Electronic Colloquium on Computational Complexity, Report No. 3 (2013)

Shielding circuits with groups

Eric Miles* Emanuele Viola*

December 17, 2012

Abstract

We show how to efficiently compile any given circuit C into a leakage-resistant circuit $\widehat{C}$ such that any function on the wires of $\widehat{C}$ that leaks information during a computation $\widehat{C}(x)$ yields advantage in computing the product of $|\widehat{C}|^{\Omega(1)}$ elements of the alternating group A_u. In combination with new compression bounds for A_u products, also obtained here, $\widehat{C}$ withstands leakage from virtually any class of functions against which average-case lower bounds are known. This includes communication protocols, and AC^0 circuits augmented with few arbitrary symmetric gates. If $\mathrm{NC}^1 \neq \mathrm{TC}^0$ then the construction resists TC^0 leakage as well. We also conjecture that our construction resists NC^1 leakage. In addition, we extend the construction to the multi-query setting by relying on a simple secure hardware component.

We build on Barrington's theorem [JCSS '89] and on the previous leakage-resistant constructions by Ishai et al. [Crypto '03] and Faust et al. [Eurocrypt '10]. Our construction exploits properties of A_u beyond what is sufficient for Barrington's theorem.

*Supported by NSF grant CCF-0845003. Email: {enmiles,viola}@ccs.neu.edu

ISSN 1433-8092

1 Introduction

Motivated by successful attacks on cryptographic hardware, a recent, exciting line of work known as *leakage-resistant cryptography* considers models in which the adversary obtains more information from cryptographic algorithms than just their input/output behavior. A general goal in this area is to compile any circuit into a new "shielded" circuit such that any attack exploiting this extra information can in fact be carried out just using input/output access (and hence does not succeed under standard hardness assumptions). However, the seminal impossibility result on obfuscation [BGI+01] implies that one cannot shield circuits against an attack that obtains just one extra bit of information about the circuit, if this bit is computed as an arbitrary efficient *leakage function* of the wires of the circuit. More specifically it is sufficient that the leakage function is powerful enough to evaluate the shielded circuit on its own description. Still, this negative result does not necessarily hinder the scope of a theoretical study of leakage-resistant cryptography, because in practice this extra information is quite difficult to obtain and is typically limited to some simple-to-compute functions such as the Hamming weight of the bits carried on the wires. Thus, it makes sense to focus our attention on attacks where the extra information is obtained from the circuit by evaluating a computationally restricted leakage function.

One line of works considers leakage functions that operate on disjoint sets of the wires of the circuit, where the sets are chosen by the compiler (as opposed to the adversary). This setting has become known as the "only computation leaks" model, after Micali and Reyzin [MR04]. Ishai, Sahai, and Wagner in [ISW03] allow the leakage function to output projections of few of (the values carried on) the wires in each set. Their result is greatly generalized by a series of works [GR10, JV10, DF12, GR12] culminating in the construction by Goldwasser and Rothblum [GR12] which allows any arbitrary function of the sets, as long as the function has bounded output length.

In a different direction, Faust et al. [FRR+10] allow leakage functions that are computable by small, bounded-depth circuits with And, Or, and Not gates (AC^0). In contrast to the previous setting, here the leakage function accesses all wires simultaneously. In the case of an unbounded number of queries from the adversary – so-called "continual leakage" – the compiled circuits in [FRR+10], unlike [ISW03, GR12], utilize a secure hardware component. The latter is a simple gadget to which the leakage function does not have access. This use of secure hardware was removed recently by Rothblum [Rot12] at the expense of introducing a computational assumption.

In this work we present a new construction which significantly extends both lines of work, except that in the case of an unbounded number of queries we have to rely on a secure hardware component, similarly to [FRR+10].

1.1 Our results

We show how to efficiently compile any given circuit C into a leakage-resistant circuit $\widehat{C}$ such that any function on the wires of $\widehat{C}$ that leaks information during some computation $\widehat{C}(x)$ yields advantage in computing iterated group products over the alternating group A_u,

which recall is the group of even permutations of a set of size u. (For background on this group, see e.g. [KS04, §4.3].)

For simplicity, we first focus on the setting where the adversary makes a single query to the circuit, and we do not use any secure hardware. Defined next, our compiler is randomized and takes two inputs: a circuit $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and a value $k \in \{0,1\}^n$ for C's second input. It outputs a circuit $\widehat{C} : \{0,1\}^n \to \{0,1\}^n$ that is functionally equivalent to $C(\cdot, k)$. The only parts of $\widehat{C}$ that depend on k and the random coins are the values of its constant gates; the rest is determined by C. The adversary depends on C and thus knows everything about $\widehat{C}$ except the values of certain constant gates. The adversary then selects both an input x to the circuit and a leakage function to be evaluated on the wires of the circuit. The requirement that the adversary "learns nothing" from the output of the leakage function is formalized by providing an efficient *simulator* S. S sees only the input x and output $\widehat{C}(x)$ of the circuit, and produces a set of wire values that is indistinguishable from the real set of wire values by the leakage function. Throughout the paper we will use $|C|$ to denote the number of wires in a circuit C, which is the input length of the leakage functions.

Definition 1.1 (Leakage-secure compiler)**.** Let $\mathsf{Comp}(\cdot,\cdot)$ denote a randomized algorithm that takes as input a circuit $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and a string $k \in \{0,1\}^n$. For a class of leakage functions $\mathcal{L}$, Comp is an $(\mathcal{L}, \epsilon)$-*leakage-secure compiler* if the following properties hold.

1. (Structure.) For every C and k, $\mathsf{Comp}(C, k)$ outputs a circuit $\widehat{C} : \{0,1\}^n \to \{0,1\}^n$ which is completely determined by C except for the values of its constant gates.
2. (Correctness.) For every C and k and every $x \in \{0,1\}^n$, $\widehat{C}(x) = C(x,k)$ with probability 1 over the choice of $\widehat{C} \leftarrow \mathsf{Comp}(C,k)$.
3. (Security.) There exists a randomized polynomial-time algorithm S such that for every C and k, every $x \in \{0,1\}^n$, and every $\ell \in \mathcal{L}$ with domain $\{0,1\}^{|\widehat{C}|}$:

$$\Delta(\ell(\widehat{W}_x), \ell(S(C, x, \widehat{C}(x)))) < \epsilon$$

where $\widehat{W}_x \in \{0,1\}^{|\widehat{C}|}$ denotes the values carried by the wires of $\widehat{C}(x)$, and the statistical distance Δ is over the choice of $\widehat{C} \leftarrow \mathsf{Comp}(C,k)$ and the random coins of S.

The security of our construction is proved against leakage classes $\mathcal{L}$ for which iterated products over A_u are hard in the following sense. As discussed later, we exploit specific properties of A_u. However, when possible we present things over any group G.

Definition 1.2 (ϵ-fooled)**.** Let G be a group (whose operation is written multiplicatively). For $\alpha \in G$ and $t \in \mathbb{N}$, let D_α denote the uniform distribution over $\{(x_1, \ldots, x_t) \in G^t \mid \prod_i x_i = \alpha\}$, and let U_{G^t} denote the uniform distribution over G^t.

Then a set of functions $\mathcal{L}$ is ϵ-*fooled by* G^t if $\Delta(\ell(D_\alpha), \ell(U_{G^t})) < \epsilon$ for every $\alpha \in G$ and every $\ell \in \mathcal{L}$ with domain G^t.

We will use the notation D_α, U_{G^t} throughout the paper. Our security reductions are computable by simple, local (a.k.a. NC^0) functions.

Definition 1.3 (Local extension)**.** A function $f : G^t \to G^*$ is a *d-local function* if each output element depends on at most d input elements. If $d = O(1)$ as a function of t, we simply say *local*. For a set of functions $\mathcal{L}$, the *d-local extension of* $\mathcal{L}$ is the set of all functions $\ell(f(\cdot))$ where $\ell \in \mathcal{L}$ and f is a d-local function.

Note that 1-local functions are also known as projections.

Our compiler is given by the following main theorem.

Theorem 1.4. *Let G be a group. For every polynomial-time computable function $t = t(n)$, there is a compiler* Comp *for which the following holds.*

1. *For every $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $k \in \{0,1\}^n$, $\mathsf{Comp}(C, k)$ runs in time $\mathrm{poly}(|C|, t)$ and outputs a circuit $\widehat{C}$ of size $O(t^2 \cdot |C|)$ and depth $O(t \cdot depth(C))$.*
2. *For every set of functions $\mathcal{L}$ and every $\epsilon > 0$, if the 4-local extension of $\mathcal{L}$ is ϵ-fooled by G^t then* Comp *is an $(\mathcal{L}, \epsilon \cdot t \cdot |C|)$-leakage-secure compiler.*

Note that making t smaller reduces the size overhead of $\widehat{C}$, but that larger values of t are necessary to find rich classes $\mathcal{L}$ that are fooled by G^t.

To instantiate our construction we prove in §3 that $(A_u)^t$ fools a number of well-studied classes of functions (with parameters polynomially related to t). For all these results we can and will choose $u = 5$. One class is that of number-on-forehead multiparty protocols introduced by Chandra, Furst, and Lipton [CFL83]; here our result relies on the long-standing lower bound by Babai, Nisan, and Szegedy [BNS92], whose proof is increasingly streamlined in [CT93, Raz00, VW08]. Another is the class AC^0 of bounded-depth And/Or/Not circuits augmented with few gates computing arbitrary symmetric functions, such as parity and majority. This is the richest circuit class for which super-polynomial average-case lower bounds are known [Vio07]. In fact, one can allow few gates whose local extension has low number-on-forehead communication under any partition, such as polynomial threshold functions [Nis93, Vio11]. We also consider the class TC^0 of bounded-depth circuits of majority gates; for this class no lower bound is known, and our results rely on the standard complexity assumption $\mathrm{TC}^0 \neq \mathrm{NC}^1$.

Obviously $(A_u)^t$ does not fool NC^1 circuits of size $\mathrm{poly}(t)$ when $u = O(1)$, for such circuits can simply compute the product. However it is not clear how such a computation would go when $u = \omega(1)$. Indeed, Cook and McKenzie in [CM87] show that computing the product of n given permutations of a set of n elements is complete for L=Space($\log n$). We conjecture the stronger (assuming $\mathrm{L} \neq \mathrm{NC}^1$) result that $(A_n)^n$ fools NC^1 circuits of size $\mathrm{poly}(n)$.

The following theorem summarizes the results above. They can also be seen as giving *compression bounds* (cf. [HN10, DI06, Dru12] among others). In fact, we essentially recover for A_5-products the parameters of the AC^0 compression bound by Dubrov and Ishai [DI06] (building on their result). In the following, $O_d(\cdot)$ and $\Omega_d(\cdot)$ hide constants that depend only on d.

Theorem 1.5. *$(A_5)^t$ ϵ-fools $\mathcal{L}$ for:*

1. *$\mathcal{L}$ = number-on-forehead protocols with s parties communicating $\leq c$ bits, under a specific partition of the input; $\epsilon = 2^{c - \Omega(t/(s^2 4^s))}$.*

2. $\mathcal{L} = \mathrm{AC}^0$ *circuits with depth* $\leq d$, *size* $\leq t^{O_d(\log t)}$, *an additional* $O_d(\log^2 t)$ *arbitrary symmetric gates, and* $t^{0.1}$ *bits of output;* $\epsilon = t^{-\Omega_d(\log t)}$.
3. *If* $\mathrm{TC}^0 \neq \mathrm{NC}^1$ *then for every* k *and infinitely many* t, $\mathcal{L} = \mathrm{TC}^0$ *circuits with size* $\leq t^k$ *and* $k \log t$ *bits of output;* $\epsilon = t^{-k}$.
4. $\mathcal{L} = \mathrm{AC}^0$ *circuits with depth* $\leq d$, *size* $\leq 2^{O_d(t^{(1-\delta)/d})}$, *and* t^δ *bits of output, for any* $\delta < 1$; $\epsilon = 2^{-\Omega_d(t^{(1-\delta)/d})}$.

The straightforward combination of Theorems 1.4 and 1.5 gives an $(\mathcal{L}, \epsilon)$-secure compiler for the circuit classes listed in items 2-4 of the latter, choosing $t = |C|$ for the following corollary. The combination is less straightforward for protocols, which are not closed under composition with arbitrary local functions. We obtain item 1 of the following corollary by showing (in §4) that the local extension of a number-in-hand protocol is computable by a number-on-forehead protocol.

