Constant-Depth Circuits for Arithmetic in Finite Fields of Characteristic Two

Alexander Healy* Emanuele Viola†

August 8, 2005

Abstract

We study the complexity of arithmetic in finite fields of characteristic two, $\mathbb{F}_{2^n}$. We concentrate on the following two problems:

- Iterated Multiplication: Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_{2^n}$.
- Exponentiation: Given $\alpha \in \mathbb{F}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_{2^n}$.

First, we consider the explicit realization of the field $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(x^{2\cdot 3^l} + x^{3^l} + 1) \simeq \mathbb{F}_{2^n}$, where $n = 2\cdot 3^l$. In this setting, we exhibit *Dlogtime*-uniform poly(n, t)-size TC^0 circuits computing exponentiation. To the best of our knowledge, prior to this work it was not even known how to compute exponentiation in logarithmic space, i.e. space $O(\log(n + t))$, over any finite field of size $2^{\Omega(n)}$. We also exhibit, for every $\epsilon > 0$, *Dlogtime*-uniform poly$(n, 2^{t^\epsilon})$-size $AC^0[\text{mod } 2]$ circuits computing iterated multiplication and exponentiation, which we prove is optimal.

Second, we consider arbitrary realizations of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$, for an irreducible $f(x) \in \mathbb{F}_2[x]$ that is given as part of the input. In this setting, we exhibit, for every $\epsilon > 0$, *Dlogtime*-uniform poly$(n, 2^{t^\epsilon})$-size $AC^0[\text{mod } 2]$ circuits computing iterated multiplication, which is again tight. We also exhibit *Dlogtime*-uniform poly$(n, 2^t)$-size $AC^0[\text{mod } 2]$ circuits computing exponentiation.

Our results over $\mathbb{F}_2[x]/(x^{2\cdot 3^l} + x^{3^l} + 1)$ have several consequences:

We prove that *Dlogtime*-uniform TC^0 equals the class AE of functions computable by certain arithmetic expressions. This answers a question raised by Frandsen, Valence and Barrington (Mathematical Systems Theory '94). We also show how certain optimal constructions of k-wise independent and ϵ-biased generators are explicitly computable in *Dlogtime*-uniform $AC^0[\oplus]$ and TC^0. This addresses a question raised by Gutfreund and Viola (RANDOM '04).

*Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, ahealy@fas.harvard.edu. Research supported by NSF grant CCR-0205423 and a Sandia Fellowship.

†Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, viola@eecs.harvard.edu. Research supported by NSF grant CCR-0133096, US-Israel BSF grant 2002246, ONR grant N-00014-04-1-0478.

1 Introduction

Finite fields have a wide variety of applications in computer science, ranging from Coding Theory to Cryptography to Complexity Theory. In this work we study the complexity of arithmetic operations in finite fields.

When considering the complexity of finite field arithmetic, there are two distinct problems one must consider. The first is the problem of actually *constructing* the desired finite field, $\mathbb{F}$; for example, one must find a prime p in order to realize the field $\mathbb{F}_p$ as $\mathbb{Z}/p\mathbb{Z}$. The second is the problem of performing arithmetic operations, such as addition, multiplication and exponentiation in the field $\mathbb{F}$. In this work, we focus on this second problem, and restrict our attention to fields $\mathbb{F}$ where a realization of the field can be found very easily, or where a realization of $\mathbb{F}$ is given as part of the input.

Specifically, we will focus on finite fields of characteristic two; that is, finite fields $\mathbb{F}_{2^n}$ having 2^n elements. In particular, the question we address is: *To what extent can basic field operations (e.g., multiplication, exponentiation) in $\mathbb{F}_{2^n}$ be computed by constant-depth circuits?* In our work, we consider three natural classes of unbounded fan-in constant-depth circuits: circuits over the bases $\{\wedge, \vee\}$ (i.e., AC^0), $\{\wedge, \vee, \mathit{Parity}\}$ (i.e., $AC^0[\oplus]$), and $\{\wedge, \vee, \mathit{Majority}\}$ (i.e., TC^0). Moreover, we will focus on *uniform* constant-depth circuits, although we defer the discussion of uniformity until the paragraph "Uniformity" later in this section. Recall that, for polynomial-size circuits, $AC^0 \subsetneq AC^0[\oplus] \subsetneq TC^0 \subseteq$ logarithmic space, where the last inclusion holds under logarithmic-space uniformity and the separations follow from works by Furst et al. [FSS] and Razborov [Raz], respectively (and hold even for non-uniform circuits).

Field Operations. Recall that the finite field $\mathbb{F}_{2^n}$ of characteristic two is generally realized as $\mathbb{F}_2[x]/(f(x))$ where $f(x) \in \mathbb{F}_2[x]$ is an irreducible polynomial of degree n. Thus, field elements are polynomials of degree at most $n-1$ over $\mathbb{F}_2$, addition of two field elements is addition in $\mathbb{F}_2[x]$ and multiplication of field elements is carried out modulo the irreducible polynomial $f(x)$. Throughout, we identify a field element $\alpha = a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{F}_{2^n}$ with the n-dimensional bit-vector $(a_0, a_1, \ldots, a_{n-1}) \in \{0,1\}^n$, and we assume that all field elements that are given as inputs or returned as outputs of some computation are of this form.

In such a realization of $\mathbb{F}_{2^n}$, addition of two field elements is just component-wise XOR and therefore trivial, even for AC^0 circuits. It is also easy to establish the complexity of Iterated Addition, i.e. given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{2^n}$, computing $\alpha_1 + \alpha_2 + \cdots + \alpha_t \in \mathbb{F}_{2^n}$. This is easily seen to be computable by $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n,t)$. On the other hand, since parity is a special case of Iterated Addition, the latter requires AC^0 circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$ (see e.g. [Hås]). Thus, we concentrate on the following *multiplicative* field operations:

- Iterated Multiplication: Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_{2^n}$.
- Exponentiation: Given $\alpha \in \mathbb{F}_{2^n}$, and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_{2^n}$.

The goal is to compute these functions as efficiently as possible for given parameters n, t. We note that these functions can be computed in time $\mathrm{poly}(n,t)$ (using the repeated squaring algorithm for exponentiation). In this work we ask what the smallest constant-depth circuits are for computing these functions. Note that computing Iterated Multiplication immediately implies being able to compute the product of two given field elements. While solving this latter problem is already non-trivial (for *Dlogtime*-, or even logspace-uniform constant-depth circuits), we will not address it separately.

Our Results. We present two different types of results. The first concerns field operations in a *specific* realization of $\mathbb{F}_{2^n}$, which we denote $\tilde{\mathbb{F}}_{2^n}$. The second type concerns field operations in an *arbitrary* realization of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$, where we assume that the irreducible polynomial $f(x)$ is given as part of the input. We describe both of these kinds of results in more detail below. Then we discuss some applications of our results.

Results in the specific representation $\tilde{\mathbb{F}}_{2^n}$: In this setting, we assume that n is of the form $n = 2 \cdot 3^l$, for some non-negative integer l, and we employ the explicit realization of $\mathbb{F}_{2^n}$ given by $\mathbb{F}_2[x]/(f(x))$ where $f(x)$ is the irreducible polynomial $x^{2 \cdot 3^l} + x^{3^l} + 1 \in \mathbb{F}_2[x]$. Our results are summarized in Table 1.

We show that exponentiation can be computed by uniform TC^0 circuits of size $\text{poly}(n, t)$ (i.e. what is achievable by standard unbounded-depth circuits). To the best of our knowledge, prior to this work it was not even known how to compute exponentiation in logarithmic space, i.e. space $O(\log(n + t))$, over any finite field of size $2^{\Omega(n)}$. As a corollary, we improve upon a theorem of Agrawal et al. [AAI+] concerning exponentiation in uniform AC^0. In the case of iterated multiplication of t field elements, results of Hesse et al. [HAB] imply that this problem can be solved by uniform TC^0 circuits of size $\text{poly}(n, t)$.

We also show that, for every $\epsilon > 0$, iterated multiplication and exponentiation can be computed by uniform $AC^0[\oplus]$ circuits of size $\text{poly}(n, 2^{t^\epsilon})$. Moreover, we show that this is tight: neither iterated multiplication nor exponentiation can be computed by (nonuniform) $AC^0[\oplus]$ circuits of size $\text{poly}(n, 2^{t^{o(1)}})$.

Results in arbitrary representation $\mathbb{F}_2[x]/(f(x))$: In this setting we assume that the irreducible polynomial $f(x)$ is arbitrary, but is given to the circuit as part of the input. Our results are summarized in Table 2.

We show (with a more complicated proof than in the specific representation case) that iterated multiplication can be computed by uniform $AC^0[\oplus]$ circuits of size $\text{poly}(n, 2^{t^\epsilon})$, and this is again tight. We show that exponentiation can be computed by uniform $AC^0[\oplus]$ circuits of size $\text{poly}(n, 2^t)$, but we do not know how to match the size $\text{poly}(n, 2^{t^\epsilon})$ achieved in the specific representation case. More dramatically, we do not know if there exist $\text{poly}(n, 2^{o(t)})$-size TC^0 circuits for exponentiation. While we cannot establish a lower bound for exponentiation, we observe that testing whether a given $\mathbb{F}_2[x]$ polynomial of degree n is irreducible can be $AC^0[\oplus]$ reduced to computing exponentiation in a given representation of $\mathbb{F}_{2^n}$, for exponents with $t = n$ bits. Specifically, a modification of Rabin's irreducibility test [Rab, MS] gives a TC^0 reduction; we show a finer analysis that gives a $AC^0[\oplus]$ reduction. Thus, any improvement on our results for exponentiation modulo a given (irreducible) polynomial of degree at most n would yield an upper bound on the complexity of testing irreducibility of a given $\mathbb{F}_2[x]$ polynomial. Some lower bounds for the latter problem are given in the recent work of Allender et. al. [ABD+]. However, it is still open whether irreducibility of a given degree-n polynomial in $\mathbb{F}_2[x]$ can be decided by $AC^0[\oplus]$ circuits of size $\text{poly}(n)$.

