], Definition 4.1). *Let $\epsilon > 0$. We say that a randomized circuit $\widehat{\mathsf{C}} : \mathbb{F}^n \to \mathbb{F}^t \times \mathbb{F}^k$ is an* ϵ-additively-correct implementation *of a function $f : \mathbb{F}^n \to \mathbb{F}^k$ if the following holds:*

- ***Completeness.*** *For all $\mathbf{x} \in \mathbb{F}^n$ it holds that $\Pr[\widehat{\mathsf{C}}(\mathbf{x}) = (0^t, f(\mathbf{x}))] = 1$.*
- ***Additive correctness.*** *For any additive attack $\mathbf{A}$ there exists $\mathbf{a}^{\mathsf{in}} \in \mathbb{F}^n$, and $\mathbf{a}^{\mathsf{out}} \in \mathbb{F}^k$, such that for every input $\mathbf{x}$ it holds that $\Pr[\widehat{\mathsf{C}}^{\mathbf{A}}(\mathbf{x}) \notin \mathsf{ERR} \cup \{(0^t, f(\mathbf{x} + \mathbf{a}^{\mathsf{in}}) + \mathbf{a}^{\mathsf{out}})\}] \leq \epsilon$, where $C^{\mathbf{A}}$ is the circuit obtained by subjecting C to the additive attack $\mathbf{A}$, and $\mathsf{ERR} = (\mathbb{F}^t \setminus \{0^t\}) \times \mathbb{F}^k$.*

We say that $\widehat{\mathsf{C}}$ is an ϵ-additively-correct implementation of a circuit C if $\widehat{\mathsf{C}}$ is an ϵ-additively-correct implementation of the function f_C computed by C.

Previous works [11,10] constructed additively correct implementations for arithmetic circuits over any finite field $\mathbb{F}$, with constant overhead, and $\epsilon = O(1/|\mathbb{F}|)$. In particular, for $\mathbb{F} = \mathbb{F}_2$ the error is constant.

1.1.1 Correctness Amplification via Correct-Only MPC

For any function f, and security parameter σ, we show the first $2^{-\sigma}$-additively-correct implementation of f, with polylogarithmic blowup:

[5] For a formal definition of additive attacks, see Definition 3.

Theorem 1 (Cf. Theorem 11). *For any depth-d arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and any security parameter σ, there exists a $2^{-\sigma}$-additively-correct implementation $\widehat{\mathsf{C}}$ of C, where $|\widehat{\mathsf{C}}| = |C| \cdot polylog(|C|, \sigma) + poly(n, k, d, \sigma)$.*

To prove Theorem 1, we present a general method of amplifying additive correctness based on "correct-only" MPC protocols. Such protocols enable a single client, aided by m servers, to evaluate an arithmetic circuit C on its input, while guaranteeing correctness of the computation in the presence of an active adversary that corrupts a constant fraction of the servers. Moreover, the only interaction between the client and servers is in the first and last rounds.

More specifically, for m servers, and some constant c, let π be a d-round cm-correct MPC protocol, namely correctness holds even if cm servers are corrupted. Let $\mathsf{InpEnc}, \mathsf{OutDec}$ denote the functions used by the client in the first and last rounds (respectively) to compute its messages to the servers, and its output (respectively). Let $\mathsf{NextMSG}$ denote the function used by the servers to compute their messages in each round of the protocol. The naive approach towards implementing the circuit $\widehat{\mathsf{C}}$ using π is to implement every sub-circuit (namely, each of $\mathsf{NextMSG}$, InpEnc, and OutDec) using an ϵ-additively-correct implementation. This naive approach fails because an additive attack may influence the computation of *all* $\mathsf{NextMSG}$ *functions*, which corresponds to *actively corrupting all servers in* π, whereas the correctness of the protocol only holds when at most cm servers are corrupted. Consequently, additive attacks on $\widehat{\mathsf{C}}$ can be divided into two categories:

1. **"Small" Attacks.** The sub-circuits of $\widehat{\mathsf{C}}$ that these attacks influence correspond to at most cm servers of π, so by the cm-correctness of π, such attacks cannot affect the output.
2. **"Large" Attacks.** The sub-circuits of $\widehat{\mathsf{C}}$ that these attacks affect correspond to *more* than cm servers of π. Since each sub-circuit (computing $\mathsf{NextMSG}$) is implemented using an ϵ-additively-correct implementation, then except with probability ϵ^{cm} at least one of these attacks is detected, or their effect on the computations in the sub-circuits is equivalent to additive attacks on the inputs and outputs of the sub-circuits.

Additionally, we notice that any additive attack on π consists of sub-attacks of one of three types:

1. **Attacks on communication channels.** These attacks only affect the messages that parties receive in π, but do not modify the $\mathsf{NextMSG}$ functions. By encoding all messages sent in the protocol using an AMD encoding scheme (and altering $\mathsf{InpEnc}, \mathsf{NextMSG}, \mathsf{OutDec}$ to operate on AMD codewords) we can guarantee that such attacks are detected with high probability.
2. **Attacks on $\mathsf{NextMSG}$ functions.** These attacks arbitrary modify the $\mathsf{NextMSG}$ function of the corresponding server, but (as noted above) can be protected against by replacing all $\mathsf{NextMSG}$ functions with their ϵ-additively-correct implementations.
3. **Attacks on client functions.** Since π is correct only as long as the client is honest, such attacks may arbitrarily affect the outputs. Therefore, to guarantee that such attacks are detected except with negligible probability, InpEnc

and OutDec should be replaced with their $2^{-\sigma}$-additively-correct implementation. The crucial point here is that since $|\mathsf{InpEnc}| + |\mathsf{OutDec}|$ is polynomial in the inputs and outputs, but otherwise independent of $|C|$, then *any* efficient $2^{-\sigma}$-additively-correct implementation will do, and the resultant overhead would still be $\mathrm{polylog}(m\,|C|)$. (We show an example of a $2^{-\sigma}$-additively-correct implementation in Appendix A.)

Consequently, we implement the circuit $\widehat{\mathsf{C}}$ using π as follows. We first replace the $\mathsf{NextMSG}$ functions of π with the functions $\mathsf{NextMSG}'$ that operate on AMD codewords, and replace $\mathsf{NextMSG}'$ with its ϵ-additively correct implementation, $\widehat{\mathsf{NextMSG}'}$, such that $\left|\widehat{\mathsf{NextMSG}'}\right| = O\left(\left|\mathsf{NextMSG}'\right|\right)$, and ϵ is constant. Additionally, we replace InpEnc (resp., OutDec) with the function InpEnc' (resp., OutDec') which outputs (resp., takes as input) AMD codewords, and replace $\mathsf{InpEnc}', \mathsf{OutDec}'$ with their $2^{-\sigma}$-additively correct implementations $\widehat{\mathsf{InpEnc}}, \widehat{\mathsf{OutDec}}$. Thus, $|\widehat{\mathsf{C}}| = |\widehat{\mathsf{InpEnc}}| + |\widehat{\mathsf{OutDec}}| + \sum_{i=1}^{m}\sum_{j=1}^{d}|\widehat{\mathsf{NextMSG}}_i^j|$. We use an efficient correct-only MPC protocol π (e.g., a slightly simplified version of [6]) to guarantee that when $m = \sigma$, the multiplicative computational overhead is only $\mathrm{polylog}(\sigma, |C|)$. (Since we would like the overhead to be sublinear in σ, we cannot use a trivial correct-only MPC protocol for evaluating C on input x.) For this choice of π, $|\mathsf{InpEnc}| + |\mathsf{OutDec}| = \mathrm{poly}(n,k)$, so $|\widehat{\mathsf{InpEnc}}| + |\widehat{\mathsf{OutDec}}| = \mathrm{poly}(n,k)$. Similarly, $\sum_{i=1}^{\sigma}\sum_{j=1}^{d}|\mathsf{NextMSG}_i^j| = |C| \cdot \mathrm{polylog}(|C|,\sigma) + \mathrm{poly}(n,k,d,\sigma)$, so $\sum_{i=1}^{\sigma}\sum_{j=1}^{d}|\widehat{\mathsf{NextMSG}}_i^j| = |C| \cdot \mathrm{polylog}(|C|,\sigma) + \mathrm{poly}(n,k,d,\sigma)$. (See section 4 for a more complete discussion.)

1.1.2 From Correctness to Security via Passive-Secure MPC

Additive correctness (as guaranteed by Theorem 1) does not rule out the possibility that the probability of ERR (due to set flags) is correlated with the inputs of $\widehat{\mathsf{C}}$. Thus, additive attacks on additively-correct circuits may leak information about the inputs to $\widehat{\mathsf{C}}$, making additive correctness insufficient for applications to secure multiparty computation (as described in, e.g., [11]) that require that no such correlations exist. This stronger property is achieved by the following *additive security* property which, intuitively, guarantees that any additive attack on $\widehat{\mathsf{C}}$ is equivalent (up to a small statistical distance) to an additive attack on the inputs and outputs of the function that $\widehat{\mathsf{C}}$ computes. Formally,

Definition 2 (Additively-secure implementation). *Let $\epsilon > 0$. We say that 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 $\mathcal{A}$ there exist $\mathbf{a}^{\mathsf{in}} \in \mathbb{F}^n$, and a distribution $\mathcal{A}^{\mathsf{Out}}$ over $\mathbb{F}^k$, such that for every $\mathbf{x} \in \mathbb{F}^n$, $\mathsf{SD}(C^{\mathbf{A}}\left(\mathbf{x}\right), f\left(\mathbf{x} + \mathbf{a}^{\mathsf{in}}\right) + \mathcal{A}^{\mathsf{out}}) \leq \epsilon$.*

As in the case of additive correctness, previous works [11,10] constructed additively-secure implementations for arithmetic circuits over any finite field $\mathbb{F}$, with constant overhead, and $\epsilon = O\left(1/|\mathbb{F}|\right)$. Unfortunately, their results and

techniques are of little use in the binary case, since the error is too large. We present the first additively-secure circuits with negligible error probability over the binary field. Formally:

Theorem 2 (Cf. Theorem 14). *For any depth-d arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and security parameter σ, there exists a $2^{-\sigma}$-additively-secure implementation $\widehat{\mathsf{C}}$ of C, where $|\widehat{\mathsf{C}}| = |C| \cdot polylog(|C|, \sigma) + poly(n, k, d, \sigma)$.*

As in Section 1.1.1, the high-level idea is to implement C using an m-party protocol (in the standard model, namely not in the server-client model), where the functions computed by the parties are replaced with additively-correct implementations that operate over AMD encodings. However, since our main concern now is privacy, and not correctness, we use *passive-secure* protocols which *only guarantee privacy* against a constant fraction c of passively-corrupted parties. This privacy guarantee allows us to decouple the probability of ERR of the additively *correct* circuits from their inputs, resulting in additively *secure* circuits.

More specifically, the input of the circuit C is shared between the parties using an additive secret-sharing, and the d-round passive-secure protocol π computes the functionality that reconstructs the input from the shares, evaluates C, and outputs an additive secret-sharing of the output. The privacy property of π, together with the secrecy property of the secret-sharing scheme, guarantee that the joint view of a constant fraction of passively-corrupted parties reveals no information about the inputs, or outputs, of the computation. As in Section 1.1.1, $\widehat{\mathsf{C}}$ is obtained from π by first replacing all $\mathsf{NextMSG}$ functions with the functions $\mathsf{NextMSG}'$ that operate on AMD encodings, and then implementing each $\mathsf{NextMSG}'$ using a $2^{-\sigma}$-additively-correct implementation $\widehat{\mathsf{NextMSG}'}$ with constant overhead (such as the one from Theorem 1). As $\widehat{\mathsf{C}}$ should emulate C (rather than output a secret sharing of the output of C), the output is reconstructed from the outputs of the parties in π by summing their shares, and is then combined with the flags generated by *all* the additively-correct implementations, such that if *any* of the flags were set then the output of $\widehat{\mathsf{C}}$ is random.

Using a union-bound over the additive-correctness property of the additively-correct implementations, except with probability at most $|C| \cdot 2^{-\sigma}$ any additive attack on the $\widehat{\mathsf{NextMSG}'}$ functions either sets a flag, or is equivalent to an attack on the inputs and outputs of $\mathsf{NextMSG}'$. Except for the inputs, and output, of $\widehat{\mathsf{C}}$, the inputs and outputs of the $\mathsf{NextMSG}'$ functions are protected by the AMD encoding scheme, so by the additive soundness of the AMD encoding scheme, any attack (except for an attack on the inputs, and output, of $\widehat{\mathsf{C}}$) will set a flag with overwhelming probability. Thus, the only additive attacks that do not set a flag (with overwhelming probability) are attacks on the inputs and outputs of $\widehat{\mathsf{C}}$, which are equivalent to attacks on the inputs and outputs of C. Thus, with overwhelming probability the execution of π is correct *even in the presence of additive attacks*.

