ON THE LOGARITHMIC DEPTH CIRCUITS FOR INVERSION IN FINITE FIELDS OF CHARACTERISTIC 2[*]

I. S. SERGEEV
(MOSCOW)

1 Introduction

In the last 30 years, finite field arithmetic has received a strong impact for development, primarily from cryptographic applications. Of particular interest is the implementation of operations in finite fields of characteristic 2 (see, for example, [5]).

This paper considers the operation of calculating an inverse element (that is, *inversion* operation) in the field $GF(2^n)$ (this notation is accepted for the Galois field of order 2^n). It is usually a part (and the most significant from the point of view of circuit performance) of the division operation.

To implement the above operations, we will employ circuits of functional elements over the basis of all two-input Boolean functions. For the most frequently used operations of negation, conjunction, disjunction, and addition modulo 2, we use the notations $\bar{\ }, \cdot, \vee, \oplus$, respectively.

The most important measures of efficiency of a circuit are its complexity (the number of elements) and depth (the maximum number of elements in an input-output chain). The concepts of complexity and depth are extended to Boolean functions. The complexity (depth) of a function is the minimum possible complexity (depth) of a circuit implementing this function[1)]. The complexity and depth of a function f are denoted by $L(f)$ and $D(f)$. A more detailed exposition of the concepts of depth and complexity can be found in [25].

A *finite field* is a finite ring with unity such that the set of its nonzero elements forms an abelian group under the operation of multiplication. The finite field $GF(2^n)$ is an n-dimensional vector space over the two-element field $GF(2)$ (with the operation of vector multiplication). Different representations of the field are related to the choice of different bases in it. The most commonly used are *standard* (or *polynomial*) and *normal* bases. In the main text, only polynomial bases and the polynomial representation of field elements will be considered. A thorough exposition of the theory of finite fields can be found in [23], and algorithmic aspects are discussed in [5, 17].

[*]**Translated version.** Originally published in: Matematicheskie Voprosy Kibernetiki [Mathematical Problems of Cybernetics]. Vol. 15. Moscow: Fizmatlit, 2006, 35–64 (in Russian). The work was supported by RFBR, project 05–01–00994, program "Leading scientific schools", project NSh–5400.2006.1, and the fundamental research program of the Department of Mathematical Sciences of the Russian Academy of Sciences "Algebraic and combinatorial methods of mathematical cybernetics" (project "Synthesis and complexity of control systems").

[1)]For a multidimensional Boolean function, we will use the term "operator" or "mapping".

In calculations in a polynomial basis, elements of $GF(2^n)$ are interpreted as polynomials of degree $n-1$ over $GF(2)$, and arithmetic operations are performed modulo some irreducible polynomial $m_n(t)$ of degree n. As a rule, a characteristic polynomial of a field is chosen to be an irreducible polynomial containing the smallest number of nonzero coefficients, usually a trinomial or pentatomial.

In practice, two groups of algorithms are used for inversion in finite fields $GF(2^n)$. The algorithms of the first group are based on the method of addition chains. Since for any $x \in GF(2^n)$ the Fermat identity $x = x^{2^n}$ holds, then inversion coincides with raising to the power $2^n - 2$, which can be performed by constructing an addition chain for the exponent $2^n - 2$. Brauer's method reduces inversion to performing $O(\log n)$ multiplications and Frobenius operations (i.e., raising to powers of the form 2^k). By performing the last operations via the Brent—Kung method [6] (see §6), one can obtain estimates $O(n^{1.667})$ for the complexity and $O(\log^2 n)$ for the depth of inversion (for more details on the application of addition chains to inversion, see [5]). In practice, methods with depth $O(\log^2 n)$ and complexity $O(n^2)$ are applied.

The second group of methods is based on the extended Euclidean algorithm for compuing the GCD of polynomials. A fast version of this algorithm, due to Schönhage and Moenck (see [11, Ch. 11]), in combination with Schönhage's method of multiplying binary polynomials [29] leads to estimates $O(n \log^2 n \log\log n)$ for the complexity and $O(\log^2 n)$ for the depth[2)] of inversion. In practice, however, asymptotically more complex methods are used. Since the Euclidean algorithm isn't adapted for circuits, the algorithms of this group are usually used in software implementation.

In [24] a circuit for inversion in the field $GF(2^n)$ of depth $O(\log n)$ was constructed. Another way of constructing such a circuit was proposed in [10]. Neither the exponent in the complexity estimate $n^{O(1)}$, nor the multiplicative coefficient in the depth estimate in [10, 24] are specified (see §7.1). Probably, the present work is chronologically the next after the two mentioned ones, dedicated to inversion in finite fields with logarithmic depth (not counting a short note [30]).

The main result consists in proving the following theorem.

Theorem. *Inversion and division in the field $GF(2^n)$ may be implemented with depth $6.44 \log n + o(\log n)$ and complexity $\frac{2}{3}n^4 + o(n^4)$; or with complexity $O(n^{1.667})$ and depth $O(\log n)$.*

Here and everywhere below we omit the base symbol for binary logarithms.

This work is purely theoretical: despite the fact that the proposed method for sufficiently large n can outperform known methods, at least in depth, in fields with dimensions $n < 1000$, which have practical significance, it can hardly be in demand. For practical construction of circuits for inversion, see [9, 19].

The main tool for constructing division and inversion circuits is a circuit implementing raising to an arbitrary power M in the field $GF(2^n)$. It is described in §3, where a depth estimate for inversion is also obtained. Before that, in §2, we consider auxiliary finite fields operations, which are used in further constructions. In §4, we discuss in detail a circuit for discrete logarithm, necessary for justifying the result of §3. A method for reducing the asymptotic complexity of inversion is described in §5. In §6, the final result of the work on the complexity of inversion is presented. In §7, some remarks concerning related issues are collected.

[2)] Note from 31.03.2025: an erroneous statement: actually, the depth can be estimated only as $O(n)$.

2 Auxiliary finite fields operations

An operator $\boldsymbol{U} : GF(2)^n \to GF(2)^m$ is called *linear* if for any vectors $x, y \in GF(2)^n$ $\boldsymbol{U}(x+y) = \boldsymbol{U}(x)+\boldsymbol{U}(y)$. The last condition is equivalent to the existence of an $m\times n$-matrix U over $GF(2)$ such that for any vector $x \in GF(2)^n$, we have $\boldsymbol{U}(x) = Ux$. In this case we say that the operator $\boldsymbol{U}$ has dimension $m \times n$.

2.1 Examples of linear operations

We list some operations of finite field arithmetic used below, for which the linearity property holds.

