Efficient Multi-Party Computation: from Passive to Active Security via Secure SIMD Circuits

Daniel Genkin[1 2], Yuval Ishai[1], and Antigoni Polychroniadou[3]

[1] Technion
{danielg3,yuvali}@cs.technion.ac.il
[2] Tel Aviv University
[3] Aarhus University
antigoni@cs.au.dk

Abstract. A central problem in cryptography is that of converting protocols that offer security against passive (or semi-honest) adversaries into ones that offer security against active (or malicious) adversaries. This problem has been the topic of a large body of work in the area of secure multiparty computation (MPC). Despite these efforts, there are still big efficiency gaps between the best protocols in these two settings. In two recent works, Genkin et al. (STOC 2014) and Ikarashi et al. (ePrint 2014) suggested the following new paradigm for efficiently transforming passive-secure MPC protocols into active-secure ones. They start by observing that in several natural information-theoretic MPC protocols, an arbitrary active attack on the protocol can be perfectly simulated in an ideal model that allows for *additive* attacks on the arithmetic circuit being evaluated. That is, the simulator is allowed to (blindly) modify the original circuit by adding an arbitrary field element to each wire. To protect against such attacks, the original circuit is replaced by a so-called *AMD circuit*, which can offer protection against such attacks with constant multiplicative overhead to the size.

Our motivating observation is that in the most efficient known information-theoretic MPC protocols, which are based on packed secret sharing, it is *not* the case that general attacks reduce to additive attacks. Instead, the corresponding ideal attack can include limited forms of linear combinations of wire values. We extend the AMD circuit methodology to so-called *secure SIMD circuits,* which offer protection against this more general class of attacks.

We apply secure SIMD circuits to obtain several asymptotic and concrete efficiency improvements over the current state of the art. In particular, we improve the additive per-layer overhead of the current best protocols from $O(n^2)$ to $O(n)$, where n is the number of parties, and obtain the first protocols based on packed secret sharing that "natively" achieve near-optimal security without incurring the high concrete cost of Bracha's committee-based security amplification method.

Our analysis is based on a new modular framework for proving reductions from general attacks to algebraic attacks. This framework allows us to reprove previous results in a conceptually simpler and more unified way, as well as obtain our new results.

1 Introduction

1.1 Overview

Secure multiparty computation (MPC) is a central research area in cryptography. An MPC protocol allows $n \geq 2$ parties to compute a function of their inputs without compromising the privacy of the inputs or the correctness of the outputs. This should hold even if some of the parties are corrupted by an adversary. Since its introduction in the 1980s [20, 12, 7, 2], there has been a rich body of work dealing with many aspects of the problem, with a major focus on efficiency.

The difficulty of designing MPC protocols depends largely on the power of the adversary. An important distinction is between MPC protocols that offer security against *passive* (or semi-honest) adversaries, who follow the protocol's specification but try to learn information from messages they receive, and security against *active* (or malicious) adversaries, who are allowed to deviate from the protocol's specification in arbitrary ways. The security guarantees in the passive case are weaker, but the protocols are simpler and more efficient.

A common paradigm for designing actively secure MPC protocols (namely, ones that are secure against active adversaries) is to start with a passively secure protocol and then convert it into an actively secure protocol. Some relevant techniques include general-purpose "GMW-style" compilers that employ zero-knowledge proofs [12, 6], ad-hoc protocols for verifying the correct execution of subprotocols [2, 7], cut-and-choose techniques [18], or "MPC in the head" [15, 16]. These techniques typically involve a significant overhead.

A different technique, which in some cases provides better results, was recently proposed independently by Genkin et al. [11] and Ikarashi et al. [14]. These works observe that in several known passively secure protocols for evaluating arithmetic circuits, the effect of any active adversary is limited to an *additive attack* on the circuit wires. That is, everything that an adversary can achieve by attacking the real protocol for evaluating C he could have also achieved by attacking an ideal circuit evaluation process in which he can *blindly* add a field element of his choice to each wire in C. In the following, we refer to such protocols as *additively corruptible protocols*. To secure such a protocol against active adversaries, it is enough to run it on a so-called *AMD circuit* $\overline{\mathsf{C}}$ – a randomized circuit which is functionally equivalent to C but additionally offers resistance against additive attacks.[4] The results of [11, 14] simplify feasibility results in the information-theoretic setting and obtain efficiency improvements, closing some previous asymptotic efficiency gaps between passively secure and actively secure protocols. This applies to the best known protocols that tolerate an optimal number of corrupted parties (i.e., $t < n/2$ parties using secure point-to-point channels or $t < n$ parties in a suitable hybrid model).

Our motivating observation is that the best information-theoretic MPC protocols that tolerate a slightly sub-optimal number of corrupted parties (e.g.,

[4] The work of [14] does not explicitly construct AMD circuits, but implicitly relies on a simple construction of AMD circuits that tolerate a restricted class of additive attacks which suffices in some cases.

$t < 0.49n$) are *not* additively corruptible. These protocols replace the standard secret sharing used in optimally resilient protocols by a more efficient *packed secret sharing* technique, and as a result provide better asymptotic efficiency. The ideal attack corresponding to an active adversary attacking these protocols can include a limited form of linear combinations that combine multiple wire values.[5] As a result, the techniques of [11, 14] do not apply to such protocols. In the following, we refer to such protocols *linearly corruptible protocols.*

A second disadvantage of the techniques of [11, 14] is that they are tailored to specific protocols. In particular, the part of the analysis that maps general attacks to additive attacks is done in an ad-hoc way per protocol without a unified framework that captures all additively corruptible protocols.

1.2 Our Contribution

In this paper we address both issues outlined above. First, we present a new general framework for proving that a passively secure protocol is additively or linearly corruptible. This framework is used to reprove previous results from [11] in a more unified way, and is also used to prove our new results. Second, we extend the AMD circuit constructions from [11] to offer security against linear attacks. We use these two types of results to close previous efficiency gaps between passively secure and actively secure information-theoretic protocols based on packed secret sharing.

We consider two regimes for such protocols: the *single input, single circuit* regime and the *Franklin and Yung (FY)* [10] regime for simultaneously evaluating ℓ copies of the circuit on different inputs. Notice that the latter is a special case of the former that allows for simpler and more efficient solutions. Currently, all actively secure protocols that rely on packed secret sharing (in both regimes) employ verification methods that introduce at least a quadratic overhead in the number of parties n, for each circuit layer. We reduce this overhead to quasi-linear (or linear in the FY regime), as in the best previous passively secure protocols. In the FY regime, by evaluating the circuit on $\ell = \Omega(n)$ inputs, the amortized per-layer overhead is reduced to constant, leading to the first actively secure protocols whose amortized communication complexity is only $O(|C| + n)$ even for circuits that are very narrow and deep. See Table 1 for a more detailed account of our results and a comparison with previous results.

In addition, we point out that the concrete efficiency of DIK-style protocols [8, 1] (see [17]c entry of Table 1) involves prohibitively large constants when applied with near-optimal security threshold. Indeed, the threshold obtained directly by [8] is $t < n/4$ which is quite far from the optimal bound of $n/2$. To improve on this threshold, a general technique due to Bracha [5] is applied for boosting the resilience. The basic idea is that a constant-size committee runs an optimally resilient protocol to emulate the role of each server in the low-threshold protocol. While this technique can be implemented with a constant

[5] In the full version we illustrate the necessity of extending the attack model to linear.

[6] In the client-server model, where m (n) is the number of clients (servers).

Ref.	**Adv.**	**Copies**	**Resilience**	**Communication complexity**
[12]	passive	1	$\|\mathcal{T}\| < n$	$O(n^2\|\mathsf{C}\|)$ for boolean circuits
[17]	passive	1	$\|\mathcal{T}\| < n$	$O(n^2\|\mathsf{C}\|)$
[2]	passive	1	$\|\mathcal{T}\| < n/2$	$O(n^2\|\mathsf{C}\|)$
[9]	passive	1	$\|\mathcal{T}\| < n/2$	$O(n\|\mathsf{C}\| + n^2)$
[10]	passive	$\Theta(n)$	$\|\mathcal{T}\| < (1/2 - \epsilon)n$	$O(n\|\mathsf{C}\|)$
[8]a	passive	$\Theta(n)$	$\|\mathcal{T}\| < (1/2 - \epsilon)n$	$O(\|\mathsf{C}\| + n)$
[8]b	passive	1	$\|\mathcal{T}\| < (1/2 - \epsilon)n$	$\widetilde{O}(\|\mathsf{C}\| + n \cdot d_{\mathsf{C}})$
[17]a	active	1	$\|\mathcal{T}\| < n$	$O(n^2\|\mathsf{C}\| + \log\|\mathbb{F}\| \cdot d_{\mathsf{C}})$
[11]	active	1	$\|\mathcal{T}\| < n$	$O(n^2\|\mathsf{C}\|)$
this work	active	1	$\|\mathcal{T}\| < n$	$O(n^2\|\mathsf{C}\|)$
[3]	active	1	$\|\mathcal{T}\| < n/2$	$O(n\|\mathsf{C}\| + n^2 \log n \cdot d_{\mathsf{C}}) +$ poly(n)
[11]	active	1	$\|\mathcal{T}\| < n/2$	$O(n\|\mathsf{C}\| + n^2)$
this work	active	1	$\|\mathcal{T}\| < n/2$	$O(n\|\mathsf{C}\| + n^2)$
[17]b	active	$\Theta(n)$	$\|\mathcal{T}\| < (1/2 - \epsilon)n$	$O(\|\mathsf{C}\| + n \cdot d_{\mathsf{C}})$
[17]c	active	$\Theta(n)$	$\|\mathcal{T}\| < (1/2 - \epsilon)n$	$O(\|\mathsf{C}\| + m \cdot d_{\mathsf{C}})^6$
this work	**active**	$\boldsymbol{\Theta(n)}$	$\boldsymbol{\|\mathcal{T}\| < (1/2 - \epsilon)n}$	$\boldsymbol{O(\|\mathsf{C}\| + n)}$
[8]c	active	1	$\|\mathcal{T}\| < (1/2 - \epsilon)n$	$\widetilde{O}(\|\mathsf{C}\| + n^2 \cdot d_{\mathsf{C}})$
this work	**active**	**1**	$\boldsymbol{\|\mathcal{T}\| < (1/2 - \epsilon)n}$	$\boldsymbol{\widetilde{O}(\|\mathsf{C}\| + n \cdot d_{\mathsf{C}} + n^2)}$

Table 1. Comparison of information-theoretic MPC protocols for arithmetic circuits. In the above, n is the number of parties, ϵ is an arbitrary small positive constant, C is an arithmetic circuit or an SIMD circuit, d_{C} is the multiplicative depth of C, and $\mathcal{T}$ is the set of corrupted parties such that $|\mathcal{T}| \leq t$. The copies column indicates the number of simultaneously evaluated circuit copies. Passively secure protocols achieve perfect security while actively secure protocols realize C (with abort) with at most $O(1/|\mathbb{F}|)$ simulation error. The communication complexity column counts the total number of field elements exchanged between the parties. For the case of simultaneous evaluation of multiple copies, we count the amortized cost for evaluating a single copy of C. The protocols having resilience $|\mathcal{T}| < n$ are constructed on the OT or OLE hybrid model. Note that the $\widetilde{O}$ notation suppresses logarithmic factors.

multiplicative overhead, this constant is very large. Our actively secure protocols natively achieve a near-optimal security threshold with a low overhead, inheriting this feature from the passively secure protocols on which they are based.