It remains to show that the probability of setting a flag in $\widehat{\mathsf{C}}$, thus causing the output to be random, is input independent. We use the fact that the probability

that a subset of $\widehat{\mathsf{NextMSG}'}$ implementations set their flags depends only on their joint inputs and outputs, and distinguish between two types of attacks.

1. **"Small" attacks.** These attacks attempt to corrupt less than cm parties. Therefore, the probability that a flag is set depends only on the inputs and outputs of these parties which, by the privacy of π, and the secrecy of the secret-sharing scheme, is independent of the inputs of $\widehat{\mathsf{C}}$.
2. **"Large" attacks.** These attacks attempt to corrupt more than cm parties, and so we can no longer use the privacy of π. However, notice that in this case the output of $\widehat{\mathsf{C}}$ is random if and only if at least one additively-correct implementation set a flag (regardless of the identity or number of flags that were set). That is, the output is random if and only if the OR of the flags is 1. Using a recent lemma of [1] (stated as Lemma 1 below), the correlation of the OR with the input is negligible, because the OR is computed over a large fraction of the flags.

As for the size of $\widehat{\mathsf{C}}$, notice that $|\widehat{\mathsf{C}}| = \sum_{i=1}^{m}\sum_{j=1}^{d}|\widehat{\mathsf{NextMSG}_i^j}|$. To obtain the small overhead guaranteed by Theorem 2, we use a cm private (for some constant $c > 0$), m-party protocol of [6] in which the total circuit size of all the $\mathsf{NextMSG}$ functions is $|C|\cdot\mathrm{polylog}(|C|,m) + \mathrm{poly}(m,n,k,d,\log|C|)$. Setting $m = \mathrm{poly}(\sigma)$, $\sum_{i=1}^{\sigma}\sum_{j=1}^{d}|\mathsf{NextMSG}_i^j| = |C|\cdot\mathrm{polylog}(|C|,\sigma)+\mathrm{poly}(n,k,d,\sigma)$, and so if all the $\widehat{\mathsf{NextMSG}_i^j}$ are generated using Theorem 1, $|\widehat{\mathsf{C}}| = |C|\cdot\mathrm{polylog}(|C|,\sigma)+\mathrm{poly}(n,k,d,\sigma)$. (See Section 5 for a more detailed analysis.)

1.2 On the Difference Between Additive Correctness and Additive Security

As noted in Section 1.1.2, Definition 1 is weaker than Definition 2. In particular, the correctness guarantee of Definition 1 is insufficient for many MPC applications, since the probability of ERR (due to set flags) might be correlated with the inputs, and consequently reveal information regarding the inputs of $\widehat{\mathsf{C}}$. As we now show, such correlations exist in many natural constructions of additively correct implementations (and, in particular, in all additivity correct constructions discussed in this paper as well as the constructions in [11,10]).

As a typical example of correlations between inputs and the probability of ERR created by additive attacks, consider the simpler case of an AMD code. Specifically, consider the code which encodes a field element $x \in \mathbb{F}$ as $(x, v_1, \cdots, v_\sigma, r_1, \cdots, r_\sigma)$, where $v_1, \cdots, v_\sigma \in_R \mathbb{F}$ are uniformly random, and $r_i = v_i \cdot x$ for all $1 \leq i \leq \sigma$. To decode $(x, v_1, \cdots, v_\sigma, r_1, \cdots, r_\sigma)$, the decoder verifies that $x \cdot v_i = r_i$ for all $1 \leq i \leq \sigma$. Consider the additive attack that adds the same arbitrary constant $\delta \neq 0$ to all the v_i's. If $x = 0$ then $r_i = 0$ for every $1 \leq i \leq \sigma$, thus the test $0 \cdot (v_i + \delta) = 0$ passes for all i, and decoding succeeds. However, if $x \neq 0$ then every $x \cdot v_i = r_i$ test fails except with probability $1/|\mathbb{F}|$. Since decoding succeeds only if *all* tests succeed, decoding fails in this case with probability at least $1 - 1/|\mathbb{F}|^\sigma$.

Overall, this attack leaks information regarding the value of x because if $x = 0$ then the decoder aborts with probability zero, whereas if $x \neq 0$ then the

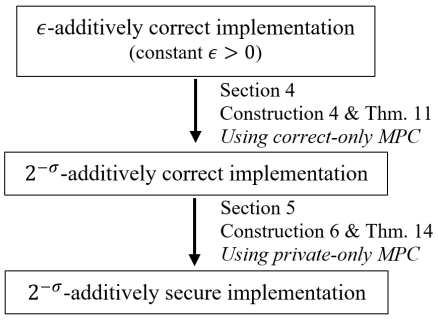

Fig. 1: Additive security from weak additive correctness (both steps use AMD codes)

decoder aborts with probability almost 1. Similar attacks apply to all additively-correct constructions presented in this paper, thus requiring the transformation of Section 5.

2 Preliminaries

In the following, $\mathbb{F}$ will denote a finite field, 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. 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, Y.

The following lemma regarding k-wise indistinguishable distributions over $\{0,1\}^n$ will be used to construct additively-secure circuits.

Lemma 1 (Cf. Claim 3.9 in [1]). *Let n, k be positive integers, and $\mathcal{X}$, $\mathcal{Y}$ be k-wise indistinguishable distributions over $\{0,1\}^n$. Then*

$$|\Pr[(x_1, \cdots, x_n) \leftarrow \mathcal{X} : \vee_{i=1}^{n} x_i = 1] - \Pr[(y_1, \cdots, y_n) \leftarrow \mathcal{Y} : \vee_{i=1}^{n} y_i = 1]| \leq 2^{-\Omega(k/\sqrt{n})}.$$

Additive Attacks. We follow the terminology of [10].

Definition 3 (Additive attack). *An additive attack $\mathbf{A}$ on a circuit C is a fixed vector of field elements which is independent from the inputs and internal values of C. $\mathbf{A}$ contains an entry for every wire, and every output gate, of C, and has the following effect on the evaluation of the circuit. For every wire ω connecting gates a and b in C, the entry of $\mathbf{A}$ that corresponds to ω is added to the output of a, and the computation of the gate b uses the derived value. Similarly, for every output gate o, the entry of $\mathbf{A}$ that corresponds to the wire in the output of o is added to the value of this output.*

Notation 3 *For a (possibly randomized) circuit C and for a gate g of C, we denote by $g_{\mathbf{x}}$ the distribution of the output value of g (defined in a natural way) when C is evaluated on an input $\mathbf{x}$.*

Notation 4 *Let C be a (possibly randomized) circuit, and $\mathbf{A}$ be an additive attack on C. We denote by $\mathbf{A}_{c,c'}$ the attack $\mathbf{A}$ restricted to the wire connecting the gates c, c' of C. Similarly we denote by $\mathbf{A}^{\mathsf{out}}$ the restriction of $\mathbf{A}$ to all the outputs of C.*[6]

Encoding schemes. An encoding scheme E over a set Σ of symbols (called "the alphabet") 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 $\hat{n} = \hat{n}(n)$; 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, $\Pr[\mathsf{Dec}(\mathsf{Enc}(x)) = x] = 1$ for every $x \in \Sigma^n$. We will assume that when $n > 1$, Enc encodes every symbol of x separately, and in particular $\hat{n}(n) = n \cdot \hat{n}(1)$.

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., $\hat{n} = \hat{n}(n,t)$, and for every t the resultant scheme is an encoding scheme (in particular, for every $x \in \Sigma^n$ *and every* $t \in \mathbb{N}$, $\Pr[\mathsf{Dec}(\mathsf{Enc}(x, 1^t), 1^t) = x] = 1$). We call such schemes *parameterized*. We will only consider parameterized encoding schemes, and therefore when we say "encoding scheme" we mean a *parameterized* encoding scheme.

Algebraic Manipulation Detection (AMD) Encoding Schemes. Informally, AMD encoding schemes over a finite field $\mathbb{F}$ guarantee that additive attacks on codewords are detected by the decoder with some non-zero probability:

Definition 4 (AMD encoding scheme, [3,11]). *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 $\mathbf{x} \in \mathbb{F}^n$,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(\mathbf{x}, 1^t), 1^t) = (0, \mathbf{x})] = 1$.
- ***Additive soundness.*** *For every $0^{\hat{n}(n,t)} \neq \mathbf{a} \in \mathbb{F}^{\hat{n}(n,t)}$, and every $\mathbf{x} \in \mathbb{F}^n$,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(\mathbf{x}, 1^t) + \mathbf{a}, 1^t) \notin \mathsf{ERR}] \leq \epsilon(n,t)$ *where $\mathsf{ERR} = (\mathbb{F} \setminus \{0\}) \times \mathbb{F}^n$, and the probability is over the randomness of* Enc.

Remark 1 *It will sometime be useful to represent* $(\mathsf{Enc}, \mathsf{Dec})$ *as families of arithmetic circuits (instead of polynomial-time algorithms) that are parameterized by the security parameter t. That is, $(\mathsf{Enc} = \{\mathsf{Enc}_n\}, \mathsf{Dec} = \{\mathsf{Dec}_n\})$ are families of arithmetic circuits over $\mathbb{F}$, where $\mathsf{Enc}_n : \mathbb{F}^n \to \mathbb{F}^{\hat{n}}$ is randomized, and $\mathsf{Dec}_n : \mathbb{F}^{\hat{n}} \to \mathbb{F} \times \mathbb{F}^n$ is deterministic. (Here, the security parameter t is "hardwired" into the circuits.) Somewhat abusing notation, we use* $\mathsf{Enc}, \mathsf{Dec}$ *to denote*

[6] Note that $\mathbf{A}_{c,c'}$ is a single field element whereas $\mathbf{A}^{\mathsf{out}}$ is a vector of field elements.

both the families *of circuits, and the circuits* $\mathsf{Enc}_n, \mathsf{Dec}_n$ *for a* specific n*, omitting the subscript (when* n *is clear from the context).*

We will sometimes need AMD codes with a stronger *robustness* guarantee which, roughly speaking, guarantees additive correctness *even in the presence of additive attacks on the internal wires of the encoding procedure*, where the ideal additive attack on the output is independent of the additive attack:

Definition 5 (Robust AMD encoding schemes). *Let* $\mathbb{F}$ *be a finite field,* $n \in \mathbb{N}$ *be an input length parameter,* $\hat{n} \in \mathbb{N}$ *be an output length parameter,* $t \in \mathbb{N}$ *be a security parameter, and* $\epsilon(n,t) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$*. We say that an encoding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *over* $\mathbb{F}$ *is an* $(n, \hat{n}, t, \epsilon(n,t))$-robust AMD encoding scheme*, if it is an* $(n, t, \epsilon(n,t))$*-AMD encoding scheme in which the additive soundness property is replaced with the following* additive robustness *property. Let* $\mathsf{Enc} : \mathbb{F}^n \to \mathbb{F}^{\hat{n}}$*,* $\mathsf{Dec} : \mathbb{F}^{\hat{n}} \to \mathbb{F} \times \mathbb{F}^n$*, then for any additive attack* $\mathbf{A}$ *on* Enc *there exists an ideal attack* $\mathbf{a}^{\mathsf{in}} \in \mathbb{F}^n$ *such that for any* $\mathbf{b} \in \mathbb{F}^{\hat{n}}$*, and any* $\mathbf{x} \in \mathbb{F}^n$*, it holds that* $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}^{\mathbf{A}}(\mathbf{x}, 1^t) + \mathbf{b}, 1^t\right) \notin \mathsf{ERR} \cup \left\{\left(0, \mathbf{x} + \mathbf{a}^{\mathsf{in}}\right)\right\}\right] \leq \epsilon$*, where* $\mathsf{ERR} = (\mathbb{F} \setminus \{0\}) \times \mathbb{F}^n$*, and the probability is over the randomness of* Enc*.*

Secure Multiparty Computation. We recall a few standard definitions that will be used in subsequent sections.

We view an MPC protocol π as a collection of $\mathsf{NextMSG}$ functions. The protocol proceeds in rounds, where in round j, the description of π contains a *next message function* $\mathsf{NextMSG}_i^j$ *of round* j *for party* P_i, defined as follows. $\mathsf{NextMSG}_i^j$ takes as input all the messages m_i^{j-1} that P_i received before round j, its input x_i, and its randomness r_i; and outputs the messages that P_i sends in round j. If j is the last round of π, then for every party P_i, $\mathsf{NextMSG}_i^j$ outputs the output of P_i in π.

The client-server model. The client-server model (see [2,4,5] for a more detailed discussion) is a refinement of the standard MPC model in which each party has one of two possible roles: *clients* hold inputs and receive outputs; and *servers* have no inputs and receive no outputs, but may participate in the computation. Notice that every protocol in the client-server model can be converted to a protocol in the standard MPC model by asking every party to emulate a single server and a single client (assuming the protocol has the same number of clients and servers). See Figure 2.