Let $\boldsymbol{S_{n,l}}$ denote the operator of raising to the power 2^l in the field $GF(2^n)$. It is called the *Frobenius operator* and is linear: for any $a, b \in GF(2^n)$ the Frobenius identity holds:

$$(a + b)^{2^l} = a^{2^l} + b^{2^l}.$$

When working in a polynomial basis, we often use the operation of calculating the remainder of division of an arbitrary polynomial $h(t)$ by a (irreducible) polynomial $m_n(t)$ defining the field basis under consideration. We introduce the notation $\boldsymbol{B_{n,p}}$ for the transform that reduces a polynomial of degree at most $p - 1$ over $GF(2)$ modulo $m_n(t)$. This operation is linear, since for any binary polynomials $h_1(t)$ and $h_2(t)$,

$$(h_1(t) + h_2(t)) \bmod m_n(t) = (h_1(t) \bmod m_n(t)) + (h_2(t) \bmod m_n(t)).$$

Let us also consider the operation of evaluating an arbitrary polynomial of degree at most $p-1$ over $GF(2)$ in a fixed point $a \in GF(2^n)$. This operation is the so-called *modular composition* of polynomials, denoted by $\boldsymbol{C_{n,p,a}}$. It is linear, since for any pair of polynomials $h_1(x)$ and $h_2(x)$ the identity $(h_1+h_2)(a) = h_1(a)+h_2(a)$ holds, which follows directly from the rule for adding coefficients at similar powers of a:

$$(h_{1,i} + h_{2,i})a^i = h_{1,i}a^i + h_{2,i}a^i,$$

where $h_{1,i}$ and $h_{2,i}$ denote the coefficients of the corresponding polynomials at x^i.

In the field $GF(2^k)$ we fix a set of elements $\{\alpha_j \mid j = 1, \dots, p\}$. The operator $\boldsymbol{F_{k,s,p}}$ maps an arbitrary polynomial of degree at most $s - 1$ over $GF(2)$ to the vector of its values on this set.

Let $s = p$. According to the main property of interpolation, the operator $\boldsymbol{F_{k,p,p}}$ establishes a one-to-one correspondence between $GF(2)^p$ and $\mathrm{Im}(\boldsymbol{F_{k,p,p}}) \subset GF(2^k)^p$, hence, there exists an inverse mapping $\boldsymbol{F^{-1}_{k,p}}$.

In fact, the mappings $\boldsymbol{F_{k,s,p}}$ and $\boldsymbol{F^{-1}_{k,p}}$ perform the inverse and forward interpolation operations, respectively, for the set of polynomials with coefficients in $GF(2)$ (one could consider the coefficients of the polynomials as belonging to the field $GF(2^k)$, which is more common, but we will not need such a generalization).

The operator $\boldsymbol{F_{k,s,p}}$ is the union of p linear mappings $\boldsymbol{C_{k,s,\alpha_j}}$, $j = 1, \dots, p$, and is thus linear. The operator $\boldsymbol{F^{-1}_{k,p}}$, as the inverse of $\boldsymbol{F_{k,p,p}}$, is also linear.

2.2 Complexity of linear operations

Consider an arbitrary linear mapping $\boldsymbol{A_{m,n}}$ of dimension $m \times n$ with matrix A. Multiplication of such a matrix by an arbitrary vector of length n can be interpreted as computing a system of m linear combinations of the vector components over the field $GF(2)$.

The computational complexity of one linear combination of n variables does not exceed $n-1$ while the depth is $\lceil \log n\rceil$. Consequently, for independent computation of m combinations, no more than $m(n-1)$ functional elements are required. By the method of O. B. Lupanov [26] (see also [25]), which is presented below, one can construct a circuit of complexity $O(mn/\log n)$ and depth $\lceil \log n\rceil + 1$, i.e., asymptotically optimal in depth and complexity with respect to the class of all linear mappings of dimension $m \times n$.

Lemma 1. *There is a circuit computing all linear combinations of s variables, minimal in depth and complexity.*

Proof. In parallel, all possible sums of two variables are calculated. At the next level — sums of various triplets and quadruplets of variables (using what has already been calculated). Then sums of sets of 5, 6, 7 and 8 variables, etc. All elements, as well as all inputs of the constructed circuit are its outputs, which implies minimal complexity. The number of elements in the circuit is $2^s - s - 1$, and the depth is $\lceil \log s\rceil$ and is minimal possible. □

Theorem 1 (O. B. Lupanov, 1956)**.** *The complexity of multiplying a binary matrix A of size $m \times n$ by a binary vector x of length n does not exceed $\frac{mn}{\log m}(1+\frac{\log\log m+2}{\log m-\log\log m})$, and the depth does not exceed*[3] $\lceil \log n\rceil + 1$.

Proof. Divide the components $x_0, \dots, x_{n-1}$ of the vector x into groups of size s. For each of the groups, implement all linear combinations by the method of Lemma 1. An arbitrary linear combination of all components of the vector x is constructed from "short" group sums, additionally requiring $\lceil n/s\rceil - 1$ elementary additions. The complexity of the entire circuit can be estimated as

$$L(\boldsymbol{A_{m,n}}) < \frac{n}{s}(2^s - s - 1) + m\frac{n}{s}.$$

After choosing $s = \lceil \log m - \log\log m\rceil$, the obtained estimate takes the form

$$L(\boldsymbol{A_{m,n}}) < \frac{n}{\log m - \log\log m}\left(\frac{2m}{\log m} + m\right) = \frac{mn}{\log m}\cdot\frac{\log m + 2}{\log m - \log\log m},$$

which proves the theorem in part of complexity.

The circuit depth is estimated as

$$D(\mathbf{A_{m,n}}) \le \lceil \log s\rceil + \lceil \log\lceil n/s\rceil\rceil \le \lceil \log n\rceil + 1.$$

□

2.3 Nonlinear operations

Note that each of the linear operations given as examples can be implemented with smaller complexity than a linear mapping in general.

For instance, by the method [6], the modular composition operation $\boldsymbol{C_{n,p,a}}$ can be reduced to multiplying a matrix of size $\sqrt{p} \times \sqrt{p}$ by a matrix of size $\sqrt{p} \times n$, which can be performed in $O(pn/\log p)$ operations (see, e.g., [16]). The Frobenius

[3] Here and in all subsequent formulations, estimates of the complexity and depth of functions are given for implementation by a single circuit.

operation $\boldsymbol{S_{n,l}}$ is a special case of modular composition and also requires less than $O(n^2/\log n)$ operations and, in particular, can be implemented with complexity $O(n^{1.667})$ (see §6).