Corollary 1.6. *There is a single efficient compiler* Comp, *outputting a circuit* $\widehat{C}$ *of size* $|\widehat{C}| = O(|C|^3)$, *that is an* $(\mathcal{L}, \epsilon)$*-leakage secure compiler for each of the following.*

1. $\mathcal{L}$ = *number-in-hand protocols with* s *parties communicating* $\leq \delta \cdot |\widehat{C}|^{1/3}$ *bits, for a fixed* $\delta > 0$ *and a fixed partition of* $\widehat{C}$ *into* $s = O(1)$ *sets;* $\epsilon = 2^{-\Omega(|\widehat{C}|^{1/3})}$.
2. $\mathcal{L} = \mathrm{AC}^0$ *circuits with depth* $\leq d$, *size* $\leq |\widehat{C}|^{O_d(\log |\widehat{C}|)}$, *an additional* $O_d(\log^2 |\widehat{C}|)$ *arbitrary symmetric gates, and* $|\widehat{C}|^{0.01}$ *bits of output;* $\epsilon = |\widehat{C}|^{-\Omega_d(\log |\widehat{C}|)}$.
3. *If* $\mathrm{TC}^0 \neq \mathrm{NC}^1$ *then for every* k *and infinitely many* $|C|$, $\mathcal{L} = \mathrm{TC}^0$ *circuits with size* $\leq |\widehat{C}|^k$ *and* $k \log |\widehat{C}|$ *bits of output;* $\epsilon = |\widehat{C}|^{-k}$.
4. $\mathcal{L} = \mathrm{AC}^0$ *circuits with depth* $\leq d$, *size* $\leq 2^{O_d(|\widehat{C}|^{(1-\delta)/3d})}$, *and* $|\widehat{C}|^{\delta/3}$ *bits of output, for any* $\delta < 1$; $\epsilon = 2^{-\Omega_d(|\widehat{C}|^{(1-\delta)/3d})}$.

In particular, our construction resists leakage from functions such as parity, majority, inner product, and polynomial thresholds. Besides being well-studied, these functions break most previous constructions. For example, inner product breaks [DF12, GR12], and parity breaks [ISW03, FRR+10, Rot12]. Also, small TC^0 circuits can be shown to break at least one instantiation of the construction [JV10] using the fact that such circuits may compute division, cf. [All01]. In fact, we are only aware of one construction that is not easily broken in TC^0. This is the construction [GR10] which relies on the Decisional-Diffie-Hellman assumption. It is broken by any leakage function that can decrypt a certain public-key cryptosystem based on it, but decryption here involves modular exponentiation (to a poly-length exponent); whether this is doable in small depth is an open problem.

Assuming the conjecture we made above that $(A_n)^n$ fools NC^1 circuits of size $\mathrm{poly}(n)$, the compiled circuit $\widehat{C}$ also withstands NC^1 leakage.

Finally, note that the last item shows that we recover the security of [FRR+10] against AC^0 leakage functions.

Multiple queries. We also consider the setting in which the adversary makes multiple, adaptive queries to the circuit $\widehat{C}$. As in the previous setting, each query consists of both an input to the circuit and a leakage function. The adversary is assumed to be computationally

unbounded, except for the restriction on the leakage functions. We defer until §5 the formal definition of security in this setting, but it is a natural extension of Definition 1.1.

If the number of queries q is fixed in advance and known to the compiler, then our construction in Theorem 1.4 can be extended with little difficulty to this setting. The resulting construction increases the size of $\widehat{C}$ by a factor of $O(q)$ and likewise the security degrades by a factor of q (details omitted).

When the number of queries q is not a priori bounded, we adopt the approach of [FRR+10] and augment $\widehat{C}$ with a so-called *secure hardware component*. In our construction, this component is a randomized, inputless gate that on each execution outputs a sample from D_{id}, where id denotes the identity element of A_5. We refer to such gates as D_{id}*-gates*, and any circuit that contains one as a D_{id}*-circuit*. The complexity of this component is comparable to the one in [FRR+10] which outputs a uniform bit vector with parity 0. (Secure hardware components are also used in [GR10, JV10], but there the components are not inputless and furthermore the distribution sampled is significantly more complex.)

To prove security in this setting, a slightly stronger property is required of $(A_5)^t$ than what is given by Theorem 1.5. Specifically we require that, for every $\ell \in \mathcal{L}$ and every $\ell' \in \mathcal{L}$ that is chosen adaptively based on the output of ℓ, the distribution $(\ell(x), \ell'(x))$ when $x \leftarrow D_\alpha$ has statistical distance $< \epsilon$ from the corresponding distribution when $x \leftarrow U_{G^t}$. We show in §5.1 that each of the classes $\mathcal{L}$ listed in Theorem 1.5 has this property; the only difference is that for AC^0 circuits with symmetric gates, we restrict the output length to $O(\log^2 t)$.

We defer the details for now and simply state our result for multiple queries.

Corollary 1.7. *There is a single efficient compiler* Comp*, outputting a* D_{id}*-circuit* $\widehat{C}$ *of size* $|\widehat{C}| = O(|C|^3)$*, that is a* q*-query* $(\mathcal{L}, \epsilon)$*-leakage secure compiler for any* q *and each of the following.*

1. $\mathcal{L}$ *= number-in-hand protocols with* s *parties communicating* $\leq \delta \cdot |\widehat{C}|^{1/3}$ *bits, for a fixed* $\delta > 0$ *and a fixed partition of* $\widehat{C}$ *into* $s = O(1)$ *sets;* $\epsilon = q \cdot 2^{-\Omega(|\widehat{C}|^{1/3})}$*.*
2. $\mathcal{L} = \mathrm{AC}^0$ *circuits with depth* $\leq d$*, size* $\leq |\widehat{C}|^{O_d(\log |\widehat{C}|)}$*, an additional* $O_d(\log^2 |\widehat{C}|)$ *arbitrary symmetric gates, and* $O_d(\log^2 |\widehat{C}|)$ *bits of output;* $\epsilon = q \cdot |\widehat{C}|^{-\Omega_d(\log |\widehat{C}|)}$*.*
3. *If* $\mathrm{TC}^0 \neq \mathrm{NC}^1$ *then for every* k *and infinitely many* $|C|$*,* $\mathcal{L} = \mathrm{TC}^0$ *circuits with size* $\leq |\widehat{C}|^k$ *and* $k \log |\widehat{C}|$ *bits of output;* $\epsilon = q \cdot |\widehat{C}|^{-k}$*.*
4. $\mathcal{L} = \mathrm{AC}^0$ *circuits with depth* $\leq d$*, size* $\leq 2^{O_d(|\widehat{C}|^{(1-\delta)/3d})}$*, and* $|\widehat{C}|^{\delta/3}$ *bits of output, for any* $\delta < 1$*;* $\epsilon = q \cdot 2^{-\Omega_d(|\widehat{C}|^{(1-\delta)/3d})}$*.*

Organization. In §2 we describe our construction and prove the key lemma that enables the proof of Theorem 1.4. In §3 we show that various computational models are fooled by $(A_5)^t$, proving Theorem 1.5. In §4 we prove Theorem 1.4 (and Corollary 1.6), and in §5 we extend our construction to multiple queries.

2 The construction

In this section we describe our main construction. Our compiler uses the general framework of the works [ISW03, FRR+10]. In this framework, to every wire of C there corresponds in the compiled circuit $\widehat{C}$ a "bundle" of wires which encode the same information. (In [ISW03, FRR+10] a bit b is encoded by a bundle x whose parity is b.) One then uses appropriate gadgets to simulate the computation of C on the bundles. Note the distinction between *gates* and *gadgets* in $\widehat{C}$: gadgets operate on bundles of wires, and are composed of gates that operate on individual wires.

The main differences between our construction and the ones in [ISW03, FRR+10] are in the encoding and in the gadgets. A side-benefit of our gadgets is that they allow for a more modular construction yielding an arguably more intuitive proof of security. Next we describe our encoding, our gadgets, and the proof of security. But first we make some remarks on the group used throughout.

The choice of the group. This work exploits 3 properties of the alternating group A_5.

(i) It fools various classes in the sense of Definition 1.2 (see Theorem 1.5). We show that this is implied by the fact that every element of A_5 is a commutator; such groups are known as *perfect* [HP89].

(ii) It supports Barrington's encoding of NC^1 computation [Bar89], which we use in the construction of the NAND gadget below. (This is implied by the group being non-solvable, which in turn is implied by it being perfect.)

(iii) It has specific elements that support a more efficient encoding of certain functions such as parity, improving on (ii). This is used in Theorem 1.5 to obtain improved parameters and in particular to match the parameters of the previous compression bound in [DI06].

We point out that (i) is not implied by (ii). Indeed, for (ii) the group S_5 is typically chosen. However $(S_5)^t$ does not even $(1/2)$-fool the 1-local extension of parity, which can compute the sign of the product permutation. This is because the sign of D_α always equals the sign of α, whereas the sign of $U_{(S_5)^t}$ is equidistributed over $\{-1, 1\}$.

The group encoding. We encode a bit $b \in \{0, 1\}$ by a tuple of elements over a group G as follows. Let id denote G's identity element, and fix an element $\mathsf{id} \neq \alpha \in G$. Then $(x_1, \ldots, x_t) \in G^t$ encodes b when

$$\prod_i x_i = \begin{cases} \mathsf{id} & \text{if } b = 0 \\ \alpha & \text{if } b = 1. \end{cases}$$

As in [Bar89], we can use any α for which there exists an element $\beta \in G$ such that α, β, and $\alpha\beta\alpha^{-1}\beta^{-1}$ are in the same conjugacy class. Equivalently, there must exist three elements $\beta, \gamma, \rho \in G$ such that the following two equations hold.

$$\gamma\alpha\gamma^{-1} = \beta \qquad\qquad \rho\alpha\beta\alpha^{-1}\beta^{-1}\rho^{-1} = \alpha. \tag{1}$$

For $G = A_5$ and using cycle notation, these values can be set as follows: $\alpha = (12345)$, $\beta = \rho = (14235)$, $\gamma = (12354)$.

For convenience we present the construction over G as opposed to $\{0,1\}$ and using gates for group multiplication and inversion. It is straightforward to obtain a construction over $\{0,1\}$ and any standard basis by implementing group operations via bit operations.

The NAND gadget. We assume without loss of generality that C, the circuit input to the compiler, contains only fan-in-2 gates that compute the Nand function. We now describe the NAND gadget that simulates each Nand gate in C. Given as input two bundles $x, y \in G^t$ with products in $\{\mathsf{id}, \alpha\}$, the NAND gadget outputs a bundle $z \in G^t$ that encodes the Nand of x and y, i.e., that satisfies:

$$\prod_i z = \begin{cases} \mathsf{id} & \text{if } \prod_i x_i = \prod_i y_i = \alpha \\ \alpha & \text{otherwise.} \end{cases} \tag{2}$$

The output bundle $z \in G^t$ is computed by the following steps.

1. Set $y \leftarrow (\gamma \cdot y_1, y_2, \ldots, y_{t-1}, y_t \cdot \gamma^{-1})$. (This gives $\prod_i y_i \in \{\mathsf{id}, \beta\}$.)
2. Compute $x^{-1} := (x_t^{-1}, \ldots, x_1^{-1})$ and similarly y^{-1}.
3. Compute $\overline{z} \in G^{4t}$ by concatenating (x, y, x^{-1}, y^{-1}). (This gives $\prod_i \overline{z}_i \in \{\mathsf{id}, \alpha\beta\alpha^{-1}\beta^{-1}\}$.)
4. Set $\overline{z} \leftarrow (\rho \cdot \overline{z}_1, \overline{z}_2, \ldots, \overline{z}_{4t-1}, \overline{z}_{4t} \cdot \rho^{-1})$. (This gives $\prod_i \overline{z}_i \in \{\mathsf{id}, \alpha\}$.)
5. Set $\overline{z} \leftarrow (\overline{z}_{4t}^{-1}, \ldots, \overline{z}_1^{-1} \cdot \alpha)$. (This maintains $\prod_i \overline{z}_i \in \{\mathsf{id}, \alpha\}$ but if the product in step 4 was α it is now id, and vice versa.)
6. Compute and output $z \in G^t$ by multiplying consecutive groups of 4 elements in $\overline{z}$:

$$z := \left(\prod_{i=1}^{4} \overline{z}_i, \prod_{i=5}^{8} \overline{z}_i, \ldots, \prod_{i=4t-3}^{4t} \overline{z}_i \right).$$

From the equations (1) it can be verified that (2) is satisfied.