A key ingredient in our results is an extension of the additive attacks model considered in [11, 14], which we now explain in more detail. Protocols that utilize packed secret sharing typically operate on SIMD *circuits*. An SIMD circuit is a generalization of arithmetic circuits, composed by ℓ-gates which get as input two wire *bundles* of size ℓ output a wire output *bundle* of size ℓ obtained by performing ℓ *point-wise* multiplications, additions and subtractions in parallel. Thus, SIMD circuits simultaneously evaluate ℓ copies of the same arithmetic circuit, on different inputs. Next, for protocols based on packed secret sharing, the ideal attack corresponding to deviations made by an active adversary can include a limited form of linear combinations of wire values. Thus, we extend the additive attacks considered in [11] to capture a stronger class of attacks, called *linear* attacks, applied to SIMD circuits.

A *linear* attack on an SIMD circuit changes the computation of a *multiplication* ℓ-gate by adding to the gate's output bundle a linear function $f : \mathbb{F}^{2\ell} \rightarrow \mathbb{F}^{\ell}$ of all the wires in the gate's two input bundles. In addition, we also allow a linear attack to specify an *additive* attack on all wire bundles inside the SIMD circuit. We note that for the case where $\ell = 1$ linear attacks are equivalent to additive attacks (see Section 2.2 for details).

In the sequel, we prove that for natural protocols based on packed secret sharing, any deviation made by an active adversary actually corresponds to a *linear* attack on the underlying SIMD circuit.

2 Detailed Overview of Results

2.1 Actively Secure MPC Protocols from AMD/SIMD Circuits

Our approach for constructing actively secure MPC protocols is as follows. We present a general *framework* and prove that any passively secure protocol π, satisfying the framework's requirements is indeed additively or linearly corruptible depending on whether π uses packed secret sharing or not. Next, in order to transform any passively secure protocol for evaluating a circuit C, which meets the framework's requirement, into an actively secure protocol, we apply the *same* passive protocol on a different circuit $\overline{\mathsf{C}}^{\mathsf{AUG}}$ which is essentially the secure version of C. We thus transfer the responsibility of handling the consequences resulting from an active adversary deviating from the protocol, to $\overline{\mathsf{C}}^{\mathsf{AUG}}$.

We now describe different applications of our framework for existing MPC protocols. See Table 1 for a concise summary.

Applying our framework to an arithmetic version of the passively secure GMW protocol [12, 17], in Theorem 10 we match the results of [11, Theorem 1.5] obtaining an actively secure protocol for computing a circuit C, without an honest majority, using $O(n^2|C|)$ calls to an OLE-oracle.[7] In the honest majority setting, applying our framework to the passively secure DN protocol [9], we match the results of [11, Theorem 1.4] and [14] obtaining an actively secure protocol with communication complexity of $O(n|C| + n^2)$ field elements.

Next, in the FY regime, by applying our framework to the passively secure DIK protocol ([8]a), we improve the result of [17]b by eliminating the dependence of the additive term on the depth of the circuit.

Theorem 1. *Let n, t, ℓ be positive integers such that $n = 2t + 2\ell - 1$ and let C be an n-party SIMD ℓ-circuit over a finite field $\mathbb{F}$. Then, there exists a protocol π, in the FY regime, that (t, ϵ)-securely computes C with abort for $\epsilon = O(\ell \log \ell / |\mathbb{F}|)$ and with communication complexity of $O(n|\mathsf{C}| + n^2)$ field elements. Setting $\ell = \Theta(n)$ yields an amortized communication complexity of $O(|\mathsf{C}| + n)$ field elements.*

Finally, applying our framework to the passively secure DIK protocol in the single input single circuit regime ([8]b), we improve the actively secure protocol of [8]c by reducing its additive term from $\widetilde{O}(n^2 \cdot d_{\mathsf{C}})$ to $\widetilde{O}(n \cdot d_{\mathsf{C}} + n^2)$.

[7] In fact, we slightly improve the construction of [11] by reducing the statistical simulation error from $O(|\mathsf{C}|/|\mathbb{F}|)$ to $O(1/|\mathbb{F}|)$.

Theorem 2. *Let n, t, ℓ be positive integers such that $n = 2t + 2\ell - 1$ and let C be an n-party circuit over a finite field $\mathbb{F}$. Then there exists an n-party protocol π, in the single circuit single input regime, that (t, ϵ)-securely computes C with abort for $\epsilon = O(\ell \log \ell / |\mathbb{F}|)$ and with communication complexity $\widetilde{O}\left((|\mathsf{C}|n + n^2 \cdot d_{\mathsf{C}})/\ell + n^2\right)$ field elements. By setting $\ell = \Theta(n)$ we obtain that the communication complexity of π is $\widetilde{O}\left(|\mathsf{C}| + n \cdot d_{\mathsf{C}} + n^2\right)$ field elements.*

2.2 Additive and Linear Attack Secure AMD/SIMD Circuits

We now define the notion of linear-attack security. Let C be a circuit to be computed. We say that a randomized SIMD circuit $\overline{\mathsf{C}}$ is an ϵ-linear-attack secure implementation of C if $\overline{\mathsf{C}}$ correctly computes C, when not attacked, and moreover any linear attack on $\overline{\mathsf{C}}$ has the same effect on the outputs of $\overline{\mathsf{C}}$ (up to ϵ statical error) as applying some *additive attack* on the inputs and outputs of $\overline{\mathsf{C}}$ alone.

Definition 1 (Linear-attack and additive-attack security). *A randomized SIMD circuit $\overline{\mathsf{C}}$ is said to be an ϵ-*linear-attack secure implementation *of a (possibly randomized) circuit $\mathsf{C} : (\mathbb{F}^\ell)^n \to (\mathbb{F}^\ell)^k$ if the following holds:*

- Completeness. *For all $\mathbf{x} \in (\mathbb{F}^\ell)^n$ it holds that $\overline{\mathsf{C}}(\mathbf{x}) \equiv \mathsf{C}(\mathbf{x})$.*
- Linear-Attack security. *For any circuit $\overline{\mathsf{C}}^{\mathbf{L}}$ obtained by subjecting $\overline{\mathsf{C}}$ to a linear attack $\mathbf{L}$, there exists $\mathbf{a}^{\mathsf{in}} \in (\mathbb{F}^\ell)^n$ and a distribution $\mathcal{A}^{\mathsf{out}}$ over $(\mathbb{F}^\ell)^k$ such that for any $\mathbf{x} \in (\mathbb{F}^\ell)^n$ it holds that $SD(\overline{\mathsf{C}}^{\mathbf{L}}(\mathbf{x}), \mathsf{C}(\mathbf{x} + \mathbf{a}^{\mathsf{in}}) + \mathcal{A}^{\mathsf{out}}) \leq \epsilon$, where SD denotes statistical distance between two distributions.*

Finally, we say that $\overline{\mathsf{C}}$ is an additive-attack-secure *implementation of C if $\overline{\mathsf{C}}$ has the same completeness property as above as well as the same security property with the linear attack $\mathbf{L}$ replaced by an additive attack $\mathbf{A}$.*

Notice that restricting Definition 1 for $\ell = 1$ yields exactly the model considered in [11]. This is the case since for non-SIMD circuits, any additive attack can be converted into a linear attack. Conversely, we notice that a linear attack on the output of a multiplication gate can be easily converted to an additive attack on its two inputs. Notice that this equivalence does not hold when $\ell > 1$.

In Section 5, we present a construction for securing circuits against additive attacks. While our construction has the same asymptotic efficiency as the construction of [11], it has much better *concrete* efficiency, as well as an improved soundness error of $O(1/|\mathbb{F}|)$ (compared to $O(|C|/|\mathbb{F}|)$ in [11]).

Theorem 3 (Cf. Theorem 6). *For any arithmetic circuit $\mathsf{C} : \mathbb{F}^n \to \mathbb{F}^k$ there exists a randomized circuit $\overline{\mathsf{C}} : \mathbb{F}^n \to \mathbb{F}^k$ such that $\overline{\mathsf{C}}$ is an ϵ-additive-attack secure implementation of C where $\epsilon = O(1/|\mathbb{F}|)$ and $|\overline{\mathsf{C}}| = O(|\mathsf{C}|)$.*

Next, departing from the case of $\ell = 1$, in the full version we present a construction for securing SIMD circuits against linear attacks.

Theorem 4. *For any SIMD circuit $\mathsf{C} : \left(\mathbb{F}^\ell\right)^n \to \left(\mathbb{F}^\ell\right)^k$ there exists a randomized SIMD circuit $\overline{\mathsf{C}}$ such that $\overline{\mathsf{C}}$ is an ϵ-linear-attack secure implementation of C where $\epsilon = O\left(\ell \log \ell / |\mathbb{F}|\right)$ and $|\overline{\mathsf{C}}| = O(|\mathsf{C}| + \log \ell)$.*

3 Our Techniques

3.1 Constructing Actively Secure MPC Protocols

Our framework for proving that a passively secure protocol π is in fact additively or linearly corruptible, consists of three steps. We point out that while these steps modify the original protocols, are only a *thought-experiment* used to prove the main claim about the effect of an active adversary on the underlying circuit that parties try to evaluate. The only *real* modification required to the protocol in order to transform it to an actively secure protocol, is to execute it on an additive-attack or linear-attack secure circuit (see below).

Step 1: Protocol randomization. In order to convert an active adversary controlling a set of parties $\mathcal{T}$ to an additive attack, we first transform a protocol π to another protocol $\pi^{\mathcal{T}}$ such that all the messages $m_{\overline{\mathcal{T}},\mathcal{T}}$ sent by the parties in $\overline{\mathcal{T}}$ to the parties in $\mathcal{T}$ (except during the last communication round) syntactically depend only on the randomness of π. In particular, we require that $m_{\overline{\mathcal{T}},\mathcal{T}}$ does not depend on the inputs $\mathbf{x}_{\overline{\mathcal{T}}}$ of the parties in $\overline{\mathcal{T}}$ or on the messages that the parties in $\overline{\mathcal{T}}$ receive during the protocol. In such case we say that $\pi^{\mathcal{T}}$ is $\mathcal{T}$-randomized.