A fundamental nonlinear operation in the field $GF(2^n)$ is multiplication, denoted by $\boldsymbol{M_n}$. Field multiplication is usually implemented in two steps: multiplication of polynomials representing the elements being multiplied, and reduction of the result modulo $m_n(t)$. The "school" algorithm for multiplying polynomials of degree $n-1$ has complexity $2n^2$ and depth $\lceil \log n \rceil + 1$. Modulo reduction is a linear operation of type $\boldsymbol{B_{n,2n-1}}$ and can be performed with the same depth and complexity.

However, the Schönhage method [29] allows to multiply binary polynomials with complexity $O(n \log n \log\log n)$ and depth $O(\log n)$. In turn, modulo reduction can be reduced to multiplication of polynomials. We describe this reduction following [11]. Write a polynomial $h(t)$ of degree at most $2n-1$ as $a(t)t^n + b(t)$, where $\deg a, b < n$. We introduce the notation $\widetilde{c}(t) = t^{\deg c} c(1/t)$, i.e., the coefficients of the polynomial $\widetilde{c}(t)$ are the coefficients of $c(t)$ written in reverse order. Let $a(t)t^n = q(t)m_n(t) + r(t)$, where $\deg q, r < n$, then $h(t) \bmod m_n(t) = r(t) + b(t)$. The remainder $r(t)$ is computed by two multiplications as follows. We have,

$$\widetilde{a}(t) = \widetilde{q}(t)\widetilde{m}_n(t) + t^n \widetilde{r}(t).$$

If $i(t)$ is the inverse polynomial to $\widetilde{m}_n(t)$ modulo t^n, i.e., $i(t)\widetilde{m}_n(t) = 1 \bmod t^n$ (it exists because the free coefficient of $\widetilde{m}_n(t)$ is 1), then

$$\widetilde{q}(t) = \widetilde{a}(t)i(t) \bmod t^n \qquad \text{and} \qquad r(t) = q(t)m_n(t) \bmod t^n.$$

Thus,

$$L(\boldsymbol{B_{n,2n}}) \le 2M(n) + n, \qquad D(\boldsymbol{B_{n,2n}}) \le 2D(n) + 1,$$

and consequently

$$L(\boldsymbol{M_n}) \le 3M(n) + n, \qquad D(\boldsymbol{M_n}) \le 3D(n) + 1,$$

where $M(n), D(n)$ are the complexity and depth of multiplication of binary polynomials of degree $n-1$.

The operation $\boldsymbol{F_{k,s,p}}$, i.e., the evaluation of a polynomial of degree $s-1$ at p points of the field $GF(2^k)$ in the case $s \le p$ (of interest to us), can be performed by an algorithm [1] in $O(M(p)\log p)$ operations in the field $GF(2^k)$ with depth $O(\log^2 p)$ over the same field (for comparison, the complexity estimate $O(skp/\log(kp))$ follows from Theorem 1). Therefore,

$$L(\boldsymbol{F_{k,p,p}}) \le O(M(p)\log p)L(\boldsymbol{M_k}), \qquad D(\boldsymbol{F_{k,p,p}}) \le O(\log^2 p)D(\boldsymbol{M_k}).$$

A similar algorithm (see [1]) allows us to obtain the same estimates for the interpolation operation $\boldsymbol{F^{-1}_{k,p}}$. In §6, a "parallel" version of this algorithm will be described, with depth of logarithmic order.

Note that Lupanov's method allows to implement an arbitrary linear mapping with an asymptotically optimal depth, so it will be used primarily in the part devoted to minimizing the depth of inversion; in the part related to the complexity of inversion with logarithmic depth (§6), alternative methods will be used, including those mentioned above.

3 Exponentiation algorithm

The subsequent constructions will be based on the algorithm for raising a field element to an arbitrary (but fixed for the algorithm) power. Here is a brief description of it.

Let $x \in GF(2^n)$ be an element in the polynomial representation, and we need to compute x^M, where $M = 2^{e_1} + 2^{e_2} + \ldots + 2^{e_m}$. We can assume that $M < 2^n - 1$ (due to the Fermat identity $x^{2^n} = x$) and, therefore, $m < n$.

1. Compute the powers $x^{2^{e_1}}, \ldots, x^{2^{e_m}}$. Let $x^{2^{e_i}}$ correspond to a polynomial $f_i(t)$ in the field representation. Let further $f(t) = f_1(t) \cdot \ldots \cdot f_m(t)$. Set $p = m(n-1)+1$, and choose a field $GF(2^k)$ containing at least p elements; in it, choose a set of elements $\alpha_1, \ldots, \alpha_p$.

2. Compute all possible $f_i(\alpha_j) \in GF(2^k)$, where $i = 1, \ldots, m$, $j = 1, \ldots, p$.

3. For all j, compute the products $f_1(\alpha_j) \cdot \ldots \cdot f_m(\alpha_j) = f(\alpha_j)$. To do this, in the field $GF(2^k)$, choose a primitive element α. If $f_i(\alpha_j) \neq 0$ for all i, then

3.1. Compute the discrete logarithms, $\log_\alpha f_i(\alpha_j)$.

3.2. Compute $\sum_{i=1}^{m} \log_\alpha f_i(\alpha_j) \bmod (2^k - 1) = \log_\alpha f(\alpha_j)$.

3.3. Compute $f(\alpha_j) = \alpha^{\log_\alpha f(\alpha_j)}$.

4. Given the values $f(\alpha_j)$, $j = 1, \ldots, p$, reconstruct the polynomial $f(t)$ of degree at most $p - 1$.

5. The element x^M corresponds to the polynomial $f(t) \bmod m_n(t)$.

It is easy to see that the algorithm is based on the idea of interpolation, which goes back to the work of A. L. Toom [33]. It allows to reduce iterated multiplication in the field $GF(2^n)$ to multiplication in a field of lower dimension $GF(2^k)$. For multiplication in the "small" field $GF(2^k)$, discrete logarithm is used, the idea of which is taken from [7].

The operation of raising to a power of weight m (weight is the number of ones in binary notation) in the field $GF(2^n)$ is denoted by $\boldsymbol{E_{n,m}}$. For brevity, we omit information about the power to which the field element is raised — this will be clear from the context. Let $L(\boldsymbol{E_{n,m}})$ and $D(\boldsymbol{E_{n,m}})$ denote the complexity and depth of implementation of the most complex (deep) of the exponentiation operations with powers of weight m.