In the following, we assume that the protocol consists of a single input client, a single output client, and m_S servers. We call such protocols m_S-server protocols. We use the simulation-based paradigm, and say that a protocol π in the client-server model is (s,ϵ)*-secure* ((s,ϵ)-private) if it is secure (up to distance ϵ) against all active (passive) adversaries corrupting at most s servers, and no clients. We assume that the description of a protocol in the client-server model consists of the following:

1. **Input Encoding.** A description of a function InpEnc whose input is the input of the input client, and whose output is the messages that the input client sends to the servers.

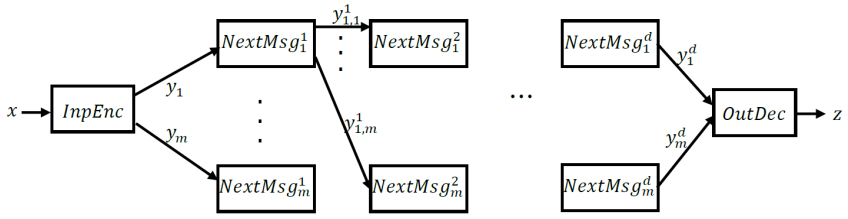

Fig. 2: MPC protocol with a single client and m servers

2. **Circuit Evaluation.** For every server S_i, and every round j, a description of a function $\mathsf{NextMSG}_i^j$ which specifies the messages that S_i sends to all the servers (to the output client) in round j (in the last round).
3. **Output Decoding.** A description of a function OutDec whose input is the messages sent to the output client (from the servers) in the last round, and whose output is the output of π.

We will use a relaxed notion of security, which we call *correct-only* MPC. Intuitively, it guarantees output correctness even in the presence of an active adversary that corrupts a "small" subset of the servers. This notion relaxes the standard security notion because it does not guarantee input privacy. We formalize correct-only MPC as follows, where for a protocol π, and an adversary Adv, $\pi^{\mathsf{Adv}}(\mathbf{x})$ denotes the outputs (of the clients) in an execution of π on inputs $\mathbf{x}$ in the presence of Adv.

Definition 6. *Let $f : X \to Y$ be a function, and π be a single client, m_S-server protocol. We say that π (t,ϵ)-*correctly computes *f if for every active adversary Adv corrupting a set $T, |T| \leq t$ of servers, and every client input $\mathbf{x} \in X$, it holds that $\Pr\left[f(\mathbf{x}) \neq \pi^{\mathsf{Adv}}(\mathbf{x})\right] \leq \epsilon$. We say that π t-*correctly computes *f if it (t,ϵ)-correctly computes f for $\epsilon = 0$.*

Remark 2 *Notice that any protocol π for t-correctly computing f in the client-server model can be assumed to be deterministic without loss of generality. This is because the adversary Adv has no effect on the randomness used by the input clients. Therefore, any π can be de-randomized by fixing its randomness to some arbitrary value.*

Next, we describe a simple replication-based m-server protocol for $(\lceil m/2 \rceil - 1)$-correctly computing a function f.

Theorem 5. *Let $\mathbb{F}$ be a finite field. Then for every arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and $m \in \mathbb{N}$, there exists an m-server protocol for $(\lceil m/2 \rceil - 1)$-correctly computing f. Moreover, the computational complexity (in field operations) of π is $|C| \cdot m$.*

Proof. The input client replicates the input $\mathbf{x}$ among all the servers, who locally compute $\mathbf{z}_i \leftarrow C(\mathbf{x})$ and send $\mathbf{z}_i$ to the output client, who outputs $\text{maj}\{\mathbf{z}_1, \cdots, \mathbf{z}_m\}$. □

We will use the following theorem regarding the existence of correct-only MPC protocols.

Theorem 6 (Implicit in [6]). *Let σ be a security parameter, $m \in \mathbb{N}$, $\mathbb{F}$ be a finite field, and $C : \mathbb{F}^n \to \mathbb{F}^k$ be a depth-d arithmetic circuit. Then there exists a d-round, m-server protocol π that $m/10$-correctly computes C, where:*

- *The total circuit size of the input encoding function* InpEnc*, and the output decoding function* OutDec*, is poly(n, k, m).*
- *The total circuit size of all the* $\mathsf{NextMSG}$ *functions is $|C| \cdot$ polylog$(|C|, \sigma) +$ poly$(m, d, n, k, \log|C|)$.*
- *In each round of π, the messages sent by each party contain in total at most poly$(n, k, \log|C|)$ field elements.*

3 Circuit Transformations

In this section we describe a few circuit transformations which will be used in Sections 4 and 5 to construct additively-correct and additively-secure circuits. At a high level, these transformations replace a given circuit C over field $\mathbb{F}$ with a new circuit that operates on AMD encodings. We first describe a randomized gadget that combines and amplifies error flags. This gadget will be used in the following constructions to combine error flags obtained from AMD decoding of several codewords.

Construction 1 *Let $n_f \in \mathbb{N}$ be an input length parameter, and $\sigma \in \mathbb{N}$ be a security parameter. The* flag combining *gadget $\mathcal{F}_{\mathsf{comb}} : \mathbb{F}^{n_f} \to \mathbb{F}^{\sigma}$, on input $f_1, \cdots, f_{n_f} \in \mathbb{F}$, operates as follows.*

1. *Generates n_f random vectors $\mathbf{r}_1, \cdots, \mathbf{r}_{n_f} \in_R \mathbb{F}^{\sigma}$.*
2. *Outputs $\mathbf{f} \leftarrow \sum_{i=1}^{n_f} \mathbf{r}_i \cdot f_i$.*

Observation 7 *If $(f_1, \cdots, f_{n_f}) \neq \mathbf{0}$ then $\mathbb{F}_{\mathsf{comb}}(f_1, \cdots, f_{n_f}) \neq \mathbf{0}$ except with probability at most $2^{-\sigma}$.*

Next, we describe a circuit transformation $\mathcal{T}_{\mathsf{inter}}$ that will be used to replace intermediate rounds in secure protocols. Intuitively, given a circuit C, the transformed circuit $\mathcal{T}_{\mathsf{inter}}(C)$ takes AMD encodings of the inputs of C, decodes them, uses the flag combining gadget $\mathcal{F}_{\mathsf{comb}}$ of Construction 1 to combine the error flags generated during decoding, evaluates the circuit C, and outputs AMD encodings of the output, concatenated with the combined error flag.

Construction 2 *Given a circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and an AMD encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ that outputs encodings of length $\hat{n}(n)$, the circuit $\mathcal{T}_{\mathsf{inter}}(C) : \mathbb{F}^{\hat{n}(n)} \to \mathbb{F}^{\sigma} \times \mathbb{F}^{\hat{n}(k)}$, on input $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$, operates as follows.*

1. *For every $1 \leq i \leq n$, computes $(f_i, \mathbf{x}'_i) \leftarrow \mathsf{Dec}(\mathbf{x}_i)$*

2. *Computes* $(y_1, \cdots, y_k) \leftarrow C(\mathbf{x}'_1, \cdots, \mathbf{x}'_n)$.
3. *Computes* $\mathbf{f} \leftarrow \mathcal{F}_{\mathsf{comb}}(f_1, \cdots, f_n)$.
4. *Outputs* $(\mathbf{f}, \mathsf{Enc}(y_1), \cdots, \mathsf{Enc}(y_k))$.

Finally, we describe a circuit transformation $\mathcal{T}_{\mathsf{fin}}$ that will be used to replace the output generation rounds. This transformation differs from the transformation $\mathcal{T}_{\mathsf{inter}}$ of Construction 2 only in the fact that it does not encode the outputs.

Construction 3 *Given a circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and an AMD encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ that outputs encodings of length $\hat{n}(n)$, the circuit $\mathcal{T}_{\mathsf{fin}}(C) : \mathbb{F}^{\hat{n}(n)} \to \mathbb{F}^\sigma \times \mathbb{F}^k$, on input $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$, operates as follows.*

1. *Performs Steps 1- 3 of Construction 2, and let $(y_1, \cdots, y_k), \mathbf{f}$ denote the outputs of Steps 2 and 3, respectively.*
2. *Outputs $(\mathbf{f}, y_1, \cdots, y_k)$.*

Finally, we will use the following notation.

Notation 8 *Given a circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and an AMD encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ that outputs encodings of length $\hat{n}(n)$, we use $(\mathsf{Enc} \circ C) : \mathbb{F}^n \to \mathbb{F}^{\hat{n}(k)}$ to denote the circuit that on input $\mathbf{x} \in \mathbb{F}^n$, computes $(y_1, \cdots, y_k) \leftarrow C(\mathbf{x})$, and outputs $(\mathsf{Enc}(y_1), \cdots, \mathsf{Enc}(y_k))$.*

4 Efficient Additive Correctness Using Correct-Only MPC

In this section we construct a $2^{-\sigma}$-additively-correct circuit with $\mathrm{polylog}(|C|, \sigma)$ overhead. Specifically, for every depth-d arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ we construct a $2^{-\sigma}$-additively correct implementation $\widehat{\mathsf{C}}$, where $\left|\widehat{\mathsf{C}}\right| = |C| \cdot \mathrm{polylog}(|C|, \sigma) + \mathrm{poly}(n, k, d, \sigma)$, thus proving Theorem 1.

Recall that when $\widehat{\mathsf{C}}$ is constructed from a correct-only MPC protocol π then each attack on $\widehat{\mathsf{C}}$ can be divided into three "parts". The first "part" attacks connecting wires between sub-circuits of $\widehat{\mathsf{C}}$ (these are $\mathsf{InpEnc}, \mathsf{OutDec}$ and $\mathsf{NextMSG}$), and we protect against such attacks by having these sub-circuits operate on AMD codewords. The second "part" attacks the $\mathsf{NextMSG}$ functions, and we protect against such attacks by replacing $\mathsf{NextMSG}$ with its ϵ-additively correct implementation. Thus, every such attack either affects only few $\mathsf{NextMSG}$ functions, in which case the correctness of π guarantees that it does not affect the outputs; or it affects many $\mathsf{NextMSG}$ functions, in which case ϵ-additive correctness guarantees that (except with negligible probability) the attack is either detected, or corresponds to an additive attack on the inputs and outputs of $\mathsf{NextMSG}$. (Additive attacks on the inputs and outputs correspond to the first type of attacks, namely attacks on the connecting wires, which are detected by the AMD encoding scheme.) The third and final "part" attacks the clients, and we protect against such attacks by replacing $\mathsf{InpEnc}, \mathsf{OutDec}$ with their $2^{-\sigma}$-additively-correct implementations (e.g., Construction 9, Appendix A). This is formalized in the following construction, and described in Figure 3.

Construction 4 *Let $\mathbb{F}$ be a finite field, $C : \mathbb{F}^n \to \mathbb{F}^k$ be an arithmetic circuit over $\mathbb{F}$, σ be a security parameter, and π be a d-round, σ-correct m-server protocol for computing C using only point-to-point channels. We assume (without loss of generality) that every message sent in π consists of exactly s field elements, for some $s \in \mathbb{N}$. Let $(\mathsf{Enc}, \mathsf{Dec})$ be an $(s, \sigma, 2^{-\sigma})$-AMD encoding scheme that outputs encodings of length $\hat{n}(s)$. The circuit $\widehat{\mathsf{C}}$ will use the following ingredients.*

1. ***Input Encoding.*** *Let h denote the number of messages sent by the input client in the first round, namely $\mathsf{InpEnc} : \mathbb{F}^n \to (\mathbb{F}^s)^h$. Let $\widehat{\mathsf{InpEnc}} : \mathbb{F}^n \to \mathbb{F}^{t'} \times \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ denote the $2^{-\sigma}$-additively correct implementation, with t' flags, of the circuit $(\mathsf{Enc} \circ \mathsf{InpEnc}) : \mathbb{F}^n \to \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ (as defined in Notation 8).*
2. ***Message Generation.*** *For every $1 \le i \le m$, and $2 \le j \le d-1$, let g (h) denote the number of messages received (sent) by the i'th server in round $j-1$ (j).[7] That is, $\mathsf{NextMSG}_i^j : (\mathbb{F}^s)^g \to (\mathbb{F}^s)^h$. Let $\widehat{\mathsf{NextMSG}}_i^j : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^t \times \mathbb{F}^\sigma \times \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ denote the ϵ-additively correct implementation, with t flags, of the circuit $\mathcal{T}_{\mathsf{inter}}\left(\mathsf{NextMSG}_i^j\right) : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^\sigma \times \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ (see Construction 2).*
3. ***Output Generation.*** *Let g denote the number of messages received by the output client in the final round, namely $\mathsf{OutDec} : (\mathbb{F}^s)^g \to \mathbb{F}^k$. Let $\widehat{\mathsf{OutDec}} : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^{t''} \times \mathbb{F}^\sigma \times \mathbb{F}^k$ denote the $2^{-\sigma}$-additively correct implementation, with t'' flags, of the circuit $\mathcal{T}_{\mathsf{fin}}(\mathsf{OutDec}) : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^\sigma \times \mathbb{F}^k$ (see Construction 3).*
4. ***Circuit Construction.*** *The circuit $\widehat{\mathsf{C}}$, on input $\mathbf{x} \in \mathbb{F}^n$:*
   (a) *Emulates π, with x as the input of the client, and where $\widehat{\mathsf{InpEnc}}$, $\widehat{\mathsf{NextMSG}}_i^j$ and $\widehat{\mathsf{OutDec}}$ of Steps 1- 3 above (connected in the natural way) replace InpEnc, $\mathsf{NextMSG}_i^j$ and OutDec. That is, for every round $1 \le j \le d$, if server S_i sends a message to server $S_{i'}$, then the corresponding output of $\widehat{\mathsf{NextMSG}}_i^j$ is wired to the corresponding input of $\widehat{\mathsf{NextMSG}}_{i'}^{j+1}$.*
   Denote the output of the client in the above execution by $\mathbf{z}$.
   (b) *For every $1 \le i \le m$, and every $1 \le j \le d$, let $f'^j_{i,1}, \cdots, f'^j_{i,t}$ be the first t outputs of $\widehat{\mathsf{NextMSG}}_i^j$, and let $f^j_{i,1}, \cdots, f^j_{i,\sigma}$ be the next σ outputs of $\widehat{\mathsf{NextMSG}}_i^j$. (The $f'^j_{i,w}$'s are the flags of the ϵ-correct implementation, and the $f^j_{i,w}$'s are the flags generated during the AMD decoding.)*
   (c) *Let $f'^1_1, \cdots, f'^1_{t'}$ be the first t' outputs of $\widehat{\mathsf{InpEnc}}$. (These are the flags of the $2^{-\sigma}$-correct implementation.)*

[7] We assume each server transfers its internal state from one round to the next by sending a message to itself.