Warm-up for the RANDOM gadget. The second and last gadget that we need is called RANDOM and is essentially applied to every bundle in $\widehat{C}$ that corresponds to a wire in C. This gadget has to satisfy two properties. First we need that on input $x \in G^t$, the RANDOM gadget outputs a bundle $z \in G^t$ that is distributed uniformly over $\{z \in G^t \mid \prod_i z_i = \prod_i x_i\}$. This is necessary both for the correctness and security of the construction. The second property, necessary only for the security, is that given an input-output pair (x, z) for this gadget, we should be able to compute *locally* a distribution on the gadget's wires that is indistinguishable from the real distribution. (This allows us to replace the real distribution on the wires of $\widehat{C}$ with the one in which each RANDOM gadget is reconstructed. Then we

can replace each bundle of wires in $\widehat{C}$ with a uniform bundle, which the simulator can do by itself, and blame any inconsistency on the reconstructor.) This property is called *local reconstructibility* and is a variant of the one in [FRR+10].

Before describing our gadget, we note that there is a simple gadget that satisfies the first property but not the second. Namely, choose $r_1, \ldots, r_{t-1} \in G$ uniformly at random, and output

$$z = (x_1 \cdot r_1,\ r_1^{-1} \cdot x_2 \cdot r_2,\ \ldots,\ r_{t-1}^{-1} \cdot x_t). \tag{3}$$

Indeed, this basic re-randomization technique has been used to great effect in a number of works, e.g. [AIK06, GGH+08, AAW10]. However, this simple gadget does not satisfy local reconstructibility. One reason is that given x, z, one can come up with values for the r_i that are consistent with each gate in the circuit if and only if $\prod x_i = \prod z_i$. However, the latter is an NC^1-hard question, whereas consistency of the r may be checked by, say, a DNF.

By contrast, one feature of our gadget is that given x, z one can produce consistent values for the wires even if $\prod x_i \neq \prod z_i$. The catch is that in the latter case the values of certain constant gates are not chosen as in the correct implementation, but the leakage functions will not be able to distinguish this.

The RANDOM **gadget.** We now describe our gadget. The computation corresponds to replacing each pair (r_i, r_i^{-1}) in (3) with a pair $(R, L) \in G^t \times G^t$ such that $(\prod_j R_j) \cdot (\prod_j L_j) = \mathsf{id}$, and then computing the multiplications in a specific order.

First, choose $R^{(1)}, \ldots, R^{(t-1)} \in G^t$ uniformly at random. Next, choose $L^{(2)}, \ldots, L^{(t)} \in G^t$ at random conditioned on

$$\prod_j L_j^{(i)} = \left(\prod_j R_j^{(i-1)} \right)^{-1} \tag{4}$$

for $1 < i \leq t$. In the single-query setting, we think of Comp choosing these values and hardwiring them into $\widehat{C}$; in the multi-query setting, each pair $(R^{(i-1)}, L^{(i)})$ will be output by a secure hardware component. We will drop the superscripts on R and L when they are clear from context. Condition (4) implies the following equation.

$$\left(x_1 \cdot \prod_j R_j^{(1)} \right) \cdot \left(\prod_j L_j^{(2)} \cdot x_2 \cdot \prod_j R_j^{(2)} \right) \cdots \left(\prod_j L_j^{(t)} \cdot x_t \right) = \prod_i x_i. \tag{5}$$

So, we compute z by letting $z_i \in G$ be the result of the ith parenthesized expression in (5). Clearly this z has the correct distribution. We perform each iterated multiplication by a depth-$O(t)$ tree of fan-in-2 multiplication gates in a specific way, described now.

For z_1, the product is computed in the straightforward way from left to right by a depth-t tree that computes each prefix product

$$\lambda_m := x_1 \cdot \prod_{j=1}^{m} R_j = \lambda_{m-1} \cdot R_m$$

for $m = 1, \ldots, t$ in order, and outputs $z_1 := \lambda_t$. The product for z_t is computed in the straightforward way from left to right as well.

Now let $1 < i < t$. The product for z_i is computed by a depth-$2t$ tree that multiplies "from the inside out". That is, it computes in order a sequence $\lambda_1, \ldots, \lambda_{2t-1}$ defined by $\lambda_1 := L_t \cdot x_i$ and recursively for $j = 1, \ldots, t-1$ by

$$\begin{aligned} \lambda_{2j} &:= \lambda_{2j-1} \cdot R_j \\ \lambda_{2j+1} &:= L_{t-j} \cdot \lambda_{2j} \end{aligned}$$

and then outputs $z_i := \lambda_{2t-1} \cdot R_t$.

By way of illustration, when $t = 3$ the sequence is computed as follows.

$$\begin{array}{rlllllllll} & \lambda_1 = & & & L_3 & x_i & & & & \\ & \lambda_2 = & & & L_3 & x_i & R_1 & & & \\ & \lambda_3 = & & L_2 & L_3 & x_i & R_1 & & & \\ & \lambda_4 = & & L_2 & L_3 & x_i & R_1 & R_2 & & \\ & \lambda_5 = & L_1 & L_2 & L_3 & x_i & R_1 & R_2 & & \\ z_i = & \lambda_6 = & L_1 & L_2 & L_3 & x_i & R_1 & R_2 & R_3. \end{array}$$

The following key lemma in this work shows that the RANDOM gadget is locally reconstructible. We say that $x, z \in G^t$ are *plausible* if it is possible for $\text{RANDOM}(x)$ to output z, i.e. if $\prod_i x_i = \prod_i z_i$.

Lemma 2.1. *There is a* $\text{poly}(t)$*-time computable distribution on 1-local functions* $\mathsf{R}_{\text{RANDOM}} : G^t \times G^t \to G^{|\text{RANDOM}|}$ *for which the following holds. Let* $W_{x \to z}$ *denote the distribution on the wires of* $z = \text{RANDOM}(x)$*. For any* ℓ *with domain* $G^{|\text{RANDOM}|}$ *and any plausible* $x, z \in G^t$*, if*

$$\Delta(\ell(\mathsf{R}_{\text{RANDOM}}(x, z)), \ell(W_{x \to z})) \geq \epsilon \cdot (t-1)$$

then some 1-local extension of ℓ *is not* ϵ*-fooled by* G*, i.e., there is a* $g \in G$ *and a* 1*-local function* $f : G^t \to G^{|\text{RANDOM}|}$ *such that*

$$\Delta(\ell(f(D_g)), \ell(f(U_{G^t}))) \geq \epsilon.$$

Proof. We first describe an alternate procedure for generating $W_{x \to z}$. Fix any plausible $x, z \in G^t$. For the tree computing z_1, choose each λ_j uniformly at random for $j = 1, \ldots, t-1$, and compute each $R_j^{(1)} := \lambda_{j-1}^{-1} \cdot \lambda_j$ for $j = 1, \ldots, t$, defining $\lambda_t := z_1$ and $\lambda_0 := x_1$. Choose the wires for the tree computing z_t analogously. Then for $i = 2, \ldots, t-1$ in order, choose the wires for the tree computing z_i as follows.

1. Choose $L^{(i)} \in G^t$ uniformly over vectors with product $= (\prod_j R_j^{(i-1)})^{-1}$.
2. For $j = 1, \ldots, t-1$ choose $\lambda_{2j} \in G$ uniformly at random.

3. For $j = 0, \ldots t-1$ compute $\lambda_{2j+1} := L^{(i)}_{t-j} \cdot \lambda_{2j}$, where $\lambda_0 := x_i$.

4. For $j = 1, \ldots, t$ compute $R^{(i)}_j := \lambda^{-1}_{2j-1} \cdot \lambda_{2j}$, where $\lambda_{2t} := z_i$.

To show that this distribution is identical to $W_{x\to z}$, it is enough to observe that the vectors $R^{(i)}, L^{(i)}$ are distributed correctly, i.e. uniformly conditioned on (4) and on the ith parenthesized expression in (5) equalling z_i for all i. This is because the above procedure computes consistent wire values, and the $R^{(i)}, L^{(i)}$ (along with x, z) determine the values of all other wires. $R^{(1)}$ and $L^{(t)}$ are clearly distributed correctly. Then for each $1 < i < t$, $L^{(i)}$ is distributed correctly assuming that $R^{(i-1)}$ is, $R^{(i)}_j$ is uniform for $1 \le j < t$, and $R^{(i)}_t$ takes the unique consistent value.

This computation is sequential, due to the selection of $L^{(i)}$ based on $R^{(i-1)}$ in step 1. This selection is there to ensure condition (4). However by dropping this condition, we can break the dependencies between multiplication trees and give a local reconstructor. Namely, we define $\mathsf{R}_{\text{RANDOM}}(x, z)$ to be the above computation except that $L^{(i)}$ is chosen uniformly at random in step 1. Note that $\mathsf{R}_{\text{RANDOM}}$ is a distribution on 1-local functions.

To prove the lemma, we define a set of hybrid distributions H_m on the wires of RANDOM for $m = 1, \ldots, t-1$. Fix any plausible $x, z \in G^t$. In H_m, the wires in the tree computing z_i for $i \le m$ are chosen as in $W_{x\to z}$, and for $i > m$ the wires are chosen as in $\mathsf{R}_{\text{RANDOM}}(x, z)$. Then, we have $H_1 \equiv \mathsf{R}_{\text{RANDOM}}(x, z)$ and $H_{t-1} \equiv W_{x\to z}$ (note that the first and last trees are distributed identically in $W_{x\to z}$ and $\mathsf{R}_{\text{RANDOM}}(x, z)$). Thus if there is a function ℓ such that

$$\Delta(\ell(\mathsf{R}_{\text{RANDOM}}(x, z)), \ell(W_{x\to z})) \ge \epsilon \cdot (t-1)$$

then there is an $m \in [2, t-1]$ such that

$$\Delta(\ell(H_{m-1}), \ell(H_m)) \ge \epsilon. \tag{6}$$

Now let $g \in G$ be the fixed value (depending on x, z) such that $\prod_j R^{(m-1)}_j = g$ with probability 1 in both H_m and H_{m-1}. Thus in H_{m-1} (resp. H_m), $L^{(m)}$ is distributed according to U_{G^t} (resp. $D_{g^{-1}}$). (Note that this g is arbitrary, i.e. not only α or id, and hence we are using the full generality of Definition 1.2.) Because H_{m-1} and H_m differ only in the tree computing z_m, and because the distribution on these wires is independent of all other wires when x and z are fixed, by an averaging argument we can fix all wires outside this tree while preserving (6). Then given a vector $v \in G^t$ distributed according to either U_{G^t} or $D_{g^{-1}}$, a 1-local function (of v) can generate H_{m-1} or H_m by setting $L^{(m)} := v$ and performing steps 2-4 in the above computation. Fixing the randomness of this function by an averaging argument, we obtain the function $f : G^t \to G^{|\text{RANDOM}|}$ in the statement of the lemma. □

Using the above gadgets and Lemma 2.1 one may complete the proof of Theorem 1.4 essentially following [FRR+10, Lemma 13]. For completeness we include a proof in §4.

3 On compressing group products

In this section we show that $(A_5)^t$ fools in the sense of Definition 1.2 a number of computational models, proving Theorem 1.5. We start this section by recalling a number of facts related to groups and computation. Then in each of the subsections we analyze each computational model in turn.

First, it will be convenient later to prove that $\ell(D_\alpha)$ and $\ell(D_{\mathsf{id}})$ are close for every $\alpha \in G$, $\ell \in \mathcal{L}$, and we observe that this is sufficient.