Note that at step 1 of the algorithm the operations $\boldsymbol{S_{n,e_i}}$, $i = 1, \ldots, m$, are performed, at step 2 — m operations $\boldsymbol{F_{k,n,p}}$, at step 4 — operation $\boldsymbol{F^{-1}_{k,p}}$, at step 5 — operation $\boldsymbol{B_{n,p}}$. All listed operations are linear (see §2).

We also introduce the following notation: $\boldsymbol{\Lambda_k}$ for the operation of discrete logarithm with base α in the field $GF(2^k)$; $\boldsymbol{\Sigma_{m,k}}$ for the operation of summing m k-digit numbers modulo $2^k - 1$; $\boldsymbol{\Lambda_k^{-1}}$ for the operation of raising a primitive element $\alpha \in GF(2^k)$ to a power that is a k-digit number; $\boldsymbol{X_{k,m}}$ to indicate that m elements of the field $GF(2^k)$ are nonzero.

Step 3 consists of performing p operations of m-fold multiplication in the field $GF(2^k)$ (we introduce the notation $\boldsymbol{\Phi_{k,m}}$ for one such multiplication) according to the scheme

$$\boldsymbol{\Phi_{k,m}}(y_1, \ldots, y_m) = \boldsymbol{X_{k,m}}(y_1, \ldots, y_m) \times \boldsymbol{\Lambda_k^{-1}} \cdot \boldsymbol{\Sigma_{m,k}}(\boldsymbol{\Lambda_k}(y_1), \ldots, \boldsymbol{\Lambda_k}(y_m)),$$

where the symbol $\cdot$ denotes the composition of mappings, and the symbol $\times$ — the usual multiplication of a scalar value by a vector.

The operation $\boldsymbol{X_{k,m}}$ is implemented by the conjunction of disjunctions of the digits of the arguments (field elements) with the minimum possible complexity $mk - 1$ for a function that essentially depends on all its variables, and depth

$\lceil \log m \rceil + \lceil \log k \rceil$, which is neglible in the context of the parallel iterated multiplication.

The discrete logarithm operation $\boldsymbol{\Lambda_k}$ is discussed in detail in §4, where it is shown that for an arbitrary $\varepsilon > 0$ one can choose $k = \log p + C_0(\varepsilon)$ such that

$$L(\boldsymbol{\Lambda_k}) \leq C_1(\varepsilon)O(p^{\varepsilon}), \qquad D(\boldsymbol{\Lambda_k}) \leq \varepsilon \log p + C_2(\varepsilon) + O(\log^2 \log p).$$

Consider the operation of m-fold summation $\boldsymbol{\Sigma_{m,k}}$. One way to construct low-depth circuits for this operation is as follows. It is known (see, e.g., [18]) that summation of several numbers can be reduced to summation of a smaller number of summands with depth $O(1)$ via a circuit called *compressor*.

The simplest example of such a circuit is a (3,2)-compressor. If three k-digit numbers are given: $a = (a_{k-1}, \ldots, a_0)$, $b = (b_{k-1}, \ldots, b_0)$, $c = (c_{k-1}, \ldots, c_0)$ (the seniority of the digits increases from right to left), then the sum $a_i + b_i + c_i$ can be represented as $2u_i + v_i$, where

$$v_i = a_i \oplus b_i \oplus c_i, \quad u_i = a_i \cdot b_i \oplus b_i \cdot c_i \oplus a_i \cdot c_i.$$

Thus the number of terms is reduced from 3 to 2: $a + b + c = u + v$, where $u = (u_{k-1}, \ldots, u_0, 0)$, $v = (v_{k-1}, \ldots, v_0)$. The pair of bits (u_i, v_i) is computed with complexity 5 and depth 3. Finally, the complexity of the compressor is $5k$, and the depth is 3.

From such subcircuits-compressors, one can compose a circuit that with depth $O(\log m)$ transforms m input numbers into $O(1)$ output numbers while preserving the sum. Finally, the resulting numbers can be summed using ordinary adders.

If (as in our case) k-digit numbers are summed modulo $2^k - 1$, then the most significant digits obtained in the process of calculations should be moved to the place of the least significant ones. For example, a modular (3,2)-compressor should return numbers $u' = (u_{k-2}, \ldots, u_0, u_{k-1})$ and $v = (v_{k-1}, \ldots, v_0)$, where u_i, v_i are defined as above.

Probably the best theoretical estimate of the depth of a compressor circuit that reduces m-fold summation to addition of two numbers, $3.44 \log m + O(1)$, is obtained in [14] using the method from [27]. The complexity of such a circuit is $O(mk)$. The constants hidden under the O sign in these estimates are quite large — in practice, one can construct circuits of depth at most $3.71 \log m$ and complexity $5mk$ from (3,2)-compressors.

A usual k-bit adder can be implemented by a circuit of linear complexity and depth $\log k + O(\sqrt{\log k})$ by the method of V. M. Khrapchenko [20]. However, on a practical range of k values, other methods work better. For example, the method of M. I. Grinchuk [13] has a depth estimate of $1.27(\log k + 1) + 3$. The complexity of a circuit directly constructed by the Grinchuk method is $O(k \log k)$, but it can be reduced to linear using the standard linearization procedure (see, for example, [20]) with an increase in depth by $O(\log \log k)$.

Addition of two k-digit numbers modulo $2^k - 1$ can be reduced to the usual addition of $2k$-digit numbers. Indeed, if $a, b \leq 2^k - 1$, then $a + b \bmod (2^k - 1) = c + d$, where $a + b = c2^k + d$ and $c + d \leq 2^k - 1$. The result $a + b \bmod (2^k - 1)$ is contained in the k-th through $(2k - 1)$-th digits (numbering starting from zero) of the sum of the numbers $(2^k + 1)a$ and $(2^k + 1)b$.

Finally, we have

$$L(\boldsymbol{\Sigma_{m,k}}) = O(mk), \qquad D(\boldsymbol{\Sigma_{m,k}}) \leq 3.44 \log m + O(\log k).$$

We will roughly estimate the complexity and depth of the exponentiation circuit (implementing the $\boldsymbol{\Lambda_k^{-1}}$ operation), but this will be sufficient for our purposes. If $(b_{k-1}, b_{k-2}, \dots, b_0)$ is the binary notation of a number b, then

$$\alpha^b = \alpha^{b_0}\alpha^{2b_1} \cdot \ldots \cdot \alpha^{2^{k-1}b_{k-1}}.$$

To compute each of the factors on the right-hand side of the formula, k functional elements ($k-1$ conjunctors and an element implementing $\overline{x} \vee y$ function) are sufficient, since

$$\alpha^{2^i b_i} = \overline{b_i} \cdot 1 \vee b_i \alpha^{2^i},$$

where 1 is the unit of the field $GF(2^k)$, and the symbol $\vee$ means bitwise disjunction. We assume that the powers of α have been computed in advance.