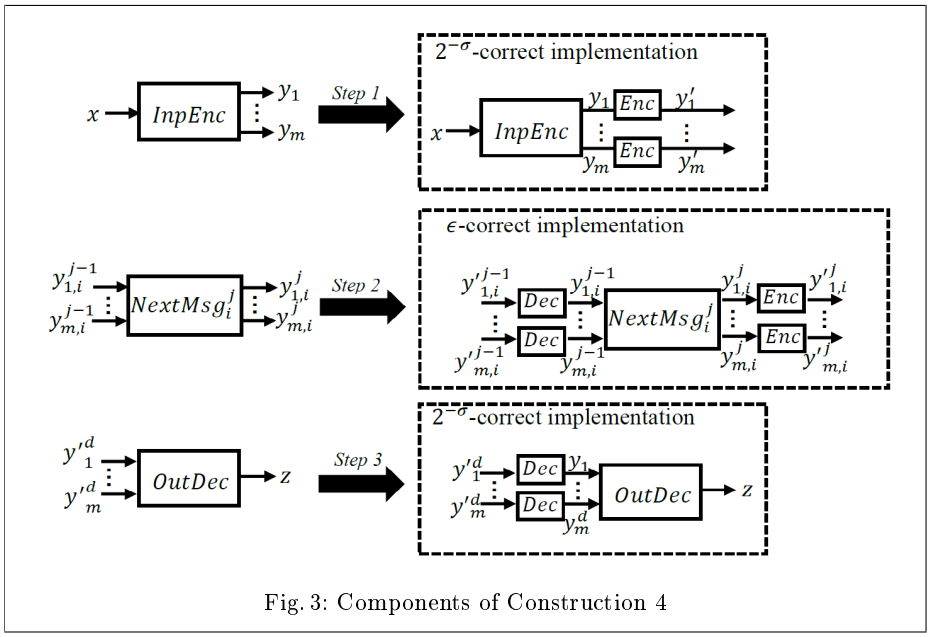

Fig. 3: Components of Construction 4

(d) *Let* $f'^d_1, \cdots, f'^d_{t''}$ *be the first* t'' *outputs of* $\widehat{\mathsf{OutDec}}$ *and let* $f^d_1, \cdots, f^d_\sigma$ *be the next* σ *outputs of* $\widehat{\mathsf{OutDec}}$. *(The* f'^d_i*'s are the flags of the* $2^{-\sigma}$*-correct implementation, and the* f^d_i*'s are the flags generated during the AMD decoding.)*

(e) *For every* $1 \leq w' \leq \sigma$, *compute* $f''_{w'} \leftarrow \sum_{i=1}^{m} \sum_{j=2}^{d-1} \left(\sum_{w=1}^{t} f'^j_{i,w} \cdot r_{i,j,w,w'} + \sum_{w=1}^{\sigma} f^j_{i,w} \cdot r_{i,j,t+w,w'} \right) + \sum_{w=1}^{t''} f'^d_w \cdot r_{1,d,w,w'} + \sum_{w=1}^{\sigma} f^d_w \cdot r_{1,d,t+w,w'} + \sum_{w=1}^{t''} f'^1_w \cdot r_{1,1,w,w'}$ *where* $r_{i,j,w,w'} \in_R \mathbb{F}$.

(f) *Output* $\mathbf{z} + \sum_{w=1}^{\sigma} f''_w \cdot \mathbf{r}'_w$ *where* $\mathbf{r}'_w \in_R \mathbb{F}^k$.

We now analyze the properties of Construction 4. The following notation will be useful.

Notation 9 *We denote the ingredients of Construction 4 as follows.*

- *We use* InpEnc' *to denote the circuit* $(\mathsf{Enc} \circ \mathsf{InpEnc})$ *obtained in Step 1.*
- *For every* $1 \leq i \leq m$, *and* $2 \leq j \leq d-1$, *we use* $\mathsf{NextMSG}'^j_i$ *to denote the circuit* $\mathcal{T}_{\mathsf{inter}}\left(\mathsf{NextMSG}^j_i\right)$ *obtained in Step 2.*
- *We use* OutDec' *to denote the circuit* $\mathcal{T}_{\mathsf{fin}}(\mathsf{OutDec})$ *obtained in Step 3.*

The next theorem shows that Construction 4 produces a $2^{-\Omega(\sigma)}$-additively-correct implementation.

Theorem 10. *Let* σ *be a security parameter,* $C : \mathbb{F}^n \to \mathbb{F}^k$ *be an arithmetic circuit, and* π *be an* m*-party,* d*-round protocol for* $(\sigma, 2^{-\sigma})$*-correctly computing*

C. Then the circuit $\widehat{\mathsf{C}}$ obtained by applying Construction 4 to C is a $2^{-\Omega(\sigma)}$-additively-correct implementation of C.

Proof. The completeness property of $\widehat{\mathsf{C}}$ immediately follows from Construction 4, the correctness of π, and the perfect completeness of the underlying AMD code. We now proceed to proving additive correctness. Let $\mathbf{A}$ be an additive attack on $\widehat{\mathsf{C}}$, and let $\mathbf{A}^{\mathsf{out}}$ denote the attacks on the outputs of $\widehat{\mathsf{C}}$ as specified by $\mathbf{A}$. Let $\mathbf{A}_{\mathsf{InpEnc}}$, $\mathbf{A}_{\mathsf{OutDec}}$ denote the restrictions of $\mathbf{A}$ to the wires of $\widehat{\mathsf{InpEnc}}$ and $\widehat{\mathsf{OutDec}}$ respectively. Additionally, for every $1 \leq i \leq m$ and every $2 \leq j \leq d-1$ let $\mathbf{A}_i^j$ denote the restriction of $\mathbf{A}$ to $\widehat{\mathsf{NextMSG}}_i^j$. Let $(\mathbf{a}^{\mathsf{in},1}, \mathbf{a}^{\mathsf{out},1})$ and $(\mathbf{a}^{\mathsf{in},d}, \mathbf{a}^{\mathsf{out},d})$ be the ideal additive attacks on the inputs and outputs of $\widehat{\mathsf{InpEnc}}$ and $\widehat{\mathsf{OutDec}}$ corresponding to $\mathbf{A}_{\mathsf{InpEnc}}$, $\mathbf{A}_{\mathsf{OutDec}}$. Similarly, for every $1 \leq i \leq m$ and every $2 \leq j \leq d-1$, let $\mathbf{a}_i^{\mathsf{in},j}$, and $\mathbf{a}_i^{\mathsf{out},j}$ be the ideal additive attacks on the inputs and outputs of $\widehat{\mathsf{NextMSG}}_i^j$ corresponding to $\mathbf{A}_i^j$. Define $\mathbf{a}^{\mathsf{in}} = \mathbf{a}^{\mathsf{in},1}$ and $\mathbf{a}^{\mathsf{out}} = \mathbf{a}^{\mathsf{out},d} + \mathbf{A}^{\mathsf{out}}$. We claim that for every input $\mathbf{x}$ it holds that

$$\Pr[\widehat{\mathsf{C}}^{\mathbf{A}}(\mathbf{x}) \notin \mathsf{ERR} \cup \{(0^\sigma, C(\mathbf{x}+\mathbf{a}^{\mathsf{in}}) + \mathbf{a}^{\mathsf{out}})\}] \leq 2^{-\Omega(\sigma)}$$

where $\mathsf{ERR} = (\mathbb{F}^\sigma \setminus \{0^\sigma\}) \times \mathbb{F}^k$.

Indeed, let $\mathbf{x} \in \mathbb{F}^n$ be an input to $\widehat{\mathsf{C}}$, and define P_{bad} as the event that $\widehat{\mathsf{C}}^{\mathbf{A}}(\mathbf{x}) \notin \mathsf{ERR} \cup \{(0^\sigma, C(\mathbf{x}+\mathbf{a}^{\mathsf{in}}) + \mathbf{a}^{\mathsf{out}})\}$, namely

$$\Pr[\widehat{\mathsf{C}}^{\mathbf{A}}(\mathbf{x}) \notin \mathsf{ERR} \cup \{(0^\sigma, C(\mathbf{x}+\mathbf{a}^{\mathsf{in}}) + \mathbf{a}^{\mathsf{out}})\}] = \Pr[P_{\mathsf{bad}}].$$

Next, denote by P_f the event that

$$\bigwedge_{i=1}^{m} \bigwedge_{j=2}^{d-1} \bigwedge_{w=1}^{t} (f'^{j\mathbf{A}}_{i,w,\mathbf{x}} = f^{j\mathbf{A}}_{i,w,\mathbf{x}} = 0) \bigwedge \bigwedge_{w=1}^{t'} f'^1_w = 0 \bigwedge \bigwedge_{w=1}^{t''} f'^d_w = 0.$$

Notice that by construction of $\widehat{\mathsf{C}}$ we obtain that

$$\Pr[\widehat{\mathsf{C}}^{\mathbf{A}}(\mathbf{x}) \notin \mathsf{ERR} \cup \{(0^\sigma, C(\mathbf{x}+\mathbf{a}^{\mathsf{in}}) + \mathbf{a}^{\mathsf{out}})\}] \leq 2^{-\Omega(\sigma)} + \Pr[P_{\mathsf{bad}} \wedge P_f].$$

We proceed by defining the event $P^{1,1}_{\mathsf{OK}}$ as $\widehat{\mathsf{InpEnc}}^{\mathbf{A}}(\mathbf{x}) \in \mathsf{ERR} \cup \{\mathsf{InpEnc}(\mathbf{x} + \mathbf{a}^{\mathsf{in},1}) + \mathbf{a}^{\mathsf{out},1}\}$ and $P^{d,d}_{\mathsf{OK}}$ as $\widehat{\mathsf{OutDec}}^{\mathbf{A}}(\mathbf{y}^{\mathbf{A}}_{\mathbf{x}}) \in \mathsf{ERR} \cup \{\mathsf{OutDec}(\mathbf{y}^{\mathbf{A}}_{\mathbf{x}} + \mathbf{a}^{\mathsf{in},d}) + \mathbf{a}^{\mathsf{out},d}\}$, where $\mathbf{y}^{\mathbf{A}}_{\mathbf{x}}$ is the random variable corresponding to the messages received by the client from the servers during the last round of π inside $\widehat{\mathsf{C}}^{\mathbf{A}}(\mathbf{x})$. We notice that by the $2^{-\sigma}$-correctness of $\widehat{\mathsf{InpEnc}}$ and $\widehat{\mathsf{OutDec}}$ it holds that

$$\Pr[P_{\mathsf{bad}} \wedge P_f] \leq 2^{-\Omega(\sigma)} + \Pr\left[P_{\mathsf{bad}} \wedge P_f \wedge P^{1,1}_{\mathsf{OK}} \wedge P^{d,d}_{\mathsf{OK}}\right].$$

Next, for every round $2 \leq j \leq d-1$ and party $1 \leq i \leq m$, denote by In_i^j the set of servers which send messages to the ith server during the jth round, and denote by $\mathbf{a}^{\mathsf{in},j}_{i,i'}$ the ideal additive attacks on the inputs of $\widehat{\mathsf{NextMSG}}_i^j$ which

correspond to the message received by server i from server i' during the jth round. Similarly, denote by Out_i^j the set of servers to which the ith server sends messages during the jth round, and denote by $\mathbf{a}_{i,i'}^{\mathsf{out},j}$ the ideal additive attacks on the outputs of $\widehat{\mathsf{NextMSG}}_i^j$ which correspond to the message sent by server i to server i' during the jth round. In addition, we assume without loss of generality that the client sends a message to all the servers during the first round, and receives a message from all the servers during the last round. Finally, for every server $1 \leq i \leq m$, we denote by $\mathbf{a}_i^{\mathsf{out},1}$ the restriction of $\mathbf{a}^{\mathsf{out},1}$ to the messages that the client sends to the ith server during the first round and by $\mathbf{a}_i^{\mathsf{in},d}$ the restriction of $\mathbf{a}^{\mathsf{in},d}$ to the messages that the client receives from the ith server during the dth round. Finally, we denote by $\mathbf{a}_i^{\mathsf{in},2}$ the messages received by the ith server from the client, and we denote by $\mathbf{a}_{i'}^{\mathsf{out},d-1}$ the messages sent by the i'th server to the client.