Lemma 3.1. *If $\Delta(\ell(D_\alpha), \ell(D_{\mathsf{id}})) < \epsilon$ for every $\alpha \in G$ and every ℓ in the 1-local extension of $\mathcal{L}$, then $\mathcal{L}$ is ϵ-fooled by G^t.*

Proof. If $\mathcal{L}$ is not ϵ-fooled by G^t then there exists $\alpha \in G$, $\ell \in \mathcal{L}$ such that $\Delta(\ell(D_\alpha), \ell(U_{G^t})) \geq \epsilon$. Then by an averaging argument there exists $\beta \in G$ such that $\Delta(\ell(D_\alpha), \ell(D_\beta)) \geq \epsilon$. Finally defining $\ell'(x_1, \ldots, x_t) := \ell(x_1, \ldots, x_t \cdot \beta)$ in the 1-local extension of $\mathcal{L}$, we have $\Delta(\ell'(D_{\alpha\beta^{-1}}), \ell'(D_{\mathsf{id}})) \geq \epsilon$. □

We will also make use of the random self-reducibility of the distributions D_α.

Lemma 3.2. *There exists a distribution on 1-local functions $R : G^t \to G^t$ such that for any $\alpha \in G$ and any x in the support of D_α, $R(x) \equiv D_\alpha$.*

Proof. R chooses $r_1, \ldots, r_{t-1} \in G$ uniformly at random, and outputs

$$(x_1 \cdot r_1,\ r_1^{-1} \cdot x_2 \cdot r_2,\ \ldots,\ r_{t-1}^{-1} \cdot x_t). \quad \square$$

Recall the following standard terminology: α is an *involution* if $\alpha = \alpha^{-1}$, and α is the *commutator* of β and γ if $\alpha = \beta\gamma\beta^{-1}\gamma^{-1}$. We say that $M : \{0,1\}^n \to (A_5)^*$ *α-computes* a function $f : \{0,1\}^n \to \{0,1\}$ if for every x, $\prod_i M(x)_i = \alpha^{f(x)}$ (where $\alpha^0 = \mathsf{id}$ and $\alpha^1 = \alpha$).

The next theorem follows from [Bar89, Theorem 5] because every element of A_5 is a commutator.

Theorem 3.3 ([Bar89])**.** *For every $\alpha \in A_5$ and every fan-in-2 circuit $C : \{0,1\}^n \to \{0,1\}$ of depth d, there is a 1-local function $M : \{0,1\}^n \to (A_5)^{O(4^d)}$ that α-computes C.*

Moreover, let $f_1, \ldots, f_m : \{0,1\}^n \to \{0,1\}$ be functions such that for each $i \leq m$ there is a 1-local function $M_i' : \{0,1\}^n \to (A_5)^n$ that α-computes f_i. Then, for every fan-in-2 circuit $C : \{0,1\}^m \to \{0,1\}$ of depth d there is a 1-local function $M : \{0,1\}^{mn} \to (A_5)^{O(n \cdot 4^d)}$ that α-computes $C(f_1(\cdot), \ldots, f_m(\cdot))$.

The next lemma allows certain functions to be more efficiently α-computed. This can be compared with the works by Cai and Lipton [yCL94] and Cleve [Cle91] which give increasingly efficient versions of Barrington's construction (here efficiency is measured in the length of M's output). Our construction is simpler than the ones given in these works, but also less general.

Lemma 3.4. *For every involution $\alpha \in A_5$, the following holds.*

1. *There is a 1-local function* $M : \{0,1\}^n \to (A_5)^n$ *that* α*-computes* $\bigoplus_{i=1}^n x_i$.
2. *If* M *and* M' α*-compute some Boolean functions* f *and* f'*, then their concatenation* (M, M') α*-computes the function* $f \oplus f'$.

Proof. For the first item: given input $x \in \{0,1\}^n$, output $y \in (A_5)^n$ such that $y_i = \alpha^{x_i}$. The correctness of this construction, as well as the second item, follows from the isomorphism between the group $\{0,1\}$ (under $\oplus$) and the subgroup $\{\mathsf{id}, \alpha\} \subset A_5$. □

We need the following fact about A_5. Note that because every k-cycle $(a_1\ a_2\ \cdots\ a_k)$ can be written as a product of $k-1$ transpositions $(a_1\ a_2)(a_1\ a_3)\cdots(a_1\ a_k)$, every element of A_5 is either a 3-cycle, a 5-cycle, the product of two disjoint transpositions, or the identity.

Fact 3.5. *Every non-involution in* A_5 *is the commutator of two involutions.*

Proof. Let a, b, c, d, e denote arbitrary, distinct elements of $\{1, \ldots, 5\}$. Every non-involution in A_5 is either a 3-cycle $(a\ b\ c)$ which is the commutator of involutions $(a\ b)(d\ e)$ and $(b\ c)(d\ e)$, or a 5-cycle $(a\ b\ c\ d\ e)$ which is the commutator of involutions $(b\ e)(c\ d)$ and $(a\ d)(b\ c)$. □

3.1 Multi-party protocols

In this section we consider functions computable by a multi-party communication protocol in the "number on forehead" model [CFL83], defined as follows. A protocol P with n-bit inputs consists of $k = k(n)$ parties, each with unlimited computational power. The input $x \in \{0,1\}^{kn}$ is partitioned into k blocks, and party i sees all input bits except those in the ith block. The parties communicate in the broadcast model, so every bit sent is seen by all parties. The $(m = m(n))$-bit output of P is defined to be the final m bits that are broadcast, and the cost of P is the total number of bits broadcast by all parties. When P's input comes from a group G, we assume some canonical representation of G's elements as $(\log |G|)$-bit strings.

We prove the following compression bound for such protocols.

Theorem 3.6. *There is a partition of the inputs in* $(A_5)^t$ *into* k *pieces such that any* k*-party protocol communicating* c *bits and outputting* $\leq c$ *bits is* ϵ*-fooled by* $(A_5)^t$ *for* $\epsilon = 2^{c-\Omega(t/(k^2 4^k))}$.

We prove this theorem by combining an efficient translation from bits to group products with the following lower bound. Define the generalized inner-product function $GIP_{n,k} : \{0,1\}^{nk} \to \{0,1\}$ as

$$GIP_{n,k}(x) := \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{k} x_{i,j}.$$

Then we have the following lemma, originally due to Babai, Nisan, and Szegedy [BNS92] and with increasingly streamlined proofs in [CT93, Raz00, VW08].

Lemma 3.7 ([BNS92])**.** *There is a partition of the inputs to* $GIP_{n,k}$ *into* k *blocks such that for every protocol* $P : \{0,1\}^{nk} \to \{0,1\}$ *with* k *parties that communicates at most* c *bits,* $\Pr_x[P(x) = GIP_{n,k}(x)] \leq \frac{1}{2} + 2^{c-\Omega(n/4^k)}$.

We give the following translation to bits from group products.

Lemma 3.8. *For every $\alpha \in A_5$, there is a 1-local function $M : \{0,1\}^{nk} \to (A_5)^{O(nk^2)}$ that α-computes $GIP_{n,k}$.*

Proof. Assume α is an involution. Let $M' : \{0,1\}^k \to (A_5)^{O(k^2)}$ be the function guaranteed by Theorem 3.3 that α-computes the k-wise AND of its input. Then letting $x^{(i)} := (x_{i,1}, \ldots, x_{i,k})$ for each $i \leq n$, the function

$$M(x) := (M'(x^{(1)}), \ldots, M'(x^{(n)}))$$

α-computes $GIP_{n,k}$ by the second item of Lemma 3.4.

If α is not an involution, let $\beta, \gamma \in A_5$ be the involutions guaranteed by Fact 3.5 such that $\alpha = \beta\gamma\beta\gamma$ (note that $\beta = \beta^{-1}$ and $\gamma = \gamma^{-1}$). Then let M' instead β-compute the AND of its input, and compute M as

$$M(x) := (M'(x^{(1)}), \ldots, M'(x^{(n)}), \gamma, M'(x^{(1)}), \ldots, M'(x^{(n)}), \gamma). \quad \square$$

We now give the proof of Theorem 3.6.

Proof of Theorem 3.6. For an appropriate $n = \Omega(t/k^2)$, let $M : \{0,1\}^{n \cdot k} \to (A_5)^t$ be the 1-local function guaranteed by Lemma 3.8 that α-computes $GIP_{n,k}$. Consider the partition on the t elements of the input from $(A_5)^t$ that is induced by M from the partition on $GIP_{n,k}$ guaranteed by Lemma 3.7.

Assume for contradiction that some protocol on this partition is not ϵ-fooled. Without loss of generality the protocol outputs 1 bit. (The last player can simulate whatever set maximizes the statistical distance of the multi-bit protocol output distributions.) By Lemma 3.1, there is an $\alpha \in A_5$ and a protocol $P : (A_5)^t \to \{0,1\}$ with k parties communicating $\leq c$ bits such that $\Delta(P(D_\alpha), P(D_{\mathsf{id}})) \geq 2^{c - \beta t/(k^2 4^k)}$, for a suitable constant β.

By combining the P with M we now give a distribution on protocols $P' : \{0,1\}^{n \cdot k} \to \{0,1\}$ for $GIP_{n,k}$ with the same number of parties, the same communication, and the same advantage up to the constant β. This contradicts Lemma 3.7.

On input x, each party in P' first computes the portion of $y := M(x) \in (A_5)^t$ that depends on the input bits it can see; this is done with no communication as M is 1-local. Next each party computes the portion of $z := (y_1 \cdot r_1, r_1^{-1} \cdot y_2 \cdot r_2, \ldots, r_{t-1}^{-1} \cdot y_t)$ that depends on the input bits it can see, again with no communication. (r is a public random string.) Finally the parties compute and output $P(z)$ using the protocol P.

Note for every x such that $GIP_{n,k}(x) = 1$ (resp. $GIP_{n,k}(x) = 0$), z is distributed according to D_α (resp. D_{id}) over the choice of r. The proof is now completed using the fact that $\Pr_x[GIP_{n,k}(x) = 0] > \Pr_x[GIP_{n,k}(x) = 1] \geq 1/2 - 2^{-\Omega(n/2^k)}$ [VW08, Claim 2.11]. $\square$

3.2 TC^0

In this section we observe that, because computing products over A_5 is complete for NC^1, if $\mathrm{TC}^0 \neq \mathrm{NC}^1$ then the set of TC^0 circuits with $O(\log t)$ bits of output is $t^{-\omega(1)}$-fooled by $(A_5)^t$.

Recall that NC^1 is the class of poly-size fan-in-2 And/Or/Not circuits with depth $O(\log n)$, and TC^0 is the class of poly-size unbounded-fan-in constant-depth circuits where each gate computes, for some c, the c-threshold function which is 1 iff $\geq c$ inputs are 1.

The high-level idea behind the next theorem is the following. Assume there is an $\alpha \in A_5$ and a TC^0 circuit C that can distinguish between D_α and D_{id} with advantage $\geq t^{-k}$ for some k. Then for any NC^1 circuit B we construct a TC^0 circuit that, on input x, chooses $m = t^{O(k)}$ samples from the distribution $D_{\alpha^{B(x)}}$ and outputs 1 iff the number of samples on which C outputs 1 is sufficiently close to $m \cdot \Pr[C(D_\alpha) = 1]$. This last check can be computed with threshold gates, and we can sample from $D_{\alpha^{B(x)}}$ by using Theorem 3.3 to obtain a single element in its support and then relying on the random self-reducibility of this distribution.

Theorem 3.9. *If $\mathrm{TC}^0 \neq \mathrm{NC}^1$ then $\forall k$ and infinitely many t, the class $\mathcal{L}$ of TC^0 circuits with size $\leq t^k$ and output length $k \log t$ is t^{-k}-fooled by $(A_5)^t$.*

Proof. Assume that $\exists k$ such that for sufficiently large t, $\mathcal{L}$ is not t^{-k}-fooled by $(A_5)^t$. Then by Lemma 3.1 there exists an $\alpha \in A_5$ and a TC^0 circuit $C : (A_5)^t \to \{0,1\}^{k \log t}$ with size $\leq t^k$ such that $\Delta(C(D_\alpha), C(D_{\mathsf{id}})) \geq t^{-k}$. Let $S \subseteq \{0,1\}^{k \log t}$ be the set that maximizes $\Pr[C(D_\alpha) \in S] - \Pr[C(D_{\mathsf{id}}) \in S]$, and note that checking $x \in S$ can be done by a TC^0 circuit of size $t^{O(k)}$. Thus, there is a TC^0 circuit $C' : (A_5)^t \to \{0,1\}$ of size $t^{O(k)}$ such that

$$\Pr[C'(D_\alpha) = 1] - \Pr[C'(D_{\mathsf{id}}) = 1] \geq t^{-k}. \tag{7}$$

Define $\epsilon_\alpha := \Pr[C'(D_\alpha) = 1]$ and $\epsilon_{\mathsf{id}} := \Pr[C'(D_{\mathsf{id}}) = 1]$, and note that $\epsilon_\alpha \geq t^{-k}$.