	AC^0	$AC^0[\oplus]$	TC^0
Addition: $\alpha, \beta \in \tilde{\mathbb{F}}_{2^n} \to \alpha + \beta \in \tilde{\mathbb{F}}_{2^n}$	poly(n) [Folklore]	poly(n) [Folklore]	poly(n) [Folklore]
Iterated Addition: $\alpha_1, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n} \to \sum_{i \le t} \alpha_i \in \tilde{\mathbb{F}}_{2^n}$	poly$(n, 2^{t^\epsilon})$ [Folklore]	poly(n, t) [Folklore]	poly(n, t) [Folklore]
Multiplication: $\alpha, \beta \in \tilde{\mathbb{F}}_{2^n} \to \alpha \cdot \beta \in \tilde{\mathbb{F}}_{2^n}$	poly(2^{n^ϵ}) [Cor. 6 (1)]	poly(n) [Thm. 4 (1)]	poly(n) [HAB]
Iterated Multiplication: $\alpha_1, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n} \to \prod_{i \le t} \alpha_i \in \tilde{\mathbb{F}}_{2^n}$	poly$(2^{n^\epsilon}, 2^{t^\epsilon})$ [Cor. 6 (1)]	poly$(n, 2^{t^\epsilon})$ [Thm. 4 (1)]	poly(n, t) [HAB]
Exponentiation: $\alpha \in \tilde{\mathbb{F}}_{2^n}$, t-bit $k \in \mathbb{Z} \to \alpha^k \in \tilde{\mathbb{F}}_{2^n}$	poly$(2^{n^\epsilon}, 2^{t^\epsilon})$ [Cor. 6 (2)]	poly$(n, 2^{t^\epsilon})$ [Thm. 4 (2)]	poly(n, t) [Thm. 3 (2)]
In the above, $\epsilon > 0$ is arbitrary, but the circuits have depth $O(1/\epsilon)$.			

Table 1: Complexity of Operations in $\tilde{\mathbb{F}}_{2^n} \equiv \mathbb{F}_2[x]/(x^{2 \cdot 3^l} + x^{3^l} + 1)$.

	AC^0	$AC^0[\oplus]$	TC^0
Addition: $\alpha, \beta \in \mathbb{F}_{2^n} \to \alpha + \beta \in \mathbb{F}_{2^n}$	poly(n) [Folklore]	poly(n) [Folklore]	poly(n) [Folklore]
Iterated Addition: $\alpha_1, \ldots, \alpha_t \in \mathbb{F}_{2^n} \to \sum_{i \le t} \alpha_i \in \mathbb{F}_{2^n}$	poly$(n, 2^{t^\epsilon})$ [Folklore]	poly(n, t) [Folklore]	poly(n, t) [Folklore]
Multiplication: $\alpha, \beta \in \mathbb{F}_{2^n} \to \alpha \cdot \beta \in \mathbb{F}_{2^n}$	poly(2^{n^ϵ}) Cor. to [HAB]	poly(n) [Thm. 7 (1)]	poly(n) [HAB]
Iterated Multiplication: $\alpha_1, \ldots, \alpha_t \in \mathbb{F}_{2^n} \to \prod_{i \le t} \alpha_i \in \mathbb{F}_{2^n}$	poly$(2^{n^\epsilon}, 2^{t^\epsilon})$ Cor. to [HAB]	poly$(n, 2^{t^\epsilon})$ [Thm. 7 (1)]	poly(n, t) [HAB]
Exponentiation: $\alpha \in \mathbb{F}_{2^n}$, t-bit $k \in \mathbb{Z} \to \alpha^k \in \mathbb{F}_{2^n}$	poly$(2^{n^\epsilon}, 2^{2^{\epsilon t}})$ Cor. to [HAB]	poly$(n, 2^t)$ [Thm. 7 (2)]	poly$(n, 2^t)$ [HAB]
In the above, $\epsilon > 0$ is arbitrary, but the circuits have depth $O(1/\epsilon)$.			

Table 2: Complexity of Operations in $\mathbb{F}_{2^n} \equiv \mathbb{F}_2[x]/(f(x))$ for given $f(x)$ of degree n.

We now discuss several applications of our results in the specific representation $\tilde{\mathbb{F}}_{2^n}$.

AE = Dlogtime-uniform TC^0: Frandsen, Valence and Barrington [FVB] study the relationship between uniform TC^0 and the class AE of functions computable by certain arithmetic expressions (defined in Section 2.3). Remarkably, they show that *Dlogtime*-uniform TC^0 is contained in AE. Conversely, they show that AE is contained in P-uniform TC^0, but they leave open whether the inclusion holds under *Dlogtime* uniformity. We show that AE is in fact contained in *Dlogtime*-uniform TC^0, thus proving that $AE =$ *Dlogtime*-uniform TC^0. (See paragraph "Uniformity" for a discussion of *Dlogtime*-uniformity.)

"Pseudorandom" Generators: We implement certain "pseudorandom" generators in *Dlogtime*-uniform constant-depth circuits. Specifically, we show how a construction of k-wise independent generators from [CG, ABI] can be implemented in uniform $AC^0[\oplus]$, and how a construction of ϵ-biased generators from [AGHP] can be implemented in uniform TC^0. These constructions address a problem posed by Gutfreund and Viola [GV].

Overview of Techniques. Our results for the specific representation $\tilde{\mathbb{F}}_{2^n} = \mathbb{F}_2[x]/(x^{2\cdot 3^l} + x^{3^l} + 1)$ exploit the special structure of the irreducible polynomial $x^{2\cdot 3^l} + x^{3^l} + 1 \in \mathbb{F}_2[x]$. The crucial observation (Fact 17) is that the order of x modulo $x^{2\cdot 3^l} + x^{3^l} + 1$ is small and is easily computed, namely it is 3^{l+1}. Thus, we are able to compute large powers of the element $x \in \tilde{\mathbb{F}}_{2^n}$ by considering the exponent k modulo the order of x. To better illustrate this idea we now sketch a proof of the fact that exponentiation over $\tilde{\mathbb{F}}_{2^n}$ can be computed by uniform TC^0 circuits of size poly(n,t). Let $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and an exponent $0 \leq k < 2^t$ be given. We think of α as a polynomial $\alpha(x) \in \mathbb{F}_2[x]$. Writing k in binary as $k = k_{t-1}k_{t-2}\cdots k_0 = \sum_{i<t} k_i 2^i$ where $k_i \in \{0,1\}$, we have:

$$\alpha(x)^k = \alpha(x)^{\sum\limits_{i<t} k_i 2^i} = \prod_{i<t} \alpha(x)^{k_i 2^i} = \prod_{i<t} \alpha\left(x^{2^i}\right)^{k_i}$$

where the last equality follows from the fact that we are working in characteristic 2. Using the fact that the iterated product of t field elements is computable by uniform TC^0 circuits of size poly(n,t) (which follows from results in [HAB]), all that is left to do is to show how to compute $\alpha(x^{2^i})^{k_i}$. Since $k_i \in \{0,1\}$, the only hard step of this is computing x^{2^i} which can be done using the fact, discussed above, that the order of x is 3^{l+1}. Specifically, first we reduce $2^i \bmod 3^{l+1}$ using results about the complexity of integer arithmetic by Hesse et. al. [HAB]. After the exponent is reduced, we show that computing the corresponding power of x is easy.

To prove that $AE =$ *Dlogtime*-uniform TC^0 we also show that $\tilde{\mathbb{F}}_{2^n}$ has an easily computable *dual basis* (as a vector-space over $\mathbb{F}_2$).

The other techniques we use are based on existing algorithms in the literature, e.g. [Kun, Sie, Rei, Ebe]. Our main contribution here is noticing that for some settings of parameters they can be implemented in $AC^0[\oplus]$ and moreover that they give tight results for $AC^0[\oplus]$. We now describe these techniques in more detail.

In the case of arbitrary realizations of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$, the main technical challenge is reducing polynomials modulo $f(x)$. Previous work has addressed this problem and shown how (over arbitrary fields) this can be solved by uniform log-depth circuits (of fan-in 2) [Rei, Ebe], and even by uniform TC^0 circuits [HAB]. The approach that is usually taken is to give a parallel implementation of the Kung-Sievking algorithm [Kun, Sie] to reduce polynomial division to the problem of computing small powers of polynomials. However, this reduction requires summations of poly(n) polynomials, which is why previous results only give implementations in log-depth or by TC^0 circuits. We take the same approach in our Lemma 20; however, we observe that in our setting we may compute these large summations using parity gates. This allows us to implement polynomial division over $\mathbb{F}_2[x]$ in $AC^0[\oplus]$.

Both in our results for $\tilde{\mathbb{F}}_{2^n}$ and for arbitrary realizations of $\mathbb{F}_{2^n}$, we make use of the Discrete Fourier Transform (DFT). This allows us to reduce the problem of multiplication or exponentiation of polynomials to the problem of multiplying or exponentiating field elements in fields of size poly(n) (and these problems are feasible for AC^0 circuits). Eberly [Ebe] and Reif [Rei] have also employed the DFT in their works on performing polynomial arithmetic in log-depth circuits. However, as with polynomial division in $\mathbb{F}_2[x]$, the fact that we are working with polynomials over $\mathbb{F}_2$ allows us to compute the DFT and inverse DFT in uniform $AC^0[\oplus]$ (and not just in log-depth or TC^0).

Other Related Work: Works by Reif [Rei] and Eberly [Ebe] show how basic field arithmetic can be computed by log-depth circuits, and the results of Hesse, Allender and Barrington [HAB] imply that some field arithmetic can be accomplished by uniform TC^0. Indeed, the main result of [HAB] states that integer division can be computed by (uniform) TC^0 circuits, and hence addition and multiplication in the field $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$ can be accomplished (in TC^0) by adding or multiplying elements as integers and then reducing the result modulo p using the division result. Other results from [HAB] imply that uniform TC^0 circuits can compute iterated multiplication in (arbitrary realizations of) $\mathbb{F}_{2^n}$. Some results on the complexity of arithmetic in finite fields of *unbounded* characteristic are given in [SF].