For any $1 \leq i, i' \leq m$ and $2 \leq j \leq d-1$, we say that a tuple (i', i, j) is *problematic* if one of the following three conditions hold.

1. **Input Corruption.** It holds that $\mathbf{a}_i^{\mathsf{in},2} + \mathbf{a}_i^{\mathsf{out},1} \neq 0$ and $i' = j = 1$.
2. **Intermediate Corruption.** It holds that $\mathbf{a}_{i,i'}^{\mathsf{in},j} + \mathbf{a}_{i',i}^{\mathsf{out},j-1} \neq 0$.
3. **Output Corruption.** It holds that $\mathbf{a}_{i'}^{\mathsf{out},d-1} + \mathbf{a}_{i'}^{\mathsf{in},d} \neq 0$ and $i = j = d$.

Next, we define the set $\mathcal{A} = \{(i', i, j) : \text{the tuple } (i', i, j) \text{ is problematic}\}$ and we split the proof into two cases.

– **Case 1: $|\mathcal{A}| > \sigma$.** Intuitively, in this case a large portion of $\widehat{\mathsf{C}}$ was corrupted. We show that in this case $\widehat{\mathsf{C}}$ will almost always abort the computation by setting at least one of the flags to a non zero value, namely the probability of an incorrect output (i.e, not in $\mathsf{ERR} \cup \{(0^\sigma, C(\mathbf{x} + \mathbf{a}^{\mathsf{in}}) + \mathbf{a}^{\mathsf{out}})\}$) is low.

  We denote the random variables describing the messages exchanged during the evaluation of $\widehat{\mathsf{C}}^{\mathbf{A}}$ on input $\mathbf{x}$ as follows: for every $1 \leq i \leq m$ and $2 \leq j \leq d-2$, $\widehat{y}_{i,i',\mathbf{x}}^{\mathbf{A},j}$ corresponds to the message sent by the ith server to the i'th server in round j; $\widehat{y}_{i,\mathbf{x}}^{\mathbf{A},1}$ corresponds to the messages sent by the client to the ith server in the first round; and $\widehat{y}_{i,\mathbf{x}}^{\mathbf{A},d-1}$ corresponds to the message sent by the ith server to the client in round $d-1$.

  Next, for any $1 \leq i \leq m$ and $2 \leq j \leq d-1$ denote by $P_{\mathsf{OK}}^{i,j}$ the event that $\widehat{\mathsf{NextMSG}}_i^{\mathbf{A},j}\left(\left(\widehat{y}_{i',i,\mathbf{x}}^{\mathbf{A},j-1}\right)_{i' \in \mathsf{In}_i^j}\right)$ is in $\mathsf{ERR} \cup \left\{\left(0^t, \mathsf{NextMSG}_i'^j\left(\left(\widehat{y}_{i',i,\mathbf{x}}^{\mathbf{A},j-1}\right)_{i' \in \mathsf{In}_i^j} + \mathbf{a}_i^{\mathsf{in},j}\right) + \mathbf{a}_i^{\mathsf{out},j}\right)\right\}$, where $\mathsf{ERR} = (\{\mathbb{F}^t\} \setminus \{0^t\}) \times \mathbb{F}^{o_i^j}$, and o_i^j is the output length of $\mathsf{NextMSG}_i'^j$.

  Next, notice that for every tuple (i', i, j) the randomness of $\widehat{\mathsf{NextMSG}}_i^{\mathbf{A},j}$ is independent from the randomness of $\widehat{\mathsf{NextMSG}}_{i'}^{\mathbf{A},j-1}$. Thus, it holds that $\Pr\left[P_{\mathsf{OK}}^{i',j-1} \wedge P_{\mathsf{OK}}^{i,j}\right] \geq (1-\epsilon)^2$, yielding $\Pr\left[P_{\mathsf{OK}}^{i',j-1} \wedge P_{\mathsf{OK}}^{i,j}\right] \leq 1 - (1-\epsilon)^2$. Next, across all the problematic tuples in $\mathcal{A}$ we obtain that

$\Pr\left[P_{\mathsf{bad}} \wedge P_f \wedge P_{\mathsf{OK}}^{1,1} \wedge P_{\mathsf{OK}}^{d,d}\right]$ is at most

$$\left(1-(1-\epsilon)^2\right)^{\sigma} + \Pr\left[\begin{array}{c} P_{\mathsf{bad}} \wedge P_f \wedge P_{\mathsf{OK}}^{1,1} \wedge P_{\mathsf{OK}}^{d,d} \wedge \\ \left(\exists (i',i,j) \in \mathcal{A} : (P_{\mathsf{OK}}^{i',j-1} \wedge P_{\mathsf{OK}}^{i,j})\right) \end{array}\right].$$

Finally, the fact that $P_{\mathsf{OK}}^{i',j-1} \wedge P_{\mathsf{OK}}^{i,j}$ for some problematic tuple $(i',i,j) \in \mathcal{A}$ implies that there is a non-zero additive attack on the wires between server i' (or the client in case $j=1$) and server i (again, or the client in case $j=d$) during the jth round. Thus, by the additive soundness of $(\mathsf{Enc},\mathsf{Dec})$ we obtain that except with probability $2^{-\sigma}$, $(f_{i,1}^j,\cdots,f_{i,\sigma}^j) \neq 0$, namely P_f does not hold. Consequently,

$$\Pr\left[\begin{array}{c} P_{\mathsf{bad}} \wedge P_f \wedge P_{\mathsf{OK}}^{1,1} \wedge P_{\mathsf{OK}}^{d,d} \wedge \\ \left(\exists (i',i,j) \in \mathcal{A} : (P_{\mathsf{OK}}^{i',j-1} \wedge P_{\mathsf{OK}}^{i,j})\right) \end{array}\right] \leq 2^{-\Omega(\sigma)}.$$

– **Case 2: $|\mathcal{A}| \leq \sigma$.** Notice that having less than σ problematic tuples implies that for the protocol π inside $\widehat{\mathsf{C}}$, the additive attack $\mathbf{A}$ only corrupted less than σ parties. In this case we get that except with probability $2^{-\sigma}$, the protocol π manages to correctly compute C. Thus, in this case

$$\Pr\left[P_{\mathsf{bad}} \wedge P_f \wedge P_{\mathsf{OK}}^{1,1} \wedge P_{\mathsf{OK}}^{d,d}\right] \leq 2^{-\Omega(\sigma)}.$$

□

We show that for an appropriate choice of parameters, Construction 4 is a $2^{-\sigma}$-additively correct implementation. This is formalized in the next Theorem.

Theorem 11. *For any depth-d arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and any security parameter σ, there exists a $2^{-\Omega(\sigma)}$-additively-correct implementation $\widehat{\mathsf{C}}$ of C where $|\widehat{\mathsf{C}}| = |C| \cdot polylog(|C|,\sigma) + poly(n,k,d,\sigma)$.*

We first state several results regarding AMD encoding schemes, which will be used in the proof.

Asymptotically optimal constructions of AMD encoding schemes have been presented by [8] and [3]. In fact, [3] consider a slightly weaker definition of AMD codes which guarantees that $\Pr[\mathsf{Dec}(\mathsf{Enc}(\mathbf{x})+\mathbf{a}) \notin \mathsf{ERR} \cup \{(0,\mathbf{x})\}] \leq \epsilon$, allowing for ERR on some inputs and correct output on others (see Definition 7 below). However, their construction actually possesses the stronger security property of Definition 4.

Theorem 12 (Implicit in [3], Corollary 1). *For any $n,\sigma \in \mathbb{N}$, and field $\mathbb{F}$, there exists a pair of families of circuits $(\mathsf{Enc},\mathsf{Dec})$ over $\mathbb{F}$ that is an $(n,\sigma,\frac{1}{|\mathbb{F}|^\sigma})$-AMD encoding scheme with encodings of length $n+\sigma$. Moreover, the size of Enc and Dec is $\widetilde{O}(n+\sigma)$.*

Theorem 13 (Implicit in [10]). *There exists a constant $\epsilon \in (0,1)$ such that for any field $\mathbb{F}$ and arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ there exist a circuit $\widehat{\mathsf{C}} : \mathbb{F}^n \to \mathbb{F} \times \mathbb{F}^k$ which is an ϵ-additively-correct implementation of C. Moreover, $\left|\widehat{\mathsf{C}}\right| = O(|C|)$.*

Proof (of Theorem 11). Apply Construction 4 to C using an AMD code of Theorem 12, the ϵ-additively-correct construction from Theorem 13 and the σ-server protocol π from Theorem 6. To obtain the $2^{-\sigma}$-additively-correct implementation of $\widehat{\mathsf{InpEnc}}$ and $\widehat{\mathsf{OutDec}}$ used in Steps 1 and 3 of Construction 4, we use an additively-correct circuit compiler $\mathsf{Comp}^{\mathsf{In}}$ that on input a circuit C outputs a circuit $\widehat{\mathsf{C}}$ such that $\left|\widehat{\mathsf{C}}\right| = \sigma \cdot |C|$ (e.g., Construction 9 of Appendix A). Since π $(\sigma/10)$-correctly computes C we obtain that $\widehat{\mathsf{C}}$ is a $2^{-\Omega(\sigma)}$-additively-correct implementation of C.

Next, we proceed to analyze the size of $\widehat{\mathsf{C}}$. By the construction of $\widehat{\mathsf{C}}$ we have that $|\widehat{\mathsf{C}}| = |\widehat{\mathsf{InpEnc}}| + |\widehat{\mathsf{OutDec}}| + \sum_{i=1}^{\sigma}\sum_{j=1}^{d}|\widehat{\mathsf{NextMSG}}_i^j|$. From Theorem 6 we obtain that $|\mathsf{InpEnc}| + |\mathsf{OutDec}|$ is $\mathrm{poly}(n,k,\sigma)$. Thus, when InpEnc and OutDec are implemented using Construction 9 (Appendix A, $|\widehat{\mathsf{InpEnc}}| + |\widehat{\mathsf{OutDec}}|$ is also $\mathrm{poly}(n,k,\sigma)$. We now proceed to analyze $\sum_{i=1}^{\sigma}\sum_{j=1}^{d}|\widehat{\mathsf{NextMSG}}_i^j|$.

We begin by noticing that in each round of π, each server sends messages containing a total of $\mathrm{poly}(n,k,\log|C|)$ field elements. Thus, by having $\mathsf{NextMSG}'$ encode every message sent during the execution of π with the AMD codes from Theorem 12 we obtain that the circuit size of every $\mathsf{NextMSG}'$ function increases by an additive term which is $\mathrm{poly}(n,k,\log|C|,\sigma)$ compared to $\mathsf{NextMSG}$. Next, since the overall circuit size of all the $\mathsf{NextMSG}$ functions is $|C| \cdot \mathrm{polylog}(|C|,\sigma) + \mathrm{poly}(\sigma,d,n,k,\log|C|)$ and since $|\widehat{\mathsf{NextMSG}}| = O(\mathsf{NextMSG}')$ we obtain that the total circuit size of all the $\widehat{\mathsf{NextMSG}}$ circuits inside $\widehat{\mathsf{C}}$ is also $|C| \cdot \mathrm{polylog}(|C|,\sigma) + \mathrm{poly}(\sigma,d,n,k,\log|C|)$. □

Remark 3 *The proof of Theorem 11 uses an ad-hoc "feasibility" construction to achieve polylog(σ) overhead. However, it is possible to improve the simplicity, and concrete efficiency, of the construction by replacing the feasibility construction with simpler gadgets implementing the input encoder and output decoder. We now outline a more direct construction (which matches the complexity of Theorem 11). We begin by observing that for the protocol of Theorem 6, we can assume (without loss of generality) that $\mathsf{InpEnc}(\mathbf{x}) = (\mathbf{x},\cdots,\mathbf{x})$, and $\mathsf{OutDec}(\mathbf{y}_1,\cdots,\mathbf{y}_m)$ outputs $(0^\sigma,\mathbf{y}_1)$ if $\mathbf{y}_1 = \cdots = \mathbf{y}_m$, otherwise it outputs a random value in $(\mathbb{F}^\sigma \setminus \{0^\sigma\}) \times \mathbb{F}^k$. Next, we implement $\widehat{\mathsf{InpEnc}}$ and $\widehat{\mathsf{OutDec}}$ directly using the following simple gadgets.*

- ***Implementing $\widehat{\mathsf{InpEnc}}$.*** *We define $\widehat{\mathsf{InpEnc}}(\mathbf{x}) = (\mathsf{Enc}(\mathbf{x}),\cdots,\mathsf{Enc}(\mathbf{x}))$, where Enc is the encoding procedure of a $2^{-\sigma}$-robust AMD code (as in Definition 5). The stronger robustness property guarantees the existence of* a single consistent value *such that (with high probability) every server either decodes to it, or aborts.*

– ***Implementing*** $\widehat{\mathsf{OutDec}}$**.** *We modify each server to compute a MAC value of its outputs. In addition, C is evaluated in the clear: $\mathbf{z} \leftarrow C(\mathbf{x})$, and the output $\mathbf{z}$ is MACed to obtain $\tilde{\mathbf{z}}$. Finally, $\widehat{\mathsf{OutDec}}$ contains a gadget that compares all MACed outputs of the servers to $\tilde{\mathbf{z}}$, and outputs $\mathbf{z}$ if the test passes, otherwise it outputs* ERR.[8]

5 From Additive Correctness to Additive Security via Passive-Secure MPC

In this section we combine additively-correct circuits with passive-secure MPC protocols to construct binary additively-secure circuits with a negligible error, thus proving Theorem 2.