We first show that for many natural MPC protocols, for any set of parties $\mathcal{T}$, such that $|\mathcal{T}|$ is below the privacy threshold of a protocol, it is possible to construct a $\mathcal{T}$-randomized protocol, $\pi^{\mathcal{T}}$, such that any deviation from π made by an active adversary has the same effect as performing the same deviation from $\pi^{\mathcal{T}}$. In this case we say that $\pi^{\mathcal{T}}$ is $\mathcal{T}$-equivalent to π. See Definition 3.

Notice that $\mathcal{T}$-randomization requirement is stronger than privacy. This is since $\mathcal{T}$-randomization requires that the *values* of $m_{\overline{\mathcal{T}},\mathcal{T}}$ do not depend on the inputs of the parties in $\overline{\mathcal{T}}$ or on messages that parties in $\overline{\mathcal{T}}$ received as opposed to privacy which makes a similar requirement on the *distribution* of $m_{\overline{\mathcal{T}},\mathcal{T}}$. See Step 2 for the necessity of the $\mathcal{T}$-randomization requirement.

Step 2: From general adversaries to additive attacks. We now reduce any adversary controlling a set of parties $\mathcal{T}$, attacking a $\mathcal{T}$-randomized protocol π, to an *additive* attack on the protocol circuit $\mathsf{C}_\pi$where C_π is a direct implementation of the arithmetic operations performed by π. C_π gets as input the inputs $\mathbf{x}$ of the parties in π as well the randomness $\mathbf{r}$ used in π. It then evaluates π on inputs $(\mathbf{x};\mathbf{r})$ and outputs the outputs of all the parties following an execution of $\pi(\mathbf{x};\mathbf{r})$.

Since π is $\mathcal{T}$-randomized we can simulate from the randomness $\mathbf{r}$ for π and from the inputs $\mathbf{x}_{\mathcal{T}}$, the view $\widetilde{u}_{\mathcal{T}}$ (except during the last round) of the parties in $\mathcal{T}$. Next, we determine the additive attack on C_π corresponding to an adversary Adv controlling the parties in $\mathcal{T}$ as follows. We first honestly simulate the parties in $\mathcal{T}$ on their view $\widetilde{u}_T$ and obtain the messages $\widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}}$ sent by the parties in $\mathcal{T}$ to the parties in $\overline{\mathcal{T}}$ during an honest execution of π. Next, we invoke Adv on the view $\widetilde{u}_{\mathcal{T}}$ and obtain the messages $\widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}}$ sent by the parties in $\mathcal{T}$ to the parties in $\overline{\mathcal{T}}$ during a real execution of π in the presence of Adv. Finally we determine the additive attack $\mathbf{A}$ on C_π by computing $\mathbf{A} \leftarrow \widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}} - \widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}}$.

Since π is $\mathcal{T}$-randomized, it is the case that inside C_π under the additive attack $\mathbf{A}$ it holds that $\widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}} = \widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}} + \mathbf{A}$, for any input $\mathbf{x}_{\overline{\mathcal{T}}}$ of the parties in $\overline{\mathcal{T}}$

as well as for any messages that these parties receive during π. We thus correctly simulate, inside C_π, the effect of Adv on π. Notice that this is not necessary true in case π is $\mathcal{T}$-private since for any selection of the randomness $\mathbf{r}$, the specific values of the messages sent by the parties in $\overline{\mathcal{T}}$ to Adv might depend on their inputs $\mathbf{x}_{\overline{\mathcal{T}}}$ to π. Since $\mathbf{x}_{\overline{\mathcal{T}}}$ is not known to the simulator, it cannot generate the correct view $\widetilde{u}_{\mathcal{T}}$ required in order to compute $\widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}}$ and $\widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}}$.

Step 3: Translate attacks on C_π to attacks on C. We translate additive attacks on C_π to equivalent attack on C. In Section 7, we present the notion of *homomorphic circuits* and prove that if a circuit C' is homomorphic to a circuit C then for any additive attack $\mathbf{A}'$ on C' there exists an equivalent additive attack $\mathbf{A}$ on C such that $\mathsf{C}^{\mathbf{A}}(\mathbf{x}) = \mathsf{C}'^{\mathbf{A}'}(\mathbf{x})$, for all $\mathbf{x}$. Next, extending the notion of circuit homomorphism to SIMD circuits, in the full version we define the notion of ℓ-homomorphic circuits and prove that if a circuit C' is ℓ-homomorphic to an SIMD circuit C, then for any additive attack $\mathbf{A}'$ on C' there exists an equivalent *linear* attack on C such that $\mathsf{C}^{\mathbf{L}}(\mathbf{x}) = \mathsf{C}'^{\mathbf{A}'}(\mathbf{x})$ for all $\mathbf{x}$.

Application to natural MPC protocols. In Section 8 we apply the above transformations on the arithmetic version of the passively secure GMW protocol, proving that it is additively corruptible. Next, in the full version we apply the above transformations to the passively secure DN and DIK protocols, proving that these protocols are additively and linear corruptible, respectively.

MPC protocols using linear or additive attack secure circuits. The notions of linear and additive-attack security only require that any attack will be equivalent to an additive attack on the inputs and the outputs of the evaluated circuit. Thus, directly executing an additively or linearly corruptible MPC protocol over an additive-attack secure or linear-attack secure circuit C still leaves the inputs and the outputs of C unprotected. Instead, before securing C against additive or linear attacks, we first modify C to $\mathsf{C}^{\mathsf{AUG}}$ which gets as inputs an AMD encoding of C's inputs and produces an encoding of C's outputs. We then transform $\mathsf{C}^{\mathsf{AUG}}$ to an additive-attack or linear-attack secure circuit $\overline{\mathsf{C}}^{\mathsf{AUG}}$ and evaluate $\overline{\mathsf{C}}^{\mathsf{AUG}}$ using a passively secure protocol, asking each party to locally compute an AMD encoding of the inputs as well as locally decode the outputs.

3.2 Securing Circuits Against Additive and Linear Attacks

We first present our techniques for additive-attack security (see Section 5). Given a circuit C, in the additive-attack secure version $\overline{\mathsf{C}}$ of C, every wire of C is paired with a wire that carries a corresponding MAC tag. Next, each gate in C is replaced by a small gadget computing the gate's result as well as its corresponding MAC tag. In addition, this gadget also gets as inputs the MAC tags corresponding to the gate's inputs. Using these tags, the gadget verifies that the gate's result was computed correctly. Notice that the MAC tag verification itself is also vulnerable to additive (and later linear) attacks. However, we construct the verification circuit in such a way that even in the presence of attacks, it outputs a random value if the MAC computation or MAC verification fails for some gate.

The basic additive-attack secure circuit compiler. Similar to [4, 11] we use a multiplicative MAC in order to additive-attack secure the output of each gate. Concretely, for each input gate $\mathtt{a}$, its corresponding MAC tag will be $\mathtt{a}' = \mathtt{a} \cdot \mathtt{v}$ where $\mathtt{v}$ is a randomly selected field element acting as the MAC key (fixed to be the same value for all gates). Next, for every addition gate $\mathtt{c} = \mathtt{a} + \mathtt{b}$ with inputs $\mathtt{a}$, $\mathtt{b}$ and associated MAC tags $\mathtt{a}'$, $\mathtt{b}'$, we compute the MAC tag $\mathtt{c}'$ associated with $\mathtt{c}$ directly by computing $\mathtt{c}' = \mathtt{a}' + \mathtt{b}'$.

For every multiplication gate $\mathtt{c} = \mathtt{a} \cdot \mathtt{b}$ with inputs $\mathtt{a}$, $\mathtt{b}$ and associated MAC tags $\mathtt{a}'$, $\mathtt{b}'$, we need to ensure the correct computation of $\mathtt{c} = \mathtt{a} \cdot \mathtt{b}$. Given a MAC tag of the expected result of $\mathtt{c}$ and the MAC tags of $\mathtt{a}, \mathtt{b}$, we could have verified that under an additive attack indeed $\mathtt{c} \cdot \mathtt{v} = \mathtt{a} \cdot \mathtt{b} \cdot \mathtt{v}$. Thus, we must somehow combine the (assumed to be correct) MAC tag values $\mathtt{a}' = \mathtt{a} \cdot \mathtt{v}$ and $\mathtt{b}' = \mathtt{b} \cdot \mathtt{v}$ in order to generate the tag of the expected result $\mathtt{a} \cdot \mathtt{b} \cdot \mathtt{v}$. Moreover, this tag generation must be done in such a way that ensures that no combination of attacks on the tag generation circuit and on the multiplication gate's actual computation, can produce an incorrect result without being detected.

In [11], this was solved by setting the MAC tag $\mathtt{c}'$ to be $\mathtt{c}' = \mathtt{a}' \cdot \mathtt{b}' = \mathtt{a} \cdot \mathtt{b} \cdot \mathtt{v}^2$. The construction of [11] was based on the fact that an additive attack $\mathbf{A}$ on the computation of $\mathtt{c}'^{\mathbf{A}} = (\mathtt{a}' + \mathbf{A}_{\mathtt{a}',\mathtt{c}'})(\mathtt{b}' + \mathbf{A}_{\mathtt{b}',\mathtt{c}'})$ introduces additional monomials of the form $\mathbf{A}_{\mathtt{b}',\mathtt{c}'} \cdot \mathtt{a}' = \mathbf{A}_{\mathtt{b}',\mathtt{c}'} \cdot \mathtt{av}$ or $\mathbf{A}_{\mathtt{a}',\mathtt{c}'} \cdot \mathtt{b}' = \mathbf{A}_{\mathtt{a}',\mathtt{c}'} \cdot \mathtt{bv}$ and it cannot introduce additional monomials of the form $\mathtt{a}' \cdot \mathtt{b}' = \mathtt{a} \cdot \mathtt{b} \cdot \mathtt{v}^2$, where $\mathbf{A}_{\mathtt{a}',\mathtt{c}'}$ denotes the attack $\mathbf{A}$ restricted to the wire connecting the gates $\mathtt{a}', \mathtt{c}'$ inside C. Next, in [11] it was shown that in case these additional monomials are present, they cannot be canceled out by any other combination of additive attacks, thereby making $\overline{\mathsf{C}}$ abort the computation by masking its results with a completely random value.