Uniformity. In the previous discussion we refer to uniform circuits for the various problems we consider. When working with restricted circuit classes, such as AC^0, $AC^0[\oplus]$ and TC^0, one must be careful not to allow the machine constructing the circuits to be more powerful than the circuits themselves. Indeed, one of the significant technical contributions of [HAB] is showing that integer division is in uniform TC^0 under a very strong notion of uniformity, namely *Dlogtime*-uniformity [BIS]. *Dlogtime*-uniformity, which is described briefly in section 3, has become the generally-accepted convention for uniformity in constant-depth circuits [BIS, FVB, HAB]. One reason for this is that *Dlogtime*-uniform constant-depth circuits have several elegant characterizations (see, e.g., [BIS]); in fact, our results will prove yet another such characterization, namely *Dlogtime*-uniform $TC^0 = AE$. *Unless otherwise specified, in this work "uniform" always means "Dlogtime-uniform".*

If one is willing to relax the uniformity condition to polynomial-time-uniformity, then some of our results on arithmetic in $\tilde{\mathbb{F}}_{2^n}$ can be proved more easily. For instance, the exponentiation result requires computing $x^{2^i} \in \tilde{\mathbb{F}}_{2^n}$ for a given i. Instead of actually computing x^{2^i} in the circuit, these values could be computed in polynomial time and then hardwired into the circuit. In contrast, in the case of our results in arbitrary realizations of $\mathbb{F}_{2^n}$, we do not know how to improve any of our results, even if we allow non-uniform circuits. If on the other hand, one allows non-uniformity that *depends on the irreducible polynomial* $f(x)$, then one can simplify some the proofs, and can actually improve the exponentiation result to match the parameters that we achieve in $\tilde{\mathbb{F}}_{2^n}$ (by hardwiring the values x^{2^i} into the circuit, as above).

Organization. This paper is organized as follows. In Section 2 we formally state our results. In Section 3 we discuss some preliminaries. In Sections 4-7 we give the proofs of our results. In Section 8 we discuss some open problems.

2 Our Results

In this section we formally state our results. In Section 2.1 we discuss our results in the specific case where n is of the form $n = 2 \cdot 3^l$, and $\mathbb{F}_{2^n}$ is realized as $\mathbb{F}_2[x]/(x^{2\cdot 3^l} + x^{3^l} + 1)$, i.e. using the explicit irreducible polynomial $x^{2\cdot 3^l} + x^{3^l} + 1 \in \mathbb{F}_2[x]$. In Section 2.2 we discuss our results in realizations of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$ for an *arbitrary* irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ that is given as part of the input. Then we discuss applications of our results. In Section 2.3 we prove that uniform $TC^0 = AE$. In Section 2.4 we exhibit k-wise independent and ϵ-biased generators explicitly computable in uniform $AC^0[\oplus]$ and TC^0.

2.1 Field Arithmetic in $\tilde{\mathbb{F}}_{2^n}$

Below we summarize our main results concerning arithmetic in the field $\tilde{\mathbb{F}}_{2^n}$, defined below.

Fact 1 ([vL], Theorem 1.1.28)**.** *For all integers $l \geq 0$, the polynomial $x^{2\cdot 3^l} + x^{3^l} + 1 \in \mathbb{F}_2[x]$ is irreducible.*

Definition 2. *For n of the form $n = 2 \cdot 3^l$, we define $\tilde{\mathbb{F}}_{2^n}$ to be the specific realization of $\mathbb{F}_{2^n}$ given by*

$$\tilde{\mathbb{F}}_{2^n} \stackrel{\text{def}}{=} \mathbb{F}_2[x]/(x^{2\cdot 3^l} + x^{3^l} + 1).$$

The next theorem states our results about field arithmetic over $\tilde{\mathbb{F}}_{2^n}$ in uniform TC^0. The first item follows from results of Hesse, Allender and Barrington [HAB]; nonetheless, we state it for the sake of comparison with our other results.

Theorem 3. *Let $n = 2 \cdot 3^l$. There exist uniform TC^0 circuits of size* $\mathrm{poly}(n, t)$ *that perform the following:*

1. *[HAB] Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

In particular, uniform TC^0 circuits of polynomial size are capable of performing iterated multiplication and exponentiation in $\tilde{\mathbb{F}}_{2^n}$ that match the parameters that can be achieved by standard unbounded-depth circuits.

The next theorem states our results about field arithmetic over $\tilde{\mathbb{F}}_{2^n}$ in uniform $AC^0[\oplus]$.

Theorem 4. *Let $n = 2 \cdot 3^l$. Then, for every constant $\epsilon > 0$, there exist uniform $AC^0[\oplus]$ circuits of size* $\mathrm{poly}(n, 2^{t^\epsilon})$ *that perform the following:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*

2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

While these parameters are considerably worse than for TC^0 circuits, they are tight:

Theorem 5. *For every constant d there is $\epsilon > 0$ such that, for sufficiently large t and $n = 2 \cdot 3^l$, the following cannot be computed by (nonuniform) $AC^0[\oplus]$ circuits of depth d and size $2^{2^{\epsilon n}} \cdot 2^{t^\epsilon}$:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*

2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

In fact, Item (1) in the above negative result holds for any sufficiently large field (i.e. not only $\tilde{\mathbb{F}}_{2^n}$); and Item (2) holds for fields of a variety of different sizes. Both of these generalizations will be apparent from the proof.

By "scaling down" Theorem 3 (as described in Section 3) we obtain the following:

Corollary 6. *Let $n = 2 \cdot 3^l$. Then, for every constant $\epsilon > 0$, there exist uniform AC^0 circuits of size $\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ that perform the following:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*

2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

This improves upon a theorem of Agrawal et al. [AAI+] showing that field exponentiation is computable by uniform AC^0 circuits of size $\mathrm{poly}(2^n, 2^t)$ (as opposed to $\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ in our result). Corollary 6 is also tight for many settings of parameters (see Theorem 5).

2.2 Field Arithmetic in Arbitrary Realizations of $\mathbb{F}_{2^n}$

As noted above, one of the advantages of working with the field $\tilde{\mathbb{F}}_{2^n}$ is that we achieve tight results for TC^0, $AC^0[\oplus]$ and AC^0. However, the use of $\tilde{\mathbb{F}}_{2^n}$ requires that $n = 2 \cdot 3^l$, and thus does not allow for the construction of $\mathbb{F}_{2^n}$ for all n; moreover some applications may require field computations in a specific field $\mathbb{F}_2[x]/(f(x))$ for some given irreducible polynomial $f(x)$ other than $x^{2 \cdot 3^l} + x^{3^l} + 1$. Thus we are led to study the complexity of arithmetic in the ring $\mathbb{F}_2[x]/(f(x))$ where the polynomial $f(x) \in \mathbb{F}_2[x]$ is *given as part of the input*. If, in addition, we have the promise that $f(x)$ is irreducible, then this corresponds to arithmetic in the field $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[x]/(f(x))$.

Theorem 7.

1. *For every constant $\epsilon > 0$, there exist uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$ that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n and $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_2[x]/(f(x))$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_2[x]/(f(x))$.*

2. *There exist uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$ that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n, $\alpha \in \mathbb{F}_2[x]/(f(x))$ and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_2[x]/(f(x))$.*

Since Item 1 of Theorem 5 actually holds for any realization of $\mathbb{F}_{2^n}$, and not just for $\tilde{\mathbb{F}}_{2^n}$ (as noted in the proof), Item 1 of Theorem 7 is tight.

Unlike Item 2 in Theorem 4, Exponentiation now requires size $\mathrm{poly}(n, 2^t)$, instead of $\mathrm{poly}(n, 2^{t^\epsilon})$. We do not know how to improve this to size $\mathrm{poly}(n, 2^{o(t)})$, even for TC^0 circuits. On the other hand, we show that testing irreducibility of a given $\mathbb{F}_2[x]$ polynomial is $AC^0[\oplus]$ reducible to computing exponentiation modulo a given irreducible polynomial.

Theorem 8. *The problem of determining whether a given polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n is irreducible, is* $\mathrm{poly}(n)$*-size $AC^0[\oplus]$-reducible to the following problem: Given an irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n, compute the conjugates $x, x^2, x^{2^2}, \ldots, x^{2^{n-1}}$* $\pmod{f(x)}$.

2.3 *AE* = *Dlogtime* uniform TC^0

Frandsen, Valence and Barrington [FVB] study the relationship between uniform TC^0 and the class *AE* of functions computable by certain arithmetic expressions (defined below). Remarkably, they show that *Dlogtime*-uniform TC^0 is contained in *AE*. Conversely, they show that *AE* is contained in *P*-uniform TC^0, but they leave open whether the inclusion holds for *Dlogtime* uniformity. We show that *AE* is in fact contained in *Dlogtime*-uniform TC^0, thus proving that *AE* = *Dlogtime*-uniform TC^0. (All these inclusions between classes hold in a certain technical sense that is made clear below.)

We now briefly review the definition of *AE* and then state our results.

Definition 9 ([FVB]). *Let I be an infinite set of formal indices. The set of formal* arithmetic expressions *is defined as follows. The basic expressions are x (we think of this as the field element x), and* Input *(we think of this as the input field element). If e, e' are expressions (possibly containing the unbound index $i \in I$), then we may form new composite expressions $\sum_{i=1}^{u} e, \prod_{i=1}^{u} e, e + e', e \cdot e', e^{2^i}$, where $i \in I$ and u is either an index, i.e. $u \in I$, or is any polynomial in n (we think of n as the input length).*

An arithmetic expression is well-formed *if all indices are bound and they are bound in a semantically sound way (we omit details).*

We associate to every well-formed arithmetic expression e a family of functions $f_n^e : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$, for every n of the form $n = 2 \cdot 3^l$ (note that all computations are performed over the field $\tilde{\mathbb{F}}_{2^n}$).

The complexity class AE consists of those families of functions $f_n : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ that are described by arithmetic expressions (for every n of the form $n = 2 \cdot 3^l$).

For example, the *trace* function, $\mathrm{tr} : \tilde{\mathbb{F}}_{2^n} \to \mathbb{F}_2 \subseteq \tilde{\mathbb{F}}_{2^n}$, defined by $\mathrm{tr}(\mathrm{Input}) \stackrel{\mathrm{def}}{=} \sum_{i=0}^{n-1} \mathrm{Input}^{2^i}$, is in *AE*.

Theorem 10. *AE = Dlogtime−uniform TC^0 in the following sense:*

Let $f : \{0,1\}^n \to \{0,1\}^n$ be in Dlogtime-uniform TC^0. Then there is $f' : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ in AE such that for every n of the form $2 \cdot 3^l$, and for every x of length n, $f(x) = f'(x)$.

Conversely, let $f : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ be in AE. Then there is $f' : \{0,1\}^n \to \{0,1\}^n$ in Dlogtime-uniform TC^0 such that for every n of the form $2 \cdot 3^l$, and for every x of length n, $f(x) = f'(x)$.

