Circuits Resilient to Additive Attacks with Applications to Secure Computation

Daniel Genkin
Technion
danielg3@cs.technion.ac.il

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

Manoj M. Prabhakaran
University of Illinois, Urbana-Champaign
mmp@uiuc.edu

Amit Sahai
University of California, Los Angeles
sahai@cs.ucla.edu

Eran Tromer
Tel Aviv University
tromer@cs.tau.ac.il

ABSTRACT

We study the question of protecting arithmetic circuits against *additive* attacks, which can add an arbitrary fixed value to each wire in the circuit. This extends the notion of algebraic manipulation detection (AMD) codes, which protect *information* against additive attacks, to that of *AMD circuits* which protect *computation*.

We present a construction of such AMD circuits: any arithmetic circuit C over a finite field $\mathbb{F}$ can be converted into a functionally-equivalent randomized arithmetic circuit $\widehat{C}$ of size $O(|C|)$ that is fault-tolerant in the following sense. For any additive attack on the wires of $\widehat{C}$, its effect on the output of $\widehat{C}$ can be simulated, up to $O(|C|/|\mathbb{F}|)$ statistical distance, by an additive attack on just the input and output. Given a small tamper-proof encoder/decoder for AMD codes, the input and output can be protected as well.

We also give an alternative construction, applicable to small fields (for example, to protect Boolean circuits against wire-toggling attacks). It uses a small tamper-proof decoder to ensure that, except with negligible failure probability, either the output is correct or tampering is detected.

Our study of AMD circuits is motivated by simplifying and improving protocols for secure multiparty computation (MPC). Typically, securing MPC protocols against *active* adversaries is much more difficult than securing them against *passive* adversaries. We observe that in simple *passive-secure* MPC protocols for circuit evaluation, the effect of any *active* adversary corresponds precisely to an additive attack on the original circuit's wires. Thus, to securely evaluate a circuit C in the presence of active adversaries, it suffices to apply the passive-secure protocol to $\widehat{C}$. We use this methodology to simplify feasibility results and attain efficiency improvements in several standard MPC models.

STOC '14, May 31 - Jun 03 2014, New York, NY, USA
ACM 978-1-4503-2710-7/14/05.
http://dx.doi.org/10.1145/2591796.2591861

Keywords

Cryptography, fault tolerant circuits, secure computation

Categories and Subject Descriptors

F.1.2 [**Modes of Computation**]: Interactive and reactive computation

General Terms

Theory

1. INTRODUCTION

1.1 Overview

The study of fault-tolerant circuits dates back to the work of von Neumann [33], who considered a model where every gate in a circuit can fail with some constant, and independent, probability. Subsequent works of Dobrushin and Ortyukov [12] and Pippenger [31] showed how to construct fault-tolerant circuits in this model with only a logarithmic overhead in the worst case and a constant overhead in the typical case. Other models for fault-tolerant circuits, protecting against a bounded number of adversarial faults, were studied in [28, 14, 15, 22, 13, 29, 8, 25, 9].

In the present work we consider the goal of protecting boolean and arithmetic circuits against adversarial faults that may apply to *all* wires in the circuit. Even if one settles for *detecting* faults rather than fully protecting against faults, this goal would be too ambitious. Indeed, an attacker can simply rewrite the input or the output of the circuit without being detected. But there is a natural model, which is also motivated by the cryptographic applications discussed later, where achieving this goal is conceivable. In this model we limit the attacker in two ways:

1. The attacker cannot directly attack the input and the output to the circuit; instead, the input is fed to a small (randomized) tamper-proof input encoder and the output is obtained from a small tamper-proof output decoder.[1]

[1]By "small" we mean independent of the circuit complexity of the function being computed (but possibly depending polynomially on the input/output size). This rules out a trivial solution where the entire computation is carried out by tamper-proof hardware.

2. The class of attacks – i.e., mappings from the original wire values to the new wire values – is restricted.

Note that (1) alone is insufficient to remove the impossibility, since it does not rule out completely rewriting the output of the input encoder or the input to the output decoder, and (2) alone is insufficient since it does not rule out direct (albeit restricted) attacks on the input or output.

We instantiate (2) by considering *additive attacks*. Given an arithmetic circuit over a finite field $\mathbb{F}$, we allow an adversary to "blindly" add a field element of his choice to each wire in the circuit. In the case of boolean circuits, this amounts to toggling an arbitrary subset of the wires. Such additive attacks were previously considered in the context of error-correcting codes by Karpovsky et al. [26] and more recently by Cramer et al. [7], who constructed *algebraic manipulation detection* (AMD) codes which that such attacks.[2] In this work we extend the notion of AMD codes, which protect *information* against additive attacks, to *AMD circuits*, which protect *computations* against such attacks.

We will start by defining a simpler notion of security against additive attacks (see Definition 1.1) that does not use any tamper-proof components (i.e., only the restriction (2) from above is used), but (inevitably) allows additive attacks on the input and output of the circuit. We show how to compile any arithmetic circuit C over a large finite field $\mathbb{F}$ into a functionally equivalent randomized arithmetic circuit $\widehat{C}$ of size $O(|C|)$ which is secure in this sense. The effect any additive attack has on the output of $\widehat{C}$ can be simulated, up to $O(|C|/|\mathbb{F}|)$ statistical distance, by applying a (randomized) additive attack to the input and output alone. Thus, as far as additive attacks are concerned, $\widehat{C}$ is essentially as good as a tamper-proof implementation of C in which only the input and output are exposed.

Combining the above construction with small tamper-proof encoder and decoder for AMD codes, the input and output can be protected as well. That is, any arithmetic circuit C over a large finite field can be compiled into a functionally equivalent randomized circuit of comparable size that uses small tamper-proof input encoder and output decoder, and is guaranteed to either produce the correct output of C or set an error flag, except with negligible failure probability. This construction has an additional security feature that will be useful for our motivating applications: Even in the presence of an additive attack, the field elements fed into the output decoder (and in particular the final output) reveal essentially nothing about the input x beyond $C(x)$.

The above construction offers no security guarantees when $\mathbb{F}$ is small. For the general case we present a more complex construction which uses small tamper-proof encoder and decoder to ensure that, except with negligible failure probability, either the output is correct or tampering is detected. More concretely, to achieve $2^{-\sigma}$ error probability, the size of the AMD circuit $\widehat{C}$ is $|C| \cdot \mathrm{poly}(\sigma)$. This construction can be used for protecting boolean circuits against wire-toggling attacks. However, here we do not realize the stronger security feature discussed above.

[2]In [7], algebraic manipulation detection codes were defined over an Abelian group, where the only manipulation allowed is an additive attack. We too are considering additive attacks, but since we work over a field (which contains a multiplication operation as well), a more appropriate term in our context would be "*Additive* Manipulation Detection" codes.

Cryptographic applications of AMD circuits. Our study of AMD circuits is further motivated by observing that they are useful for the design of protocols for *secure multi-party computation* (MPC). An MPC protocol allows two or more mutually distrusting parties to perform a distributed computation on their local inputs without compromising the secrecy of the inputs or the correctness of the outputs. Following the seminal works from the 1980s that established the general *feasibility* of secure computation [34, 17, 3, 6, 32], significant research efforts have been invested into studying *efficiency* questions in this area.