Thus, the simplest way to compute α^b is to perform $k-1$ field multiplications, which yields the estimates

$$L(\boldsymbol{\Lambda_k^{-1}}) \le kL(\boldsymbol{M_k}) = O(k^2 \log k \log\log k),$$
$$D(\boldsymbol{\Lambda_k^{-1}}) \le \lceil \log k \rceil D(\boldsymbol{M_k}) + 1 = O(\log^2 k).$$

Theorem 2. *Let m be the weight of a number M. Then for the operation of raising to the power M in $GF(2^n)$ the following estimates hold (for $\varepsilon > 0$):*

$$L(\boldsymbol{E_{n,m}}) \lesssim \frac{\log(mn) + C_0(\varepsilon)}{\log(m^2 n)} m^2 n^2 + C_1(\varepsilon) m^{2+\varepsilon} n^{1+\varepsilon}, \tag{1}$$
$$D(\boldsymbol{E_{n,m}}) \lesssim (2+\varepsilon)\log n + 4.44 \log m + D_0(\varepsilon). \tag{2}$$

Proof. Choose $k = \log(mn) + C_0(\varepsilon)$ such that the complexity and depth estimates for the discrete logarithm from Corollary 3 (see §4) hold.

Considering the composition of linear mappings $\boldsymbol{S_{n,e_i}}$ and $\boldsymbol{F_{k,n,p}}$, $i = 1, \dots, m$, as a single linear mapping of dimension $kmp \times n$, from Lupanov's method we derive the complexity estimate for the corresponding subcircuit $(1 + o(1))(kmpn/\log(kmp)) \le \frac{\log(mn)+C_0(\varepsilon)}{\log(m^2 n)} m^2 n^2$. The depth of the subcircuit does not exceed $\log n + 2$.

Another linear operator $\boldsymbol{B_{n,p}} \cdot \boldsymbol{F_{k,p}^{-1}}$ of dimension $n \times kp$ is implemented by a subcircuit of complexity $O(pkn/\log n) \sim O(mn^2)$ and depth $\log(kp) + 2 \sim \log(mn) + D_1(\varepsilon)$.

The subcircuit for computing discrete logarithms in an auxiliary field consists of mp parallel blocks implementing operations of type $\boldsymbol{\Lambda_k}$. The complexity of this subcircuit (see §4) is estimated as $C_1(\varepsilon) mp^{1+\varepsilon} \sim C_1(\varepsilon) m^{2+\varepsilon} n^{1+\varepsilon}$. The depth is $(\varepsilon + o(1)) \log p + D_2(\varepsilon) \sim (\varepsilon + o(1)) \log(mn) + D_2(\varepsilon)$.

Further, the complexity of implementing p adders of type $\boldsymbol{\Sigma_{m,k}}$ is estimated as $O(mkp) \sim O(m^2 n \log n)$. The depth can be estimated as $3.44 \log m + O(\log k) = 3.44 \log m + O(\log\log n) + D_3(\varepsilon)$.

Thus, the complexity of the entire circuit is determined by the complexity estimate for the first block of linear mappings. The depth asymptotics is composed of the depths of the four subcircuits,

$$D(\boldsymbol{E_{n,m}}) \lesssim \log n + \log(mn) + D_1(\varepsilon) + \varepsilon \log(mn) + D_2(\varepsilon) + 3.44 \log m + D_3(\varepsilon) \sim$$
$$\sim (2+\varepsilon)\log n + 4.44 \log m + D_0(\varepsilon).$$

□

Theorem 3. *Inversion in the field $GF(2^n)$ may be implemented by a circuit with the depth and complexity*

$$D(\boldsymbol{I_n}) \le (6.44 + o(1)) \log n, \qquad L(\boldsymbol{I_n}) \le (2/3 + o(1))n^4.$$

Proof. The inversion operation corresponds to raising to the power $2^n - 2$, having weight $n - 1$. Indeed,

$$x^{-1} = x^{2^n-2} = x^2 x^{2^2} \cdot \ldots \cdot x^{2^{n-1}}.$$

This theorem is a direct consequence of Theorem 2. It suffices to set $m = n - 1$, and to choose ε within the rounding error of the constant from [14] up to 3.44. □

Let $\boldsymbol{\Delta_n}$ denote the division operation in the field $GF(2^n)$. Obviously, the division is reduced to one inversion and one multiplication in the field, but the multiplication can be integrated into the inversion circuit described above, since

$$\frac{y}{x} = yx^2 x^{2^2} \cdot \ldots \cdot x^{2^{n-1}}.$$

Let's set $m = n$ in the algorithm from the beginning of the section. Performing step 2 for y separately and in parallel with combined steps 1, 2 for x, and then applying the remaining steps of the exponentiation algorithm with a power of weight n, we obtain the following result.

Theorem 4. *For the division operation in $GF(2^n)$, we have*

$$D(\boldsymbol{\Delta_n}) \le (6.44 + o(1)) \log n, \qquad L(\boldsymbol{\Delta_n}) \le (2/3 + o(1))n^4.$$

From a practical point of view, the proposed circuit is hardly of interest. The obtained estimates show that for n of size of not more than several hundred thousand, the standard inversion method apparently has better depth. At the same time, the complexity of the standard method is always $O(n^3)$, and actually for most fields is $O(n^2)$.

4 Discrete logarithm

Fix α — a generating element of the multiplicative group of the field $GF(2^k)$ (the multiplicative group is denoted by $GF(2^k)^*$ and consists of all nonzero elements of the field). Then for any element $\beta \in GF(2^k)^*$ a number $b \in 0, \ldots, 2^k - 2$ is uniquely determined such that $\beta = \alpha^b$. It is called the *discrete logarithm of the element β to the base α*. We are going to estimate the parameters of a circuit implementing the discrete logarithm operation $\boldsymbol{\Lambda_k}$, where $\boldsymbol{\Lambda_k}(\beta) = b$.

The method considered below is an adaptation of the Silver—Pohlig—Hellman algorithm (see, e.g., [22]) to the model of circuits of functional elements.