Remark 11. *Our definition of arithmetic expressions is slightly different from the definition in [FVB], which we denote [FVB]-arithmetic expressions. We now argue that our definition only makes our results hold in a sense that is stronger than that in [FVB]. From a syntactical point of view, [FVB]-arithmetic expressions may use a special element g, which intuitively corresponds to our x. Then, from a semantical point of view, to define the class of functions [FVB]-AE computed by [FVB]-arithmetic expressions, they fix a certain* representation *of finite fields, which in particular fixes the element g. Their inclusion "uniform TC^0 is contained in [FVB]-AE" only holds* after a particular representation has been fixed. *Roughly, the representation fixes g such that $g, g^2, g^4, \ldots, g^{2^{n-1}}$ is a self-dual normal basis for the field. We note that computing such a g requires a lot of machinery, and it is not known (to the best of our knowledge) how to do it in, say, Dlogtime-uniform TC^0. On the other hand, in our results we work over the standard representation of finite fields modulo the irreducible polynomial $x^{2\cdot 3^l} + x^{3^l} + 1$, and we set $g = x$; it is easy to see that our representation is easily computable.*

Remark 12. *It is perhaps unsatisfactory that Theorem 10 only holds for certain input lengths. However, even the results in [FVB] only hold for certain input lengths, though they are more "dense" than ours.*

2.4 k-wise and ϵ-biased generators

We use our results on computing field operations to give constant-depth implementations of certain "pseudorandom" generators, namely k-wise independent and ϵ-biased generators. The complexity of these generators is also studied by Gutfreund and Viola [GV]. Our results will complement some of the results in [GV] (see the remark at the end of this section).

We now give some definitions and then we state our results. We say a generator $G : \{0,1\}^s \to \{0,1\}^m$ is *explicitly computable* in uniform TC^0 (resp., $AC^0[\oplus]$) if there is a uniform TC^0 (resp., $AC^0[\oplus]$) circuit of size $\mathrm{poly}(s, \log m)$ that, given $x \in \{0,1\}^s$ and $i \le m$, computes the i-th output bit of $G(x)$. We now define k-wise independent and ϵ-biased generators. We refer the reader to the works [CG, ABI, NN, AGHP] and the book by Goldreich [Gol] for background and discussion of these generators. Denote the set $\{1, \ldots, m\}$ by $[m]$. For $I \subseteq [m]$ and $G(x) \in \{0,1\}^m$ we denote by $G(x)|_I \in \{0,1\}^{|I|}$ the projection of $G(x)$ on the bits specified by I.

Definition 13. *Let $G : \{0,1\}^s \to \{0,1\}^m$ be a generator.*

- *G is* k-wise independent *if for every $M : \{0,1\}^k \to \{0,1\}$ and $I \subseteq [m]$ such that $|I| = k$: $\Pr_{y \in \{0,1\}^k}[M(y) = 1] = \Pr_{x \in \{0,1\}^s}[M(G(x)|_I) = 1]$.*
- *G is* ϵ-biased *if for every $\emptyset \ne I \subseteq [m]$: $\left| \Pr_{x \in \{0,1\}^s}[\bigoplus_{i \in I} G(x)_i = 0] - \frac{1}{2} \right| \le \epsilon$.*

Using our results on field operations we obtain the following results. Note both constructions are optimal up to constant factors (cf. [CGH+, AGHP]).

Theorem 14.

1. *For every k and m there is a k-wise independent generator $G : \{0,1\}^s \to \{0,1\}^m$, with $s = O(k \log m)$ that is explicitly computable by uniform $AC^0[\oplus]$ circuits of size* $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.

2. *For every ϵ and m, there is an ϵ-biased generator $G : \{0,1\}^s \to \{0,1\}^m$ with $s = O(\log m + \log(1/\epsilon))$ that is explicitly computable by uniform TC^0 circuits of size* $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.

Remark 15. *A previous and different construction of k-wise independent generators in [GV] matches (up to constant factors) Item 1 in Theorem 14 for the special case $k = O(1)$. The construction in Item 1 in Theorem 14 improves on the construction in [GV] for $k = \omega(1)$. Also, in [GV] they exhibit a construction of ϵ-biased generators computable by uniform $AC^0[\oplus]$ circuits (while the construction in Item 2 in Theorem 14 uses TC^0 circuits). However, the construction in [GV] has worse dependence on ϵ.*

3 Background

In this section we give some background about constant-depth circuits.

Constant-Depth Circuits: The three main complexity classes that we study in this work are as follows:

- AC^0 : The class of circuits having AND and OR gates of unbounded fan-in, NOT gates and depth $O(1)$.
- $AC^0[\oplus]$: The class of circuits having AND, OR and XOR gates of unbounded fan-in, NOT gates and depth $O(1)$.
- TC^0 : The class of circuits having AND, OR and MAJORITY gates of unbounded fan-in, NOT gates and depth $O(1)$.

We will routinely abuse language and refer to *functions* f that can be computed by AC^0 (respectively $AC^0[\oplus]$ and TC^0) circuits (of a certain size s); by this we simply mean that, given x and $i \leq |x|$, computing the i-th bit of $f(x)$ can be performed by AC^0 (resp. $AC^0[\oplus]$ and TC^0) circuits (of size s).

Uniformity: When referring to the *uniformity* of a family of circuits, we mean the complexity of the *uniform* algorithm that "constructs" the n-th circuit, given input n. As mentioned in the introduction, when working with constant-depth circuits, the issue of uniformity can be a delicate one. Nonetheless, there is a single notion of uniformity that is generally accepted to be the most appropriate for these classes, namely *Dlogtime*-uniformity. A detailed description of *Dlogtime*-uniformity can be found in [BIS] (see also [Vol]); below we give a more informal description.

A family of circuits $\{C_n\}_{n=1}^{\infty}$ of size $s(n)$ is said to be *Dlogtime*-uniform if there exists a random-access Turing machine that:

- On input n and $i \leq s$ determines in time $O(\log n + \log i)$ the type of gate i (e.g., AND, OR, NOT, XOR, MAJ) in the circuit C_n.
- On input n and $i, j \leq s$ decides in time $O(\log n + \log i)$ whether the output of gate i is joined to the input of gate j in the circuit C_n.

This restrictive notion of uniformity is more than adequate to ensure that the class of functions computed by uniform poly(n)-size TC^0 circuits is contained in logarithmic space.

Scaling Down TC^0: It is well-known that uniform poly(n)-size AC^0 circuits can compute the MAJORITY function on polylog(n) bits. In particular, this means that any problem that is solved by uniform poly(n)-size TC^0 circuits on inputs of length n can, on inputs of length polylog(n), be solved by uniform poly(n)-size AC^0 circuits that simulate the MAJORITY gates of the TC^0 circuits. We will use these facts frequently and will often simply refer to "scaling down" a given family of uniform TC^0 circuits to obtain the appropriate uniform AC^0 circuits. For example, since both iterated integer multiplication of n n-bit numbers and division of n-bit numbers are in uniform poly(n)-size TC^0 [HAB], we have the following lemma about performing these operations by uniform AC^0 circuits.

Lemma 16 ([HAB], Theorem 5.1)**.** *For every constant $c > 1$, the following can be computed by Dlogtime-uniform AC^0 circuits of size* poly(n)*:*

- *Given integers $a_1, a_2, \ldots, a_{\log^c n}$, each of length at most $\log^c n$ bits, compute $\prod_{i \leq \log^c n} a_i$.*
- *Given integers a, b, each of length at most $\log^c n$ bits, compute $\lfloor a/b \rfloor$.*

4 Arithmetic in $\tilde{\mathbb{F}}_{2^n}$

In this section we prove our results about field arithmetic in the field $\tilde{\mathbb{F}}_{2^n} = \mathbb{F}_2[x]/(x^{2\cdot 3^l} + x^{3^l} + 1)$.

One useful property of $\tilde{\mathbb{F}}_{2^n}$ is that the order of $x \in \tilde{\mathbb{F}}_{2^n}$ is small, specifically it is $3^{l+1} = O(n)$. (A priori, it could have been as large as $2^n - 1$.)

Fact 17. *The order of $x \in \tilde{\mathbb{F}}_{2^n}$ is 3^{l+1}.*

Proof. First we show that $x^{3^{l+1}} \equiv 1 \pmod{x^{2\cdot 3^l} + x^{3^l} + 1}$. This holds because

$$x^{3^{l+1}} = x^{2\cdot 3^l} \cdot x^{3^l} \equiv \left(x^{3^l} + 1\right) \cdot x^{3^l} = \left(x^{2\cdot 3^l} + x^{3^l}\right) \equiv 1 \pmod{x^{2\cdot 3^l} + x^{3^l} + 1}.$$

Thus the order of x has to divide 3^{l+1}. Noting that $x^{3^l} \not\equiv 1 \pmod{x^{2\cdot 3^l} + x^{3^l} + 1}$, the result follows. $\square$

One way in which Fact 17 is useful is that it allows us to easily reduce a given polynomial modulo $x^{2\cdot 3^l} + x^{3^l} + 1$.

Lemma 18. *Let $n = 2 \cdot 3^l$. Then there exist uniform $AC^0[\oplus]$ circuits of size* poly(n, d) *that, on input $g(x) \in \mathbb{F}_2[x]$ of degree at most d, compute $g(x) \pmod{x^{2\cdot 3^l} + x^{3^l} + 1}$.*

We will ultimately prove a much more general statement (Lemma 20), namely that one can reduce $g(x) \in \mathbb{F}_2[x]$ modulo any given polynomial (and not only $x^{2\cdot 3^l}+x^{3^l}+1$). However, the proof of this more general result is also much more complicated, and so we now give an easier proof for the special case of reducing modulo $x^{2\cdot 3^l}+x^{3^l}+1$.

Proof of Lemma 18. First we show that, given $k \le d$, we can compute $x^d \in \tilde{\mathbb{F}}_{2^n}$ by uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n,d)$. The circuit will first use the fact that division of integers of $O(\log n + \log d)$ bits is computable by uniform AC^0 circuits of size $\mathrm{poly}(n,d)$ (see Lemma 16) to reduce k modulo 3^{l+1} and obtain $0 \le k' < 3^{l+1}$ such that $k' \equiv k \pmod{3^{l+1}}$. By Fact 17, $x^{k'} \equiv x^k \pmod{x^{2\cdot 3^l}+x^{3^l}+1}$. Clearly, if $k' < 2\cdot 3^l$, then the result is simply $x^{k'}$. On the other hand, if $2\cdot 3^l \le k' < 3\cdot 3^l$, then $x^{k'} \equiv x^{k'-3^l}+x^{k'-2\cdot 3^l}$.