It is typically much easier to secure MPC protocols against *passive* adversaries, who may try to learn information about secret inputs but do not otherwise deviate from the protocol, than against *active* adversaries who may arbitrarily deviate from the protocol. The security of protocols that were only designed to withstand passive attacks may break down completely if the adversary is active. While there are general techniques for strengthening security against passive attacks into security against active attacks (most notably, the "GMW paradigm" [17]), these involve a considerable overhead and do not apply at all to the type of protocols considered here.

Our key observation is that in natural MPC protocols that offer information-theoretic security against passive attacks, any cheating strategy of an active adversary can be modeled as an *additive* attack on the underlying circuit. This holds both for protocols in the setting of an honest majority, such as the "BGW protocol" [3] and its more efficient variant from [11], and for protocols in the setting of no honest majority, such as variants of the "GMW protocol" over an ideal oblivious transfer oracle [17, 16] or an OLE oracle[3] [21].

The above observation gives rise to a novel methodology for the design of MPC protocols with security against active adversaries. Instead of designing a complex protocol for evaluating f that explicitly protects against active attacks, apply a simple protocol, which was only designed to protect against passive attacks, to evaluate an AMD circuit for f. (The role of the input encoder and the output decoder can be emulated via local computation and does not require interaction.) Thus, the most challenging aspect of MPC protocol design is reduced to the arguably cleaner problem of AMD circuit design.

We demonstrate the usefulness of this methodology by applying it to simplify and improve on previous results in the area of MPC. We derive the feasibility of active-secure MPC in the presence of an honest majority [32] from the much simpler passive-secure BGW protocol [3], as well as the feasibility of active-secure MPC protocols with no honest majority [17, 27, 20, 21] (given an OLE oracle) from their much simpler passive-secure counterparts. We also obtain a new feasibility result for MPC with no honest majority using a corruptible source of correlated randomness. On the efficiency front, we apply our methodology to a simplified variant of a passive-secure protocol from [11] to obtain a simpler and more efficient alternative to a recent protocol from [4]. We also obtain the first active-secure two-party protocol for evaluating an arbitrary arithmetic circuit over a large field using only a constant number of calls to an OLE oracle for each gate in the circuit.

[3]An OLE oracle receives $a, b \in \mathbb{F}$ from one party and $x \in \mathbb{F}$ from another, and returns $ax + b$ to the latter. OLE can be viewed as an arithmetic generalization of oblivious transfer.

1.2 Our contribution

We now give a more detailed outline of our results. In Section 1.2.1 we summarize results on protecting circuits against additive attacks and in Section 1.2.2 we summarize the applications to secure multiparty computation.

1.2.1 Protecting circuits against additive attacks

We start by defining our main notion of security with respect to additive attacks. Let $f : \mathbb{F}^n \to \mathbb{F}^k$ be a function to be computed. We say that a randomized arithmetic circuit[4] $\widehat{C}$ is an ϵ-secure implementation of f if $\widehat{C}$ correctly computes f when it is not attacked, and moreover any additive attack on $\widehat{C}$ has the same effect on the output of $\widehat{C}$ (up to an ϵ statistical error) as applying some additive attack to the inputs and outputs alone:

DEFINITION 1.1 (ADDITIVE-ATTACK SECURITY). *A randomized circuit $\widehat{C} : \mathbb{F}^n \to \mathbb{F}^k$ is an* ϵ-secure implementation *of a function $f : \mathbb{F}^n \to \mathbb{F}^k$ if the following holds:*

- Completeness. *For all $\mathbf{x} \in \mathbb{F}^n$,* $\Pr[\widehat{C}(\mathbf{x}) = f(\mathbf{x})] = 1$.
- Additive-attack security. *For any circuit $\widetilde{C}$ obtained by subjecting $\widehat{C}$ to an additive attack, there exists $\mathbf{a}^{\mathsf{in}} \in \mathbb{F}^n$ and a distribution $\mathcal{A}^{\mathsf{out}}$ over $\mathbb{F}^k$ such that for any $\mathbf{x} \in \mathbb{F}^n$ it holds that*

$$SD\left(\widetilde{C}(\mathbf{x}), f(\mathbf{x} + \mathbf{a}^{\mathsf{in}}) + \mathcal{A}^{\mathsf{out}}\right) \leq \epsilon,$$

where SD denotes statistical distance between two distributions.

We extend the definition to the case where f is a randomized *function by requiring that the output distribution of $\widehat{C}(\mathbf{x})$ and $f(\mathbf{x})$ be identical. We say that $\widehat{C}$ is an ϵ-secure implementation of a circuit C if $\widehat{C}$ is an ϵ-secure implementation of the (possibly randomized) function f computed by C.*

In Sections 4 and 5 we prove that every circuit C over a large finite field can be compiled into a circuit $\widehat{C}$ that is secure against additive attacks.

THEOREM 1.1. *For any field $\mathbb{F}$ and (possibly randomized) arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ there exists a randomized circuit $\widehat{C} : \mathbb{F}^n \to \mathbb{F}^k$ such that $\widehat{C}$ is an ϵ-secure implementation of C where $\epsilon = O(|C|/|\mathbb{F}|)$. Moreover, $|\widehat{C}| = O(|C|)$.*

The notion of additive-attack security in Definition 1.1 above still allows for an attack on the inputs and outputs of the circuit. This is because the adversary is allowed to attack every wire in the circuit, and in particular input and output wires. Thus, we need a randomized, *tamper-proof* input encoder Enc and output decoder Dec in order to prevent attacks against the inputs and outputs. We would like the size of Enc and Dec to be kept as small as possible (and in particular much smaller than the circuit being computed).

Notice that even in the presence of a decoder that cannot be attacked, the adversary is still allowed to attack all the wires leading from the circuit to the decoder. Thus, we cannot hope for *correcting* the result following an additive attack but must settle for a weaker guarantee of *detecting* the attack. We capture this by allowing Dec to have a special output, denoted *flag*, where if this output is nonzero this means that an attack has been detected.

The circuits Enc and Dec will perform an AMD encoding of the input and an AMD decoding of the output, respectively. The circuit $\widehat{C}$, which gets input from Enc and produces output for Dec, is obtained by applying Theorem 1.1 to the circuit C' obtained from C by applying an AMD decoder to its input and an AMD encoder to its output.

Theorem 1.1 does not provide any security guarantees for circuits over small fields. In particular, it cannot be used to protect boolean circuits. To handle general fields, we need to rely on a small, tamper-proof output decoder. Moreover, unlike the previous construction, here we only guarantee the *correctness* of the output and do not provide any guarantees regarding the secrecy of the input in the presence of additive attacks. Below we define the stronger notion of correctness-without a tamper-proof input encoder. (As before, the input can be protected by an input encoder.) This feature will be useful when applying a composition-based approach for constructing AMD circuits in this setting.