Let $B : \{0,1\}^n \to \{0,1\}$ be any NC^1 circuit, and for an appropriate $t = n^{O(1)}$ let $M : \{0,1\}^n \to (A_5)^t$ be the 1-local function (guaranteed by Theorem 3.3) that α-computes B. Let $C'' : \{0,1\}^n \to \{0,1\}$ be the randomized TC^0 circuit that performs the following steps on input $x \in \{0,1\}^n$.

1. Compute $y = M(x) \in (A_5)^t$.
2. For $m := t^{3k}(n+2)/\epsilon_\alpha = n^{O(k)}$, sample $z_1, \ldots, z_m \in (A_5)^t$ independently from $D_{\alpha^{B(x)}}$ by computing $R(y)$ where R is the 1-local function from Lemma 3.2.
3. Use two layers of threshold gates to output 1 iff

$$(1 - 1/(2t^k)) \cdot m\epsilon_\alpha \;\leq\; \sum_{i=1}^{m} C'(z_i) \;\leq\; (1 + 1/(2t^k)) \cdot m\epsilon_\alpha.$$

We now prove the following claim.

Claim. *$\forall x \in \{0,1\}^n$: $\Pr[C''(x) = B(x)] \geq 1 - 2^{-n-1}$ over the random coins of C''.*

This implies the theorem, as follows. By a union bound there is a way to fix the random coins of C'' such that $C''(x) = B(x)$ for every x. Then because $B \in \mathrm{NC}^1$ was arbitrary and $C'' \in \mathrm{TC}^0$, we have $\mathrm{TC}^0 = \mathrm{NC}^1$.

Proof of Claim. Denote $X := \sum_{i=1}^{m} C'(z_i)$, and $\mu := \mathbb{E}[X]$.

Fix x, and first assume $B(x) = 1$ which means $\mu = m\epsilon_\alpha = t^{3k}(n+2)$. Then

$$\Pr[C''(x) = B(x)] = \Pr\left[|X - \mu| \le \mu/(2t^k)\right] \ge 1 - 2e^{-\mu \cdot t^{-3k}} \ge 1 - 2^{-n-1}$$

by a Chernoff bound.

Now assume $B(x) = 0$. Then $\mu = m\epsilon_{\mathsf{id}}$, and since $\epsilon_\alpha/\epsilon_{\mathsf{id}} \ge 1 + 1/t^k$ by (7), we have $(1 - 1/(2t^k)) \cdot m\epsilon_\alpha \ge \mu(1 + 1/(3t^k))$. Then using another Chernoff bound, we have

$$\Pr[C''(x) = B(x)] \ge 1 - \Pr[X \ge \mu(1 + 1/(3t^k))] \ge 1 - e^{-\mu \cdot t^{-3k}} \ge 1 - 2^{-n-1}. \quad \square$$

This completes the proof of the theorem. $\square$

3.3 AC^0

In this section, we observe that by combining the following compression bound against AC^0 due to Dubrov and Ishai [DI06] with our Lemma 3.4 above, we can obtain a quantitatively identical compression bound for A_5-products against AC^0.

Let $\oplus^{-1}(b)$ denote the uniform distribution over n-bit strings with parity $= b$.

Theorem 3.10 ([DI06])**.** *For every $0 < \delta < 1$ and every $d \in \mathbb{N}$, there is a constant $\epsilon > 0$ such that the following holds. For every unbounded-fan-in circuit $C : \{0,1\}^n \to \{0,1\}^{n^\delta}$ of depth $\le d$ and size $\le 2^{\epsilon \cdot n^{(1-\delta)/d}}$:*

$$\Delta(C(\oplus^{-1}(0)), C(\oplus^{-1}(1))) < 2^{-\epsilon \cdot n^{(1-\delta)/d}}.$$

Theorem 3.11. *For every $0 < \delta < 1$ and every $d \in \mathbb{N}$, there is a constant $\epsilon > 0$ such that the following holds. Let $\mathcal{L}$ be the class of unbounded-fan-in circuits $C : (A_5)^t \to \{0,1\}^{t^\delta}$ of depth $\le d$ and size $\le 2^{\epsilon \cdot t^{(1-\delta)/d}}$. Then, $\mathcal{L}$ is $2^{-\epsilon \cdot t^{(1-\delta)/d}}$-fooled by $(A_5)^t$.*

Proof. Assume for contradiction that $\mathcal{L}$ is not $2^{-\epsilon \cdot t^{(1-\delta)/d}}$-fooled by $(A_5)^t$. Then by Lemma 3.1 there is a circuit $C \in \mathcal{L}$ and an $\alpha \in A_5$ such that

$$\Delta(C(D_\alpha), C(D_{\mathsf{id}})) \ge 2^{-\epsilon \cdot t^{(1-\delta)/d}}.$$

For $n = \Omega(t)$, let $M : \{0,1\}^n \to (A_5)^t$ be the 1-local function guaranteed by Lemma 3.4 and Fact 3.5 such that

$$\prod_{i=1}^{t} M(x)_i = \begin{cases} \mathsf{id} & \text{if } \oplus_{j=1}^{n} x_j = 0 \\ \alpha & \text{if } \oplus_{j=1}^{n} x_j = 1. \end{cases}$$

Let R be the function from Lemma 3.2. Then we have

$$\Delta(C(R(M(\oplus^{-1}(0)))), C(R(M(\oplus^{-1}(1))))) \ge 2^{-\epsilon \cdot t^{(1-\delta)/d}} = 2^{-\epsilon' \cdot n^{(1-\delta')/(d+1)}}$$

for appropriately chosen $\epsilon' = \epsilon'(\epsilon)$ and $\delta' = \delta'(\delta)$. Noting that the depth of $C(R(M(\cdot)))$ is $d+1$ and using an averaging argument to fix the randomness of R, this contradicts Theorem 3.10. $\square$

3.4 AC^0 with symmetric gates

Here we show that our hardness assumption holds when $\mathcal{L}$ is the class of unbounded-fan-in constant-depth circuits that contain $t^{O(\log t)}$ And/Or/Not gates and $O(\log^2 t)$ gates that each compute an arbitrary symmetric function. Specifically, we prove the following.

Theorem 3.12. *For every d, there is an $\epsilon > 0$ such that the following holds for every t.*

Let $\mathcal{L}$ be the set of functions $\ell : (A_5)^t \to \{0,1\}^{t^{0.1}}$ where each output bit of ℓ is computable by an unbounded-fan-in circuit of depth $\leq d$ that contains $\leq t^{\epsilon \log t}$ And/Or/Not gates and $\leq \epsilon \log^2 t$ arbitrary symmetric gates.

Then, $\mathcal{L}$ is $t^{-\epsilon \log t}$-fooled by $(A_5)^t$.

Note that in fact these circuits have up to $O(t^{0.1} \cdot \log^2 t)$ arbitrary symmetric gates, though each output bit only depends on $O(\log^2 t)$ of them.

The proof of Theorem 3.12 extends a lower bound due to Viola [Vio07] and combines it with an efficient translation from bits to group products. Viola's lower bound shows that the following function $PAP_{n,m} : \{0,1\}^{n^2 m} \to \{0,1\}$ (for "parity-and-parity") is hard on average for this circuit class to compute.

$$PAP_{n,m}(x) := \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{m} \bigoplus_{k=1}^{n} x_{i,j,k}$$

We use the following translation from PAP-inputs to group products.

Theorem 3.13. *For every $\alpha \in A_5$, there is a 1-local function $M : \{0,1\}^{n^2 m} \to (A_5)^{O(n^2 m^2)}$ that α-computes $PAP_{n,m}$.*

The proof of this theorem is analogous to that of Lemma 3.8, using the "moreover" part of Theorem 3.3. We omit the details.

For the remainder of this section, we let PAP denote $PAP_{n,0.3 \log n}$. By combining Theorem 3.13 with the random self-reducibility of the distributions D_α, we prove the following.

Lemma 3.14. *Let $\mathcal{L}$ be as in Theorem 3.12, and assume that $\mathcal{L}$ is not $t^{-\epsilon \log t}$-fooled by $(A_5)^t$. Then there is an $n = \Omega(\sqrt{t}/\log t)$, an $\epsilon' = \epsilon'(\epsilon) > 0$, and a function $\ell' : \{0,1\}^{n^{2 \cdot 0.3 \log n}} \to \{0,1\}^{t^{0.1}}$ such that each output bit of ℓ' is computable by an unbounded-fan-in circuit of depth $\leq d+1$ that contains $\leq n^{\epsilon' \log n}$ And/Or/Not gates and $\leq \epsilon' \log^2 n$ arbitrary symmetric gates, and*

$$\Delta(\ell'(PAP^{-1}(1)),\ \ell'(PAP^{-1}(0))) \geq n^{-\epsilon' \log n}.$$

Proof. If $\mathcal{L}$ is not $t^{-\epsilon \log t}$-fooled by $(A_5)^t$, then there is an $\alpha \in A_5$ and an $\ell \in \mathcal{L}$ such that $\Delta(\ell(D_\alpha), \ell(D_{\mathsf{id}})) \geq t^{-\epsilon \log t}$. For an appropriate $n = \Omega(\sqrt{t}/\log t)$, let $M : \{0,1\}^{n^{2 \cdot 0.3 \log n}} \to (A_5)^t$ be the 1-local function guaranteed by Theorem 3.13 that α-computes PAP. Let $R : (A_5)^t \to (A_5)^t$ be the randomized 1-local function from Lemma 3.2. Then $\ell' := \ell(R(M(\cdot)))$ satisfies the lemma, and by an averaging argument we can fix the randomness of R so that ℓ' is deterministic. □

We now prove Theorem 3.12. The lower bound in [Vio07] shows that, when restricted to one output bit and with domain $\{0,1\}^{n^2 \cdot 0.3 \log n}$, circuits in $\mathcal{L}$ have correlation $n^{-\Omega(\log n)}$ with PAP. This is done by showing that with probability $1 - n^{-\Omega(\log n)}$ over a random restriction ρ to the input bits of a circuit $C \in \mathcal{L}$, we have both that $PAP|_\rho = GIP$ and that $C|_\rho$ is computable by a $(0.3 \log n)$-party protocol communicating $\log^5 n$ bits, which triggers the lower bound of Lemma 3.7.

In addition to the translation to group products above, we extend this argument to $t^{0.1}$ output bits by using a union bound to show that ρ satisfies these properties simultaneously for all output bits, again with probability $1 - n^{-\Omega(\log n)}$. The protocol now exchanges $t^{0.1} \cdot \log^5 n < n^{0.21}$ bits which is still sufficiently small to use Lemma 3.7.

Proof of Theorem 3.12. Assume that $\mathcal{L}$ is not $t^{-\epsilon \log t}$-fooled by $(A_5)^t$, and let n, ϵ', and $\ell' = (\ell'_1, \ldots, \ell'_{t^{0.1}})$ be given by Lemma 3.14. Let R be the following distribution over restrictions ρ on $n^2 \cdot 0.3 \log n$ bits that leave $n \cdot 0.3 \log n$ bits unrestricted:

- Choose ρ' uniformly over all restrictions that leave $(n^2 \cdot 0.3 \log n)^{0.9}$ bits unset.
- If $PAP|_{\rho'}$ has ≥ 1 input unrestricted per bottom $\oplus$ gate, then choose ρ'' uniformly over restrictions to the remaining bits that leave exactly 1 input unrestricted per bottom $\oplus$ gate.
- Else, choose ρ'' uniformly over all restrictions to the remaining bits that leave exactly $n \cdot 0.3 \log n$ bits unrestricted.
- Output $\rho = \rho' \circ \rho''$.