It follows that any given polynomial $g(x) \in \mathbb{F}_2[x]$ of degree d can be reduced modulo $x^{2\cdot 3^l}+x^{3^l}+1$ by uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n,d)$; indeed, the circuit needs only reduce each term x^i of $g(x)$ modulo $x^{2\cdot 3^l}+x^{3^l}+1$, and then compute the sum of all the terms (using parities of d bits). $\square$

A crucial way in which Fact 17 is useful is that it allows us to compute high powers, α^k, of field elements $\alpha \in \tilde{\mathbb{F}}_{2^n}$, in the special case when k is a power of 2.

Lemma 19. *Let n be of the form $n = 2\cdot 3^l$. Then there exist uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n,i)$ that, on input $\alpha \in \tilde{\mathbb{F}}_{2^n}$, computes $\alpha^{2^i} \in \tilde{\mathbb{F}}_{2^n}$.*

Proof. Since $\alpha^{2^n} = \alpha$ for all $\alpha \in \tilde{\mathbb{F}}_{2^n}$, we first reduce i modulo n. This can be accomplished by uniform AC^0 circuits of size $\mathrm{poly}(n,i)$ by Lemma 16. From this point on we assume w.l.o.g. that $i \le n$.

Let $\alpha(x) \in \mathbb{F}_2[x]$ be the polynomial representing α. Thus, it suffices to compute $\alpha(x)^{2^i} \equiv \alpha(x^{2^i})$ modulo $x^{2\cdot 3^l}+x^{3^l}+1$. In particular, it suffices to compute $x^{h\cdot 2^i}$ in $\tilde{\mathbb{F}}_{2^n}$ for every $h, i \le n$, since then we can then compute $\alpha(x^{2^i})$ by simply summing the appropriate terms.

We show that each $x^{h\cdot 2^i} \in \tilde{\mathbb{F}}_{2^n}$ can actually be computed in uniform AC^0: Recall that the order of x modulo $f(x) = x^{2\cdot 3^l}+x^{3^l}+1$ is 3^{l+1} by Fact 17. Therefore it suffices to be able to reduce $h\cdot 2^i$ modulo 3^{l+1}, and then we can apply Lemma 18. The only hard part of this is reducing 2^i modulo 3^{l+1}, since we can then multiply the result by h and divide by 3^{l+1} using Lemma 16.

We now show how to reduce 2^i modulo 3^{l+1}. By the binomial theorem,

$$2^i \equiv (3-1)^i \equiv \sum_{j=0}^{i} \binom{i}{j} 3^j (-1)^{i-j} \bmod 3^{l+1}.$$

Noting that all the terms of this sum vanish for $j \ge l+1$ (thanks to the 3^j factor), this sum is actually congruent to

$$\sum_{j=0}^{l} \binom{i}{j} 3^j (-1)^{i-j} \bmod 3^{l+1} \equiv \sum_{j=0}^{l} \frac{i(i-1)\cdots(i-j+1)}{j(j-1)\cdots 1} \cdot 3^j (-1)^{i-j} \bmod 3^{l+1}.$$

Since $l = O(\log n)$ and $|i| = O(\log n)$, we can compute, for every j, $\frac{i(i-1)\cdots(i-j+1)}{j(j-1)\cdots 1}$ by using an iterated product (of $O(\log n)$ integers of $O(\log n)$ bits) for the numerator and denominator, and then performing a division of the results (i.e., of integers having polylog(n) bits); both of these can be done by uniform AC^0 circuits of size poly(n) by Lemma 16. Additionally, the 3^j term can be computed (using iterated multiplication, say), and the $(-1)^{(i-j)}$ is easy to compute.

Finally, since $l = O(\log n)$, the sum can be computed by uniform AC^0 circuits of size poly(n) using an iterated sum of integers having polylog(n) bits. Clearly, the result only has polylog(n) bits, and so we may reduce modulo 3^{l+1} one last time to find $2^i \pmod{3^{l+1}}$. $\square$

We note that the above lemma is easier to prove if one is willing to settle for either TC^0 circuits (as opposed to $AC^0[\oplus]$) or size poly$(n, 2^{i^\epsilon})$ (as opposed to poly(n, i)), which is all that is needed for our other theorems. Nonetheless, we prefer to state and prove this single more general result.

We now prove our main theorems about field operations in $\tilde{\mathbb{F}}_{2^n}$. We repeat the theorem statements for the reader's convenience.

Theorem (3, restated)**.** *Let $n = 2 \cdot 3^l$. There exist uniform TC^0 circuits of size* poly(n, t) *that perform the following:*

1. *[HAB] Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

Proof of Theorem 3. (1) Each field element α_i is represented by a polynomial $\alpha_i(x) \in \mathbb{F}_2[x]$. For the moment, we will actually consider the polynomials $\alpha_i(x)$ as polynomials $\alpha'_i(x)$ over the integers, i.e. as polynomials with coefficients in $\{0, 1\} \subset \mathbb{Z}$. It is proved in [HAB] that the product of t polynomials of degree n over $\mathbb{Z}$ can be computed by uniform TC^0 circuits of size poly(n, t). Thus, the product $A'(x) \stackrel{\text{def}}{=} \alpha'_1(x) \cdots \alpha'_t(x)$ can be computed by uniform TC^0 circuits of size poly(n, t). Clearly, $A(x) \stackrel{\text{def}}{=} \alpha_1(x) \cdots \alpha_t(x) \equiv A'(x) \pmod 2$, and so it remains to reduce $A(x)$ modulo $x^{2\cdot 3^l} + x^{3^l} + 1$; however, this follows from Lemma 18 (or by results in [HAB]).

(2) We reduce the computation of α^k to the computation of a product $\alpha_1 \cdot \alpha_2 \cdots \alpha_t$ and apply Part (1). The integer $k = \sum_{i=0}^{t-1} k_i 2^i$ is given in binary, as $k_{t-1} \cdots k_1 k_0$, $k_i \in \{0, 1\}$, and thus

$$\alpha^k = \alpha^{\sum_i k_i 2^i} = \left(\alpha^{2^0}\right)^{k_0} \cdot \left(\alpha^{2^1}\right)^{k_1} \cdots \left(\alpha^{2^{t-1}}\right)^{k_{t-1}}.$$

Hence, to apply part(1), it suffices to show that each term $\left(\alpha^{2^i}\right)^{k_i}$ can be computed by TC^0 circuits of size poly(n, t). Computing α^{2^i} follows from Lemma 19 and, since $k_i \in \{0, 1\}$, the exponentiation by k_i is easy. $\square$

Theorem (4, restated)**.** *Let $n = 2 \cdot 3^l$. Then, for every constant $\epsilon > 0$, there exist uniform $AC^0[\oplus]$ circuits of size* poly$(n, 2^{t^\epsilon})$ *that perform the following:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

Proof of Theorem 4. (1) The idea is to reduce the problem to computing iterated multiplication over an exponentially smaller field $\mathbb{F}'$ via the Discrete Fourier Transform. We can compute iterated multiplication over $\mathbb{F}'$ in uniform AC^0 by scaling down the TC^0 result (Theorem 3, Item 1). Details follow.

Consider an iterated multiplication instance $(\alpha_1, \ldots, \alpha_t)$. Recalling that $\tilde{\mathbb{F}}_{2^n} = \mathbb{F}_2[x]/(f(x))$ (where $f(x) = x^{2\cdot 3^l} + x^{3^l} + 1$ is the irreducible polynomial) we may view each α_i as a polynomial $\alpha_i(x)$ of degree at most $n-1$ in $\mathbb{F}_2[x]$. To compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$ it will then suffice to compute the *polynomial* product $A(x) \stackrel{\text{def}}{=} \alpha_1(x)\alpha_2(x)\cdots\alpha_t(x) \in \mathbb{F}_2[x]$, and then apply Lemma 18 to reduce this polynomial modulo $f(x)$.

Let $m \in \{\log n + \log t, \ldots, 3(\log n + \log t)\}$ be of the form $m = 2 \cdot 3^{l'}$ for some l' (such an m can be found by uniform AC^0 circuits of size $\text{poly}(2^m) = \text{poly}(n,t)$), and consider the field $\tilde{\mathbb{F}}_{2^m}$. To compute the polynomial product $A(x)$ we will first evaluate each polynomial $\alpha_i(x)$ at every element $\gamma_i, 1 \le i < 2^m$, of $\tilde{\mathbb{F}}_{2^m}^{\times}$. Next, we will compute $A(\gamma_1), \ldots, A(\gamma_{2^m-1})$ by using iterated product over the field $\tilde{\mathbb{F}}_{2^m}$ to compute $A(\gamma_i) = \alpha_1(\gamma_i)\cdots\alpha_t(\gamma_i)$. Then, since $A(x)$ has degree at most $(n-1)\cdot t < 2^{\log n + \log t} - 1$, the values $A(\gamma_1), \ldots, A(\gamma_{2^m-1})$ uniquely determine $A(x)$, and we will show how to interpolate in uniform $AC^0[\oplus]$ to recover $A(x)$.

To accomplish these steps we will use the Discrete Fourier Transform matrix. That is, let $g \in \tilde{\mathbb{F}}_{2^m}$ be a generator of $\tilde{\mathbb{F}}_{2^m}$ and note that such a generator can be found in uniform AC^0 by brute force (by computing exponentiation over $\tilde{\mathbb{F}}_{2^m}$, which can be done by scaling down the TC^0 result, i.e. Theorem 3, Item 2. Alternatively, one can use Theorem 3.2 in [AAI$^+$]). Now define the matrix $D = (d_{i,j})_{0 \le i,j \le 2^m-2}$, where $d_{i,j} \stackrel{\text{def}}{=} g^{i\cdot j}$ and note that $D^{-1} = (d_{i,j}^{-1})_{0 \le i,j \le 2^m-2}$. If we view α_i as a (2^m-1)-dimensional vector $\vec{\alpha_i} = (\alpha_i^{(0)}, \ldots, \alpha_i^{(2^m-2)}) \in \mathbb{F}_2^{2^m-1}$, where $\alpha_i^{(j)}$ is the coefficient of x^j in $\alpha_i(x)$ (either 0 or 1), then $D\vec{\alpha_i} = (\alpha_i(g^0), \alpha_i(g^1), \ldots, \alpha_i(g^{2^m-2}))$. The matrix-vector product $D\vec{\alpha_i}$ can be computed by uniform $AC^0[\oplus]$ circuits of size $\text{poly}(2^m)$ because it only involves computing parities (of fan-in $2^m - 1$) and multiplications in the field $\tilde{\mathbb{F}}_{2^m}$, which we can do by uniform AC^0 circuits of size $\text{poly}(2^m)$ by scaling down the TC^0 result, i.e. Theorem 3, Item (1).