DEFINITION 1.2. *Let $\mathbb{F}$ be a finite field and let $f : \mathbb{F}^n \to \mathbb{F}^k$. We say that a pair of circuits $(\widehat{C}, D)$ are an ϵ-correct implementation of f with a decoder if the following holds:*

- Completeness. *For all $\mathbf{x} \in \mathbb{F}^n$, we have* $\Pr[D(\widehat{C}(\mathbf{x})) = (0, f(\mathbf{x}))] = 1$.
- Additive-attack correctness. *For any circuit $\widetilde{C}$ obtained by subjecting $\widehat{C}$ to an additive attack there exists $\mathbf{a}^{\mathsf{in}} \in \mathbb{F}^n$ such that for all $\mathbf{x} \in \mathbb{F}^n$*

$$\Pr\left[D(\widetilde{C}(\mathbf{x})) \notin \mathsf{ERR} \cup \{(0, f(\mathbf{x} + \mathbf{a}^{\mathsf{in}}))\}\right] \leq \epsilon$$

where $\mathsf{ERR} = \{(z', z) : z' \in \mathbb{F} \setminus \{0\}, z \in \mathbb{F}^k\}$ and the probability is taken over the internal randomness of $\widetilde{C}$.

In the full version we prove the following theorem.

THEOREM 1.2. *For any field $\mathbb{F}$, positive integer σ and arithmetic circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ there exist $(\widehat{C}, D)$ that form an ϵ-correct implementation of C with a decoder, where $\epsilon = 2^{-\sigma} \cdot |C|$, $|\widehat{C}| = |C| \cdot poly(\sigma)$, and $|D| = k \cdot poly(\sigma)$.*

Notice the differences between Theorems 1.1 and 1.2 above. Theorem 1.1 guarantees security for arithmetic circuits over large fields while Theorem 1.2 achieves the weaker notion of additive-attack correctness, which allows information to leak via the error flag, but without requiring the underlying field to be large. In particular, Theorem 1.2 can be used over the binary field.

1.2.2 Multiparty computation via AMD circuits

The notion of AMD circuits is motivated by the following application to secure multiparty computation (MPC). Our goal is to construct MPC protocols that are secure against active adversaries, starting from those which are secure only against passive adversaries. Unlike the prevalent approach of modifying the protocol itself to directly handle any deviations of an active adversary, our approach is to use the protocol as it is, but apply it to a modified circuit. That is, given an MPC protocol, secure against passive adversaries, for a function f computed by a circuit C, we apply the *same*

[4] An arithmetic circuit consists of field addition, subtraction, and multiplication gates. If it is randomized, it may also include randomness gates that output uniformly random field elements. We write $\widehat{C} : \mathbb{F}^n \to \mathbb{F}^k$ to indicate that the input of $\widehat{C}$ consists of n field elements (not including randomness gates) and its output consists of k field elements.

Adv	Resilience	Security	Communication complexity	Model	Ref
passive	$\|T\| < n/2$	perfect	$O(n^2\|C\|)$	plain	[3]
passive	$\|T\| < n/2$	perfect	$O(n\|C\| + n^2)$	plain	[11]
passive	$\|T\| < n$	perfect	$O(n^2\|C\|)$ for boolean circuits	OT	[17]
passive	$\|T\| < n$	perfect	$O(n^2\|C\|)$	OLE	[21]
active	$\|T\| < n/2$	statistical	$\text{poly}(n) \cdot \|C\|$	plain	[32]
active	$\|T\| < n/2$	statistical	$O(n\|C\| + n^2 \log n \cdot d_C) + \text{poly}(n)$	plain	[4]
active	$\boldsymbol{\|T\| < n/2}$	**statistical**	$\boldsymbol{O(n^2\|C\|)}$	**plain**	**Theorem 1.3**
active	$\boldsymbol{\|T\| < n/2}$	**statistical**	$\boldsymbol{O(n\|C\| + n^2)}$	**plain**	**Theorem 1.4**
active	$\|T\| < n$	statistical	$O(n^2\|C\| + \log \|\mathbb{F}\| \cdot d_C)$	OT+OLE	[21]
active	$\|T\| < (1/2 - \epsilon)n$	statistical	$O(\log n \cdot \|C\|) + \text{poly}(n, d_C)$	plain	[10]
active	$\boldsymbol{\|T\| < n}$	**statistical**	$\boldsymbol{O(n^2\|C\|)}$	**OLE**	**Theorem 1.5**
active	$\boldsymbol{\|T\| < n}$ **or** $\boldsymbol{T = \{\text{dealer}\}}$	**statistical**	$\boldsymbol{O(n^2\|C\|)}$	**plain**	**Theorem 1.6**

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 over a finite field $\mathbb{F}$, d_C is the multiplicative depth of C, and T is the set of parties corrupted by the adversary. Statistical security means that the protocol securely realizes C (with abort) with at most $O(|C|/|\mathbb{F}|)$ simulation error. The communication complexity column counts the total number of field elements that are communicated between the parties (where in the plain model we assume only the availability of secure point-to-point channels). An OLE oracle (an arithmetic generalization of OT) receives $a, b \in \mathbb{F}$ from one party and x from another, and returns $ax + b$ to the latter. The results highlighted in boldface are new.

protocol to a modified circuit $\widehat{C}^{\mathsf{AMD}}$. The circuit $\widehat{C}^{\mathsf{AMD}}$, in addition to computing the function, is also responsible for handling any consequences resulting from the adversary's deviations from the protocol. The circuit $\widehat{C}^{\mathsf{AMD}}$ will be essentially an additively secure version of the original circuit; we show that, for several simple MPC protocols from the literature that were only designed to provide security against passive adversaries, this approach suffices to handle general active adversaries. In the following we describe different applications of this methodology in the context of prior results (see summary in Table 1).

For simplicity we consider an MPC model where the adversary can abort the execution of the protocol, and do not attempt to provide guaranteed output delivery. The latter can be achieved when there is an honest majority and a broadcast channel is available [32].[5] We note, however, that protecting against active attacks is highly nontrivial even in this setting, and moreover the efficiency comparison with previous works takes this simpler model into account.

We begin by deriving a simple version of the result of [32], for MPC in the presence of an honest majority, from the passive-secure BGW protocol with $n = 2t + 1$ parties.

THEOREM 1.3. *For any n-party functionality f represented by an arithmetic circuit C over a sufficiently large $\mathbb{F}$ there exists a protocol π that ϵ-securely computes f with abort in the presence of an honest majority for $\epsilon = O(|C|/|\mathbb{F}|)$. The protocol involves communication of $O(n^2|C|)$ field elements.*

Here and in the following, one can eliminate the dependence of the error on the field size by using an extension field. This results in a multiplicative overhead of at most σ for reducing the error to $2^{-\sigma}$.

Next, we obtain a more efficient variant which has a communication complexity of $O(n|C| + n^2)$ field elements. This asymptotically matches the communication complexity of the best known *passive-secure* protocol from [11], and is obtained by applying our methodology to this protocol.

[5]Our protocols can be modified to have this feature, whenever it is achievable, by using standard techniques; however, the details are beyond the scope of this work.

THEOREM 1.4. *For any n-party functionality f represented by an arithmetic circuit C over a sufficiently large $\mathbb{F}$ there exists a protocol π with communication complexity of $O(n|C| + n^2)$ field elements, where π ϵ-securely computes f with abort in the presence of an honest majority for $\epsilon = O(|C|/|\mathbb{F}|)$.*

This gives a simpler alternative to a recent protocol from [4] and improves its complexity by eliminating a quadratic overhead for each layer of the circuit, as well as a large polynomial additive term. See Table 1.