Let

$$2^k - 1 = r_1 r_2 \cdot \ldots \cdot r_w$$

be some known decomposition of $2^k - 1$ into coprime factors. We introduce the notation $q_i = (2^k - 1)/r_i$, where $i = 1 \ldots w$. Note that

$$\beta^{q_i} = (\alpha^b)^{q_i} = (\alpha^{q_i})^{b \bmod r_i}.$$

Thus, by comparing β^{q_i} with all possible powers of the element α^{q_i} (a total of r_i such comparisons are needed), we can determine the remainder $b_i = b \bmod r_i$. From the set of remainders $b_i, i = 1 \dots w$, the number b is restored uniquely. Consider the following computation scheme.

1. Compute all $\beta_i = \beta^{q_i}$, $i = 1, \dots, w$.

2. For each i, among $j = 0, \dots, r_i - 1$, by coefficient-by-coefficient comparison of elements β_i and α^{jq_i} (the latter are calculated in advance), find b_i satisfying $\beta_i = \alpha^{b_i q_i}$.

3. The number $b = \log_\alpha \beta$ is restored by its remainders $b_i = b \bmod r_i$.

The operation of computing w powers of an element $\beta \in GF(2^k)$ is denoted by $\boldsymbol{H_{k,w}}$. Obvious estimates for the depth and complexity of this operation are stated in the following lemma.

Lemma 2. *For the complexity and depth of the operator $\boldsymbol{H_{k,w}}$ the following estimates hold:*

$$L(\boldsymbol{H_{k,w}}) \le w(k-3)L(\boldsymbol{M_k}) + O(k^3/\log k), \tag{3}$$

$$D(\boldsymbol{H_{k,w}}) \le \lceil \log(k-2) \rceil D(\boldsymbol{M_k}) + \lceil \log k \rceil + 1. \tag{4}$$

Proof. All powers of the form β^{2^l}, $l = 0, \dots, k-1$, are computed by a circuit implementing the corresponding linear operator of dimension $k^2 \times k$ with complexity and depth $O(k^3/\log k)$ and $\lceil \log k \rceil + 1$, respectively.

An arbitrary power can be represented as a product of at most $k-2$ factors of the form β^{2^l}, since obviously $q_i < 2^{k-1} - 1$. To compute w such products, each involving at most $k-2$ factors, at most $w(k-3)$ field multiplications are required. $\square$

If the standard multiplication algorithm is applied, we obtain

$$L(\boldsymbol{H_{k,w}}) \le 2wk^3 + O(wk^3/\log k), \quad D(\boldsymbol{H_{k,w}}) < 2(\lceil \log k \rceil + 1)^2.$$

In general, the calculation of the system of powers can be performed more economically exploiting techiques from the theory of addition chains (see, e.g., [5, 21]).

The product of k polynomials of degree $k-1$ may be implemented by a circuit of depth $O(\log k)$. To construct it, one can employ an analogous algorithm for multiplying numbers from [2] (the product of polynomials is reduced to a numerical product, see, e.g., [7]). Another way is to recursively apply the method of the present work. Thus, actually, the operator $\boldsymbol{H_{k,w}}$ can be implemented by a circuit of logarithmic depth.

Let us introduce some additional notation. Let $qr = 2^k - 1$, where $(q, r) = 1$. On a subgroup of r-th roots of unity of the field $GF(2^k)$, the operation $\boldsymbol{\Lambda'_{k,r}}$ of taking the logarithm to the base of the subgroup generating element, which is α^q, can be well defined.

Note that $\boldsymbol{\Lambda'_{k,r}}(\beta^q) = b \bmod r$, where $b = \log_\alpha \beta$. The mapping $\boldsymbol{\Lambda'}$ can be formally extended to the entire field; outside the indicated subgroup of roots of unity, define it arbitrarily.

The mapping $\boldsymbol{R_{k,w}}$ restores a number that has given remainders modulo r_i, $i = 1, \dots, w$, namely

$$\boldsymbol{R_{k,w}}(b_1, b_2, \dots, b_w) = b, \qquad 0 \le b < \prod r_i, \quad b = b_i \bmod r_i, \quad i = 1, \dots, w.$$

Using the introduced notations, we can write

$$\boldsymbol{\Lambda_k}(\beta) = \boldsymbol{R_{k,w}}(\boldsymbol{\Lambda'_{k,r_1}}(\beta^{q_1}), \dots, \boldsymbol{\Lambda'_{k,r_w}}(\beta^{q_w})).$$

Lemma 3. *For the complexity and depth of a circuit implementing* $\boldsymbol{R_{k,w}}$*, the following estimates hold:*

$$L(\boldsymbol{R_{k,w}}) = O(k^2), \quad D(\boldsymbol{R_{k,w}}) = O(\log k).$$

Proof. According to the Chinese remainder theorem (see, e.g., [21])

$$b = b_1c_1 + b_2c_2 + \ldots + b_wc_w \bmod 2^k - 1,$$
$$c_i = \nu_i r_1 \cdot \ldots \cdot r_{i-1}r_{i+1} \cdot \ldots \cdot r_w = \frac{\nu_i(2^k-1)}{r_i},$$

where the normalizing coefficient $\nu_i \in [1, r_i - 1]$ is chosen based on the condition $c_i = 1 \bmod r_i$. By construction, the numbers c_i consist of no more than k binary digits.

Consider the following way of executing calculations (close to [2]). Let $b_i = (b_{i,j-1}, b_{i,j-2}, \ldots, b_{i,0})$ in binary representation, $j = \lceil \log r_i \rceil$. Then

$$b_ic_i = b_{i,0}c_i + 2b_{i,1}c_i + \ldots + 2^{j-1}b_{i,j-1}c_i.$$

The computation of the terms in the given formula is carried out "for free" — their reduction modulo $2^k - 1$ is also "free" (the most significant digits are substituted for the least significant ones). Apply this to every product b_ic_i, $i = 1, \ldots, w$. The number of newly formed terms is estimated as

$$\sum_{i=1}^{w} \lceil \log r_i \rceil < w + \sum_{i=1}^{w} \log r_i = w + \log(2^k - 1) < k + w.$$

The problem is reduced to summing at most $k+w$ instances of k-digit numbers modulo $2^k - 1$ (the corresponding mapping was denoted by $\boldsymbol{\Sigma_{k+w,k}}$). Therefore,

$$L(\boldsymbol{R_{k,w}}) \le L(\boldsymbol{\Sigma_{k+w,k}}), \quad D(\boldsymbol{R_{k,w}}) \le D(\boldsymbol{\Sigma_{k+w,k}}).$$

Now the assertion of the lemma follows from the estimates of §3 and an obvious observation $w < k$. $\square$

In [15] a method for constructing a circuit of complexity $O(k^{1+\varepsilon})$ and depth $O(\varepsilon^{-1}\log k)$, where $\varepsilon > 0$, is proposed. Apparently, this method does not outperform the standard method in terms of depth.