Recall that (as described in Section 1.1.2) we construct the additively-secure implementation $\widehat{\mathsf{C}}$ of C from a passive-secure MPC protocol π. More specifically, the inputs of parties in π are additive secret-shares of the input of C, and π evaluates the function that: (1) reconstructs the input from the secret shares; (2) evaluates C; and (3) outputs an additive secret-sharing of the output.

Consequently, every additive attack on $\widehat{\mathsf{C}}$ can be divided into two "parts". The first "part" targets the wires connecting different sub-circuits NextMSG of $\widehat{\mathsf{C}}$, and we protect against such attacks by having these sub-circuits operate on AMD codewords. The second "part" modifies the internal computations of the NextMSG functions, and we protect against such attacks by replacing each NextMSG with its $2^{-\sigma}$-additively-correct implementation. Thus, the resultant $\widehat{\mathsf{C}}$ is a $2^{-\Omega(\sigma)}$-additively correct implementation of C, where every attack is with overwhelming probability either "harmless" (namely, corresponds to an additive attack on the inputs and output of C), or causes the output to be random. Moreover, as we argued in Section 1.1.2, the probability that the output is random is independent of the inputs.

We start by defining the circuit C_{AUG}, which implements the functionality computed by π (namely, emulates C on secret shares).

Construction 5 *Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be an arithmetic circuit, and $m \in \mathbb{N}$. The circuit C_{AUG}, on inputs $(\mathbf{x}_1, \cdots, \mathbf{x}_m) \in (\mathbb{F}^n)^m$, performs the following.*

1. *Computes $\mathbf{x} \leftarrow \sum_{i=1}^{m} \mathbf{x}_i$, and $\mathbf{y} \leftarrow C(\mathbf{x})$. (This step reconstructs the input to C from the secret shares, and evaluates C.)*
2. *Generates $\mathbf{y}_1, \cdots, \mathbf{y}_{m-1} \in \mathbb{F}^n$ uniformly at random, and compute $\mathbf{y}_m \leftarrow \mathbf{y} - \sum_{i=1}^{m-1} \mathbf{y}_i$. ($\mathbf{y}_1, \cdots, \mathbf{y}_m$ is an additive secret sharing of the output $\mathbf{y}$.)*
3. *Outputs $(\mathbf{y}_1, \cdots, \mathbf{y}_m)$.*

Next, we use C_{AUG} to construct the circuit $\widehat{\mathsf{C}}$, see also Figure 4.

[8] To implement $\widehat{\mathsf{OutDec}}$ without leaking information regarding the outputs of C, we compare *only the MAC tags* generated by the servers (and not *the actual outputs*). This necessitates an additional evaluation of C (in the clear) to generate the output.

Construction 6 *Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be an arithmetic circuit over a finite field $\mathbb{F}$, σ be a security parameter, and π be a d-round, t-private, m-party protocol for computing the circuit C_{AUG} of Construction 5, using only point-to-point channels. We assume (without loss of generality) that every message sent in π consists of exactly s field elements, for some $s \in \mathbb{N}$. Let $(\mathsf{Enc}, \mathsf{Dec})$ be an $(s, \sigma, 2^{-\sigma})$-AMD encoding scheme that outputs encodings of length $\hat{n}(s)$, and Dec outputs σ flags during decoding. The circuit $\widehat{\mathsf{C}}$ will use the following ingredients.*

1. ***Protecting the first round.*** *For every $1 \leq i \leq m$, assume that party P_i sends h messages in the first round, namely $\mathsf{NextMSG}_i^1 : \mathbb{F}^n \to (\mathbb{F}^s)^h$. Let $\widehat{\mathsf{NextMSG}}_i^1 : \mathbb{F}^n \to \mathbb{F}^t \times \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ be the $2^{-\sigma}$-additively correct implementation, with t flags, of the circuit $\left(\mathsf{Enc} \circ \mathsf{NextMSG}_i^1\right) : \mathbb{F}^n \to \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ (see Construction 3).*
2. ***Protecting middle rounds.*** *For every $1 \leq i \leq m$, and $2 \leq j \leq d-1$, assume that in round $j-1$ (j) P_i receives (sends) g (h) messages, namely $\mathsf{NextMSG}_i^j : (\mathbb{F}^s)^g \to (\mathbb{F}^s)^h$. Let $\widehat{\mathsf{NextMSG}}_i^j : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^t \times \mathbb{F}^\sigma \times \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ be the $2^{-\sigma}$-additively correct implementation, with t flags, of the circuit $\mathcal{T}_{\mathsf{inter}}\left(\mathsf{NextMSG}_i^j\right) : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^\sigma \times \left(\mathbb{F}^{\hat{n}(s)}\right)^h$ (see Construction 2).*
3. ***Protecting the last round.*** *For every $1 \leq i \leq m$ assume that P_i receives g messages in the final round, namely $\mathsf{NextMSG}_i^d : (\mathbb{F}^s)^g \to \mathbb{F}^k$. Let $\widehat{\mathsf{NextMSG}}_i^d : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^t \times \mathbb{F}^\sigma \times \mathbb{F}^k$ be the $2^{-\sigma}$-additively correct implementation, with t flags, of the circuit $\mathcal{T}_{\mathsf{fin}}\left(\mathsf{NextMSG}_i^d\right) : \left(\mathbb{F}^{\hat{n}(s)}\right)^g \to \mathbb{F}^t \times \mathbb{F}^k$ (see Construction 3).*
4. ***Circuit construction.*** *The circuit $\widehat{\mathsf{C}}$ on input $\mathbf{x}$ performs the following.*
   (a) *Generate $\mathbf{x}_1, \cdots, \mathbf{x}_{m-1} \in \mathbb{F}^n$ uniformly at random and compute $\mathbf{x}_m \leftarrow \mathbf{x} - \sum_{i=1}^{m-1} \mathbf{x}_i$.*
   (b) *Emulates π with $\mathbf{x}_i$ as the input of party P_i, where the $\widehat{\mathsf{NextMSG}}_i^j$ described in Steps 1- 3 (connected in the natural way) replace the $\mathsf{NextMSG}_i^j$. That it, for every round $1 \leq j \leq d-1$, if party P_i sends a message to party $P_{i'}$, we wire the corresponding output of $\widehat{\mathsf{NextMSG}}_i^j$ to the corresponding input of $\widehat{\mathsf{NextMSG}}_{i'}^{j+1}$.*
   (c) *Let $\mathbf{z}_i$ denote P_i's output in the above execution. Compute $\mathbf{z} \leftarrow \sum_{i=1}^m \mathbf{z}_i$.*
   (d) *For every $1 \leq i \leq m$, and $2 \leq j \leq d$, let $f'^j_{i,1}, \cdots, f'^j_{i,t}$ denote the first t outputs of $\widehat{\mathsf{NextMSG}}_i^j$, and $f^j_{i,1}, \cdots, f^j_{i,\sigma}$ denote the $t+1$ to $t+\sigma$ outputs of $\widehat{\mathsf{NextMSG}}_i^j$. (The $f'^j_{i,w}$'s are the flags of the $2^{-\sigma}$-correct implementation, and the $f^j_{i,w}$'s are the flags generated during the AMD decoding.)*
   (e) *For every $1 \leq i \leq m$ let $f'^d_{i,1}, \cdots, f'^d_{i,t}$ denote the first t outputs of $\widehat{\mathsf{NextMSG}}_i^1$. (These are the flags of the $2^{-\sigma}$-correct implementation.)*

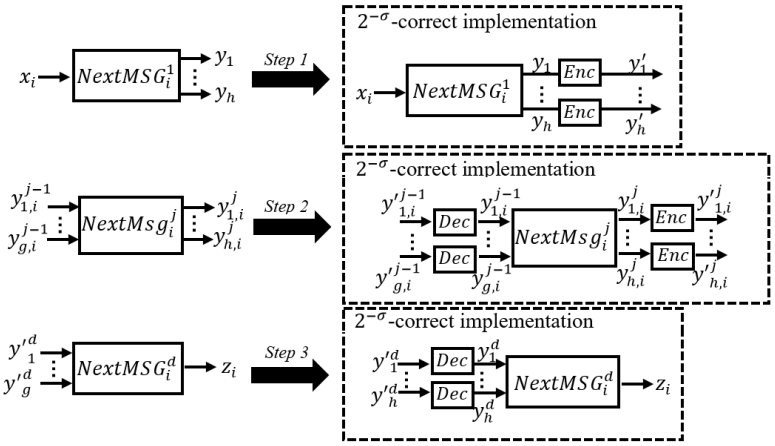

Fig. 4: Components of Construction 6

(f) *For every* $1 \leq w' \leq \sigma$, *compute* $f''_{w'} \leftarrow \sum_{i=1}^{m}\sum_{j=2}^{d}\left(\sum_{w=1}^{t} f'^{j}_{i,w}\cdot r_{i,j,w} + \sum_{w=1}^{\sigma} f^{j}_{i,w}\cdot r_{t+i,j,w}\right) + \sum_{i=1}^{m}\sum_{w=1}^{t} f'^{1}_{i,w}\cdot r_{i,d,w}$, *where* $r_{i,j,w} \in_R \mathbb{F}$.

(g) *Output* $\mathbf{z} + \sum_{w=1}^{m} f''_{w}\cdot \mathbf{r}'_{w}$, *where* $\mathbf{r}'_{w} \in_R \mathbb{F}^k$.

We show that any additive attack on $\widehat{\mathsf{C}}$ is either equivalent to an additive attack on the inputs and output of C, or will sets flags inside $\widehat{\mathsf{C}}$ to non-zero values. Moreover, the probability that a flag is set depends only on the additive attack, and is almost independent of the input. This is captured by the next theorem.

Theorem 14. *For any depth-d arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, and security parameter σ, there exists a $2^{-\Omega(\sigma)}$-additively-secure implementation $\widehat{\mathsf{C}}$ of C, where $|\widehat{\mathsf{C}}| = |C|\cdot \mathit{polylog}(|C|,\sigma) + \mathit{poly}(n,k,d,\sigma)$. Moreover, $\widehat{\mathsf{C}}$ can be constructed from C in $\mathit{poly}(|C|,\sigma,m)$ time.*

The proof of Theorem 14, which follows the outline presented in Section 1.1.2, is deferred to the full version. Here, we only outline the main points and subtle issues in the proof. We first show that with overwhelming probability any additive attack on $\widehat{\mathsf{C}}$ either sets error flags in $\widehat{\mathsf{C}}$, or is equivalent to an additive attack on its inputs and output. This is proved in two steps: first, using the additive correctness property of the $\widehat{\mathsf{NextMSG}}^{j}_{i}$ sub-circuits, except with negligible probability additive attacks on the internal wires of every $\widehat{\mathsf{NextMSG}}^{j}_{i}$ can be "pushed" to an additive attack on its inputs and outputs. Second, we examine the additive attacks obtained in this manner between every pair of adjacent $\widehat{\mathsf{NextMSG}}^{j-1}_{i'}$ and $\widehat{\mathsf{NextMSG}}^{j}_{i}$ sub-circuits. If all these attacks cancel out, then the output of $\widehat{\mathsf{C}}$ is

correct. Otherwise, the additive-security property of the AMD code protecting the communication channels between the $\widehat{\mathsf{NextMSG}}$ sub-circuits guarantees that with overwhelming probability an error flag will be set, causing $\widehat{\mathsf{C}}$ to abort.