Next, we tackle the task of secure multiparty computation in the presence of an active adversary without an honest majority. Unfortunately, this task is impossible to achieve for arbitrary circuits in the plain model. Thus, we are forced to use some kind of a hybrid model or have an honestly-executed input-independent preprocessing phase which is done before the execution of the protocol. In the full version we present results for secure multiparty computation using an arithmetic generalization of the OT-hybrid model, called the OLE-hybrid model [30, 21] (where an oracle receives $a, b \in \mathbb{F}$ from one party and x from another, and returns $ax + b$ to the latter). In addition, we also present our results in the preprocessing model.

Concretely, we use an arithmetic version of the GMW protocol [17, 21] and obtain an n-party protocol for securely computing any functionality (represented by an arithmetic circuit C), without requiring an honest majority, using $O(n^2|C|)$ calls to the OLE oracle. This improves over the protocol of [21] that inherently requires $\Omega(\sigma)$ additional oracle calls for achieving $2^{-\sigma}$-security, regardless of the field or circuit size.

THEOREM 1.5. *For any n-party functionality f represented by an arithmetic circuit C there exists a protocol π in the OLE-hybrid model that $O(|C|/|\mathbb{F}|)$-securely computes C with abort. Moreover, π invokes the OLE oracle $O(n^2|C|)$ times.*

Finally, we address the goal of secure multiparty computation in the preprocessing model. We present a protocol for securely computing an n-party functionality (again represented as an arithmetic circuit C) that utilizes a preprocessing phase which runs before the computation of f starts

and does not depend on the inputs to f. This phase can be implemented using an additional party called the *dealer* that sends correlated randomness to the parties. We strengthen previous results in the preprocessing model [21, 5, 18], which assume the dealer to be honest, by providing security when either the dealer or any subset of the other parties may be corrupted (though not both).

THEOREM 1.6. *For any n-party functionality represented by an arithmetic circuit C over $\mathbb{F}$, there exists a protocol π that uses an additional dealer, such that π is an ϵ-secure protocol for computing C with abort against an active adversary controlling either the dealer or any other parties, where $\epsilon = O(|C|/|\mathbb{F}|)$. The dealer in π only distributes correlated randomness to the other parties.*

2. OVERVIEW OF TECHNIQUES

2.1 Additive-attack security

We first present our result for additive-attack security (see Section 4 for details) for a computation over a large field $\mathbb{F}$. Suppose we are given an arithmetic circuit C. In its secure version $\widehat{C}$, every wire of C is paired with a wire that carries a "MAC tag." Each gate in C is replaced by a small gadget which computes the original gate's output as well as a MAC tag for it; further, this gadget accepts the MAC tags of the inputs to the original gate, and carries out a MAC verification computation. Note that this verification circuitry itself is open to additive attacks. Nevertheless, we can arrange that $\widehat{C}$ will produce a random output if the MAC tag verification fails for a wire anywhere in the computation.

In the following we identify the gate g with its result; the meaning will be clear from the context.

The basic construction. For a gate's output wire g, its MAC value will simply be $g \cdot v^d$, where v is a randomly generated element of the underlying field $\mathbb{F}$ (fixed to the same value for all gates), and d is the degree of the wire g (as a polynomial in the input variables). This MAC has property that multiplications can be performed on the MAC value homomorphically to obtain a value that can correspond to a MAC value of the result of a multiplication. Updating the MAC for an addition and subtraction gate is implemented using a simple gadget. The consistency check is implemented as follows. We first compute the result of the original gate, g. Next, we compute the MAC value in two ways. The first way is by directly multiplying g to v^d (in turn computed from v). The second way is using the MAC values of the inputs (homomorphically for multiplication and via a simple gadget for the case of addition and subtraction). We then check that the values are equal: more precisely, in each gate we take the difference of these two values, and linearly combine them across all gates using random coefficients; the result – which is a random field element if any inconsistency was detected, and 0 otherwise – is added to the final outcome.

We show that any additive attack on $\widehat{C}$ is either equivalent to an additive attack on the input wires and output wires only, or results in the output being random, up to $O(d/|\mathbb{F}|)$ statistical distance from the uniform distribution. Note that if the field is large (i.e., is of size exponential in the security parameter) and if d is small (for e.g., polynomial), we obtain negligible error in security. The security of the construction requires that the underlying circuit C be such that for every gate in C, the joint values of its two inputs should be almost uniformly distributed over $\mathbb{F} \times \mathbb{F}$. This is ensured by first compiling C into an appropriately randomized circuit (see below).

One problem with the above basic construction is that the security error grows with the degree of the circuit. Since the degree of a circuit can be exponential in its depth, this construction does not yield a full solution to our problem. However, we show that bootstrapping from this construction for low-degree circuits, we can indeed obtain a construction that is secure for all polynomial-sized circuits (see below).

The randomization process. As noted above, the basic construction relies on the inputs to each gate of the given circuit being uniformly random. We can enforce this as follows. Each wire a inside the circuit C will be replaced by two wires, carrying values $a+r_1$ and $a+r_2$, where the masking values r_1 and r_2 are generated as random field elements (that are the same throughout the circuit). Next, we will replace each multiplication and addition gate with a gadget that will get as input $(a + r_1, a + r_2, b + r_1, b + r_2)$ and output either $(ab + r_1, ab + r_2)$, $(a + b + r_1, a + b + r_2)$ or $(a - b + r_1, a - b + r_2)$ respectively. These gadgets have the property that the inputs of every internal gate are completely random. To complete the modification of the circuit, two additional layers are added. First, a layer of addition gates is added to the input wires to carry out the encoding. Next, a layer of subtraction gates is added to the output wires to carry out the decoding. Note that the inputs to the gates in these additional layers do not have the randomness property we set out to ensure for every gate (since the inputs and outputs are not random). However, attacks on these addition and subtraction gates are equivalent to attacks on the inputs and outputs of the circuit which are permitted by Definition 1.1.

From low-degree circuits to arbitrary circuits. Observe that additive-attack security could be easily achieved if we were allowed to use tamper-proof gadgets to implement each gate. Then each gate can be replaced by a tamper-proof component that gets two inputs encoded using an AMD code and, after decoding them, computes an AMD encoding of the gate's result. In our final construction, we implement these gadgets using the above construction for low-degree circuits and obtain a construction for arbitrary circuits.

2.2 Handling small fields

Our constructions for additive-attack security inherently fail when the underlying field is small, even if we were willing to tolerate a small constant error (see full version for details). We present an alternative construction that achieves additive-attack *correctness* over small fields, with negligible error. Recall that correctness prevents the attacker from causing the circuit to output a wrong value without being detected.

Basic construction without a decoder. Our final construction will achieve additive-attack correctness using a small tamper-proof output decoder. But first, we present a constant error construction that does not use any decoders but allows (inevitably) both inputs and outputs to be attacked. Later we will show how to amplify the correctness (and also improve the efficiency) of this construction, and meet the requirements of Definition 1.2, by relying on a small tamper-proof output decoder.