It is shown below that the subcircuits implementing $\boldsymbol{H_{k,w}}$ and $\boldsymbol{R_{k,w}}$ do not have a significant effect on the asymptotic complexity and depth of the discrete logarithm circuit as a whole.

Theorem 5. *The complexity and depth of the operator* $\boldsymbol{\Lambda'_{k,r}}$ *satisfy the estimates:*

$$L(\boldsymbol{\Lambda'_{k,r}}) < r\left(2 + \frac{k}{\log r} \cdot \frac{\log r + 6}{\log r - \log\log r}\right), \qquad D(\boldsymbol{\Lambda'_{k,r}}) \le \lceil \log k \rceil + \lceil \log r \rceil + 1.$$

The circuit is constructed from subcircuits, which are described in the next two paragraphs.

4.1 Implementing systems of comparators

Let β^q be fed to the inputs of the subcircuits of comparison with the corresponding α^{lq}, $l = 0 \ldots r-1$. Since the latter are precomputed, comparing of a k-bit element β^q with a fixed field element is represented by a generalized conjunction of the bits of β^q (here, comparing is understood as determining the coincidence or non-coincidence of two vectors).

Split the set of k variables (which encode β^q) into groups containing at most s variables. For each group, construct a circuit implementing all possible generalized conjunctions of this group of variables (it is called a decoder). To compute the required r conjunctions of k variables, we additionally need at most $r(\lceil k/s\rceil - 1)$ conjunctions to connect the corresponding outputs of the decoders. The following lemma is actually contained in [25].

Lemma 4. *The complexity of the decoder of s variables is $L(\boldsymbol{K_s}) < 2^s + 3.81\cdot 2^{\frac{s}{2}}$, and the depth is $D(\boldsymbol{K_s}) \le \lceil \log s\rceil + 1$.*

Proof. Consider the following circuit. Split the set of variables into two parts: they are equal when s is even, and differ by 1 in the odd case. Let two decoders be constructed for them. Then, using 2^s conjunctions, combine the outputs of these subcircuits in every possible way.

The lower-dimension decoders involved in this design are constructed in exactly the same way. The decoder of one variable includes only one functional element of negation: $L(\boldsymbol{K_1}) = 1$, $D(\boldsymbol{K_1}) = 1$. Let us estimate the complexity of the simplest decoders:

$$L(\boldsymbol{K_2}) = 2^2 + 2L(\boldsymbol{K_1}) = 6, \quad L(\boldsymbol{K_3}) = 2^3 + L(\boldsymbol{K_1}) + L(\boldsymbol{K_2}) = 15,$$
$$L(\boldsymbol{K_4}) = 2^4 + 2L(\boldsymbol{K_2}) = 28, \quad L(\boldsymbol{K_5}) = 2^5 + L(\boldsymbol{K_2}) + L(\boldsymbol{K_3}) = 53,$$
$$L(\boldsymbol{K_6}) = 2^6 + 2L(\boldsymbol{K_3}) = 94, \quad L(\boldsymbol{K_7}) = 2^7 + L(\boldsymbol{K_3}) + L(\boldsymbol{K_4}) = 171,$$
$$L(\boldsymbol{K_8}) = 2^8 + 2L(\boldsymbol{K_4}) = 312.$$

In these cases, the stated complexity estimate is satisfied; the constant 3.81 is obtained for $s = 7$.

Now, we will verify the assertion for $s > 8$ by induction. Note that the calculation below is correct for both the even ($\delta = 0$) and odd cases ($\delta = 0.5$).

$$\begin{aligned} L(\boldsymbol{K_s}) &\le 2^s + L(\boldsymbol{K_{\frac{s}{2}-\delta}}) + L(\boldsymbol{K_{\frac{s}{2}+\delta}}) < \\ &< 2^s + 2^{s/2}(2^{\delta} + 2^{-\delta}) + 3.81\cdot 2^{s/4}(2^{\delta/2} + 2^{-\delta/2}) < \\ &< 2^s + \frac{3}{\sqrt{2}}2^{s/2} + 3.81\cdot\frac{1+\sqrt{2}}{\sqrt[4]{2}}2^{s/4} < 2^s + 3.81\cdot 2^{s/2}, \end{aligned}$$

if $2^{s/4} > 4.6$, which holds for $s \ge 9$.

The depth of the constructed decoder circuit is $\lceil \log s\rceil + 1$. □

Corollary 1. *For the complexity and depth of a system of r k-bit comparators (with a common input set), the following estimates hold:*

$$L(\boldsymbol{Q_{k,r}}) < \frac{kr}{\log r}\cdot\frac{\log r + 6}{\log r - \log\log r}, \quad D(\boldsymbol{Q_{k,r}}) \le \lceil \log k\rceil + 2.$$

Proof. By the proven lemma, the overall complexity of the comparison circuit is estimated as

$$L(\boldsymbol{Q_{k,r}}) \le \frac{k}{s}(L(\boldsymbol{K_s}) + r) \le \frac{k}{s}\left(2^s + 3.81\cdot 2^{\frac{s}{2}}\right) + \frac{k}{s}r.$$

Let's choose the parameter $s = \lceil \log r - \log\log r \rceil$, then the estimate take the form:

$$L(\boldsymbol{Q_{k,r}}) < \frac{k}{\log r - \log\log r}\left(\frac{2r}{\log r} + 3.81\sqrt{\frac{2r}{\log r}} + r\right) <$$
$$< \frac{kr}{\log r(\log r - \log\log r)}\left(2 + 3.81\sqrt{\frac{2\log r}{r}} + \log r\right) < \frac{kr}{\log r}\cdot\frac{\log r + 6}{\log r - \log\log r},$$

since $2\log r \leq r$.

The depth of a single comparator, and hence of the entire circuit, does not exceed $\lceil \log k \rceil + 2$. □

4.2 Implementating encoders

The following circuit, given r inputs (comparator outputs), only one of which can take the value 1, calculates the number of the nonzero input. Such a circuit is called an encoder.

Let the output of each circuit comparing β^q and α^{lq} be associated with the number l, or more precisely, its binary notation. Call a *partial disjunction at digit h* the disjunction of all inputs with numbers whose h-th digit is 1. Note that a partial disjunction of the comparator outputs at arbitrary digit h calculates the h-th digit of the number $b \bmod r$. Indeed, on input β only one comparator takes the value 1, namely, the one labeled by $b \bmod r$. If the h-th digit of $b \bmod r$ is 1, then the output of the corresponding comparator participates in the partial disjunction at digit h, which is therefore 1. Otherwise, if the h-th digit of $b \bmod r$ is 0, the output of the comparator is not connected to the input of this disjunction, which therefore takes the value 0. Thus, computing the entire set of partial disjunctions yields a binary representation of the $\lceil \log r \rceil$-digit number $b \bmod r$. In other words, the outputs of the encoder implement partial disjunctions at all digits, up to the $\lceil \log r \rceil$-th.