Say that ρ is *good* if $PAP|_\rho$ has exactly 1 input unrestricted per bottom $\oplus$ gate and for every $i = 1, \ldots, t^{0.1}$, $\ell'_i|_\rho$ is computable by a $(0.3 \log n)$-party protocol (under any partitioning of the input) exchanging $\log^5 n$ bits of communication. Combining [Vio07, Claim 11 & Lemma 12] with a union bound over all ℓ'_i, we obtain

$$\Pr_{\rho \leftarrow R}[\rho \text{ is good}] \geq 1 - n^{-\Omega(\log n)}.$$

Because $\Delta(\ell'(PAP^{-1}(1)), \ell'(PAP^{-1}(0))) \geq n^{-\epsilon' \log n}$, there is a set $S \subseteq \{0,1\}^{t^{0.1}}$ such that

$$\Pr_x[\ell'(x) \in S \mid PAP(x) = 1] - \Pr_x[\ell'(x) \in S \mid PAP(x) = 0] \geq n^{-\epsilon' \log n}. \tag{8}$$

For any ρ that is good, let $P_\rho : \{0,1\}^{n \cdot 0.3 \log n} \to \{0,1\}$ be the following $(0.3 \log n)$-party protocol exchanging $t^{0.1} \cdot \log^5 n + 1 \leq n^{0.21}$ bits. On input y, the parties first compute each $\ell'_i|_\rho(y)$ by communicating $t^{0.1} \cdot \log^5 n$ bits, and then output 1 iff $\ell'|_\rho(y) \in S$ using one additional bit of communication.

For every ρ that is good, $PAP|_\rho$ is equal (up to complementing some inputs) to the generalized inner-product function GIP from §3.1. Thus, by Lemma 3.7 we have

$$\Pr_y[P_\rho(y) = PAP|_\rho(y)] < 1/2 + 2^{-n^{\Omega(1)}} \tag{9}$$

for every good ρ.

Now notice that choosing a random $x \in \{0,1\}^{n^2 \cdot 0.3 \log n}$ can be thought of as first choosing ρ from R and then choosing y uniformly over $\{0,1\}^{n \cdot 0.3 \log n}$. Then letting E_b denote the event "ρ is good and $PAP|_\rho(y) = b$", we have

$$\begin{aligned}
& \Pr_x[\ell'(x) \in S \mid PAP(x) = 1] - \Pr_x[\ell'(x) \in S \mid PAP(x) = 0] \\
= \; & \Pr_{\rho,y}[\ell'|_\rho(y) \in S \mid PAP|_\rho(y) = 1] - \Pr_{\rho,y}[\ell'|_\rho(y) \in S \mid PAP|_\rho(y) = 0] \\
\leq \; & \Pr_{\rho,y}[\ell'|_\rho(y) \in S \mid E_1] - \Pr_{\rho,y}[\ell'|_\rho(y) \in S \mid E_0] + \Pr_\rho[\rho \text{ is not good}] \\
= \; & \Pr_{\rho,y}[P_\rho(y) = 1 \mid E_1] - \Pr_{\rho,y}[P_\rho(y) = 1 \mid E_0] + \Pr_\rho[\rho \text{ is not good}] \\
= \; & \Pr_{\rho,y}[P_\rho(y) = 1 \mid E_1] + \Pr_{\rho,y}[P_\rho(y) = 0 \mid E_0] - 1 + \Pr_\rho[\rho \text{ is not good}] \\
< \; & (1/2 + 2^{-n^{\Omega(1)}})/(1/2 - 2^{-n^{\Omega(1)}}) - 1 + \Pr_\rho[\rho \text{ is not good}] \\
= \; & 2^{-n^{\Omega(1)}} + \Pr_\rho[\rho \text{ is not good}] \\
\leq \; & n^{-\Omega(\log n)}
\end{aligned}$$

which contradicts (8) for sufficiently small ϵ'. Note that the second inequality follows from (9) because $PAP|_\rho = GIP$ is balanced up to an additive factor of $2^{-n^{\Omega(1)}}$. □

4 Proof of Theorem 1.4 and Corollary 1.6

Theorem 1.4. *Let G be a group. For every polynomial-time computable function $t = t(n)$, there is a compiler* Comp *for which the following holds.*

1. *For every $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $k \in \{0,1\}^n$, $\mathsf{Comp}(C,k)$ runs in time $\mathrm{poly}(|C|,t)$ and outputs a circuit $\widehat{C}$ of size $O(t^2 \cdot |C|)$ and depth $O(t \cdot depth(C))$.*
2. *For every set of functions $\mathcal{L}$ and every $\epsilon > 0$, if the 4-local extension of $\mathcal{L}$ is ϵ-fooled by G^t then* Comp *is an $(\mathcal{L}, \epsilon \cdot t \cdot |C|)$-leakage-secure compiler.*

Proof. Let $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $k \in \{0,1\}^n$ be the input to Comp. As described in §2, Comp constructs a circuit $\widehat{C}$ by replacing each wire in C with a bundle of t wires, and replacing each gate in C with a set of gadgets. Specifically for each Nand gate in in C with two input wires and m output wires, $\widehat{C}$ contains a NAND gadget followed by m RANDOM gadgets in parallel (each of which takes as input the NAND gadget's output).

In order for $\widehat{C}(x)$ to map $\{0,1\}^n \to \{0,1\}^n$, it must encode $x \in \{0,1\}^n$ to $x' \in (G^t)^n$ as a first step, and decode $z' \in (G^t)^n$ to $z \in \{0,1\}^n$ as a final step. This is done in the following straightforward way. The input encoder sets each x'_i to be either $(\mathsf{id}, \ldots, \mathsf{id})$ or $(\mathsf{id}, \ldots, \alpha)$ depending if $x_i = 0$ or 1. The output decoder computes each product $\prod_{j=1}^t (z'_i)_j$, and sets $z_i = 0$ or 1 depending if this product is id or α. The decoder may use any correct multiplication tree, i.e. the specific tree used is not relevant to the proof of security.

The size/depth bounds of Comp are immediate. To prove that $\widehat{C}$ is a correct circuit (i.e. that $\widehat{C}(x) = C(x,k)$ for every x), one can apply an inductive argument to show that each bundle at the output of a RANDOM gadget correctly encodes the value of the corresponding wire in C, and thus the output decoder indeed produces $C(x,k)$.

In the hybrid arguments below, we will crucially use the fact that each bundle of the secret state and each bundle at the output of a RANDOM gadget is uniform (over the random coins of Comp) subject to correctly encoding the corresponding wire of C.

On input $(C, x, \widehat{C}(x))$, the simulator S computes a distribution on the wires of $\widehat{C}$ as follows.

First, S computes the wires for the encoder and decoder honestly. For the encoder this is straightforward. For the decoder, S chooses n vectors $z'_i \in G^t$ which are uniform conditioned on the correct product (determined by $\widehat{C}(x)_i$), and then computes the wires for the multiplication trees honestly. These wires are distributed identically to the real distribution on $\widehat{C}(x)$'s wires and thus will not affect the hybrid arguments that follow, which is why these multiplication trees and the complexity of S for this step are not of interest.

Next, S chooses uniformly at random the values for each wire encoding the secret input k, as well as each connecting wire at the output of a RANDOM gate (except those which touch the output decoder and have already been chosen).

Next, for each NAND gadget S computes values for its internal wires and for its output wires by simply evaluating the gadget. (Here we use the fact that the output of one NAND gadget is never the input of another, so all NAND input bundles have already been set.)

Finally, S computes internal wire values for each RANDOM gadget using $\mathsf{R}_{\text{RANDOM}}$.

Now let $\widehat{C} \leftarrow \mathsf{Comp}(C,k)$, and recall that $\widehat{W}_x$ denotes the real distribution on the wires of $\widehat{C}(x)$. We define an intermediate distribution W'_x as follows: first draw a sample from $\widehat{W}_x$, and then recompute the internal wires of each RANDOM gadget from its input/output bundles using $\mathsf{R}_{\text{RANDOM}}$.

We now show that W'_x is indistinguishable (by $\mathcal{L}$) from both $\widehat{W}_x$ and $S(C, x, \widehat{C}(x))$.

Claim 1. *If the* 1*-local extension of* $\mathcal{L}$ *is* ϵ*-fooled by* G^t*, then* $\forall \ell \in \mathcal{L}$*:*

$$\Delta(\ell(W'_x), \ell(\widehat{W}_x)) < \epsilon \cdot |C| \cdot (t-1).$$

Proof. Assume that there an $\ell \in \mathcal{L}$ such that $\Delta(\ell(W'_x), \ell(\widehat{W}_x)) \geq \epsilon \cdot |C| \cdot (t-1)$. Define some fixed ordering on the $\leq |C|$ RANDOM gadgets of $\widehat{C}$. Then by a hybrid argument, there is an $m \leq |C|$ and two distributions H and H', defined as follows, for which $\Delta(\ell(H), \ell(H')) \geq \epsilon \cdot (t-1)$. H is defined by first drawing a sample from $\widehat{W}_x$, and then recomputing the internal wires of RANDOM gadgets $1, \ldots, m$ from their input/output bundles; H' is the same except only random gadgets $1, \ldots, m-1$ are recomputed.

Now by an averaging argument, we can fix all wires in both H and H' except those internal to the mth RANDOM gadget, obtaining a function ℓ' (with domain $G^{|\text{RANDOM}|}$) in the 1-local extension of $\mathcal{L}$. Then ℓ' distinguishes the real wires of the mth RANDOM gadget from those computed by $\mathsf{R}_{\text{RANDOM}}$ with advantage $\geq \epsilon \cdot (t-1)$. In combination with Lemma 2.1, this contradicts the claim's hypothesis. □

Claim 2. *If the 4-local extension of $\mathcal{L}$ is ϵ-fooled by G^t, then $\forall \ell \in \mathcal{L}$:*

$$\Delta(\ell(S(C, x, \widehat{C}(x))), \ell(W'_x)) < \epsilon \cdot |C|.$$

Proof. Assume that there an $\ell \in \mathcal{L}$ such that $\Delta(\ell(S(C, x, \widehat{C}(x))), \ell(W'_x)) \geq \epsilon \cdot |C|$. Define some fixed ordering on the $\leq |C|$ bundles of $\widehat{C}$ that either encode a bit of the secret input k or are at the output of a RANDOM gadget but do not touch the output decoder. Then by a hybrid argument, there is an $m \leq |C|$ and two distributions H and H', defined as follows, for which $\Delta(\ell(H), \ell(H')) \geq \epsilon$. In H, bundles $1, \ldots, m$ are uniformly random and bundles $m+1, \ldots, |C|$ are random subject to correctly encoding the value of the corresponding wire in C; in H' only bundles $1, \ldots, m-1$ are uniformly random. In both, each NAND's internal wires are computed using the gadget itself, and each RANDOM's internal wires are computed using $\mathsf{R}_{\text{RANDOM}}$.

Let $g \in \{\mathsf{id}, \alpha\}$ be the value encoded by the mth bundle in W'_x (determined by C, k and x). Note that the mth bundle is necessarily the input of a NAND gadget, and is either the output of a RANDOM gadget or a bundle encoding a bit of k. By an averaging argument, we can fix all wires in H and H' while preserving $\Delta(\ell(H), \ell(H')) \geq \epsilon$, except for the following: the mth bundle, the internal and output wires of the NAND gadget that it touches, the internal wires of the RANDOM gadgets that are adjacent to the output of this NAND gadget, and the internal wires of the RANDOM gadget that outputs the mth bundle (if it exists).

Finally, a 4-local function can compute one of the two distributions from an input $v \in G^t$ distributed according to either U_{G^t} or D_g: it plugs v into the mth bundle and computes the (4-local) NAND gadget and the (1-local) $\mathsf{R}_{\text{RANDOM}}$. □

Finally, because Δ is a metric, these two claims give part 2 of the theorem. □

Corollary 1.6. *There is a single efficient compiler* Comp*, outputting a circuit $\widehat{C}$ of size $|\widehat{C}| = O(|C|^3)$, that is an $(\mathcal{L}, \epsilon)$-leakage secure compiler for each of the following.*

1. *$\mathcal{L}$ = number-in-hand protocols with s parties communicating $\leq \delta \cdot |\widehat{C}|^{1/3}$ bits, for a fixed $\delta > 0$ and a fixed partition of $\widehat{C}$ into $s = O(1)$ sets; $\epsilon = 2^{-\Omega(|\widehat{C}|^{1/3})}$.*
2. *$\mathcal{L} = \text{AC}^0$ circuits with depth $\leq d$, size $\leq |\widehat{C}|^{O_d(\log |\widehat{C}|)}$, an additional $O_d(\log^2 |\widehat{C}|)$ arbitrary symmetric gates, and $|\widehat{C}|^{0.01}$ bits of output; $\epsilon = |\widehat{C}|^{-\Omega_d(\log |\widehat{C}|)}$.*
3. *If $\text{TC}^0 \neq \text{NC}^1$ then for every k and infinitely many $|C|$, $\mathcal{L} = \text{TC}^0$ circuits with size $\leq |\widehat{C}|^k$ and $k \log |\widehat{C}|$ bits of output; $\epsilon = |\widehat{C}|^{-k}$.*
4. *$\mathcal{L} = \text{AC}^0$ circuits with depth $\leq d$, size $\leq 2^{O_d(|\widehat{C}|^{(1-\delta)/3d})}$, and $|\widehat{C}|^{\delta/3}$ bits of output, for any $\delta < 1$; $\epsilon = 2^{-\Omega_d(|\widehat{C}|^{(1-\delta)/3d})}$.*

Proof. For all four items, we choose $t = |C|$. As mentioned in §1.1, items 2-4 follow in a straightforward manner from Theorem 1.4 and the results of §3. We now give the proof of item 1.