The basic idea is that our new circuit would compute not only the output of the original circuit, but also a *proof* that the output is correct; at the end of the computation, this proof will be verified by another part of the circuit. We need a simple proof system that can be implemented in such a way that soundness holds even when *the verifier as well as the prover could be under (additive) attack.*[6] Our proof system follows in the pattern of the Hadamard PCP system of [1], which turns out to have the linearity properties suitable for our purposes. However, we cannot use this PCP system as it is, since the proof is exponentially large. We use an alternate compact representation of the proof that suffices provided the prover indeed computes prescribed linear functions of a purported witness and the verifier's queries. This condition on the prover is enforced by a "matrix multiplication gadget" (see below). The verifier's computation is simple and results in an error flag to be set (to a non-zero value) with at least a constant probability, if the proof is not valid.

It remains to ensure that under additive attack, the prover is restricted to computing the correct linear functions (but possibly using an invalid witness). This is achieved using the following gadgets.

The multiplication gadgets. We sketch our ϵ-correct implementation (without output decoder) of a matrix-by-vector multiplication. For this, first we construct a scalar-by-vector multiplication gadget.

The inputs to a scalar-by-vector multiplication gadget consist of a vector $\mathbf{v}$ and a scalar x, the output is $\mathbf{v}x$. The idea of this construction is to make the circuit compute $\mathbf{z} = \mathbf{v}x$ and $q' = (\mathbf{r} \cdot \mathbf{v})\, x$ where $\mathbf{r}$ is a random vector. To verify that $\mathbf{r} \cdot \mathbf{z} = q'$, we compute $f = \mathbf{r}\mathbf{z} - q'$ as an error flag. We show that any attack that does not correspond to an additive attack on the inputs and outputs of the scalar-by-vector multiplication gadget will cause the flag to be set randomly.

We next proceed to the matrix-by-vector multiplication gadget. The inputs to such a gadget are a matrix M and a vector $\mathbf{x}$, and the output is $\mathbf{z} = M\mathbf{x}$. We implement this gadget using the scalar-by-vector multiplication gadget. The main idea is as follows: we treat the *columns* of M as vectors and multiply each column with the required coordinate of $\mathbf{x}$ using the scalar-by-vector multiplication gadget. Afterwards, we sum up these intermediate values to obtain the output of the matrix-by-vector multiplication. Since any attack on the scalar-by-vector multiplication gadget is equivalent to an attack on its inputs and outputs and since the matrix-by-vector multiplication gadget only sums up the results of the scalar-by-vector multiplication gadget, it will be the case that any attack on the matrix-by-vector multiplication gadget is either equivalent to an attack on its inputs and outputs or it causes its error flag to become non-zero with constant probability.

Using this gadget to implement the prover in the above outline, we obtain the following result.

THEOREM 2.1 (INFORMAL). *Any circuit C over a finite field $\mathbb{F}$ admits a 0.997-correct implementation without output decoder, $\widehat{C}$, where $|\widehat{C}| = O(|C|^2)$.*

Correctness amplification. For small fields (in particular, the binary field), the above construction has a high error probability. A naive attempt at reducing the error to ϵ^σ would be to repeat the ϵ-correct construction σ times, and then use a (tamper-proof) decoder to check for consistency. However, it is possible that different instances will be operating on different inputs, and therefore no amplification will be achieved (see full version). This problem can be solved by asking each instance of the construction to output its input in addition to the result and then using a decoder to verify that all the inputs are consistent. However, in this case the complexity of the decoder will be polynomial in the input size. Keeping the tamper-proof decoder size virtually independent (up to logarithmic factors) of the input size is crucial for the efficiency improvement we discuss next. Thus, we use a family of (almost pairwise independent) hash functions such that the circuit will output a hash digest of its input instead of the actual input. Since input consistency is still verified, attacks that cause different instances of the construction to operate on different inputs will cause inconsistency in the hash digests, and the decoder will then set the error flag wire to be non-zero.

THEOREM 2.2. *Any circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ admits a $2^{-\sigma}$-correct implementation $(\widehat{C}, D)$ where $|\widehat{C}| = |C|^2 \cdot poly(\sigma)$ and $|D| = k \cdot poly(\sigma)$.*

From quadratic to linear overhead. The above construction has quadratic overhead in the circuit size, since we use parts from the PCP prover of [1]. In particular, similarly to [1], our construction will compute all possible multiplications of two intermediate wires inside the circuit. We improve this using "bootstrapping", as follows. We go over the gates of C in topological order. For each input gate, we apply the above construction to the single-wire identity circuit, yielding an ϵ-correct gadget, and a corresponding decoder. Then, for each subsequent gate g, we consider the small circuit C' consisting of the decoders corresponding to the two gates of upstream of g, along with g itself, and apply the above construction to C' to yield an ϵ-correct gadget and a new decoder, and so on. These are wired together. Finally, the decoders corresponding to the output gates, taken together, are considered the decoder for the resulting ϵ-correct implementation of C. Since the substitution replaces a gate with a small gadget whose size is independent of C, the resulting circuit size grows linearly with that of $|C|$.

THEOREM 2.3. *Any circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ admits a $(2^{-\sigma} \cdot |C|)$-correct implementation $(\widehat{C}, D)$ where $|\widehat{C}| = |C| \cdot poly(\sigma)$ and $|D| = k \cdot poly(\sigma)$.*

2.3 Secure multiparty computation

We review the main techniques used for applying AMD circuits towards secure computation in the presence of an active adversary, as discussed in Section 1.2.2.

Protecting the computation of circuits. We start from a protocol π that evaluates a circuit C with security against *passive* adversaries. We prove, for several useful protocols π, that when π is executed in the presence of an *active* adversary, π actually computes a circuit $\widetilde{C}$ that is the same as C up to some additive attack that is chosen by the adversary. Thus, by replacing the circuit C with an additive-attack secure implementation $\widehat{C}$ of C we obtain that any active attack on the protocol corresponds to an additive attack on the inputs and outputs of C.

[6] Even if we allowed a small tamper-proof decoder (which we do not for this basic construction), it would not be feasible to house the verifier there, since the verifier would be at least as large as the original circuit itself; allowing such a large tamper-proof component trivializes the problem.

Protecting the inputs and outputs. To protect the inputs and outputs of C against additive attacks, we construct another circuit C^{AMD} from C so that C^{AMD} gets its inputs in some AMD code, decodes them, and then applies C. Finally, C^{AMD} encodes the outputs of C using an AMD code. In addition, if C^{AMD} gets inputs that are not valid AMD encodings due to an additive attack by the adversary, C^{AMD} sets a special output flag to be random. This will notify the honest parties that they should abort the computation since the results might have been corrupted by the adversary.

The final protocol. We construct an active-secure MPC protocol π' for C as follows. First, all the parties locally encode their inputs using an AMD code. Then they invoke π on an additive-attack secure implementation $\widehat{C}^{\mathsf{AMD}}$ of C^{AMD}. Finally, the parties locally decode the outputs of C^{AMD} obtained from the execution of π and abort if the decoding fails or if the error flag is nonzero. The security of π' is argued as follows. Notice that by the properties of π, the adversary is limited to only performing additive attacks on $\widehat{C}^{\mathsf{AMD}}$. Since $\widehat{C}^{\mathsf{AMD}}$ is additive-attack secure, these attacks are equivalent (up to small statistical distance) to additive attacks on the inputs and outputs of C^{AMD}. Finally, notice that any additive attack on the inputs and outputs of $\widehat{C}^{\mathsf{AMD}}$ will be detected by the AMD code, causing the honest parties to abort.