The main problem with the basic construction of [11] is that even when no attacks are present, the degree of the MAC key $\mathtt{v}$ inside $\mathtt{c}'$ increases from $\mathtt{v}$ to $\mathtt{v}^2$. This limits the construction of [11] to only low-degree circuits as well as requiring complicated ad-hoc gadgets to support addition and subtraction gates with MAC tags having different degrees of $\mathtt{v}$. Finally, in order to additive-attack secure arbitrary-degree circuits, [11] employs a degree reduction procedure vastly increasing the concrete overhead of the overall construction.

An efficient MAC combination gadget. In Construction 1, we solve the problem of combining MAC tags in a different way. Let $\mathtt{a}' = \mathtt{a} \cdot \mathtt{v}$ and $\mathtt{b}' = \mathtt{b} \cdot \mathtt{v}$. We first compute $\mathtt{c}'$ as $\mathtt{c}' = \mathtt{a}' \cdot \mathtt{b} = (\mathtt{a} \cdot \mathtt{v}) \cdot \mathtt{b}$. Moreover, we also compute $\mathtt{c}'' = \mathtt{a} \cdot \mathtt{b}' = \mathtt{a} \cdot (\mathtt{b} \cdot \mathtt{v})$. Therefore, if no additive attack is present, it is always the case that $\mathtt{c}' - \mathtt{c}'' = 0$. However, notice that an additive attack on $\mathtt{c}'$ can only produce monomials of the form $\mathbf{A}_{\mathtt{a}',\mathtt{c}'} \cdot \mathtt{b}$ or of the from $\mathbf{A}_{\mathtt{b},\mathtt{c}'} \cdot (\mathtt{a} \cdot \mathtt{v})$. In contrast, notice that any additive attack on $\mathtt{c}''$ can only produce monomials of the form $\mathbf{A}_{\mathtt{a},\mathtt{c}'} \cdot (\mathtt{b} \cdot \mathtt{v})$ and $\mathbf{A}_{\mathtt{b},\mathtt{c}'} \cdot \mathtt{a}$ which cannot be canceled out by any of the monomials produced by the attack on $\mathtt{c}'$. Thus, by checking that $\mathtt{c}' - \mathtt{c}'' = 0$, we either obtain that $\mathtt{c}' - \mathtt{c}''$ is non-zero with high probability (making the entire circuit to abort) or that no attack was mounted on the circuits computing $\mathtt{c}'$ and $\mathtt{c}''$. In the latter case, we obtain that $\mathtt{c}' = \mathtt{a} \cdot \mathtt{b} \cdot \mathtt{v}$, which is the correct MAC tag of the expected result of the multiplication gate $\mathtt{c} = \mathtt{a} \cdot \mathtt{b}$ under the key $\mathtt{v}$.

Computing multiplication gates. Next, we use the MAC tag $\mathtt{c}'$ computed previously in order to verify the correct computation of $\mathtt{c}$. We achieve this by computing the output of $\mathtt{c}$ and then MAC it by multiplying with the MAC key $\mathtt{v}$. Next, we check that the above result matches the *known-good* MAC tag $\mathtt{c}'$. This last check is implemented by computing $\mathtt{c} \cdot \mathtt{v} - \mathtt{c}'$ and having $\overline{\mathsf{C}}$ abort in case $\mathtt{c} \cdot \mathtt{v} - \mathtt{c}' \neq 0$. Notice that any additive attack on $\mathtt{c}$ can only introduce (after the multiplication by the MAC key) monomials of the from $\mathtt{a} \cdot \mathtt{v}$ or $\mathtt{b} \cdot \mathtt{v}$ which cannot be canceled-out by the MAC tag $\mathtt{a} \cdot \mathtt{b} \cdot \mathtt{v}$. Hence, we conclude that in the presence of an additive attack the gate output check fails, making $\overline{\mathsf{C}}$ abort.

Computing addition gates. Notice that in the above described construction, the degree of the used MAC key $\mathtt{v}$ is always 1 and in particular it does not increase after the computation of multiplication gates. Therefore, given an addition gate $\mathtt{c} = \mathtt{a} + \mathtt{b}$, we can compute the MAC tag for $\mathtt{c}$ by computing $\mathtt{c}' = \mathtt{a}' + \mathtt{b}'$. This avoids the ad-hoc gadget of [11] for additive-attack securely computing MAC tags where the inputs of the addition gate are of different degrees. Eliminating this gadget also simplifies the circuit randomization process (see below).

Avoiding degree-reduction. Next, since the degree of the key does not increase after the execution of each multiplication, this allows us to directly additive-attack secure arbitrary circuits without the need to reduce the degree (as opposed to the construction of [11]). This, together with a simplified circuit randomization process (described below), induces a big improvement in the concrete overhead of the construction compared to the construction of [11].

Circuit randomization process. The above described construction only achieves additive-attack security for the case where the inputs to each multiplication gate are *almost* random (See Definition 5). Moreover, it is also required that each input of C is also almost random (individually). We force the inputs of a multiplication gate $\mathtt{c} = \mathtt{a} \cdot \mathtt{b}$ to be almost random as follows. First, we additively secret share $\mathtt{a}$ and $\mathtt{b}$ to $(\mathtt{a} - \mathtt{r}_1, \mathtt{r}_1)$ and $(\mathtt{b} - \mathtt{r}_2, \mathtt{r}_2)$. We then compute the output of $\mathtt{c}$ by $\mathtt{c} = (\mathtt{a} - \mathtt{r}_1)(\mathtt{b} - \mathtt{r}_2) + (\mathtt{a} - \mathtt{r}_1) \cdot \mathtt{r}_2 + \mathtt{r}_1 \cdot (\mathtt{b} - \mathtt{r}_2) + \mathtt{r}_1 \cdot \mathtt{r}_2 = \mathtt{a} \cdot \mathtt{b}$. Notice that in this case, the inputs of every multiplication gate are uniformly random. Randomizing the inputs of C is done similarly, see full version for details.

Protecting SIMD circuits against linear attacks. As described above, a linear attack $\mathbf{L}$ on a multiplication gate $\mathbf{c}$ with input gates $\mathbf{a}$ and $\mathbf{b}$ specifies a linear function $f : \mathbb{F}^{2\ell} \to \mathbb{F}^{\ell}$ (in the gate's input *bundles*) to be added to the gate's output *bundle*. We specify f using two $\ell \times \ell$ matrices $\mathbf{L}_{\mathbf{a},\mathbf{c}}$ and $\mathbf{L}_{\mathbf{b},\mathbf{c}}$, changing the computation performed by $\mathbf{c}$ to be $\mathbf{c} = \mathbf{a} \odot \mathbf{b} + \mathbf{L}_{\mathbf{a},\mathbf{c}} \cdot \mathbf{a} + \mathbf{L}_{\mathbf{b},\mathbf{c}} \cdot \mathbf{b}$. Notice that $\mathbf{L}$ only introduces monomials of the form $\mathbf{L}_{\mathbf{a},\mathbf{c}} \cdot \mathbf{a}$, $\mathbf{L}_{\mathbf{b},\mathbf{c}} \cdot \mathbf{b}$ but not of the form $\mathbf{a} \odot \mathbf{b}$, where $\odot$ denotes ℓ-wide point-wise multiplication of two wire bundles. In the full version, we extend the high-level ideas of the above described construction to handle SIMD circuits and linear attacks.

Next, our basic construction for transforming an SIMD circuit C to a functionally-equivalent linear-attack secure SIMD circuit $\overline{\mathsf{C}}$ guarantees that every linear attack on $\overline{\mathsf{C}}$ is either equivalent to an *additive* attack on the inputs and outputs of C, or some wire in a special bundle $\mathbf{f}$, which denotes an error flag inside $\overline{\mathsf{C}}$, becomes non-zero. In such a case, we would like another bundle $\mathbf{f}'$ to be almost random. In the full version, we design a special-purpose gadget, called $\overline{\mathsf{Mix}}$ circuit, which satisfies the above property, even in the presence of linear attacks.

4 Preliminaries

Arithmetic circuits. Following [11], an *arithmetic circuit* C is a directed acyclic graph whose vertices are called *gates* and whose edges are called *wires.* Every in-degree 0 gate in C is labeled by a variable from a set of variables $X = \{x_1, \cdots, x_n\}$ and is referred to as an input gate. All other gates have in-degree 2, are labeled by elements from $\{+, -, \times\}$ and referred to as $\mathsf{add}, \mathsf{sub}$ and mult gates, respectively. Every gate of out-degree 0 is called an output gate. We assume that the output gates are ordered. In some cases we also allow in-degree 0 gates labeled by rand referred to as *randomness* gates. A circuit containing rand gates is called a *randomized* circuit. For a (possibly randomized) circuit C and for a gate g of C we denote by $\mathsf{g}_{\mathbf{x}}$ the distribution of the output value of g (defined in a natural way) when C is being evaluated on an input $\mathbf{x}$.

SIMD **circuits.** An SIMD circuit with bundle size ℓ is defined similar to arithmetic circuits. We refer to the edges of an SIMD circuit C as *wire bundles* or *bundles* and to vertices of an SIMD circuits as *ℓ-gates.* We write $\mathsf{C} : \left(\mathbb{F}^\ell\right)^n \to \left(\mathbb{F}^\ell\right)^k$ to indicate that C is an SIMD circuit with n input bundles and k output bundles. Each multiplication, addition or subtraction ℓ-gate of an SIMD circuit gets as input two wire bundles of size ℓ and outputs a bundle of size ℓ obtained by performing ℓ *point-wise* multiplications, additions or subtractions in parallel.[8]

We also allow SIMD circuits to contain an additional type of ℓ-gates with in-degree and out-degree 1, referred to as routing ℓ-gates. Each routing ℓ-gate is labeled by a function $\rho : [\ell] \to [\ell]$. We shall sometimes refer to these routing ℓ-gates as ρ-gates. A ρ-gate on an input bundle $\mathbf{a} = (a_1, \cdots, a_\ell)$ outputs a bundle $\mathbf{b} = (b_1, \cdots, b_\ell)$ such that $b_i = a_{\rho(i)}$ for all $1 \leq i \leq \ell$.

Additive attacks. An additive attack $\mathbf{A}$ changes the computation performed by a circuit C by specifying for every wire in C, connecting gates $\mathtt{a}$ and $\mathtt{b}$, a value to be added to the output of $\mathtt{a}$. The derived value is then used for the computation of $\mathtt{b}$. In addition, $\mathbf{A}$ specifies values to be added to the outputs of C. Note that an additive attack on a circuit C is a fixed vector of field elements which is independent from the inputs and internal values of C.