We say that a set of functions $F = \{f : G^t \to \{0,1\}^{|\widehat{C}|}\}$ is *uniformly d-local* if for all $j \leq |\widehat{C}|$, the set $S_j := \{i \leq t \mid \exists f \in F$ whose jth output depends on its ith input$\}$ has size $|S_j| \leq d$. Then, the proof of Theorem 1.4 establishes that, for every C, there are two sets

F_1, F_2 of functions $f : G^t \to \{0,1\}^{|\widehat{C}|}$ for which the following holds. First, F_1 is uniformly 1-local and F_2 is uniformly 4-local. Second, for every $k \in \{0,1\}^n$, if there is a function ℓ on domain $\{0,1\}^{|\widehat{C}|}$ and an $x \in \{0,1\}^n$ such that

$$\Delta(\ell(\widehat{W}_x), \ell(S(C, x, \widehat{C}(x))) \geq \epsilon \cdot t \cdot |C| \tag{10}$$

for $\widehat{C} \leftarrow \mathsf{Comp}(C, k)$, then there is a $g \in G$ and a function $f \in F_1 \cup F_2$ such that

$$\Delta(\ell(f(D_g)), \ell(f(U_{G^t}))) \geq \epsilon. \tag{11}$$

Namely F_1 contains the 1-local functions computing the reduction in Claim 1 above (which essentially is the reduction from Lemma 2.1), and F_2 contains the 4-local functions computing the reduction in Claim 2 above.

Let S be the partition of strings in $(A_5)^t$ onto 6 foreheads given by Theorem 3.6. We define a partition S' of strings in $\{0,1\}^{|\widehat{C}|}$ into 6 hands as follows. For each index $j \leq |\widehat{C}|$, assign j to hand i for some $i \leq 6$ such that for every $f \in F_1 \cup F_2$, the jth output bit of f does not depend on any input element on forehead i in S. This is possible by the definition of uniformly d-local, because each output bit could possibly depend on $\leq 4 + 1 = 5$ input elements and thus ≤ 5 foreheads of S.

Now let P' be any 6-party number-in-hand protocol on domain $\{0,1\}^{|\widehat{C}|}$ under the partition S'. Assume that (10) holds for the function ℓ computed by P', and let $f \in F_1 \cup F_2$ and $g \in G$ be given that satisfy (11). We define a 6-party number-on-forehead protocol P under partition S that computes $\ell(f(\cdot))$ with the same amount of communication as P', which contradicts Theorem 3.6. Specifically, player i in $P(x)$ acts as player i in $P'(f(x))$. This is possible because, by construction, player i in $P(x)$ can compute every bit of $f(x)$ that is held by player i in $P'(f(x))$. □

5 Multi-query security via secure hardware

In this section we consider the extension of our construction to the setting where the adversary can make multiple, adaptive queries. We follow the approach of [FRR+10], using a simple secure hardware component. As mentioned in §1.1, in our setting this component has no input and outputs a sample from the distribution D_{id}.

We generalize Definition 1.1 to the multi-query setting following [FRR+10, Definition 1], beginning with an overview. As before, the adversary A is restricted to choosing leakage functions from some class $\mathcal{L}$ and remains otherwise computationally unbounded. A makes q queries to the circuit $\widehat{C}$, denoted $(x_i, \ell_i) \in \{0,1\}^n \times \mathcal{L}$ for $i \leq q$, and in response to the ith query A receives $(\widehat{C}(x_i), \ell_i(\widehat{W}_i))$ where $\widehat{W}_i$ denotes the wires of $\widehat{C}(x_i)$. A chooses its queries *adaptively*, meaning that each query can depend on all responses seen so far.

On input (C, k), the compiler from §2 implicitly computes a string $\widehat{k}_0 \in (G^t)^n$ that encodes k. In the multi-query setting, this encoding must be "refreshed" between queries, as otherwise A can learn, say, the first bit of k after $O(t)$ queries. To accomplish this, the compiler below outputs a circuit $\widehat{C} : \{0,1\}^n \times (G^t)^n \to \{0,1\}^n \times (G^t)^n$ as well as an initial

encoding $\widehat{k}_0$. Then, the second output $\widehat{k}_i$ of $\widehat{C}(x_i, \widehat{k}_{i-1})$ is used as the second input to $\widehat{C}$ in the $(i+1)$th query. (This corresponds to the notion of a stateful circuit in [FRR+10].) Crucially A does not directly obtain any $\widehat{k}_i$, but the leakage functions operate on these values as they are carried on wires of $\widehat{C}$.

For an adversary A interacting with such a circuit $\widehat{C}$ on initial encoding $\widehat{k}_0$, we let $(x_1, \ell_1) = A(\widehat{C})$ be the first query, and then inductively define

$$\begin{aligned} (y_i, \widehat{k}_i) &:= \widehat{C}(x_i, \widehat{k}_{i-1}) \\ (x_{i+1}, \ell_{i+1}) &:= A(\widehat{C}, x_1, y_1, \ell_1(\widehat{W}_1), \ldots, x_i, y_i, \ell_i(\widehat{W}_i)). \end{aligned} \tag{12}$$

We note two final differences from the single-query setting. First, the circuit $\widehat{C}$ is *randomized*, which means that it contains gates whose output comes from some distribution rather than being deterministically fixed by the input. (In our construction, the only randomized gates will be the secure hardware components which output a sample from D_{id}.) Second, the simulator S is *stateful*, which means that between subsequent evaluations it maintains some state. (In our construction the state will be an element of $(G^t)^n$.)

Definition 5.1. Let $\mathsf{Comp}(\cdot,\cdot)$ be a randomized algorithm that takes as input a circuit $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and a string $k \in \{0,1\}^n$. For a class of leakage functions $\mathcal{L}$, Comp is a *q-query $(\mathcal{L}, \epsilon)$-leakage-secure compiler* if the following properties hold.

1. (Structure.) For every C and k, $\mathsf{Comp}(C, k)$ outputs a string $\widehat{k}_0 \in (G^t)^n$, and a randomized circuit $\widehat{C} : \{0,1\}^n \times (G^t)^n \to \{0,1\}^n \times (G^t)^n$ which is completely determined by C.

2. (Correctness.) For every A as above, every C, k, and every $i \leq q$: $y_i = C(x_i, k)$ with probability 1.

3. (Security.) There is a randomized polynomial-time stateful algorithm S such that the following holds for every C, k and every A as above. Let D_{real} denote the distribution $(\ell_1(\widehat{W}_1), \ldots, \ell_q(\widehat{W}_q))$, and let D_{sim} denote the corresponding distribution $(\ell_1(S(C, x_1, y_1)), \ldots, \ell_q(S(C, x_q, y_q)))$ when each $\widehat{W}_j$ in (12) $(1 \leq j \leq i)$ is replaced with $S(C, x_j, y_j)$. Then, $\Delta(D_{\text{real}}, D_{\text{sim}}) < \epsilon$.

The construction. Recall from §1.1 that a D_{id}-gate is a gate with no input that on each execution of the circuit outputs a string of length $2t$ sampled from D_{id}, and any circuit that contains one or more D_{id}-gates is a D_{id}-circuit.

Then, Comp outputs a D_{id}-circuit $\widehat{C} : \{0,1\}^n \times (G^t)^n \to \{0,1\}^n \times (G^t)^n$ as follows. $\widehat{C}$ is identical to the construction in §2 with two exceptions. First, each pair $(R^{(i)}, L^{(i+1)})$ in each RANDOM gadget is computed by a D_{id}-gate. Second, $\widehat{C}$ computes its second output $\widehat{k}_i$ by applying a RANDOM gadget to each bundle of its second input $\widehat{k}_{i-1}$.

The simulator S then operates as follows. For the first query, S chooses $\widehat{k}_0, \widehat{k}_1 \in (G^t)^n$ uniformly at random, and produces wire values for $\widehat{C}(x_1, \widehat{k}_0)$ conditioned on output $(y_1, \widehat{k}_1)$

as described in Theorem 1.4. Between queries i and $i+1$, S stores the value $\widehat{k}_i$, and for the $(i+1)$th query it chooses $\widehat{k}_{i+1}$ uniformly at random and proceeds in the same manner.

Proving the security of this construction requires a slightly stronger property of the group encoding than what is given by Definition 1.2. Namely, it requires that the leakage class $\mathcal{L}$ cannot distinguish the distributions D_α and U_{G^t} even with two *adaptive* queries. The need for this is due to the fact that each $\widehat{k}_i$ $(1 \leq i < q)$ is given as input to two leakage functions: once when $\widehat{k}_i$ is an output of $\widehat{C}$ and once when it is an input. Formally, we require the following strengthening of Definition 1.2.

Definition 5.2. Let G be a group and $t \in \mathbb{N}$. A set of functions $\mathcal{L}$ is *2-adaptive ϵ-fooled by* G^t if for every $\alpha \in G$, every $\ell \in \mathcal{L}$, and every function $A : \mathrm{range}(\ell) \to \mathcal{L}$, the following two distributions are ϵ-close in statistical distance.

1. Sample $w \leftarrow D_\alpha$, compute $\ell' := A(\ell(w))$, and output $(\ell(w), \ell'(w))$.
2. Sample $w \leftarrow U_{G^t}$, compute $\ell' := A(\ell(w))$, and output $(\ell(w), \ell'(w))$.

With this stronger property, the following theorem can be proved by building on Theorem 1.4. Specifically, one first uses Theorem 1.4 to show that the security property of Definition 5.1 holds when the simulator chooses each $\widehat{k}_i$ as in the real execution but the internal wires of each evaluation $\widehat{C}(x_i, \widehat{k}_i)$ are reconstructed. Then, a hybrid argument over each bundle in each $\widehat{k}_i$ is used to show that the security property is satisfied even when S chooses each $\widehat{k}_i$ uniformly at random. The proof is essentially identical to the proof of [FRR+10, Lemma 15], and we omit the details.

Theorem 5.3. *Let G be a group. For every polynomial-time computable function $t = t(n, |C|)$, there is a compiler* Comp *for which the following holds.*

1. *For every $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and $k \in \{0,1\}^n$, $\mathsf{Comp}(C, k)$ runs in time $\mathrm{poly}(|C|, t)$ and outputs a D_{id}-circuit $\widehat{C}$ of size $O(t^2 \cdot |C|)$ and depth $O(t \cdot \mathit{depth}(C))$.*
2. *For every set of functions $\mathcal{L}$, every $q \in \mathbb{N}$, and every $\epsilon > 0$, if the 4-local extension of $\mathcal{L}$ is 2-adaptive ϵ-fooled by G^t then Comp is a q-query $(\mathcal{L}, \epsilon')$-leakage-secure compiler for $\epsilon' := (q+1) \cdot \epsilon \cdot (n + t \cdot |C|)$.*

Similarly to Corollary 1.6, Corollary 1.7 is derived as a result of Theorem 5.3 and the following subsection, by choosing $t = |C|$.

5.1 Adaptive compression bounds

In this section, we show that the function classes from §3 are 2-adaptive fooled by $(A_5)^t$. For the class of functions computable by AC^0 with symmetric gates, this requires asymptotically smaller output length.