Once the matrix-vector products $D\vec{\alpha_1}, \ldots, D\vec{\alpha_t}$ have been computed, the resulting vectors can be multiplied component-wise (using the scaled-down version of Theorem 3, item 1) to obtain the vector $\hat{A} = (A(g^0), \ldots, A(g^{2^m-2}))$. Next, note that $\vec{A} = D^{-1}\hat{A}$ can also be computed in $AC^0[\oplus]$, just as $D\alpha_i$ was computed above, allowing us to recover the product polynomial $A(x)$.

Finally, by Lemma 18, $A(x)$ can be reduced modulo the irreducible polynomial $f(x)$ to obtain the field element $A = \alpha_1 \cdots \alpha_t$.

(2) As in the uniform TC^0 case we can reduce this problem to the product of t field elements. Specifically, the reduction in Item 3 in Theorem 3 needs to compute α^{2^i} for $i \le t$. These can be computed in uniform $AC^0[\oplus]$ by Lemma 19. For the iterated product we use the previous item. □

Theorem (5, restated)**.** *For every constant d there is $\epsilon > 0$ such that, for sufficiently large t and $n = 2 \cdot 3^l$, the following cannot be computed by (nonuniform) $AC^0[\oplus]$ circuits of depth d and size $2^{2^{\epsilon n}} \cdot 2^{t^\epsilon}$:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

Proof of Theorem 5. (1) We reduce MAJORITY on t bits to computing $\alpha_1 \cdot \alpha_2 \cdots \alpha_t$ for given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, where $n \in \{\log(t+1), \ldots, 3\log(t+1)\}$ is of the form $n = 2 \cdot 3^l$ for some l. Since by a result of Razborov [Raz] and Smolensky [Smo] we know that for every constant d there is a constant $\epsilon > 0$ such that MAJORITY on t bits cannot be computed by $AC^0[\oplus]$ circuits of depth d and size 2^{t^ϵ}, the result follows. We now describe the reduction. Let $g \in \mathbb{F}^\times$ be a generator. Given a MAJORITY instance $x = w_1, w_2, \ldots, w_t$, consider the following instance of iterated multiplication: $\alpha_1, \alpha_2, \ldots, \alpha_t$, where $\alpha_i \stackrel{\text{def}}{=} g \in \mathbb{F}$ if $w_i = 1$, and $\alpha_i \stackrel{\text{def}}{=} 1 \in \mathbb{F}$ if $w_i = 0$. It is easy to see that $\alpha_1 \cdot \alpha_2 \cdots \alpha_t = g^j \in \mathbb{F}$ where $j = \sum_i w_i$. We can decide majority simply by checking whether $j \geq t/2$; this last step can be accomplished by a simple look-up in a (nonuniform) table of size $\mathrm{poly}(n, t)$.

(2) We reduce MAJORITY on t bits to computing $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$ for $|k| = O(t \log t)$ and $n = O(\log t)$. Since by a result of Razborov [Raz] and Smolensky [Smo] we know that for every constant d there is a constant $\epsilon > 0$ such that MAJORITY on t bits cannot be computed by $AC^0[\oplus]$ circuits of depth d and size 2^{t^ϵ}, the result follows. Let $l \stackrel{\text{def}}{=} \lceil \log_3 \log_2(t+1) \rceil$ and $m \stackrel{\text{def}}{=} 3^l$. Set $n \stackrel{\text{def}}{=} 2 \cdot m$ and consider the field $\tilde{\mathbb{F}}_{2^n}$. Note that since $|\tilde{\mathbb{F}}_{2^n}^\times| = 2^n - 1 = (2^m - 1)(2^m + 1)$, there is an element $\alpha \in \tilde{\mathbb{F}}_{2^n}$ of order $(2^m - 1)$.

The reduction works as follows. From the MAJORITY instance $z = z_0 z_1 \ldots z_{t-1}$ construct an integer k with binary representation

$$k = z_{t-1} \underbrace{00\cdots0}_{m-1 \text{ zeros}} z_{t-2} \underbrace{00\cdots0}_{m-1 \text{ zeros}} z_{t-3} \ldots z_1 \underbrace{00\cdots0}_{m-1 \text{ zeros}} z_0 = \sum_{i=0}^{t-1} z_i (2^m)^i.$$

Now observe that $k = \sum_i z_i (2^m)^i \equiv \sum_i z_i \pmod{2^m - 1}$. Therefore $\alpha^k = \alpha^{\sum_i z_i}$; since $t < 2^m - 1$, this uniquely determines $\sum_i z_i$, and so MAJORITY can now be decided via look-up in a (nonuniform) table of size $\mathrm{poly}(n, t)$. $\square$

5 Arithmetic in Other Realizations of $\mathbb{F}_{2^n}$

In this section we prove Theorem 7. An important difference between this setting and the case of field operations in $\tilde{\mathbb{F}}_{2^n}$ is that we must now be able to reduce a polynomial $g(x) \in \mathbb{F}_2[x]$ modulo an arbitrary given polynomial $f(x) \in \mathbb{F}_2[x]$, and not only modulo $x^{2 \cdot 3^l} + x^{3^l} + 1$. The next lemma states that polynomial division in $\mathbb{F}_2[x]$ can be computed in uniform $AC^0[\oplus]$.

Lemma 20. *There exist uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n, d)$ that, on input polynomials $f(x), g(x) \in \mathbb{F}_2[x]$ where $\deg(f) = n$ and $\deg(g) \leq d$, computes the unique polynomials $q(x), r(x) \in \mathbb{F}_2[x]$, such that $g(x) = q(x)f(x) + r(x)$, where $\deg(q) = \deg(g) - n$ and $\deg(r) < n$.*

The approach for proving Lemma 20 is to implement, in constant-depth, the Kung-Sieveking [Kun, Sie] algorithm, which reduces the problem of polynomial division to the problem of computing small powers of polynomials. A similar approach has been employed by Reif [Rei] and Eberly [Ebe] in constructing log-depth circuits for polynomial division. The essential difference here is the observation that log-depth is only required to compute sums of $\mathrm{poly}(n)$ polynomials and, in our setting, we may instead use parity gates to accomplish such large summations in constant depth.

Before proving Lemma 20, we show how to compute small powers of polynomials in $AC^0[\oplus]$, as this is an essential component of the proof.

Lemma 21. *There exist uniform $AC^0[\oplus]$ circuits of size* $\mathrm{poly}(n, 2^t)$ *that, on input $s(x) \in \mathbb{F}_2[x]$ of degree n and a t-bit integer k, compute the polynomial $s(x)^k$.*

Proof. As in the proof of Theorem 4, part 1, we also use the Discrete Fourier Transform to compute $s(x)^k$.

In particular, let $m \in \{\log n+t, \ldots, 3(\log n+t)\}$ be of the form $m = 2\cdot 3^{l'}$ for some l' (such an m can be found by uniform AC^0 circuits of size $\mathrm{poly}(2^m) = \mathrm{poly}(n, 2^t)$), and consider the field $\tilde{\mathbb{F}}_{2^m}$. To compute the polynomial power $s(x)^k$, we first evaluate the polynomial $s(x)$ at every element $\gamma_i, 1 \le i < 2^m$, of $\tilde{\mathbb{F}}_{2^m}^{\times}$, just as in the proof of Theorem 4, part 1. Next, we compute $s(\gamma_1)^k, \ldots, s(\gamma_{2^m-1})^k$ by using exponentiation in the field $\tilde{\mathbb{F}}_{2^m}$ (which follows from part 2 of Corollary 6; alternatively, one can use Theorem 3.2 from [AAI$^+$].) Then, by the choice of m, the values $s(\gamma_1)^k, \ldots, s(\gamma_{2^m-1})^k$ uniquely determine $s(x)^k$, and thus we can interpolate in uniform $AC^0[\oplus]$ to recover $A(x)$, using the inverse Fourier Transform just as in the proof of Theorem 4, part 1. $\square$

Proof of Lemma 20. Denote the degree of $g(x)$ by $m \le d$, and throughout we write $f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$ and $g(x) = x^m + b_{m-1}x^{n-1} + \cdots + b_0$ for $a_i, b_i \in \mathbb{F}_2$. The algorithm will proceed as follows:

(1) Construct $f_R(x) \stackrel{\text{def}}{=} a_0x^n + \cdots + a_{n-1}x + 1$ by reversing the coefficients of $f(x)$.

(2) Construct $g_R(x) \stackrel{\text{def}}{=} b_0x^m + \cdots + b_{m-1}x + 1$ by reversing the coefficients of $g(x)$.

(3) Let $\tilde{f}_R(x) \stackrel{\text{def}}{=} 1 + (1 - f_R(x)) + (1 - f_R(x))^2 + \cdots + (1 - f_R(x))^{m-n}$. (Note that $\tilde{f}_R(x)$ is simply a truncation of the power series $f_R(x)^{-1} = 1 + (1 - f_R(x)) + (1 - f_R(x))^2 + \cdots$.)

(4) Compute $h(x) \stackrel{\text{def}}{=} \tilde{f}_R(x)g_R(x) = c_0 + c_1x + c_2x^2 + \cdots$, and then the coefficients of $q(x) = q_{m-n}x^{m-n} + \cdots + q_1x + q_0$ can be read off as $q_i = c_{m-n-i}$, i.e. the reverse of the lowest $m - n + 1$ coefficients of $h(x)$.

(5) Once $q(x)$ has been computed, $r(x)$ can be found by computing $r(x) = g(x) - q(x)f(x)$.

Before proving the correctness of the algorithm, let us see why it can be performed by uniform $AC^0[\oplus]$ circuits: Steps (1) and (2) are trivial. The computation of $(1 - f_R(x))^k$ for $0 \le k \le m - n$ follows from Lemma 21, and it is clear that the summation in step (3) only requires (unbounded fan-in) parity gates. Step (4) is trivial. Step (5) only requires polynomial multiplication which is easily seen to be in uniform $AC^0[\oplus]$.