Linear attacks. A *linear attack* $\mathbf{L}$ on an SIMD circuit changes the computation of a multiplication ℓ-gate by adding to each wire in the gate's output bundle a linear function of all the wires in the gate's two input bundles. In particular, for any multiplication ℓ-gate $\mathbf{c}$ with input bundles $\mathbf{a}$ and $\mathbf{b}$, a linear attack $\mathbf{L}$ specifies a linear function $f : \mathbb{F}^{2\ell} \to \mathbb{F}^\ell$ such that the output bundle of $\mathbf{c}$ is equal to $\mathbf{c} = \mathbf{a} \odot \mathbf{b} + f(\mathbf{a}, \mathbf{b})$, where $\odot$ denotes point-wise multiplication of two wire bundles. In addition, similar to additive attacks, we allow a linear attack $\mathbf{L}$ to specify an additive attack $\mathbf{L}^{\mathsf{out}}$ on the outputs of the SIMD circuit C.

Attacks on addition and subtraction gates. We do not allow linear attacks on addition and subtraction gates. This is since mounting an attack of the form $f(\mathsf{a}, \mathsf{b}) = -\mathsf{a} - \mathsf{b}$ the adversary is able to fix an output of an addition gate

[8] Notice that for the case of SIMD circuits, the notion of in-degree of a gate corresponds to the number of its input wire-bundles (as opposed to individual wires). Thus the in-degree of $\{\times, +, -\}$ gates is 2. The notion of out-degree is defined similarly.

$\mathsf{c} = \mathsf{a} + \mathsf{b} - \mathsf{a} - \mathsf{b}$ to be always zero. Therefore, allowing for such attacks means that it is possible to override the output of these gates to be an arbitrary value. Such attacks are not supported by our constructions. [9]

Additive attacks on SIMD circuits. Note that allowing additive attacks on wire bundles of SIMD circuits (in addition to linear attacks) will not provide the adversary with additional capabilities in modifying the circuit's computation. This is since for any pair of attacks $(\mathbf{A}, \mathbf{L})$ on an SIMD circuit C where $\mathbf{A}$ is an additive attack and $\mathbf{L}$ is a linear attack there exists a functionally-equivalent linear attack $\mathbf{L}'$. The linear attack $\mathbf{L}'$ can be constructed as follows. First, the additive attacks specified by $\mathbf{A}$ can be pushed "downstream" through the circuit till the inputs of the multiplication gates and the outputs of the output gates. Next, additive attacks on inputs of a multiplication gate c, can be added to the diagonal of the appropriate matrices as specified by $\mathbf{L}$, yielding $\mathbf{L}'$.

Additive attacks in secure multi-party computation. In the following we define the notion of additively corruptible versions of a functionality. Without loss of generality, we only consider functionalities where only P_1 gets an output. That is, functionalities of the form $f : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$ where $(I_1, \cdots, I_n, O_1)$ are positive integers. Note that we can move to individual outputs using a standard transformation (See [13, Section 2.5.2]).

Definition 2. *Let C be an n-party circuit. We define the* additively corruptible *version of C to be an n-party functionality $f_{\mathsf{C}}^{\mathbf{A}}$ that takes additional input from the adversary representing an additive attack, $\mathbf{A}$, on C. For an input $\mathbf{x}$ and additive attack $\mathbf{A}$, $f_{\mathsf{C}}^{\mathbf{A}}$ outputs $\mathsf{C}^{\mathbf{A}}(x)$. The notion of a linearly corruptible circuit is defined similarly, replacing the additive attack $\mathbf{A}$ with a linear attack $\mathbf{L}$.*

Next, we define the notion of $\mathcal{T}$-equivalent protocols.

Definition 3. *Let π and π' be two protocols for computing an n-party circuit C in the f and f' hybrid models respectively. We say that π is* $\mathcal{T}$-equivalent *to π' if for any adversary Adv controlling a set of parties $\mathcal{T} \subseteq \mathcal{P}$ and for any input $\mathbf{x}$ it holds that $\mathsf{Real}_{\pi,\mathcal{T}}^{\mathsf{Adv},f}(\mathbf{x}) \equiv \mathsf{Real}_{\pi',\mathcal{T}}^{\mathsf{Adv},f'}(\mathbf{x})$.*

5 Additive Security for Arithmetic Circuits

In this section we simplify the construction of [11] improving its additive-attack security from $O(|\mathsf{C}|/|\mathbb{F}|)$ to $O(1/|\mathbb{F}|)$, as well as improving its concrete efficiency. Following the approach of [11], we first present a simpler construction whose security holds only when the circuit's wire values satisfy some local randomness property (Construction 1). In the full version, we show how to eliminate this assumption by applying general transformations to the circuit.

We begin by defining additive-attack security for specific input distributions.

Definition 4. *Let $\mathbb{F}$ be a finite field, $\mathsf{C} : \mathbb{F}^n \to \mathbb{F}^k$ an arithmetic circuit, and I a distribution over $\mathbb{F}^n$. We say that a circuit $\overline{\mathsf{C}} : \mathbb{F}^n \to \mathbb{F}^{k+1}$ is an ϵ-additive-attack secure implementation of C with respect to I if the following holds:*

[9] Note that linear attacks on multiplication gates suffice to achieve MPC tasks.

– Completeness. *For all* $\mathbf{x} \in \mathbb{F}^n$, $\overline{\mathsf{C}}(\mathbf{x}) \equiv \mathsf{C}(\mathbf{x})$.
– Security with respect to I. *For any additive attack* $\mathbf{A}$, *there exists* $\mathbf{a}^{\mathsf{in}} \in \mathbb{F}^n$ *and a distribution* $\mathcal{A}^{\mathsf{out}}$ *over* $\mathbb{F}^k$ *such that* $SD(\overline{\mathsf{C}}^{\mathbf{A}}(I), \mathsf{C}(I + \mathbf{a}^{\mathsf{in}}) + \mathcal{A}^{\mathsf{out}}) \leq \epsilon$.

The construction guarantees security as defined in Definition 1 with $\epsilon = O(1/|\mathbb{F}|)$, under the assumption that the inputs of the circuit as well as the inputs of each multiplication gate are sufficiently random. Unlike the basic construction of [11], the construction described in this section does not require the randomization of the inputs of addition and subtraction gates. Thus, below we define a weaker notion of locally random circuits compared to the one used in [11], by not imposing any requirement about the inputs of addition and subtraction gates. This also greatly simplifies the construction of such circuits.

Definition 5 (Locally Random Circuits). *Let* $\mathbb{F}$ *be a finite field,* C *be a randomized arithmetic circuit. We say that* C *is locally* ϵ*-random with respect to a distribution* I *if the following two properties hold.*

1. ***Local randomization of input gates.*** *For any* $y \in \mathbb{F}$ *and for any* $1 \leq i \leq n$ *the probability over selecting* $\mathbf{x} \leftarrow I$ *that* $x_i = y$ *is at most* $|\mathbb{F}| \cdot \epsilon$.
2. ***Local randomization of multiplication gates.*** *For any* $(y, z) \in \mathbb{F}^2$ *and any pair of gates* (a, b), *whose outputs are the inputs to some multiplication gate in* C, *it holds that the probability, over the internal randomness of* C *and the selection* $\mathbf{x} \leftarrow I$, *that* $(\mathsf{a}_{\mathbf{x}}, \mathsf{b}_{\mathbf{x}}) = (y, z)$ *is at most* ϵ.

We now present our basic construction for constructing additive-attack circuits.

Construction 1 *Let* $\mathsf{C} : \mathbb{F}^n \to \mathbb{F}^k$ *be a circuit. Define a circuit* $\overline{\mathsf{C}}$ *that on input* $\mathbf{x}$ *computes* $\mathbf{z} = \mathsf{C}(\mathbf{x})$ *and then performs the following:*

MAC generation Circuit:

1. *Generate a random elements* $\mathtt{r}, \mathtt{v} \in \mathbb{F}$ *and compute* $\mathtt{r}' \leftarrow \mathtt{r} \cdot \mathtt{v}$.
2. *For each input gate* c, *compute the value* $\mathsf{c}' \leftarrow \mathsf{c} \cdot \mathtt{v}$.
3. *For each non-input gate* c *let* a, b *be its inputs and let* a', b' *be the MAC tags corresponding to* a *and* b. *Compute the MAC tag* c' *as follows:*
   (a) *If* c *is a multiplication gate, let* $\mathsf{c}' \leftarrow \mathsf{a}' \cdot \mathsf{b}$ *and let* $\mathsf{c}'' \leftarrow \mathsf{a} \cdot \mathsf{b}'$.
   (b) *If* c *is an addition gate let* $\mathsf{c}' \leftarrow \mathsf{a}' + \mathsf{b}'$. *Similarly, if* c *is a subtraction gate let* $\mathsf{c}' \leftarrow \mathsf{a}' - \mathsf{b}'$.

MAC checking circuit:

3. *For every input gate* c *in* C, *generate a random element* $\mathtt{t}^{\mathsf{c}}$ *and compute*
   $\mathsf{g}^{\mathsf{c}} \leftarrow \mathsf{c} + \mathtt{r}$, $\mathsf{h}'^{\mathsf{c}} \leftarrow \mathsf{c}' + \mathtt{r}'$, $\mathsf{g}'^{\mathsf{c}} \leftarrow \mathsf{g}^{\mathsf{c}} \cdot \mathtt{v}$, $\mathsf{f}^{\mathsf{c}} \leftarrow \mathsf{h}'^{\mathsf{c}} - \mathsf{g}'^{\mathsf{c}}$.
4. *Compute* $\mathtt{f}_1 \leftarrow \sum_{\mathsf{c} \in \mathsf{inpt}_{\mathsf{C}}} \mathtt{t}^{\mathsf{c}} \cdot \mathtt{f}^{\mathsf{c}}$ *where* $\mathsf{inpt}_{\mathsf{C}}$ *is the set of the input gates of* C.
5. *For every multiplication gate* c, *generate two random field elements* $\mathtt{t}^{\mathsf{c}}, \mathtt{w}^{\mathsf{c}}$ *and compute* $\mathsf{f}^{\mathsf{c}} \leftarrow \mathsf{c}' - \mathsf{c}''$, $\mathsf{g}^{\mathsf{c}} \leftarrow \mathsf{c} \cdot \mathtt{v}$, $\mathsf{h}^{\mathsf{c}} \leftarrow \mathsf{g}^{\mathsf{c}} - \mathsf{c}'$.
6. *Let* $\mathsf{mul}_{\mathsf{C}}$ *be the set of all multiplication gates in* C, *compute* $\mathtt{f}_2 \leftarrow \sum_{\mathsf{c} \in \mathsf{mul}_{\mathsf{C}}} \mathtt{w}^{\mathsf{c}} \cdot \mathtt{f}^{\mathsf{c}}$ *and* $\mathtt{f}_3 \leftarrow \sum_{\mathsf{c} \in \mathsf{mul}_{\mathsf{C}}} \mathtt{t}^{\mathsf{c}} \cdot \mathsf{h}^{\mathsf{c}}$.
7. *Compute* $\mathtt{f} \leftarrow \mathtt{f}_1 \cdot \mathsf{s}_1 + \mathtt{f}_2 \cdot \mathsf{s}_2 + \mathtt{f}_3 \cdot \mathsf{s}_3$ *where* $\mathsf{s}_1, \mathsf{s}_2, \mathsf{s}_3$ *are random field elements.*

Output generation: *Output* $\mathbf{z} + \mathtt{f} \cdot \mathbf{r}$ *where* $\mathbf{r}$ *is a random vector from* $\mathbb{F}^k$.