5.1.1 Multi-party protocols

Theorem 5.4. *There is a partition of the inputs in $(A_5)^t$ into k pieces such that the set $\mathcal{L}$ of k-party number-on-forehead protocols communicating c bits and outputting $\leq c$ bits is 2-adaptive ϵ-fooled by $(A_5)^t$ for $\epsilon = 2^{c-\Omega(t/(k^2 4^k))}$.*

Proof. The partition is the same as in Theorem 3.6. Assume that $\mathcal{L}$ is not 2-adaptive ϵ-fooled by $(A_5)^t$, and let $\alpha \in A_5$, $P' \in \mathcal{L}$, and $A : \text{range}(P') \to \mathcal{L}$ violate Definition 5.2.

Consider the following k-party protocol P that communicates and outputs $2c$ bits. On input x, the parties first compute $P'(x)$ by communicating c bits. Then, they each determine $P'' := A(P'(x))$ with no communication. Finally, they compute $P''(x)$ again communicating c bits. By assumption we have $\Delta(P(D_\alpha), P(U_{(A_5)^t})) \geq \epsilon$, which contradicts Theorem 3.6. □

5.1.2 TC^0

Theorem 5.5. *If $\mathrm{TC}^0 \neq \mathrm{NC}^1$ then $\forall k$ and infinitely many t, the class $\mathcal{L}$ of TC^0 circuits with size $\leq t^k$ and output length $k \log t$ is 2-adaptive t^{-k}-fooled by $(A_5)^t$.*

Proof. Assume that there exists k such that for sufficiently large t, $\mathcal{L}$ is not 2-adaptive t^{-k}-fooled by $(A_5)^t$. Let $\alpha \in A_5$, $\ell : (A_5)^t \to \{0,1\}^{k \log t}$, and $A : \{0,1\}^{k \log t} \to \mathcal{L}$ be the choices that violate Definition 5.2. Let $\ell'_1, \ldots, \ell'_{t^k} \in \mathcal{L}$ be all possible circuits that A could output.

Let $\ell'' : (A_5)^t \to \{0,1\}^{2k \log t}$ be the following procedure. On input $x \in (A_5)^t$: first compute $\ell(x)$, then select $\ell' := \ell'_{\ell(x)}$ (identifying $\ell(x) \in \{0,1\}^{k \log t}$ with the natural number it represents), and finally output $(\ell(x), \ell'(x))$. Clearly ℓ'' is computable by a TC^0 circuit of size $t^{O(k)}$, and by assumption we have $\Delta(\ell''(D_\alpha), \ell''(U_{(A_5)^t})) \geq t^{-k}$ which contradicts Theorem 3.9. □

5.1.3 AC^0

The proof of the following theorem is essentially identical to the previous proof.

Theorem 5.6. *For every $0 < \delta < 1$ and every integer $d > 0$, there is a constant $\epsilon > 0$ such that the following holds. Let $\mathcal{L}$ be the class of unbounded-fanin circuits $C : (A_5)^t \to \{0,1\}^{t^\delta}$ of depth $\leq d$ and size $\leq 2^{\epsilon \cdot t^{(1-\delta)/d}}$. Then, $\mathcal{L}$ is 2-adaptive $2^{-\epsilon \cdot t^{(1-\delta)/d}}$-fooled by $(A_5)^t$.*

Proof. Assume that $\mathcal{L}$ is not 2-adaptive $2^{-\epsilon \cdot t^{(1-\delta)/d}}$-fooled by $(A_5)^t$, and let $\alpha \in A_5$, $\ell \in \mathcal{L}$, and $A : \{0,1\}^{t^\delta} \to \mathcal{L}$ be the choices that violate Definition 5.2. Let $\ell'_1, \ldots, \ell'_{2^{t^\delta}} \in \mathcal{L}$ be all possible circuits that A could output.

Let $\ell'' : (A_5)^t \to \{0,1\}^{2t^\delta}$ be the following procedure. On input $x \in (A_5)^t$: first compute $\ell(x)$, then select $\ell' := \ell'_{\ell(x)}$, and finally output $(\ell(x), \ell'(x))$. Clearly ℓ'' is computable by an AC^0 circuit of depth $2d + O(1)$ and size $2 \cdot 2^{\epsilon \cdot t^{(1-\delta)/d}} + 2^{t^\delta} + O(1) = 2^{\epsilon' \cdot t^{(1-\delta')/(2d+O(1))}}$ for an appropriate ϵ' and δ', which contradicts Theorem 3.11. □

5.1.4 AC^0 with symmetric gates

Recall that in §3.4, it was shown that $(A_5)^t$ $t^{-\Omega(\log t)}$-fools the class of unbounded-fan-in constant-depth circuits that contain $t^{O(\log t)}$ And/Or/Not gates and $O(\log^2 t)$ arbitrary symmetric gates and output $t^{0.1}$ bits. To apply the technique from the two preceding proofs to this class, one needs to restrict the output length to $O(1)$ to ensure that the circuit ℓ'' still contains only $O(\log^2 t)$ symmetric gates. However, by a more careful extension of Theorem 3.12 we can improve the output length to $\Omega(\log^2 t)$. In the following, we focus mainly on the necessary changes to the proof of Theorem 3.12.

Theorem 5.7. *For every d, there is an $\epsilon > 0$ such that the following holds for every t.*

Let $\mathcal{L}$ be the set of functions $\ell : (A_5)^t \to \{0,1\}^{\epsilon \log^2 t}$ where each output bit of ℓ is computable by an unbounded-fan-in circuit of depth $\leq d$ that contains $\leq t^{\epsilon \log t}$ And/Or/Not gates and $\leq \epsilon \log^2 t$ arbitrary symmetric gates.

Then, $\mathcal{L}$ is 2-adaptive $t^{-\epsilon \log t}$-fooled by $(A_5)^t$.

Proof. Assume that $\mathcal{L}$ is not 2-adaptive $t^{-\epsilon \log t}$-fooled by $(A_5)^t$, and let $\alpha \in A_5$, $\ell_0 \in \mathcal{L}$, and $A : \{0,1\}^{\epsilon \log^2 t} \to \mathcal{L}$ be the choices that violate Definition 5.2. Let $\ell_1, \ldots, \ell_{t^{\epsilon \log t}} \in \mathcal{L}$ be all possible circuits that A could output.

For an appropriate $n = \Omega(\sqrt{t}/\log t)$ and for each $i = 0, \ldots, t^{\epsilon \log t}$, let $\ell'_i : \{0,1\}^{n^2 \cdot 0.3 \log n} \to \{0,1\}^{\epsilon \log^2 t}$ be the corresponding function given by Lemma 3.14. These functions have the property that the distribution $(\ell'_0(x), \ell'_{\ell'_0(x)}(x))$ when $x \leftarrow PAP^{-1}(1)$ has statistical distance $\geq n^{-\epsilon' \log n}$ from the corresponding distribution when $x \leftarrow PAP^{-1}(0)$, for $\epsilon' = \epsilon'(\epsilon) > 0$.

Let R be the distribution on random restrictions ρ given in the proof of Theorem 3.12. Say that ρ is *good* if $PAP|_\rho = GIP$ (i.e. $PAP|_\rho$ has exactly 1 input restricted per bottom $\oplus$ gate) and for every $i \leq t^{\epsilon \log t}$ and $j \leq \epsilon \log^2 t$, $\ell'_{i,j}|_\rho$ is computable by a $(0.3 \log n)$-party protocol exchanging $\log^5 n$ bits (where $\ell'_{i,j}$ denotes the jth output bit of ℓ'_i.) Because the number of $\ell'_{i,j}$ is $\epsilon \log^2 t \cdot (t^{\epsilon \log t} + 1)$ and because $t = n^{O(1)}$, when ϵ is sufficiently small we obtain

$$\Pr_{\rho \leftarrow R}[\rho \text{ is good}] \geq 1 - n^{\Omega(\log n)}$$

by combining [Vio07, Claim 11 & Lemma 12] with a union bound.

For any ρ that is good, let $P_\rho : \{0,1\}^{n \cdot 0.3 \log n} \to \{0,1\}$ be the following $(0.3 \log n)$-party protocol exchanging $2\epsilon \log^2 t \log^5 n + 1 = \log^{O(1)} n$ bits. On input y, the parties first compute each of the $\epsilon \log^2 t$ output bits of $\ell'_0|_\rho(y)$, exchanging a total of $\epsilon \log^2 t \log^5 n$ bits. Then, each party chooses $\ell'_i := \ell'_{\ell'_0|_\rho(y)}$ with no communication. Then, the parties compute each of the $\epsilon \log^2 t$ output bits of $\ell'_i|_\rho(y)$ again exchanging $\epsilon \log^2 t \log^5 n$ bits. Finally, the parties use one additional bit of communication to output 1 iff $(\ell'_0|_\rho(y), \ell'_i|_\rho(y)) \in S$ for the appropriate set S corresponding to (8) in Theorem 3.12.

The rest of the proof follows the same argument as Theorem 3.12. □

References

[AAW10] Eric Allender, Vikraman Arvind, and Fengming Wang. Uniform derandomization from pathetic lower bounds. In *Workshop on Randomization and Computation (RANDOM)*, pages 380–393, 2010.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. on Computing*, 36(4):845–888, 2006.

[All01] Eric Allender. The division breakthroughs. *Bulletin of the EATCS*, 74:61–77, 2001.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. of Computer and System Sciences*, 38(1):150–164, 1989.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Int. Cryptology Conf. (CRYPTO)*, pages 1–18, 2001.

[BNS92] László Babai, Noam Nisan, and Márió Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. of Computer and System Sciences*, 45(2):204–232, 1992.

[CFL83] Ashok K. Chandra, Merrick L. Furst, and Richard J. Lipton. Multi-party protocols. In *15th ACM Symp. on the Theory of Computing (STOC)*, pages 94–99, 1983.

[Cle91] Richard Cleve. Towards optimal simulations of formulas by bounded-width programs. *Computational Complexity*, 1:91–105, 1991.

[CM87] Stephen A. Cook and Pierre McKenzie. Problems complete for deterministic logarithmic space. *J. Algorithms*, 8(3):385–394, 1987.

[CT93] Fan R. K. Chung and Prasad Tetali. Communication complexity and quasi randomness. *SIAM J. Discrete Math.*, 6(1):110–123, 1993.

[DF12] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In *Theory of Cryptography Conf. (TCC)*, pages 230–247, 2012.

[DI06] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *38th ACM Symposium on Theory of Computing (STOC)*, pages 711–720, 2006.

[Dru12] Andrew Drucker. New limits to classical and quantum instance compression. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, 2012.

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *Int. Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 135–156, 2010. The full version is available at `http://eprint.iacr.org/2009/379`.

[GGH+08] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy Rothblum. A (de)constructive approach to program checking. In *40th ACM Symposium on Theory of Computing (STOC)*, pages 143–152, 2008.

[GR10] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *Int. Cryptology Conf. (CRYPTO)*, pages 59–79, 2010.

[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, 2012.

[HN10] Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM J. on Computing*, 39(5):1667–1713, 2010.

[HP89] Derek Holt and W. Plesken. *Perfect Groups*. Oxford Mathematical Monographs. Clarendon Press, 1989.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Int. Cryptology Conf. (CRYPTO)*, pages 463–481, 2003.

[JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *Int. Cryptology Conf. (CRYPTO)*, pages 41–58, 2010.

[KS04] Hans Kurzweil and Bernd Stellmacher. *The Theory of Finite Groups: An Introduction*. Springer, 2004.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography Conf. (TCC)*, pages 278–296, 2004.

[Nis93] Noam Nisan. The communication complexity of threshold gates. In *Combinatorics, Paul Erdős is Eighty, number 1 in Bolyai Society Mathematical Studies*, pages 301–315, 1993.

[Raz00] Ran Raz. The BNS-Chung criterion for multi-party communication complexity. *Comput. Complexity*, 9(2):113–122, 2000.

[Rot12] Guy N. Rothblum. How to compute under AC^0 leakage without secure hardware. In *Int. Cryptology Conf. (CRYPTO)*, pages 552–569, 2012.

[Vio07] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM J. on Computing*, 36(5):1387–1403, 2007.

[Vio11] Emanuele Viola. Selected results in additive combinatorics: An exposition. *Theory of Computing Library, Graduate Surveys series*, 3:1–15, 2011.

[VW08] Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for GF(2) polynomials and multiparty protocols. *Theory of Computing*, 4:137–168, 2008.

[yCL94] Jin yi Cai and Richard J. Lipton. Subquadratic simulations of balanced formulae by branching programs. *SIAM J. on Computing*, 23(3):563–572, 1994.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de