Now we establish the correctness of the algorithm. Note that $f_R(x) = x^n f(1/x)$, $g_R(x) = x^m g(1/x)$ and define $q_R(x) = x^{m-n}q(1/x)$ and $r_R(x) = x^{n-1}r(1/x)$. Thus we have

$$g(x) = q(x)f(x) + r(x)$$

$$g(1/x) = q(1/x)f(1/x) + r(1/x)$$

$$g_R(x) = q_R(x)f_R(x) + x^{m-n+1}r_R(x)$$

Hence $h(x) \stackrel{\text{def}}{=} \tilde{f}_R(x) g_R(x) = q_R(x)(\tilde{f}_R(x) f_R(x)) + x^{m-n+1}\tilde{f}_R(x) r_R(x)$. Note, however, that

$$\tilde{f}_R(x) f_R(x) = \tilde{f}_R(x)(1-(1-f_R(x))) = 1 + (1-f_R(x))^{m-n+1},$$

and since the constant term of $f_R(x)$ is 0 (by the assumption that f has degree exactly n), we have that

$$\tilde{f}_R(x) f_R(x) = 1 + x^{m-n+1} t(x)$$

for some $t(x) \in \mathbb{F}_2[x]$. In particular,

$$h(x) \stackrel{\text{def}}{=} \tilde{f}_R(x) g_R(x) = q_R(x)(1 + x^{m-n+1} t(x)) + x^{m-n+1}\tilde{f}_R(x) r_R(x),$$

and it is clear that the lowest $m-n$ coefficients of $h(x)$ are the coefficients of $q_R(x)$ as claimed. $\square$

Now we are prepared to prove Theorem 7. We restate the theorem for the reader's convenience.

Theorem (7, restated)**.**

1. *For every constant $\epsilon > 0$, there exist uniform $AC^0[\oplus]$ circuits of size* $\mathrm{poly}(n, 2^{t^\epsilon})$ *that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n and $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_2[x]/(f(x))$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_2[x]/(f(x))$.*

2. *There exist uniform $AC^0[\oplus]$ circuits of size* $\mathrm{poly}(n, 2^t)$ *that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n, $\alpha \in \mathbb{F}_2[x]/(f(x))$ and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_2[x]/(f(x))$.*

Proof of Theorem 7. (1) It suffices to replace the use of Lemma 18 in the proof of part 1 of Theorem 4 with Lemma 20.

(2) Consider $\alpha \in \mathbb{F}_2[x]/(f(x))$ as a polynomial $\alpha(x) \in \mathbb{F}_2[x]$ of degree at most $n-1$. We may apply Lemma 21 to compute $\alpha(x)^k$ (which has degree at most $k \cdot (n-1) \leq \mathrm{poly}(n, 2^t)$) by $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$, and then apply Lemma 20, to reduce it modulo $f(x)$, again by $AC^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$. $\square$

Theorem (8, restated)**.** *The problem of determining whether a given polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n is irreducible, is* $\mathrm{poly}(n)$*-size $AC^0[\oplus]$-reducible to the following problem: Given an irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n, compute the conjugates $x, x^2, x^{2^2}, \ldots, x^{2^{n-1}}$ (mod $f(x)$).*

Proof. The reduction proceeds as follows on input $f(x) \in \mathbb{F}_2[x]$ of degree n:

(i) Use the oracle to try to compute $x, x^2, x^{2^2}, \ldots, x^{2^{n-1}} \pmod{f(x)}$. Call the resulting quantities $a_0, a_1, \ldots, a_{n-1}$.

(ii) Check that $a_0 = x$, that $a_{i+1} \equiv a_i^2 \pmod{f(x)}$ for all $0 \leq i \leq n-1$ and that $a_{n-1}^2 \equiv x \pmod{f(x)}$. Otherwise, return REDUCIBLE.

(iii) If
$$\prod_{\text{primes } p|n} \left(x^{2^{n/p}} - x\right) \equiv 0 \pmod{f(x)}$$
then return REDUCIBLE, otherwise return IRREDUCIBLE.

First we argue the correctness of the reduction. Since the analysis is similar to the approach from [Rab] and [MS], we will only give a sketch of the proof.

The proof will use the following basic facts from the theory of finite fields: (1) The roots of $x^{2^n} - x$ are precisely the elements of the field $\mathbb{F}_{2^n}$, each occurring with multiplicity 1. (2) $x^{2^n} - x$ is divisible by an irreducible polynomial $g(x)$ of degree m if and only if m divides n.

If $f(x)$ is irreducible, then the oracle call in step (i) will succeed, returning $a_i \equiv x^{2^i} \pmod{f(x)}$; step (ii) will succeed because of the assignment of the a_i's and because $a_{n-1}^2 \equiv (x^{2^{n-1}})^2 = x^{2^n} \equiv x \pmod{f(x)}$ for any irreducible $f(x)$ (since $x^{2^n} - x$ is divisible by every irreducible polynomial of degree n); finally, step (iii) succeeds because an irreducible $f(x)$ of degree n cannot divide $x^{2^m} - x$ for any $m < n$.

On the other hand, if $f(x)$ is reducible and steps (i) and (ii) succeed, then we know that $a_i \equiv x^{2^i} \pmod{f(x)}$, and moreover that $x^{2^n} \equiv x \pmod{f(x)}$. This guarantees that $f(x)$ divides $x^{2^n} - x$ and therefore that $f(x)$ is square-free and has all its roots in $\mathbb{F}_{2^n}$. Let $f(x) = h_1(x) \cdots h_l(x)$ be the factorization of f into distinct irreducibles $h_i(x)$. To show that step (iii) succeeds, it suffices, by the Chinese Remainder Theorem, to show that the product $\prod_{p|n}(x^{2^{n/p}} - x)$ is divisible by each $h_i(x)$. Fix an irreducible factor $h(x)$ of $f(x)$. The degree of $h(x)$ must divide n (since $f(x)$ divides $x^{2^n} - x$), and hence must divide some maximal proper divisor n/p of n. Therefore, $h(x)$ will divide $(x^{2^{n/p}} - x)$ for some prime $p \mid n$, and the product in part (iii) will be divisible by $h(x)$. This concludes the proof of correctness.

Next we argue that the reduction is computable by $AC^0[\oplus]$ circuits of size poly(n): Step (i) is simply an oracle query. Each check in step (ii) can be accomplished in parallel using modular multiplication (which follows from the iterated product in Theorem 7 part 1, together with Lemma 20). Each term in the product from step (iii) can be computed using the oracle responses to compute $x^{2^{n/p}}$; since there are at most $O(\log n)$ primes dividing n, step (iii) is an iterated product of $O(\log n)$ polynomials which can be computed by $AC^0[\oplus]$ circuits of size poly(n) by Theorem 7, provided that the list of primes p dividing n is known. This list of primes can either be hard-wired into the circuit (for a non-uniform reduction) or can be shown to be computable in uniform $AC^0[\oplus]$ by a more complicated proof, which we omit. Finally, the answer can be reduced modulo $f(x)$ using Lemma 20. □

6 Proof of $AE = \mathit{Dlogtime}$ uniform TC^0

In this section we prove that $AE = \mathit{Dlogtime}$ uniform TC^0. First we exhibit a dual basis for $\tilde{\mathbb{F}}_{2^n}$. Recall that the *trace* function $\mathrm{tr} : \mathbb{F}_{2^n} \to \mathbb{F}_2$ is defined by $\mathrm{tr}(\alpha) \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} \alpha^{2^i}$. Also recall that two bases $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ and $(\beta_0, \beta_1, \ldots, \beta_{n-1})$ are *dual* if for every i, j we have that $\mathrm{tr}(\alpha_i \cdot \beta_j) = 1$ when $i = j$, while $\mathrm{tr}(\alpha_i \cdot \beta_j) = 0$ when $i \neq j$.

Lemma 22. *Let n be of the form $n = 2 \cdot 3^l$. Let $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = (1, x, \ldots, x^{n-1})$ be the standard basis for $\tilde{\mathbb{F}}_{2^n}$. Then $(\beta_0, \beta_1, \ldots, \beta_{n-1}) = (x^{3^l}, x^{3^l-1}, \ldots, x^{3^l-(n-1)})$ is the dual basis of $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$.*

The proof of this lemma follows immediately from the next lemma.

Lemma 23. *Let $x \in \tilde{\mathbb{F}}_{2^n}$ be a root of $x^{2\cdot 3^l} + x^{3^l} + 1$. Then, for any $0 \le i < 3^{l+1}$, we have $tr(x^i) = 1$ if $i = 3^l$ or $i = 2 \cdot 3^l$, and $tr(x^i) = 0$ otherwise.*

Proof. Recall that $\mathrm{tr}(x^i) = \sum_{k=0}^{2\cdot 3^l - 1} x^{i\cdot 2^k}$. Thus, if $i = 0$, then $\mathrm{tr}(x^i) = \mathrm{tr}(1) = \sum_{k=0}^{2\cdot 3^l-1} 1^{2^k} = (2\cdot 3^l)\cdot 1 \equiv 0 \mod 2$.

Now suppose that $0 < i < 3^{l+1}$, and let 3^m be the largest power of 3 dividing i. We will show that $\mathrm{tr}(x^i) = 1$ if $m = l$ and $\mathrm{tr}(x^i) = 0$ otherwise.

Since 2 is a generator of $\mathbb{Z}_{3^t}^\times$ for any integer $t > 0$ (e.g., [vL] Lemma 1.1.27), and since x has order 3^{l+1} by Fact 17, we know that the exponent, $i \cdot 2^k$, of x will take on every value in the multiset $3^m\mathbb{Z}_{3^{l+1}}^\times$ (with multiplicities) as k ranges from 0 to $\varphi(3^{l+1}) - 1 = 2\cdot 3^l - 1$. Therefore, we have

$$\mathrm{tr}(x^i) \equiv \sum_{\substack{r=0\\(r,3)=1}}^{3^{l+1}-1} x^{3^m\cdot r} = \sum_{r=0}^{3^{l+1}-1} x^{3^m\cdot r} - \sum_{r=0}^{3^l-1} x^{3^{m+1}\cdot r} = \frac{1-(x^{3^m})^{3^{l+1}}}{1-x^{3^m}} - \sum_{r=0}^{3^l-1} x^{3^{m+1}\cdot r} \equiv \sum_{r=0}^{3^l-1} x^{3^{m+1}\cdot r},$$

where we use that the denominator is non-zero because $m < l+1$ and x has order 3^{l+1}.