In the full version we prove the following theorems.

Theorem 5. *Let* $\mathsf{C} : \mathbb{F}^n \to \mathbb{F}^k$ *be a randomized arithmetic circuit which is locally* ϵ*-random with respect to and input distribution* I. *Then the circuit* $\overline{\mathsf{C}}$ *obtained by applying Construction 1 to* C *is a* $(|\mathbb{F}| \cdot \epsilon + 1/|\mathbb{F}|)$*-additive-attack secure implementation of* C *with respect to* I. *Moreover,* $|\overline{\mathsf{C}}| = O(|\mathsf{C}|)$.

Theorem 6 (Additive-attack security). *For any arithmetic circuit* $\mathsf{C} : \mathbb{F}^n \to \mathbb{F}^k$ *there exists a randomized circuit* $\overline{\mathsf{C}} : \mathbb{F}^n \to \mathbb{F}^k$ *such that* $\overline{\mathsf{C}}$ *is an* ϵ*-additive-attack secure implementation of* C *where* $\epsilon = O(1/|\mathbb{F}|)$. *Moreover,* $|\overline{\mathsf{C}}| = O(|\mathsf{C}|)$.

Notice that unlike the work of [11], the error parameter of the construction is $O(1/|\mathbb{F}|)$. This matches the result of [14], but in a stronger attack model.

6 From General Adversaries to Additive Attacks

In this section we reduce any general adversary attacking a randomized protocol π to an additive attack on the protocol circuit C_π defined as follows. We compile a protocol π into a circuit, C_π, by writing all local computations performed by the parties as circuits and whenever a party P_i sends a message to P_j, we connect the corresponding parts of the circuits representing P_i and P_j using wires. Notice that for every input $\mathbf{x}$ and randomness $\mathbf{r}$, it holds that $\pi(\mathbf{x}; \mathbf{r}) = \mathsf{C}_\pi(\mathbf{x}, \mathbf{r})$.

We now define the notion of a *last-round-private* protocol.

Definition 6. *Let* $\mathcal{T}$ *be a set of corrupted parties and let* π *be a* $\mathcal{T}$*-randomized* n*-party protocol for computing an* n*-input circuit* $\mathsf{C} : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$. *We say that* π *is* $\mathcal{T}$-last-round-private *if the following hold.*

1. ***Structure of the last round.*** *During the last round, only* P_1 *computes the output vector* $\mathbf{z}$, *in the following way. Let* $\overline{\mathcal{T}}' \subseteq \overline{\mathcal{T}}$ *be the set of parties from* $\overline{\mathcal{T}}$ *sending messages to* P_1 *during the last round. Each output* $\{z_i\}_{1 \le i \le O_1}$ *is computed by* P_1 *evaluating two linear functions* $F_\mathcal{T}$ *and* $F_{\overline{\mathcal{T}}'}$ *such that* $z_i = F_\mathcal{T}(l^i_{\mathcal{T},P_1}) + F_{\overline{\mathcal{T}}'}(l^i_{\overline{\mathcal{T}}',P_1})$ *where the messages* $l^i_{\mathcal{T},P_1}, l^i_{\overline{\mathcal{T}}',P_1}$ *are the shares corresponding to* z_i *received by* P_1 *from the parties in* $\mathcal{T}$ *and* $\overline{\mathcal{T}}'$, *respectively.*
2. ***Privacy of the last round.*** *Fix an input* $\mathbf{x}_\mathcal{T}$ *and randomness* $\mathbf{r}_\mathcal{T}$ *to the circuit* C_π *for the parties in* $\mathcal{T}$. *In addition, fix an additive attack* $\mathbf{A}$ *on* C_π *and fix a view* $\widehat{u}_\mathcal{T}$ *of the parties in* $\mathcal{T}$ *during an execution of* $\mathsf{C}^{\mathbf{A}}_\pi$ *on* $(\mathbf{x}_\mathcal{T}, \mathbf{r}_\mathcal{T})$. *Let* $\mathbf{Z}$ *be the distribution of outputs in* $\mathsf{C}^{\mathbf{A}}_\pi$ *conditioned on* $(\mathbf{x}_\mathcal{T}, \mathbf{r}_\mathcal{T}, \mathbf{A}, \widehat{u}_\mathcal{T})$ *and fix* $\mathbf{z}$ *from the support of* $\mathbf{Z}$. *Finally, let* $\widehat{l}_{\mathcal{T},P_1}$ *be the messages received by* P_1 *from the parties in* $\mathcal{T}$ *during the last round of* $\mathsf{C}^{\mathbf{A}}_\pi$ *as uniquely defined by* $(\mathbf{x}_\mathcal{T}, \mathbf{r}_\mathcal{T}, \widehat{u}_\mathcal{T})$. *We require that the distribution of the messages* $\widehat{l}_{\overline{\mathcal{T}}',P_1}$, *over the unfixed randomness* $\mathbf{r}_{\overline{\mathcal{T}}}$ *is uniform conditioned on* $F_{\overline{\mathcal{T}}'}(\widehat{l}_{\overline{\mathcal{T}}',P_1}) = \mathbf{z} - F_T(\widehat{l}_{\mathcal{T},P_1})$.

In the full version we prove the following theorem.

Theorem 7. *Let* π *be a* $\mathcal{T}$-last-round-private *and* $\mathcal{T}$*-randomized protocol. Then for any active adversary* Adv *controlling the parties in* $\mathcal{T}$ *there exists a simulator* Sim *such that for any input* $\mathbf{x}$ *it holds that* $\mathsf{Ideal}^{\mathsf{Sim}}_{f^{\mathbf{A}}_{\mathsf{C}_\pi},\mathcal{T}}(\mathbf{x}) \equiv \mathsf{Real}^{\mathsf{Adv}}_{\pi,\mathcal{T}}(\mathbf{x})$.

7 Homomorphism for Standard Circuits

In this section we prove that if two circuits C and C' meet certain properties, then for any additive attack on C' there exists an equivalent additive attack on C. Applying this approach to C_π, we prove that any additive attack on C_π corresponds to an additive attack on C.

Without loss of generality, we express every multiplication gate as a product of its inputs where each input is an arbitrary fixed linear combination of the preceding addition and subtraction gates up to the depth of the preceding multiplication gate.

Definition 7. *Let C be a randomized circuit and let c be an in-degree 2 multiplication gate inside C. We define two ordered sets $\mathsf{left_c}$ and $\mathsf{right_c}$, as follows.*

$$\mathsf{left_c} = \left\{ \begin{matrix} \mathsf{a} \in \{\times, \mathsf{input}\} \\ \end{matrix} : \begin{matrix} \exists \mathsf{path} \textit{ from } \mathsf{a} \textit{ to the first input of } \mathsf{c} \\ \textit{which only contains gates from the set } \{+,-\} \end{matrix} \right\}$$

$$\mathsf{right_c} = \left\{ \begin{matrix} \mathsf{a} \in \{\times, \mathsf{input}\} \\ \end{matrix} : \begin{matrix} \exists \mathsf{path} \textit{ from } \mathsf{a} \textit{ to the second input of } \mathsf{c} \\ \textit{which only contains gates from the set } \{+,-\} \end{matrix} \right\}$$

The ordered sets $\mathsf{left_c}$ and $\mathsf{right_c}$ naturally define two linear functions $l^{\mathsf{c}} : \mathbb{F}^{|\mathsf{left_c}|} \to \mathbb{F}$ and $r^{\mathsf{c}} : \mathbb{F}^{|\mathsf{right_c}|} \to \mathbb{F}$ representing the output of c as a function of the outputs of the preceding mult and input gates. More specifically, for any input $\mathbf{x}$ to C it holds that $\boldsymbol{c}_{\mathbf{x}} = l^{\mathsf{c}}(\boldsymbol{a}_{\mathbf{x}}) \cdot r^{\mathsf{c}}(\boldsymbol{b}_{\mathbf{x}})$ where $\boldsymbol{a} = \mathsf{left_c}$ and $\boldsymbol{b} = \mathsf{right_c}$.

We now express every output gate which is an addition or subtraction gate as a fixed linear combination of the output of the proceeding multiplication gates.

Definition 8. *Let C be a deterministic circuit and let c be an output gate that is an add or sub gate. We define the ordered set $\mathsf{in_c}$ as follows.*

$$\mathsf{in_c} = \left\{ \begin{matrix} \mathsf{a} \in \{\times, \mathsf{input}\} \\ \end{matrix} : \begin{matrix} \exists \mathsf{path} \textit{ from } \mathsf{a} \textit{ to either of the two inputs of } \mathsf{c} \\ \textit{which only contains gates from the set } \{+,-\} \end{matrix} \right\}$$

The set $\mathsf{in_c}$ naturally defines a linear function $f^{\mathsf{c}} : \mathbb{F}^{|\mathsf{in_c}|} \to \mathbb{F}$ representing the output of c as a function of the outputs of the preceding mult and input gates. More specifically, for any input $\mathbf{x}$ to C it holds that $\boldsymbol{c}_{\mathbf{x}} = f^{\mathsf{c}}(\boldsymbol{a}_{\mathbf{x}})$ where $\boldsymbol{a} = \mathsf{in_c}$.

We now define the notion of circuit homomorphism. Later, we prove that if a circuit C' is homomorphic to a circuit C then any additive attack on C' can be simulated by an additive attack on C. Applying the above on MPC protocols, as long as the circuit C_π of a protocol π is homomorphic to C, then any additive attack on C_π can be simulated by an additive attack on C. Combining this with the result of Section 6, we obtain that for any protocol π computing a circuit C, which is *$\mathcal{T}$-randomized*, *$\mathcal{T}$-last-round-private* and *homomorphic* to C, any attack mounted by an active adversary is equivalent to an additive attack on C.