3. RELATED WORK

The goal of securing cryptographic hardware against active attacks has motivated different models for fault-tolerant circuits that mainly aim to protect the *secrecy* of the data stored inside the circuits. All prior works along this line somehow restrict the attacker so that some of the wires in the circuit are unaffected. This could be done by either restricting the number of attacked wires or by requiring that the attack fail with some probability. In our case, we eliminate this requirement by only considering the restricted class of additive attacks.

Gennaro et al. [15] and, more recently, Tauman-Kalai et al. [24] considered tampering attacks that apply only to the *memory* but not to the circuit logic. The work of Liu and Lysyanskaya [29] considered the question of protecting circuits against leakage and tampering in the split-state model, where the leakage and tampering functions are not allowed to operate on the entire circuit at once but only on different parts of it. Ishai et al. [22], as well Dachman-Soled and Tauman-Kalai [8, 9], considered a reactive setting where in each clock cycle, the circuit produces outputs as well as updates its internal state. In their model, no part of the circuit must be free from tampering, but the adversary is restricted to tampering with a *bounded* number of wires in each clock cycle. Finally, Faust et al. [13] considered a variant in which the adversary can attack every wire in the circuit, but each attack fails with some constant probability.

4. PROTECTING LOW-DEGREE CIRCUITS OVER LARGE FINITE FIELDS

In this section we construct ϵ-secure implementations for low-degree arithmetic circuits over large finite fields. The main idea behind the construction is as follows. We ensure that any additive attack on the circuit will have one of two consequences: it will either cause the circuit to output a random output for all inputs, or it will be equivalent to a set of wire corruptions on the inputs and the outputs of the circuit. To do so, we encode the values in the circuit and compute over encoded values. The special property of the encoding is that every additive attack on the encoded values will cause the encoding to become invalid. We first present a simpler construction whose security holds when the wire values satisfy some local randomness property (Section 4.1). Later, we eliminate this assumption by applying a general transformation to the circuit (Section 4.2). Finally, we combine the two together into a secure construction for low-degree circuits and arbitrary inputs (Section 4.3). We begin by presenting the security notion for specific input distributions.

DEFINITION 4.1. *Let $\mathbb{F}$ be a finite field, $C : \mathbb{F}^n \to \mathbb{F}^k$ an arithmetic circuit, and I a distribution over $\mathbb{F}^n$. We say that a circuit $\widehat{C}$ is an ϵ-secure implementation of C with respect to I if the following holds:*

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

4.1 Security for locally-random distributions

We now present a secure construction for constant degree circuits and specific input distributions. Similarly to the approach of [5] for secure computation with preprocessing (and somewhat similarly to the MAC-based quantum MPC protocol of [2]), our construction is based on a simple homomorphic MAC. However, in contrast to [5], we cannot rely on any tamper-proof component. The main idea is to add for every wire in the circuit another wire carrying its MAC value. When two wires enter a gate the two MAC values corresponding to them will enter a special circuit that will produce the expected MAC value of the gate's result. Afterwards, the result of the gate and the corresponding MAC value are checked. The MAC used will have the property that if an input to a gate is attacked then the MAC value produced separately for this gate will not verify with the gate's result. As soon as this situation is detected a special abort flag will become non-zero causing the entire circuit to output a random value.

The construction will guarantee security as defined in Definition 4.1 with $\epsilon = O\left(d/|\mathbb{F}|\right)$, where d is the degree of the circuit it is applied on, under two assumptions.

1. The inputs of each gate are sufficiently random. In Section 4.2 we present a transformation that will randomize the inputs of each gate in the circuit.
2. The input to the circuit is taken from a specific input distribution. In Section 4.3 we present a construction for arbitrary inputs.

Since the security of the construction depends on the degree of the circuit, the construction is only useful for low-degree circuits. We now define the required local randomness property.

DEFINITION 4.2. *Let $\mathbb{F}$ be a finite field, $C : \mathbb{F}^n \to \mathbb{F}^k$ a randomized arithmetic circuit, and I a distribution over $\mathbb{F}^n$. We say that C is locally ϵ-random with respect to I if for any $(y, z) \in \mathbb{F}^2$, and any pair of gates (g_1, g_2) whose outputs are the inputs to the same gate in C, it holds that the probability*

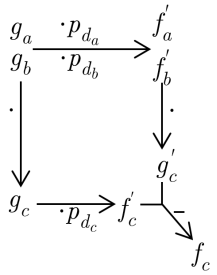

Figure 1: MAC computation for multiplication gates ($\cdot$ denotes field multiplication).

over $\mathbf{x} \leftarrow I$ *that the outputs of* (g_1, g_2) *in* $C(\mathbf{x})$ *are equal to* (y, z) *is at most* ϵ.

We now describe a construction that takes as input an ϵ-random circuit with respect to some class of input distributions and transforms it to a secure circuit with respect to the same class of distributions. The idea is as follows. For gate g_c with inputs g_a and g_b, the circuit will compute a MAC for g_c in two ways. The first way is by computing the gate's result and obtaining the MAC directly from the result. This MAC value is denoted in the construction below by f'_c. The second way is by homomorphically combining the input MACs f'_a and f'_b into a MAC for g_c. This MAC value is denoted below by g'_c. Finally, the circuit will verify that $f'_c = g'_c$. The guarantee of the MAC is that every additive attack is either harmless and will not affect the result, or it will be the case that $f'_c \neq g'_c$ with high probability. In the latter case, a special wire inside the circuit will become non-zero and will cause the entire circuit to output a random value. Intuitively, this guarantee is achieved by utilizing the fact that addition and multiplication do not commute.

See Figure 4.1 for the MAC for multiplication gates.

CONSTRUCTION 4.1. *Let* $C : \mathbb{F}^n \to \mathbb{F}^k$ *be a circuit. Let* g_i, $1 \leq i \leq |C|$, *denote the gates of* C *in some topological order. Define a circuit* $\widehat{C}$ *that on input* $\mathbf{x}$ *performs the following:*

1. *Compute* $\mathbf{z} = C(\mathbf{x})$.
2. *Generate a random field element* $v \in \mathbb{F}$.
3. *For* $i = 1, \ldots, \mathsf{deg}(C)$ *compute* $p_i = v^i$ *using multiplication gates.*
4. *For any gate* g_i *of degree* d_i, *compute* $f'_i = g_i \cdot p_{d_i}$.
5. *For each non-input gate* g_c, $c = n+1, \ldots, |C|$, *let* g_a *and* g_b *be its inputs and let* d_a *and* d_b *be their degrees. Compute the value* g'_c *as follows:*
   - *If* g_c *is a multiplication gate, let* $g'_c = f'_a \cdot f'_b$.
   - *If* g_c *is an addition gate: (1) if* $d_a > d_b$ *let* $h'_c = p_{d_a - d_b} \cdot f'_b$ *and* $g'_c = f'_a + h'_c$, *(2) if* $d_a < d_b$ *let* $h'_c = p_{d_b - d_a} \cdot f'_a$ *and* $g'_c = h'_c + f'_b$, *and (3) if* $d_a = d_b$ *let* $g'_c = f'_a + f'_b$.
   - *The case of a subtraction gate is handled similarly.*
6. *For any non-input gate* g_i, *let* $f_i = f'_i - g'_i$.
7. *Let* $f = \sum_i f_i r_i$ *where* r_i *is a random field element.*
8. *Output* $\mathbf{z} + f\mathbf{r}'$ *where* $\mathbf{r}'$ *is a random vector from* $\mathbb{F}^k$.