If $m = l$, then every term of the sum is 1, and so this is $3^l \equiv 1 \mod 2$. On the other hand, if $m < l$, then

$$\sum_{r=0}^{3^l-1} x^{3^{m+1}\cdot r} = \frac{1-(x^{3^{m+1}})^{3^{l+1}}}{1-x^{3^{m+1}}} = 0.$$

□

Theorem (10, restated)**.** *AE = Dlogtime−uniform TC^0 in the following sense:*

Let $f : \{0,1\}^n \to \{0,1\}^n$ be in Dlogtime-uniform TC^0. Then there is $f' : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ in AE such that for every n of the form $2\cdot 3^l$, and for every x of length n, $f(x) = f'(x)$.

Conversely, let $f : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ be in AE. Then there is $f' : \{0,1\}^n \to \{0,1\}^n$ in Dlogtime-uniform TC^0 such that for every n of the form $2\cdot 3^l$, and for every x of length n, $f(x) = f'(x)$.

Proof of Theorem 10. We use the following result from [FVB]: *AE* is equivalent to the class of functions computed by *Dlogtime*-uniform *arithmetic* circuits of polynomial size and constant-depth. Where an arithmetic circuit is a circuit with gates for the constant field element x, unbounded fan-in sum, unbounded fan-in product, and single input conjugation (this gate computes $\alpha \to \alpha^{2^j}$ for $0 \le j \le n$). We refer the reader to Definition 2.1 in [FVB] for more on arithmetic circuits. (While their equivalence is proved for a slightly different notion of *AE* and arithmetic circuit, it can be verified that it also applies to ours.)

AE $\subseteq$ *Dlogtime*−uniform TC^0: By the equivalence above, it is enough to show that any function computed by a *Dlogtime*−uniform arithmetic circuits of polynomial size and constant-depth is computable in *Dlogtime*−uniform TC^0. This follows by replacing the gates of the uniform circuits with the corresponding circuits as given by Theorem 3 (the iterated sum is not stated in the theorem but can be easily computed with XOR gates).

Dlogtime−uniform $TC^0 \subseteq AE$: To understand the proof of this inclusion, we first need to discuss an issue about interpretations of bit strings as field elements. Throughout the paper, and in particular in the statement of the theorem we are proving, we have interpreted a n-bit string as a field element in a field of size 2^n. Let us call this interpretation (1). Another possible interpretation, which we denote (2), is to interpret a n-bit string as a *tuple* of n field elements, the i-th field element being 0 or 1 according to the i-th bit in the string. Lemma 2.2 in [FVB] proves the inclusion *Dlogtime*−uniform $TC^0 \subseteq AE$ under interpretation (2) (to make sense of this one extends in the natural way the definition of AE to include functions mapping *tuples* of field elements to *tuples* of field elements). To prove the inclusion under interpretation (1), and thus concluding the proof of the theorem, we show how to convert back and forth between interpretations (1) and (2) in AE. Converting from (2) to (1) is relatively simple, and we omit the details that can be found in [FVB]. To convert from (1) to (2), following [BFS, FVB], we use a *dual basis* for $\tilde{\mathbb{F}}_{2^n}$. Specifically, let $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = (1, x, \ldots, x^{n-1})$ be the standard basis for $\tilde{\mathbb{F}}_{2^n}$. In other words we view an input $(c_0, c_1, \ldots, c_{n-1}) \in \{0,1\}^n$ as the field element $\gamma = \sum_i c_i \alpha_i \in \tilde{\mathbb{F}}_{2^n}$. Now let $(\beta_0, \beta_1, \ldots, \beta_{n-1}) = (x^{3^l}, x^{3^l-1}, \ldots, x^{3^l-(n-1)})$ be the dual basis of $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ as given by Lemma 22. It follows from the definition of dual basis that $c_i = \mathrm{tr}(\beta_i \cdot \gamma)$. Therefore to convert from interpretation (1) to (2) is enough to note that $\mathrm{tr}(\beta_i \cdot \gamma)$ can be computed by a *Dlogtime*-uniform arithmetic circuit, and thus is in AE by the result from [FVB] mentioned at the beginning of this proof. □

7 Proof of k-wise and ϵ-biased generator constructions

Theorem (14, restated)**.**

1. *For every k and m there is a k-wise independent generator $G : \{0,1\}^s \to \{0,1\}^m$, with $s = O(k \log m)$ that is explicitly computable by uniform $AC^0[\oplus]$ circuits of size* $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.

2. *For every ϵ and m, there is an ϵ-biased generator $G : \{0,1\}^s \to \{0,1\}^m$ with $s = O(\log m + \log(1/\epsilon))$ that is explicitly computable by uniform TC^0 circuits of size* $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.

Proof of Theorem 14. (1) We use the following construction from [CG, ABI]. Let $h = O(\log m)$ be the smallest integer bigger than $\log(m)$ of the form $h = 2 \cdot 3^l$ for some l. The generator $G : \{0,1\}^s \to \{0,1\}^m$ is defined as

$$G(\alpha_0, \alpha_1, \ldots, \alpha_{k-1})_i \stackrel{\text{def}}{=} \sum_{j<k} \alpha_j \cdot i^j, \text{ where } \alpha_0, \alpha_1, \ldots, \alpha_{k-1}, i \in \tilde{\mathbb{F}}_{2^h},$$

is a k-wise independent generator. This generator is computable by uniform $AC^0[\oplus]$ circuits of size $\mathrm{poly}(s, \log m)$ by Theorem 4.

(2) We use the following construction from [AGHP]. Let $h = O(\log m + \log(2/\epsilon))$ be the smallest integer bigger than $\log(m) + \log(2/\epsilon)$ of the form $h = 2 \cdot 3^l$ for some l. The generator $G : \{0,1\}^s \to \{0,1\}^m$ defined as

$$G(\alpha, \beta)_i \stackrel{\text{def}}{=} \langle \alpha^i, \beta \rangle \text{ where } \alpha, \beta \in \tilde{\mathbb{F}}_{2^h},$$

is an ϵ-biased generator. (Where $\langle \cdot, \cdot \rangle$ denotes inner product mod 2.) This generator is computable by uniform TC^0 circuits of size $\mathrm{poly}(s, \log m)$ by Item 2 in Theorem 3. □

8 Open Problems

Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$, can α^{-1} be computed by uniform $AC^0[\oplus]$ circuits of size poly(n)?

Given an irreducible polynomial $f(x)$ of degree n and $\alpha \in \mathbb{F}_2[x]/(f(x))$, is it possible to compute α^{2^i} for any $i = \omega(\log n)$ by uniform TC^0 circuits of size poly(n)? (cf. Lemma 19)? This is what limits our results about exponentiation in $\mathbb{F}_2[x]/(f(x))$.

Both of the above problems are also open for nonuniform circuits.

9 Acknowledgements

We would like to thank Eric Allender for various enlightening discussions and encouraging feedback concerning this work, and in particular for pointing out the fact that field multiplication over $\tilde{\mathbb{F}}_{2^n}$ is in uniform $AC^0[\oplus]$. We thank the anonymous CCC '05 referees for pointing out [FVB]. We also thank Gudmund Skovbjerg Frandsen for helpful discussions on [FVB], Kristoffer Hansen for helpful comments and Dan Gutfreund for discussions on this problem at an early stage of this research. Samir Datta independently proved some of our negative results from Theorem 5. Many thanks to Salil Vadhan for helpful comments.

References

[AAI+] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. *Comput. Complexity*, 10(2):117–138, 2001.

[ABD+] E. Allender, A. Bernasconi, C. Damm, J. von zur Gathen, M. Saks, and I. Shparlinski. Complexity of some arithmetic problems for binary polynomials. *Comput. Complexity*, 12(1-2):23–47, 2003.

[ABI] N. Alon, L. Babai, and A. Itai. A fast and simple randomized algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[AGHP] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[BIS] D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comput. System Sci.*, 41(3):274–306, 1990.

[BFS] J. Boyar, G. Frandsen, and C. Sturtivant. An arithmetic model of computation equivalent to threshold circuits. *Theoret. Comput. Sci.*, 93(2):303–319, 1992.

[CG] B. Chor and O. Goldreich. On the Power of Two-Point Based Sampling. *Journal of Complexity*, 5(1):96–106, Mar. 1989.

[CGH+] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem and t-resilient functions. In *26th Annual Symposium on Foundations of Computer Science*, pages 396–407, Portland, Oregon, 21–23 Oct. 1985. IEEE.

[Ebe] W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989.

[FVB] G. S. Frandsen, M. Valence, and D. A. M. Barrington. Some results on uniform arithmetic circuit complexity. *Math. Systems Theory*, 27(2):105–124, 1994.

[FSS] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[Gol] O. Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1999.

[GV] D. Gutfreund and E. Viola. Fooling Parity Tests with Parity Gates. In *Proceedings of the Eight International Workshop on Randomization and Computation (RANDOM)*, Lecture Notes in Computer Science, Volume 3122, pages 381–392, August 22–24 2004.

[Hås] J. Håstad. *Computational limitations of small-depth circuits.* MIT Press, 1987.

[HAB] W. Hesse, E. Allender, and D. A. M. Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. System Sci.*, 65(4):695–716, 2002. Special issue on complexity, 2001 (Chicago, IL).

[Kun] H. T. Kung. On computing reciprocals of power series. *Numerical Math*, 22:341–348, 1974.

[MS] M. Morgensteren and E. Shamir. Parallel Algorithms for Arithmetics, Irreducibility and Factoring of GFq-Polynomials. Stanford University Technical Report STAN-CS-83-991, December 1983.

[NN] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

[Rab] M. O. Rabin. Probabilistic Algorithms in Finite Fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.

[Raz] A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

[Rei] J. Reif. Logarithmic depth circuits for algebraic functions. *SIAM Journal on Computing*, 15(1):231–242, 1986.

[Sie] M. Sieveking. An algorithm for division of power series. *Computing*, 10:153–156, 1972.

[Smo] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 77–82, New York City, 25–27 May 1987.

[SF] C. Sturtivant and G. S. Frandsen. The computational efficacy of finite-field arithmetic. *Theoret. Comput. Sci.*, 112(2):291–309, 1993.

[vL] J. H. van Lint. *Introduction to coding theory*. Springer-Verlag, Berlin, third edition, 1999.

[Vol] H. Vollmer. *Introduction to circuit complexity*. Springer-Verlag, Berlin, 1999.