Definition 9 (Circuit Homomorphism). *Let C be a deterministic circuit. A circuit C' is said to be homomorphic to C if there exists a mapping $\mathcal{H}$ from the input and mult gates of C to the gates of C' such that the following properties hold. Below, for any gate c of C we denote the output of $\mathcal{H}(\mathsf{c})$ by c'.*

1. ***Inputs.*** *For any* input *gate* c *of* C *and for any input* $\mathbf{x}$ *it holds that* $\mathsf{c}_{\mathbf{x}} = \mathsf{c}'_{\mathbf{x}}$.
2. ***Multiplications.*** *For any* mult *gate* c *we require that there exists constant* $\lambda^{\mathsf{c}} \in \mathbb{F}$ *with the following properties for any input* $\mathbf{x}$*:*
   (a) *It holds that* $\mathsf{c}'_{\mathbf{x}} + \lambda^{\mathsf{c}} = l^{\mathsf{c}}\left((\mathsf{a}'_{\mathbf{x}} + \lambda^{\mathsf{a}})_{\mathsf{a}\in\mathsf{left}_{\mathsf{c}}}\right) \cdot r^{\mathsf{c}}\left((\mathsf{b}'_{\mathbf{x}} + \lambda^{\mathsf{b}})_{\mathsf{b}\in\mathsf{right}_{\mathsf{c}}}\right)$.
   (b) *For every* mult *gate used for the computation of the output of* c' *inside* C'*, the left input is a linear function of* $l^{\mathsf{c}}\left((\mathsf{a}'_{\mathbf{x}})_{\mathsf{a}\in\mathsf{left}_{\mathsf{c}}}\right)$ *and the right input is a linear function of* $r^{\mathsf{c}}\left((\mathsf{b}'_{\mathbf{x}})_{\mathsf{b}\in\mathsf{right}_{\mathsf{c}}}\right)$.
3. ***Outputs.*** *We first require that both* C *and* C' *have the same number of* output *gates. Let* c *be the* i*-th gate of* C*, we distinguish two different cases.*
   (a) *Let* o' *be the* i*-th* output *gate of* C'*. If* c *is a* mult *gate, then* $\mathsf{o}'_{\mathbf{x}} = \mathsf{c}'_{\mathbf{x}} + \lambda^{\mathsf{c}}$.[10]
   (b) *If* c *is an* add*, or* sub *gate then the* i*-th output of* C'*,* $\mathsf{o}'_{\mathbf{x}}$ *is equal to* $\mathsf{o}'_{\mathbf{x}} = f^{\mathsf{c}}\left((\mathsf{a}'_{\mathbf{x}} + \lambda^{\mathsf{a}})_{\mathsf{a}\in\mathsf{in}_{\mathsf{c}}}\right)$ *for all input* $\mathbf{x}$.

   Moreover, we require that the recovery of the output from the gates $\mathbf{o}'$ *of* C' *is performed without computing any* mult *gates.*

Remark 1 *Given two circuits* C, C'*, a mapping* $\mathcal{H}$*, a constant* λ^{c} *and functions* l^{c} *and* r^{c} *for every* mult *gate* c *in* C*, it is possible to decide in polynomial time if* C' *is homomorphic to* C*. Checking that the requirements of Definition 9 hold can be done symbolically using the gate's output as variables.*

For simplicity of exposition, Definition 9 is tailored to protocols working on additive secret sharing such as the GMW protocol. A simple generalization of Definition 9 captures protocols working on any linear secret sharing scheme, such as the DN and DIK. See full version for details.

Lemma 1. *Let* C *be a deterministic circuit and let* C' *be a circuit homomorphic to* C*. Then for any additive attack* $\mathbf{A}'$ *on* C' *there exists an additive attack* $\mathbf{A}$ *on* C *such that for any input* $\mathbf{x}$ *it holds that* $\mathsf{C}'^{\mathbf{A}'}(\mathbf{x}) = \mathsf{C}^{\mathbf{A}}(\mathbf{x})$.

We now extend Lemma 1 to handle n-party circuits computed during an MPC protocol. We begin by defining the notion of $\mathcal{T}$-homomorphic circuits.

Definition 10. *Let* π *be an* n*-party protocol,* C *be an* n*-party circuit and let* $\mathcal{T}$ *be a set of parties. We say that* C_π *is* $\mathcal{T}$*-homomorphic to* C *if for any input* $\mathbf{x}_{\mathcal{T}}$ *for the parties in* $\mathcal{T}$ *and for every randomness* $\mathbf{r}$*, the circuit* $\mathsf{C}_\pi((\mathbf{x}_{\mathcal{T}}, \cdot), \mathbf{r})$ *obtained by fixing the inputs* $\mathbf{x}_{\mathcal{T}}$ *and* $\mathbf{r}$ *inside* C_π *is homomorphic to* $\mathsf{C}(\mathbf{x}_{\mathcal{T}}, \cdot)$.

In the full version we prove the following theorem.

Theorem 8. *Let* π *be an* n*-party protocol for computing a circuit* $\mathsf{C} : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$ *in the* f*-hybrid model and let* $\mathcal{T}$ *be a set of parties such that* π *is* $\mathcal{T}$*-randomized,* $\mathcal{T}$*-last-round-private and* C_π *is* $\mathcal{T}$*-homomorphic to* C*. Then for any active adversary* Adv *controlling the parties in* $\mathcal{T}$ *there exists a simulator* Sim *such that for any input* $\mathbf{x}$ *it holds that* $\mathsf{Ideal}^{\mathsf{Sim}}_{f^{\mathbf{A}}_{\mathsf{C}},\mathcal{T}}(\mathbf{x}) \equiv \mathsf{Real}^{\mathsf{Adv},f}_{\pi,\mathcal{T}}(\mathbf{x})$.

[10] Notice that here we do not require that $\mathcal{H}(\mathsf{c}) = \mathsf{o}'$. This is since already $\mathcal{H}(\mathsf{c}) = \mathsf{c}'$ and moreover there exists a gate o' such that $\mathsf{o}' = \mathsf{c}'_{\mathbf{x}} + \lambda^{\mathsf{c}}$.

8 The GMW Protocol

In this section we prove that an arithmetic generalization of the passively secure GMW protocol [12] is additively corruptible. We first extend the GMW protocol to the arithmetic setting [17], where the OT oracle is replaced by oblivious linear function evaluation (OLE) [19].

Definition 11 (The OLE functionality). *Let $\mathbb{F}$ be a finite field. We define the functionality f_{OLE} that on inputs $(a, b) \in \mathbb{F}^2$ from the sender and $x \in \mathbb{F}$ from the receiver outputs $\perp$ to the sender and $a \cdot x + b$ to the receiver.*

We now proceed describing an arithmetic version of the GMW protocol in the OLE-hybrid model [12, 17]. We begin by describing the $\mathsf{Input\text{-}Share}_{\mathsf{GMW}}$ and $\mathsf{Mult}_{\mathsf{GMW}}$ protocols used to evaluate input and multiplication gates.

Construction 2 (Subprotocol $\mathsf{Input\text{-}Share}_{\mathsf{GMW}}$) *The subprotocol $\mathsf{Input\text{-}Share}_{\mathsf{GMW}}$ is defined as follows. Each party P_i on input x computes a random additive sharing of x, denoted by $[\mathbf{x}]_{\mathsf{add}} = (x_1, \ldots, x_n)$, and deals it among all the parties.*

Construction 3 (Subprotocol $\mathsf{Mult}_{\mathsf{GMW}}$) *The subprotocol $\mathsf{Mult}_{\mathsf{GMW}}$ gets as input additive sharings $[\mathbf{a}]_{\mathsf{add}}$, $[\mathbf{b}]_{\mathsf{add}}$ and outputs an additive sharing $[\mathbf{c}]_{\mathsf{add}}$ such that $\mathsf{c} = \mathsf{a} \cdot \mathsf{b}$. The protocol proceeds as follows.*

1. *Each ordered pair of parties P_i, P_j, such that $i \neq j$, performs the following.*
   (a) P_i generates a random value $r_{i,j}$ and acting as a sender sends $(a_i, r_{i,j})$ to the OLE oracle. P_j acting as a receiver sends b_j to the OLE oracle.
   (b) The OLE oracle responds with $s_{i,j} = a_i \cdot b_j + r_{i,j}$ to P_j.
2. *Each party P_i computes $c_i \leftarrow a_i \cdot b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} (s_{j,i} - r_{i,j})$.*

We now proceed in describing the passively secure GMW protocol.

Construction 4 (Passively secure GMW protocol) *Let $\mathsf{C} : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$ be an n-party circuit. The protocol $\mathsf{GMW}_{\mathsf{C}}$ for C proceeds as follows:*

1. ***Input sharing phase.*** *For each input gate associated to party P_i, party P_i executes the protocol $\mathsf{Input\text{-}Share}_{\mathsf{GMW}}$ described in Construction 2.*
2. ***Circuit evaluation phase.*** *For each gate c in C with input sharings $[\mathbf{a}]_{\mathsf{add}} = (a_1, \ldots, a_n)$ and $[\mathbf{b}]_{\mathsf{add}} = (b_1, \ldots, b_n)$ proceed as follows:*

   Evaluating addition and subtraction gates. *For the case of addition gates, all parties locally compute $[\mathbf{c}]_{\mathsf{add}} \leftarrow [\mathbf{a}]_{\mathsf{add}} + [\mathbf{b}]_{\mathsf{add}}$. Similarly, for subtraction gates, all parties locally compute $[\mathbf{c}]_{\mathsf{add}} \leftarrow [\mathbf{a}]_{\mathsf{add}} - [\mathbf{b}]_{\mathsf{add}}$.*

   Evaluating multiplication gates. *All the parties execute the $\mathsf{Mult}_{\mathsf{GMW}}$ protocol described in Construction 3 on inputs $[\mathbf{a}]_{\mathsf{add}}$ and $[\mathbf{b}]_{\mathsf{add}}$.*
3. ***Output recovery phase.*** *At the end of the computation, for each output gate c of C all the parties hold a sharing $[\mathbf{c}]_{\mathsf{add}}$ corresponding to its value. For each output gate c, the parties generate a random sharing $[\mathbf{z}]_{\mathsf{add}}$ of 0 and compute $[\mathbf{c}']_{\mathsf{add}} \leftarrow [\mathbf{c}]_{\mathsf{add}} + [\mathbf{z}]_{\mathsf{add}}$. Parties $\{P_2, \cdots, P_n\}$ send their shares of $[\mathbf{c}']_{\mathsf{add}}$ to P_1. Then P_1 recovers the output c by computing $c \leftarrow \sum_{i=1}^{n} c'_i$.*

The works of [12, 17] analyzed the passively secure GMW protocol.