THEOREM 4.1. *Let* $C : \mathbb{F}^n \to \mathbb{F}^k$ *be a randomized arithmetic circuit of degree* d *which is locally* ϵ*-random with respect to a distribution* I. *Then the circuit* $\widehat{C}$ *obtained by applying Construction 4.1 to* C *is a* $(|\mathbb{F}|\epsilon + (d+1)/|\mathbb{F}|)$*-secure implementation of* C *with respect to* I.

4.2 Obtaining locally-random circuits

We now present a general transformation mapping any arithmetic circuit C into a locally random circuit C' whose output encodes the output of C. Similar transformations were previously used for the purpose of protecting circuits against leakage [23, 19]. We use a natural generalization of a transformation from [19] to the arithmetic setting and show that it satisfies the required local randomness property, namely that the pair of inputs to each gate in C' have almost full entropy. Similarly to [19], each wire a inside C (including input and output wires) will be split into two wires, one masked by r_1 and the other masked by r_2. Each gate c of C with inputs a, b will be replaced by a gadget that maps $(a+r_1, a+r_2, b+r_1, b+r_2)$ to $(c+r_1, c+r_2)$. The gadget has the property that the inputs of every internal gate are almost completely random (assuming that r_1 and r_2 are random). The two random field elements r_1, r_2 will be reused for the whole circuit.

CONSTRUCTION 4.2. *The gadget* add *is the circuit that, on input* $(v_1, v_2, v_3, v_4, r_1, r_2)$ *performs the following: (the circuit will be always used where* $v_1 = a + r_1$, $v_2 = a + r_2$, $v_3 = b + r_1$, $v_4 = b + r_2$*)*

1. *Compute* $v_5 = v_1 + v_4$ *(note that* $v_5 = a + b + r_1 + r_2$*).*
2. *Compute* $v_6 = v_5 - r_2$ *(note that* $v_6 = a + b + r_1$*).*
3. *Compute* $v_7 = v_5 - r_1$ *(note that* $v_7 = a + b + r_2$*).*
4. *Output* (v_6, v_7).

The gadget sub *is defined similarly.*

We define the multiplication gadget mult as follows:

CONSTRUCTION 4.3. *The gadget* mult *is the circuit that, on input* $(v_1, v_2, v_3, v_4, r_1, r_2)$ *performs the following: (the circuit will be always used where* $v_1 = a + r_1$, $v_2 = a + r_2$, $v_3 = b + r_1$, $v_4 = b + r_2$*):*

1. *Compute* $v_5 = v_1 v_4$ *(note that* $v_5 = ab + r_1 b + r_2 a + r_1 r_2$*).*
2. *Compute* $v_6 = v_1 r_2$ *(note that* $v_6 = ar_2 + r_1 r_2$*).*
3. *Compute* $v_7 = v_4 r_1$ *(note that* $v_7 = br_1 + r_1 r_2$*).*
4. *Compute* $v_8 = v_5 + r_2$ *(note that* $v_8 = ab + r_1 b + r_2 a + r_1 r_2 + r_2$*).*
5. *Compute* $v_9 = v_8 - v_6$ *(note that* $v_9 = ab + r_1 b + r_2$*).*
6. *Compute* $v_{10} = v_9 - v_7$ *(note that* $v_{10} = ab + r_2 - r_1 r_2$*).*
7. *Compute* $v_{11} = r_1 r_2$
8. *Compute* $v_{12} = v_{10} + v_{11}$ *(note that* $v_{12} = ab + r_2$*).*
9. *Similarly, compute* $v_{13} = ab + r_1$
10. *Output* (v_{12}, v_{13})

It is not hard to verify that for any $a, b \in \mathbb{F}$, the distribution of the pair of inputs of every gate inside add, sub and mult takes each value from $\mathbb{F}^2$ with at most $O\left(1/|\mathbb{F}|^2\right)$ probability.

We will now build our locally-random circuit. We start from a circuit C and replace all of its gates with their corresponding gadgets obtaining C'. Notice that both gadgets assume that every input wire x is split into two wires $x + r_1$ and $x + r_2$. We thus add additional gates to C' that generate two new random values r_1, r_2 and encode every input x_i of C to $(x_i + r_1, x_i + r_2)$. We also require that C' will output r_1 and the first element of each wire pair. This will allow us to define a decoder circuit Dec in order to decode the outputs.

CONSTRUCTION 4.4. *Let* $C : \mathbb{F}^n \to \mathbb{F}^k$ *be a circuit. Consider the circuits* (C', Dec) *that are defined as follows.*

- *The circuit C' is constructed as follows.*
  1. *C' on input $\mathbf{x}$ generates random field elements $r_1, r_2 \in \mathbb{F}$ for every i computes $x'_i = (x_i + r_1, x_i + r_2)$.*
  2. *Every gate of C is replaced in C' by the corresponding gadget as described above.*
  3. *Let $y_1, \ldots, y_k$ be the first elements from wire pairs corresponding to the output. C' will output $(y_1, \ldots, y_k, r_1)$.*
- *The circuit* Dec *on input $(y_1, \ldots, y_k, r_1)$ outputs $(y_1 - r_1, \ldots, y_k - r_1)$.*

We now claim that the circuits resulting from Construction 4.4 are an $O\left(1/|\mathbb{F}|^2\right)$-random implementation of C with respect to all input distributions in which every input element is (individually) uniform. Formally,

THEOREM 4.2. *For any circuit $C : \mathbb{F}^n \to \mathbb{F}^k$, the circuits (C', Dec) resulting from applying Construction 4.4 to C have the following properties:*

- Completeness. *For any $\mathbf{x} \in \mathbb{F}^n$, $C(\mathbf{x}) = \mathsf{Dec}(C'(\mathbf{x}))$.*
- Randomization. *The circuit C' is a $O(1/|\mathbb{F}|^2)$-random circuit with respect to every input distribution I in which each entry I_j is distributed uniformly over $\mathbb{F}$.*
- Complexity. *The size of C' is $O(|C|)$.*

4.3 Security for arbitrary inputs

We now present our construction for additive-attack security of low-degree circuits for arbitrary inputs. We use a simple randomized input encoding to ensure that the input distribution I satisfies the required local randomness property. We then define an *augmented* circuit that will receive such an input distribution, recover the input and compute the original circuit.