Next, we prove that the probability of abort is almost independent of the inputs of $\widehat{\mathsf{C}}$. As before, we first "push" additive attacks on the $\widehat{\mathsf{NextMSG}}_i^j$ sub-circuits to additive attacks on their inputs and outputs. We then traverse the layers of $\widehat{\mathsf{C}}$ from the inputs to the output. In each layer j, a flag can be raised either by a $\widehat{\mathsf{NextMSG}}_i^j$ sub-circuit (which corresponds to the computation performed by a single party P_i), or by the AMD decoding performed in $\widehat{\mathsf{NextMSG}}_i^j$. In either case, the event that a flag is set depends only on the view of P_i which, by the t-privacy of π (and of the additive secret sharing of the input), guarantees that the distributions of the flags when evaluating $\widehat{\mathsf{C}}$ on two different inputs $\mathbf{x}, \mathbf{x}'$ are t-wise indistinguishable. Since a *single* set flag suffices to cause an abort, the "OR lemma" (Lemma 1) guarantees that the probability of abort is independent of the inputs to $\widehat{\mathsf{C}}$.

6 Constant-Overhead AMD Codes and Their Applications to Constant-Overhead MPC

In this section we use AMD codes to relate the open question of constructing actively-secure two-party protocols with constant computational overhead to the simpler questions of constructing passively-secure honest-majority MPC protocols, and correct-only honest-majority MPC protocols, with constant computational overhead. This is done by combining our constructions from Sections 4 and 5 with a (relaxed) AMD encoding scheme that has constant overhead.

More formally, we say that a secure implementation of a circuit C (e.g., an additively-secure implementation of C, or a secure protocol for evaluating C) has *constant computational overhead* if its circuit size is $O(|C|) + \mathrm{poly}(\log|C|, \sigma, d, n, k)$ where σ is the security parameter, d is the circuit depth, and n, k are the input and output lengths, respectively . (The circuit size of a protocol π is the total circuit size of all the $\mathsf{NextMSG}$ functions of π.)

We first construct relaxed AMD encoding schemes with constant overhead, namely the size of the encoding and decoding circuits is linear in the message length. At a high level, relaxed AMD encoding schemes, first considered by [3], have a weaker soundness guarantee: as long as the output is correct with high probability, (non-zero) additive attacks are allowed to pass unnoticed. This should be contrasted with (standard) AMD codes, in which *every* additive attack is guaranteed to be detected (with high probability).

Definition 7 (Relaxed AMD encoding scheme [3]). *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 $(n, t, \epsilon(n,t))$-*relaxed AMD encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ over $\mathbb{F}$ *is an encoding scheme with the following properties.*

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

– ***Relaxed additive soundness.*** *For every* $0^{\hat{n}(n,t)} \neq \mathbf{a} \in \mathbb{F}^{\hat{n}(n,t)}$*, and every* $\mathbf{x} \in \mathbb{F}^n$*,* $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(\mathbf{x}, 1^t\right) + \mathbf{a}, 1^t\right) \notin \mathsf{ERR} \cup \{(0, \mathbf{x})\}\right] \leq \epsilon\left(n, t\right)$ *where* $\mathsf{ERR} = \left(\mathbb{F} \setminus \{0\}\right) \times \mathbb{F}^n$*, and the probability is over the randomness of* Enc*.*

Roughly speaking, we construct a constant-overhead AMD encoding scheme by composing a linearly encodable and decodable AMD encoding scheme with constant additive soundness, with a linearly encodable error-correcting code with constant rate and relative distance. We will need the following notion of an $[n, k, d]$-error-correcting code.

Definition 8. *We say that a pair* $(\mathsf{Enc} : \mathbb{F}^k \to \mathbb{F}^n, \mathsf{Dec} : \mathbb{F}^n \to \mathbb{F}^k)$ *of deterministic circuits is an* $[n, k, d]$-error-correcting code (ECC) over $\mathbb{F}$ *if any* $\mathbf{x}, \mathbf{y} \in \mathbb{F}^k$ *it holds that* $\Pr\left[\mathsf{Dec}(\mathsf{Enc}(\mathbf{x})) = \mathbf{x}\right] = 1$ *and that* $|\{i : (\mathsf{Enc}(x))_i \neq (\mathsf{Enc}(y))_i\}| \geq d$*.*

The following theorem is due to Spielman [19] (see also [9]):

Theorem 15. *There exist constants* $d_1 > 1$*, and* $d_2 > 0$*, such that for any field* $\mathbb{F}$*, and any* $k \in \mathbb{N}$*, there exists a pair of circuits* $(\mathsf{Enc}_k, \mathsf{Dec}_k)$ *which is a* $[\lfloor d_1 k \rfloor, k, \lceil d_2 k \rceil]$*-ECC over* $\mathbb{F}$*. Moreover, the size of* Enc_k *is* $O(k)$*.*

We can now construct an AMD encoding scheme with constant overhead.

Construction 7 *Let* n *be a positive integer,* $\mathbb{F}$ *be a finite field, and* $(\mathsf{Enc}_n, \mathsf{Dec}_n)$ *be an* $[n', n, d]$*-ECC over* $\mathbb{F}$*. In addition, let* $(\mathsf{Enc}_{amd} : \mathbb{F} \to \mathbb{F}^k, \mathsf{Dec}_{amd} : \mathbb{F}^k \to \mathbb{F} \times \mathbb{F}$ *be a* $(1, t, \epsilon(t))$*-AMD encoding scheme. Consider the circuits* $\mathsf{Enc} : \mathbb{F}^n \to \mathbb{F}^{n + k \cdot n'}$ *and* $\mathsf{Dec} : \mathbb{F}^{n + k \cdot n'} \to \mathbb{F} \times \mathbb{F}^n$ *which are defined as follows.*

– *The circuit* Enc *on input* $\mathbf{x} \in \mathbb{F}^n$ *performs the following:*

1. *Computes* $\mathbf{x}' \leftarrow \mathsf{Enc}_n(\mathbf{x})$ *and for all* $1 \leq i \leq n'$ *computes* $\widehat{x}_i \leftarrow \mathsf{Enc}_{amd}(x'_i)$*.*
2. *Outputs* $(\mathbf{x}, \widehat{\mathbf{x}})$*.*

– *The circuit* Dec *on input* $(\mathbf{x}, \widehat{\mathbf{x}})$ *performs the following:*

1. *Computes* $\mathbf{x}' \leftarrow \mathsf{Enc}_n(\mathbf{x})$*.*
2. *For all* $1 \leq i \leq n'$ *computes* $(f_i, y'_i) \leftarrow \mathsf{Dec}_{amd}(\widehat{x}_i)$ *and* $f'_i \leftarrow x'_i - y'_i$*.*
3. *In case there exists an* $1 \leq i \leq n'$ *such that* $f_i \neq 0$ *or* $f'_i \neq 0$*, outputs* $(1, 0^n)$*. Otherwise, outputs* $(0, \mathbf{x})$*.*

Theorem 16. *For any positive integer* n*, the pair of circuits* $\mathsf{Enc}, \mathsf{Dec}$ *of Construction 7 is an* $(n, t, \epsilon(t)^d)$*-relaxed AMD encoding scheme.*

Proof. The correctness property follows directly from the construction. We now prove the relaxed additive soundness property. Let $\mathbf{x} \in \mathbb{F}^n$ be an input to Enc, and $\mathbf{A} = (\mathbf{a}, \mathbf{b}) \in \mathbb{F}^n \times \mathbb{F}^{kn'}$ be an additive attack on the outputs of Enc. We consider two possible cases.

1. $\mathbf{a}$**=0.** In this case, the additive attack does not attempt to alter the value $\mathbf{x}$ passed from Enc to Dec, so $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(\mathbf{x}, 1^t\right) + (\mathbf{a}, \mathbf{b}), 1^t\right) \notin \mathsf{ERR} \cup \{(0, \mathbf{x})\}\right] = 0$.

2. $\boldsymbol{a_i \neq 0}$ **for some** $\boldsymbol{1 \leq i \leq n}$**.** In this case, let $\mathcal{I} = \{i : (\mathsf{Enc}_n(\mathbf{x}+\mathbf{a}))_i \neq (\mathsf{Enc}_n(\mathbf{x}))_i\}$. For an additive attack to successfully cause Dec to output some $\tilde{\mathbf{x}} \neq \mathbf{x}$, it must be the case that $x_i'^{\mathbf{A}} = y_i'^{\mathbf{A}}$ for every $i \in \mathcal{I}$, where $x_i'^{\mathbf{A}} = (\mathsf{Enc}_n(\mathbf{x}+\mathbf{a}))_i$, and $y_i'^{\mathbf{A}} = \mathsf{Dec}_{amd}(\widehat{x}_i + \mathbf{b}_i) = \mathsf{Dec}_{amd}\left((\mathsf{Enc}_{amd}((\mathsf{Enc}_n(\mathbf{x}))_i)) + \mathbf{b}_i\right)$ (the right equality follows from the definition of $\widehat{\mathbf{x}}$). For every $i \in \mathcal{I}$, if $\mathbf{b}_i = 0$ then by the correctness of $(\mathsf{Enc}_{amd}, \mathsf{Dec}_{amd})$, $\mathsf{Dec}_{amd}\left(\mathsf{Enc}_{amd}((\mathsf{Enc}_n(\mathbf{x})_i)) + \mathbf{b}_i\right) = \mathsf{Dec}_{amd}\left(\mathsf{Enc}_{amd}((\mathsf{Enc}_n(\mathbf{x}))_i)\right) = (\mathsf{Enc}_n(\mathbf{x}))_i \neq (\mathsf{Enc}_n(\mathbf{x}+\mathbf{a}))_i$ (the rightmost equality holds since $i \in \mathcal{I}$), so Dec outputs ERR (with probability 1); otherwise the additive soundness of $(\mathsf{Enc}_{amd}, \mathsf{Dec}_{amd})$ guarantees that $f_i \neq 0$ only with probability $\epsilon(t)$. Moreover, the relative distance property of the ECC guarantees that $|\mathcal{I}| \geq d$. Consequently, $\Pr\left[\mathsf{Dec}\left(\mathsf{Enc}\left(\mathbf{x}, 1^t\right) + (\mathbf{a}, \mathbf{b}), 1^t\right) \notin \mathsf{ERR} \cup \{(0, \mathbf{x})\}\right] \leq \epsilon(t)^d$. □

Instantiating Construction 7 with the ECC of Theorem 15, we obtain the following result.

Theorem 17. *For any positive integer n there exists an $(n, t, 2^{-\Omega(n)})$-relaxed AMD encoding scheme with encoding and decoding circuits of size $\Theta(n)$.*

Theorem 17 can be used to relate the open question of constructing actively-secure two-party protocols with constant computational overhead to the simpler questions of constructing passively-secure honest-majority MPC protocols, and correct-only honest-majority MPC protocols, with constant computational overhead. We first show that actively secure 2-party MPC protocols in the OT-hybrid model, with constant computational overhead, can be constructed from additively-secure circuits with constant computational overhead. Formally,

Claim 18 *Assume that any boolean circuit C admits an additively-secure implementation $\widehat{\mathsf{C}}$ with constant computational overhead. Then there exists an actively secure 2-party protocol π for evaluating C in the OT-hybrid model with constant computational overhead.*

Proof (sketch). The work of [11] observed that the effect of an active attack on an arithmetic version of the *passively-secure* GMW protocol [12] π_{GMW} (in the OLE-hybrid model) corresponds to an additive attack on the underlaying circuit being evaluated. This observation holds in the binary case as well (where π' is executed in the OT-hybrid model). Thus, given an additively-secure implementation $\widehat{\mathsf{C}}$ of C with constant computational overhead, one can construct an actively secure 2-party protocol π for evaluating C in the OT-hybrid model, with constant computational overhead, simply by running π_{GMW} on $\widehat{\mathsf{C}}$. □

The following corollary reduces the task of constructing actively-secure 2-party protocols in the OT-hybrid model, with constant computational overhead, to the following simpler tasks:

1. Constructing passively-secure 2-party protocols in the OT-hybrid model with constant computational overhead.

2. Constructing correct-only (as per Definition 6) 2-party protocols in the OT-hybrid model with constant computational overhead.

Corollary 1. *If there exist both correct-only MPC protocols, and passively secure MPC protocols, with constant computational overhead, then there is a secure 2-party protocol in the OT-hybrid model with constant computational overhead.*

Proof (sketch). Let π_1, π_2 be correct-only, and passively secure, protocols (resp.) with constant computational overhead. The protocol π for evaluating a circuit C is obtained by applying Claim 18 to the circuit $\widehat{\mathsf{C}}_{\mathsf{sec}}$ constructed below.

1. Construct an additively-correct implementation $\widehat{\mathsf{C}}_{\mathsf{corr}}$ of C (as per Definition 1) with constant computational overhead using π_1, Construction 4, and the relaxed AMD codes of Theorem 17.
2. Construct an additively-secure implementation $\widehat{\mathsf{C}}_{\mathsf{sec}}$ of C (as per Definition 2) with constant computational overhead using π_2, Construction 6, and the relaxed AMD codes of Theorem 17.