Theorem 9 ([12, 17]). *For any n-party circuit $\mathsf{C} : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$, the protocol $\mathsf{GMW_C}$ in the OLE hybrid model is passively secure against any adversary controlling at most $n-1$ parties. Moreover, the communication complexity (in field elements) as well as the number of oracle calls of $\mathsf{GMW_C}$ is $O(n^2|C|)$.*

8.1 Randomizing the GMW Protocol

Note that the protocol $\mathsf{Input\text{-}Share_{GMW}}$ is already randomized. This is since additive secret sharing is done by having the party P_i, holding the input x, send random shares r_j to all other parties and then compute his share to be $x - \sum_j r_j$. Therefore, the messages exchanged during the input sharing phase are already input-independent. We now describe how to randomize the evaluation of multiplication gates in GMW protocol. In the $\mathsf{Mult_{GMW}}$ protocol, all messages received by the parties are sent by the f_{OLE} oracle. We thus construct the $f_{\mathsf{OLE}}^{\mathcal{T}}$ oracle which sends messages to the parties in $\mathcal{T}$ which only depend syntacticly on the randomness of the protocol and not on the inputs of the parties in $\overline{\mathcal{T}}$.

Construction 5 (The $f_{\mathsf{OLE}}^{\mathcal{T}}$ functionality) *Let $\mathcal{T}$ be a set of parties. We define the functionality $f_{\mathsf{OLE}}^{\mathcal{T}}$ that on inputs (a, b) from a party P_i acting as a sender and $x \in \mathbb{F}$ from a party P_j acting as a receiver performs the following.*

1. ***$P_j \in \mathcal{T}$ and $P_i \in \overline{\mathcal{T}}$.*** *Let P_h be the first party not in $\mathcal{T}$. $f_{\mathsf{OLE}}^{\mathcal{T}}$ generates a random value e, sends $\perp$ to P_i and e to P_j and $ax + b - e$ to P_h.*
2. ***Otherwise.*** *In this case $f_{\mathsf{OLE}}^{\mathcal{T}}$ sends $\perp$ to P_i and $ax + b$ to P_j.*

In the following we describe the $\mathsf{Mult_{GMW}^{\mathcal{T}}}$ protocol in the $f_{\mathsf{OLE}}^{\mathcal{T}}$ hybrid model.

Construction 6 (Subprotocol $\mathsf{Mult_{GMW}^{\mathcal{T}}}$) *Let $\mathcal{T}$ be a set of parties and let P_h be the first party not in $\mathcal{T}$. The subprotocol $\mathsf{Mult_{GMW}^{\mathcal{T}}}$, in the $f_{\mathsf{OLE}}^{\mathcal{T}}$ hybrid model, gets as input additive sharings of $[\mathbf{a}]_{\mathsf{add}}$, $[\mathbf{b}]_{\mathsf{add}}$ and outputs an additive sharing $[\mathbf{c}]_{\mathsf{add}}$ such that $\mathsf{c} = \mathsf{a} \cdot \mathsf{b}$. The protocol proceeds as follows.*

1. *Each ordered pair of parties P_i, P_j, such that $i \neq j$, performs the following.*
   (a) *P_i generates a random value $r_{i,j}$ and acting as a sender sends $(a_i, r_{i,j})$ to the $f_{\mathsf{OLE}}^{\mathcal{T}}$ oracle. P_j acting as a receiver sends b_j to the $f_{\mathsf{OLE}}^{\mathcal{T}}$ oracle.*
   (b) *The $f_{\mathsf{OLE}}^{\mathcal{T}}$ oracle responds with $s_{i,j}$ to P_j, and with $s'_{i,j}$ to P_h in case that $P_j \in \mathcal{T}$ and $P_i \in \overline{\mathcal{T}}$.*
2. *Each party $P_i \in \mathcal{T}$ computes $c_i \leftarrow a_i \cdot b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} (s_{j,i} - r_{i,j})$.*
3. *Each party $P_i \in \overline{\mathcal{T}}$, such that $P_i \neq P_h$, generates his share c_i of c uniformly at random, computes $d_i \leftarrow a_i \cdot b_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} (s_{j,i} - r_{i,j})$ and sends (c_i, d_i) to P_h.*
4. *Party P_h computes $c_h \leftarrow a_h \cdot b_h + \sum_{\substack{P_i \in \overline{\mathcal{T}} \\ P_i \neq P_h}} (d_i - c_i) + \sum_{\substack{P_i \in \overline{\mathcal{T}} \\ P_j \in \mathcal{T}}} s'_{i,j}$.*

Next, we describe the $\mathsf{GMW_C^{\mathcal{T}}}$ protocol. In the full version we prove that $\mathsf{GMW_C^{\mathcal{T}}}$ is $\mathcal{T}$-randomized and $\mathcal{T}$-equivalent to $\mathsf{GMW_C}$.

Construction 7 ($\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}$ **protocol)** *Let* $\mathsf{C} : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$ *be an* n*-party circuit and let* $\mathcal{T}$ *be a set of parties such that* $|\mathcal{T}| < n$*. The protocol* $\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}$ *for* C *is defined to be the same as the* $\mathsf{GMW}_{\mathsf{C}}$ *protocol form Construction 4 except that the parties execute the* $\mathsf{Mult}_{\mathsf{GMW}}^{\mathcal{T}}$ *protocol instead of* $\mathsf{Mult}_{\mathsf{GMW}}$*.*

Lemma 2. *Let* C *be an* n*-party circuit. For any set of parties* $\mathcal{T}$ *such that* $|\mathcal{T}| < n$ *the protocol* $\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}$ *is* $\mathcal{T}$*-randomized and is* $\mathcal{T}$*-equivalent to* $\mathsf{GMW}_{\mathsf{C}}$*.*

8.2 The GMW Protocol in the Presence of an Active Adversary

In this section we prove that the execution of the passively secure GMW protocol is additively corruptible. We begin by stating that $\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}$ defined in Construction 4 is $\mathcal{T}$-last-round-private as well as $\mathcal{T}$-homomorphic to C.

Lemma 3. *Let* n *be positive integer and let* C *be an* n*-party circuit. Then for any set of parties* $\mathcal{T}$ *such that* $|\mathcal{T}| < n$ *it holds that the protocol* $\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}$ *for computing* C *is* $\mathcal{T}$*-last-round-private as well as* $\mathcal{T}$*-homomorphic to* C*.*

Proof (sketch). The $\mathcal{T}$-last-round-private property follows from the fact that during the output recovery phase of the $\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}$, all the parties locally re-randomize their shares with random sharings of 0. We now prove that $\mathsf{C}_{\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}}$ is indeed $\mathcal{T}$-homomorphic to C. Fix randomness $\mathbf{r}$ for $\mathsf{C}_{\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}}$. Next, for any input gate c of C, we set the homomorphism $\mathcal{H}$ to map c to the corresponding input gate in $\mathsf{C}_{\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}}$. Finally, for every multiplication gate c of C, we set $\mathcal{H}$ to map c to a wire in $\mathsf{C}_{\mathsf{GMW}_{\mathsf{C}}^{\mathcal{T}}}$ corresponding to the share c_h, held by the party P_h in step 4 of the $\mathsf{Mult}_{\mathsf{GMW}}^{\mathcal{T}}$ protocol. Finally, we set λ^{c} to be the sum of all the shares c_i generated during steps 2 and 3 of $\mathsf{Mult}_{\mathsf{GMW}}^{\mathcal{T}}$. Notice that since $\mathsf{Mult}_{\mathsf{GMW}}^{\mathcal{T}}$ is $\mathcal{T}$-randomized, λ^{c} can be uniquely determined from $\mathbf{r}$. It can be easily verified that for every choice of $\mathbf{r}$ the homomorphism $\mathcal{H}$ as well as the constants λ^{c}, where c is a multiplication gate, satisfy all the requirements of Definition 9. □

Combining the results of Lemmas 2,3 and Theorem 8 with additive-attack constructions in Section 5 we obtain the following theorem.

Theorem 10 (Cf. Theorem 1.5 in [11]). *For any* n*-party circuit* $\mathsf{C} : \mathbb{F}^{I_1} \times \cdots \times \mathbb{F}^{I_n} \to \mathbb{F}^{O_1}$ *there exists a protocol* π *for* $O(1/|\mathbb{F}|)$*-securely computing* C *with abort in the OLE hybrid model. Moreover* π *invokes the OLE oracle* $O(n^2|\mathsf{C}|)$ *times and has a total communication complexity of* $O(n^2|\mathsf{C}|)$ *field elements.*

Acknowledgments

This research was supported by the European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC. The first author was also supported by the Check Point Institute for Information Security; the Israeli Centers of Research Excellence I-CORE program (center 4/11); the Israeli Ministry of Science and Technology; the Leona M. & Harry B. Helmsley Charitable Trust. The second author was also supported by ISF grant 1709/14, and BSF grant 2012378. The third author was also supported by the Danish National Research Foundation; the National Science Foundation of China (grant no. 61061130540) for the Sino-Danish CTIC; the CFEM supported by the Danish Strategic Research Council.

References

1. Baron, J., El-Defrawy, K., Lampkins, J., Ostrovsky, R.: How to withstand mobile virus attacks, revisited. In: PODC. pp. 293–302 (2014)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10 (1988)
3. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multi-party computation with a dishonest minority. In: CRYPTO. pp. 663–680 (2012)
4. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: EUROCRYPT. pp. 169–188 (2011)
5. Bracha, G.: An O(log n) expected rounds randomized byzantine generals protocol. J. ACM 34(4), 910–920 (1987)
6. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC. pp. 494–503 (2002)
7. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC. pp. 11–19 (1988)
8. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: EUROCRYPT. pp. 445–465 (2010)
9. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: CRYPTO. pp. 572–590 (2007)
10. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: STOC. pp. 699–710 (1992)
11. Genkin, D., Ishai, Y., Prabhakaran, M., Sahai, A., Tromer, E.: Circuits resilient to additive attacks with applications to secure computation. In: STOC. pp. 495–504 (2014)
12. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC. pp. 218–229 (1987)
13. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography, Springer (2010)
14. Ikarashi, D., Kikuchi, R., Hamada, K., Chida, K.: Actively private and correct mpc scheme in t<n/2 from passively secure schemes with small overhead. IACR Cryptology ePrint Archive 2014, 304 (2014)
15. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC. pp. 21–30 (2007)
16. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: CRYPTO. pp. 572–591 (2008)
17. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: TCC. pp. 294–314 (2009)
18. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: EUROCRYPT. pp. 52–78 (2007)
19. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006)
20. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS. pp. 160–164 (1982)