CONSTRUCTION 4.5. *Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be a circuit. We define the circuit $C_{\mathsf{AUG}} : \mathbb{F}^{n+1} \to \mathbb{F}^k$ that on input $x_0, \ldots, x_n$ outputs $C(x_1 - x_0, \ldots, x_n - x_0)$.*

We now present our construction for securing low-degree circuits over arbitrary inputs.

CONSTRUCTION 4.6. *Let $C : \mathbb{F}^n \to \mathbb{F}^k$ be a circuit. We construct $\widehat{C}$ from C via the following transformations.*

- *Construct C_{AUG} from C using Construction 4.5 (to unmask the inputs of C).*
- *Construct (C', Dec) from C_{AUG} using Construction 4.4 (to randomize all wires inside C_{AUG}).*
- *Construct $\widehat{C}'$ from C' using Construction 4.1 (to additively secure C').*

The circuit $\widehat{C}$ on input $\mathbf{x}$ preforms the following.

- *Generate a random field element $r \in \mathbb{F}$.*
- *Output $\mathsf{Dec}(\widehat{C}'(r, x_1 + r, \ldots, x_n + r))$.*

We claim that Construction 4.6 above transforms any low-degree circuit C to an additively secure circuit $\widehat{C}$.

THEOREM 4.3. *For any circuit C of degree d the circuit $\widehat{C}$ obtained by applying Construction 4.6 to C is an $O\left(d/|\mathbb{F}|\right)$-secure implementation of C.*

5. PROTECTING ARBITRARY CIRCUITS OVER LARGE FINITE FIELDS

In Section 4 we presented a transformation that meets the goal of additive-attack security (see Definition 1.1) for low-degree circuits. However, the field size required for obtaining security grows linearly with the degree of the circuit. Thus, in some cases, the field size must grow exponentially with the circuit size. In this section, we present a transformation where the field size can be much smaller. We first present a construction using small tamper-proof components and later eliminate these components.

5.1 A solution using tamper-proof components

Suppose we are given tamper-proof components G_{add}, G_{sub} and G_{mul}, that receive a pair of AMD-encoded inputs for a gate g, decode the inputs, compute the output of g, and finally produce a fresh AMD encoding of this output.

Such components can be used in a straightforward way to obtain AMD circuits: first, replace every addition, subtraction and multiplication gate in C with G_{add}, G_{sub} and G_{mul} respectively. Next, append to each output gate an AMD decoder circuit Dec to perform the output decoding. Finally, combine the error flags of all the Dec circuits such that in case one of the decodings fails, the output of the entire circuit will be random. Since $G_{\mathsf{add}}, G_{\mathsf{sub}}, G_{\mathsf{mul}}$ verify their inputs and encode the outputs, the security of the AMD code guarantees that any additive attack on the internal wires of C will be caught with high probability.

COROLLARY 5.1. *There exists a finite gate set $\mathcal{G}$ such that for any arithmetic circuit C over some finite field $\mathbb{F}$, there exists an arithmetic circuit $\widehat{C}$ of size $O(|C|)$ over $\mathcal{G}$ such that $\widehat{C}$ is an ϵ-secure implementation of C, where $\epsilon = 1 - (|\mathbb{F}| - 1)^2/|\mathbb{F}|^2$.*

5.2 Eliminating the tamper-proof components

The tamper-proof components of the previous construction can be eliminated in a natural way by first implementing each component using a (constant-size) arithmetic circuit and then applying the construction for low-degree circuits to protect this circuit against additive attacks. Security is implied by the following composition theorem.

CONSTRUCTION 5.1. *Let C be an arithmetic circuit, C' be an ϵ-secure implementation of C over some gate set $\mathcal{G}$ containing m components $G_1, \ldots, G_m$ from $\mathcal{G}$. In addition, for all $1 \leq i \leq m$ let $\widehat{G}_i$ be an ϵ_i-secure implementation of G_i. Consider the circuit $\widehat{C}$ over the gate set $\{+, -, \cdot\}$ where the gate G_i is replaced with $\widehat{G}_i$ for every i.*

THEOREM 5.1. *For any circuit $C : \mathbb{F}^n \to \mathbb{F}^k$ the circuit $\widehat{C} : \mathbb{F}^n \to \mathbb{F}^k$ constructed in Construction 5.1 is an $\left(\epsilon + \sum_{i=1}^{m} \epsilon_i\right)$-secure implementation of C.*

6. ACKNOWLEDGMENTS

D. Genkin and Y. Ishai were supported by European Union's Tenth Framework Programme (FP10/2010-2016) under grant agreement no. 259426 ERC-CaC. Y. Ishai was additionally supported by ISF grant 1361/10 and BSF grant 2012378. M. Prabhakaran was supported by NSF grants 07-47027 and 12-28856. A. Sahai was supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an Intel equipment grant, and an Okawa Foundation Research Grant. This material is based

upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. 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. E. Tromer was 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, ISF grant 1361/10 and NATO's Public Diplomacy Division in the Framework of "Science for Peace".

7. REFERENCES

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *FOCS*, pages 14–23, 1992.

[2] M. Ben-Or, C. Crépeau, D. Gottesman, A. Hassidim, and A. Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *FOCS*, pages 249–260, 2006.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[4] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *CRYPTO*, pages 663–680, 2012.

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

[6] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[7] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *EUROCRYPT*, pages 471–488, 2008.

[8] D. Dachman-Soled and Y. T. Kalai. Securing circuits against constant-rate tampering. In *CRYPTO*, pages 533–551, 2012.

[9] D. Dachman-Soled and Y. T. Kalai. Securing circuits and protocols against 1/poly(k) tampering rate. In *TCC*, pages 540–565, 2014.

[10] I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, pages 445–465, 2010.

[11] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, pages 572–590, 2007.

[12] 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.

[13] S. Faust, K. Pietrzak, and D. Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *ICALP*, pages 391–402, 2011.

[14] A. Gál and M. Szegedy. Fault tolerant circuits and probabilistically checkable proofs. In *Structure in Complexity Theory Conference*, pages 65–73, 1995.

[15] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[16] O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[17] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[18] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, pages 600–620, 2013.

[19] Y. Ishai, E. Kushilevitz, R. Ostrovsky, M. Prabhakaran, and A. Sahai. Efficient non-interactive secure computation. In *EUROCRYPT*, pages 406–425, 2011.

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

[21] Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[22] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.

[23] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

[24] Y. T. Kalai, B. Kanukurthi, and A. Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO*, pages 373–390, 2011.

[25] Y. T. Kalai, A. B. Lewko, and A. Rao. Formulas resilient to short-circuit errors. In *FOCS*, pages 490–499, 2012.

[26] M. G. Karpovsky and P. Nagvajara. Optimal codes for minimax criterion on error detection. *IEEE Transactions on Information Theory*, 35(6):1299–1305, 1989.

[27] J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[28] D. J. Kleitman, F. T. Leighton, and Y. Ma. On the design of reliable boolean circuits that contain partially unreliable gates. In *FOCS*, pages 332–346, 1994.

[29] F.-H. Liu and A. Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.

[30] M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.

[31] N. Pippenger. On networks of noisy gates. In *FOCS*, pages 30–38, 1985.

[32] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85, 1989.

[33] J. von Neumann. Probabilistic logics and synthesis of reliable organisms from unreliable components. *Automata Studies*, 34:43–98, 1956.

[34] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.