By repeating the analysis of Constructions 4 and 6 while replacing the protocol from [6] with π_1, π_2, we obtain that π has constant computational overhead. Regarding the security of π, the only difference from the analysis in Sections 4 and 5 is that π employs a *relaxed* AMD encoding scheme (whereas Constructions 4 and 6 used (standard) AMD encoding schemes). However, since AMD codes are used in these constructions only to protect the communication channels of π_1, π_2, then relaxed additive soundness suffices for the analysis since it guarantees that no attack can alter the values of these messages. □

Acknowledgments

The first author is a member of the Check Point Institute for Information Security and was supported by ERC starting grant 259426; by the Blavatnik Interdisciplinary Cyber Research Center; by the Israeli Centers of Research Excellence I-CORE program (center 4/11); by the Leona M. & Harry B. Helmsley Charitable Trust; and by NATO's Public Diplomacy Division in the Framework of "Science for Peace".

The second author was supported by ERC starting grant 259426, ISF grant 1709/14, BSF grant 2012378, a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

The third author was supported by ERC starting grant 259426 and a Check Point Institute for Information Security grant for graduate students and post-doctoral fellows.

References

1. Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In *CRYPTO 2016*, pages 593–618, 2016.

2. Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *TCC 2005*, pages 342–362. Springer, 2005.
3. Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Eurocrypt 2008*, pages 471–488. Springer, 2008.
4. Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO 2005*, pages 378–394, 2005.
5. Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO 2006*, pages 501–520. Springer, 2006.
6. Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Eurocrypt 2010*, pages 445–465. Springer, 2010.
7. R. Dobrushin and E. Ortyukov. Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements. *Problems of Information Transmission*, 23(2):203–218, 1977.
8. Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In *CRYPTO 2006*, pages 232–250. Springer, 2006.
9. Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In *ITCS 2014*, pages 169–182. ACM, 2014.
10. Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou. Efficient multi-party computation: From passive to active security via secure SIMD circuits. In *CRYPTO 2015*, pages 721–741, 2015.
11. Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *STOC 2014*, pages 495–504, 2014. Full version in Cryptology ePrint Archive: Report 2015/154.
12. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC 1987*, pages 218–229. ACM, 1987.
13. Dai Ikarashi, Ryo Kikuchi, Koki Hamada, and Koji Chida. Actively private and correct MPC scheme in $t \leq n/2$ from passively secure schemes with small overhead. *IACR Cryptology ePrint Archive*, 2014:304, 2014.
14. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO 2003*, pages 145–161, 2003.
15. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC 2008*, pages 433–442, 2008.
16. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO 2008*, pages 572–591, 2008.
17. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *CRYPTO 2015 Part I*, pages 724–741, 2015.
18. Nicholas Pippenger. On networks of noisy gates. In *FOCS 1985*, pages 30–38. IEEE, 1985.
19. Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.
20. J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 43–98. Princeton University Press, 1956.

A Additive Correctness Without a Decoder: Feasibility

In this section we construct a $2^{-\Omega(\sigma)}$ additively-correct circuit compiler $\mathsf{Comp}^{\mathsf{In}}$ which on input a circuit C outputs a circuit $\widehat{\mathsf{C}}$ such that $\left|\widehat{\mathsf{C}}\right| = \sigma \cdot |C|$.

We present a method of amplifying security of additively-correct constructions through repetition. The natural approach is for the compiled circuit $\widehat{\mathsf{C}}$ to contain σ copies of an ϵ-additively-correct implementation $\widehat{\mathsf{C}}_\epsilon$ that are all evaluated on the input x. This approach raises two issues. First, an additive attack $\mathbf{A}$ on $\widehat{\mathsf{C}}$ consists of σ additive attacks $\mathbf{A}_1, \cdots, \mathbf{A}_\sigma$ on the σ copies of $\widehat{\mathsf{C}}_\epsilon$. For each copy $\widehat{\mathsf{C}}_{\epsilon,i}$, the additive-security of $\widehat{\mathsf{C}}_\epsilon$ guarantees that there exists an "ideal" additive attack on the inputs and outputs of $\widehat{\mathsf{C}}_{\epsilon,i}$ such that except with probability ϵ, the output of $\widehat{\mathsf{C}}_{\epsilon,i}$ under the additive attack $\mathbf{A}_i$ equals its output under the corresponding ideal additive attack. However, if different copies are evaluated under *different* additive attacks then *the corresponding ideal additive attacks may also be different.* This in effect causes different copies to be evaluated *on different inputs.* To overcome this, before compiling C we first modify it to take inputs encoded using a robust AMD encoding scheme. Since such codes guarantee additive correctness with an ideal additive attack that is independent of the additive attack on the outputs of the encoder, this guarantees the existence of a *single* additive attack that *simultaneously* corresponds to the additive attacks on *all* copies.

The second issue is that $\widehat{\mathsf{C}}$ should verify that all copies have the same output, and this should be *performed in the presence of additive attacks.* Therefore, before compiling C we first transform it into a circuit that MACs its output. Thus, the test comparing the MACs of two inconsistent outputs will fail, *even if it is performed under an additive attack.* These alterations of C are summarized in the following construction of C_{AUG}.

Construction 8 ($C_{\mathsf{AUG}}, C_{\mathsf{MAC}}$) *Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be an arithmetic circuit over a finite field $\mathbb{F}$, $\sigma \in \mathbb{N}$ be a length parameter, and $(\mathsf{Enc}, \mathsf{Dec})$ be an $(n+\sigma, l, \sigma, \epsilon)$-robust AMD encoding scheme (as in Definition 5). The circuit $C_{\mathsf{AUG}} : \mathbb{F}^l \to \mathbb{F}^\sigma \times \mathbb{F}^k \times (\mathbb{F}^\sigma)^k$, on input $\mathbf{x}' \in \mathbb{F}^l$, performs the following.*

1. *Compute $(f, (\mathbf{u}, \mathbf{x})) \leftarrow \mathsf{Dec}(\mathbf{x}')$, where $f \in \mathbb{F}$, $\mathbf{u} = (u_1, \cdots, u_\sigma) \in \mathbb{F}^\sigma$, and $\mathbf{x} \in \mathbb{F}^n$. (Intuitively, $\mathbf{x}$ is the input to the original circuit, and $\mathbf{u}$ will be used to MAC the outputs.)*
2. *Computes $\mathbf{z} \leftarrow C(\mathbf{x})$, where $\mathbf{z} \in \mathbb{F}^k$.*
3. *For all $1 \leq i \leq k$, computes $(z'_{i,1}, \cdots, z'_{i,\sigma}) \leftarrow (u_1 \cdot z_i, \cdots, u_\sigma \cdot z_i)$. (This step MACs each output coordinate z_i.)*
4. *Outputs $\left(f \cdot \mathbf{s}, \mathbf{z}, (z'_{1,1}, \cdots, z'_{1,\sigma}), \cdots, (z'_{k,1}, \cdots, z'_{k,\sigma})\right)$ where $\mathbf{s} \in_R \mathbb{F}^\sigma$.*

The circuit $C_{\mathsf{MAC}} : \mathbb{F}^n \times \mathbb{F}^\sigma \to \mathbb{F}^k \times (\mathbb{F}^\sigma)^k$ is obtained from C in a similar manner, except that its input is $(\mathbf{u}, \mathbf{x})$ ("in the clear"), and so it does not perform the input decoding of Step 1 above, and does not output a list of flags.

Construction 9 *Let $\mathbb{F}$ be a finite field, $\sigma \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ be an input length parameter, and $k, k' \in \mathbb{N}$ be output length parameters. Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be an arithmetic circuit over $\mathbb{F}$, and $(\mathsf{Enc}, \mathsf{Dec})$ be an $(n+\sigma, l, \sigma, 2^{-\sigma})$-additively robust AMD encoding scheme. Let C_{MAC} and C_{AUG} denote the circuits obtained by applying Construction 8 to C. Notice that the inputs to C_{MAC} are $\mathbf{x} \in \mathbb{F}^n$ and a MAC key $\mathbf{u} \in \mathbb{F}^\sigma$, and its output is in $\mathbb{F}^{k+\sigma k}$, whereas the inputs to C_{AUG} are robust-AMD encodings of $\mathbf{u}, \mathbf{x}$, and its output is in $\mathbb{F}^{\sigma+k+\sigma k}$. Let $\widehat{\mathsf{C}}_{\mathsf{AUG}}$ be an ϵ-additively-correct implementation of C_{AUG} with t flags. The randomized circuit $\widehat{\mathsf{C}} : \mathbb{F}^n \to \mathbb{F}^\sigma \times \mathbb{F}^k$, on input $\mathbf{x} \in \mathbb{F}^n$, operates as follows.*

1. *Generates a random MAC key $\mathbf{u} \in \mathbb{F}^\sigma$. ($\mathbf{u}$ will be used to MAC the outputs of C in C_{AUG}.)*
2. *Computes $\mathbf{z}_{\mathsf{MAC}} \leftarrow C_{\mathsf{MAC}}(\mathbf{x}, \mathbf{u})$, and we denote $\mathbf{z}_{\mathsf{MAC}} = (\mathbf{z}, (\widetilde{z}_{1,1}, \cdots, \widetilde{z}_{1,\sigma}), \cdots, (\widetilde{z}_{k,1}, \cdots, \widetilde{z}_{k,\sigma}))$. (This step evaluates C once directly, and MACs the outputs.)*
3. *Computes $(\mathbf{x}', \mathbf{u}') \leftarrow \mathsf{Enc}((\mathbf{x}, \mathbf{u}))$. (This step encodes the inputs to C_{AUG}.)*
4. *For all $1 \le i \le \sigma$, computes $(\mathbf{f}_i, \mathbf{y}_i) \leftarrow \widehat{\mathsf{C}}_{\mathsf{AUG},i}(\mathbf{x}', \mathbf{u}')$, where $\widehat{\mathsf{C}}_{\mathsf{AUG},1}, \cdots, \widehat{\mathsf{C}}_{\mathsf{AUG},\sigma}$ denote σ separate copies of $\widehat{\mathsf{C}}_{\mathsf{AUG}}$.*
5. *For all $1 \le i \le \sigma$, interprets $\mathbf{y}_i$ as $\Big((f'_{i,1}, \cdots, f'_{i,\sigma}), \mathbf{z}'_i, (z'_{i,1,1}, \cdots, z'_{i,1,\sigma}), \cdots, (z'_{i,k,1}, \cdots, z'_{i,k,\sigma})\Big)$. (The output of each copy $\widehat{\mathsf{C}}_{\mathsf{AUG},i}$ is interpreted as σ flags $f'_{i,1}, \cdots, f'_{i,t}$ indicating whether the decoding of $\mathbf{x}', \mathbf{u}'$ succeeded, a k-length output, and σ MACs for every output coordinate.)*
6. *For all $1 \le i, j \le \sigma$, computes $f''_{i,j} \leftarrow \sum_{l=1}^{k} (\widetilde{z}_{l,i} - z'_{j,l,i}) r_{i,j,l}$ where all the $r_{i,j,l}$ are generated uniformly from $\mathbb{F}$. (This step compares the MACed outputs computed by the ϵ-additively-correct implementations in Step 3, with the MACed output computed directly in Step 2. Specifically, $f''_{i,j}$ compares the i'th MAC of the j'th copy, to the i'th MAC of Step 2.)*
7. *For all $1 \le i \le \sigma$, computes $f''_i \leftarrow \sum_{j=1}^{\sigma} f''_{i,j} r_{i,j} + \sum_{j=1}^{\sigma} r'_{i,j} f'_{j,i}$, where all the $r_{i,j}$ and $r'_{i,j}$ are generated uniformly from $\mathbb{F}$. (This step checks that all copies agree on the i'th MAC, and in addition, that the decoding of the inputs in all copies succeeded.)*
8. *For all $1 \le i \le \sigma$ compute $g_i \leftarrow \sum_{j=1}^{\sigma} (\sum_{u=1}^{t} f_{j,u} \tilde{r}_{i,j,u} + f''_j \tilde{r}'_{i,j,u})$ where all the $\tilde{r}_{i,j}$ and $\tilde{r}'_{i,j}$ are generated uniformly from $\mathbb{F}$. (This step checks that the computation in the i'th ϵ-additively-correct implementation succeeded, and in addition, that the input decoding in all copies succeeded, and they all agree on all MACs.)*
9. *Output $((g_1, \cdots, g_\sigma), \mathbf{z})$.*

In the full version of the paper we prove the following:

Theorem 19. *For any field $\mathbb{F}$, arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ and security parameter σ, the circuit $\widehat{\mathsf{C}}$ obtained by applying Construction 9 to C is a $2^{-\Omega(\sigma)}$-additively-correct implementation of C. Moreover, $|\widehat{\mathsf{C}}| = poly(\sigma, |C|)$.*