Lemma 5. *For $s > 0$, the complexity of the 2^s-input encoder is $L(\boldsymbol{V_{2^s}}) \leq 2^{s+1} - 2s - 2$ while the depth is $D(\boldsymbol{V_{2^s}}) = s - 1$.*

Proof. We construct inductively a circuit for which these estimates are satisfied. In fact, we are going to construct a circuit in which all partial disjunctions are calculated for groups of inputs with fixed senior code bits. For $s = 1$, the inputs of the circuit are coded by one bit; the only disjunction coincides in this case with the input numbered by 1, so $L(\boldsymbol{V_{2^1}}) = 0, D(\boldsymbol{V_{2^1}}) = 0$.

Consider the induction step from s to $s+1$. Depending on the value of the most significant $(s+1)$-th digit of the code (0 or 1), all inputs can be divided into two groups, each containing 2^s inputs. For each of the groups, implement a system of s partial disjunctions at the least significant digits. Connecting the corresponding outputs of these subcircuits with disjunction elements, we obtain all correct partial disjunctions for the full set of 2^{s+1} inputs, with the exception of the disjunction at the $(s+1)$-th digit.

The partial disjunction at the highest digit unites all the inputs of one of the subgroups. To calculate it, we can use the results of the preceding computations. Note that the partial disjunction at the s-th digit for a given set of inputs has already been obtained at depth $s - 1$; it calculates the disjunction of half of the inputs of the subgroup under consideration. Note further that the disjunction of half of the remaining inputs as a partial disjunction at the $(s-1)$-th digit of a group of inputs with two fixed highest digits 1 and 0 has also already been calculated at depth $s-2$, and so on.

Thus, a 2^{s+1}-input encoder is obtained from two 2^s-input encoders, and s elements are additionally required to compute the partial disjunction at the highest digit and one for each of the remaining partial disjunctions. Then, by induction, we derive

$$L(\boldsymbol{V_{2^{s+1}}}) \le 2L(\boldsymbol{V_{2^s}}) + 2s \le 2(2^{s+1} - 2s - 2) + 2s = 2^{s+2} - 2(s+1) - 2.$$

Simultaneously, it can be checked that the depth of the constructed circuit is s. The depth of the partial disjunction at $(s+1)$-th digit is s by construction. The depth of the outputs of the particular disjunctions at other digits is greater than the depth of an encoder with 2^s inputs by 1, which implies that their depth is also s. □

Corollary 2. *The complexity of an encoder with r inputs is $L(\boldsymbol{V_r}) \le 2r - 2\lceil \log r\rceil - 2$ while the depth is $D(\boldsymbol{V_r}) = \lceil \log r\rceil - 1$.*

Proof. Let $2^{s+1} \ge r = 2^s + r'$, $r' > 0$. The proof is by induction on r.

The circuit will be structured in the same way as in the special case. Split the set of inputs into two halves: 2^s inputs with a zero leading digit and r' with a leading digit 1. Calculate the partial disjunctions on these subsets (the second of them is encoded, generally speaking, by $\lceil \log r'\rceil$ least significant digits of the input code).

Next, $\lceil \log r'\rceil$ functional elements are needed to obtain the disjunctions of the least significant digits. The same number more are needed to compute the disjunction at the $(s+1)$-th digit. This leads to the recurrence relation

$$\begin{aligned}L(\boldsymbol{V_r}) &\le L(\boldsymbol{V_{2^s}}) + L(\boldsymbol{V_{r'}}) + 2\lceil \log r'\rceil \le \\ &\le (2^{s+1} - 2s - 2) + (2r' - 2\lceil \log r'\rceil - 2) + 2\lceil \log r'\rceil = 2r - 2(s+1) - 2.\end{aligned}$$

The depth of the circuit is $s = \lceil \log r\rceil - 1$. □

Proof of Theorem 5. The proof is obtained by summing up the estimates from Corollaries 1 and 2:

$$\begin{aligned}L(\boldsymbol{\Lambda'_{k,r}}) &\le L(\boldsymbol{Q_{k,r}}) + L(\boldsymbol{V_r}) < \frac{kr}{\log r} \cdot \frac{\log r + 6}{\log r - \log\log r} + 2r, \\ D(\boldsymbol{\Lambda'_{k,r}}) &\le D(\boldsymbol{Q_{k,r}}) + D(\boldsymbol{V_r}) \le (\lceil \log k\rceil + 2) + (\lceil \log r\rceil - 1).\end{aligned}$$

□

4.3 The choice of an auxiliary field

Theorem 6. *Let $2^k - 1 = r_1 r_2 \cdot \ldots \cdot r_w$, where the factors r_i are pairwise coprime, w is bounded, $\rho = \log \max_i r_i$. Then for $k \to \infty$,*

$$L(\boldsymbol{\Lambda_k}) \lesssim \sum_{i=1}^{w} (2 + k/\log r_i) r_i, \quad D(\boldsymbol{\Lambda_k}) \le \rho + O(\log^2 k).$$

P r o o f. By construction,

$$L(\boldsymbol{\Lambda_k}) \le \sum_{i=1}^{w} L(\boldsymbol{\Lambda'_{k,r_i}}) + L(\boldsymbol{H_{k,w}}) + L(\boldsymbol{R_{k,w}}).$$

Let $r_{max} = \max_i r_i$. From $r_{max} \gtrsim 2^{k/w}$ it follows that the estimate of the order of complexity $\boldsymbol{\Lambda'_{k,r_{max}}}$ from Theorem 5 is

$$r_{max}(2 + k/\log r_{max}) \gtrsim O(w2^{k/w}).$$

The complexity of the subcircuits implementing $\boldsymbol{H_{k,w}}$ and $\boldsymbol{R_{k,w}}$, according to Lemmas 2 and 3, is $O(wk^3)$. This quantity is insignificant for the asymptotics, compared to the complexity of the subcircuit implementing $\boldsymbol{\Lambda'_{k,r_{max}}}$.

The depth estimate is verified similarly,

$$D(\boldsymbol{\Lambda_k}) \le D(\boldsymbol{\Lambda'_{k,r_{max}}}) + D(\boldsymbol{H_{k,w}}) + D(\boldsymbol{R_{k,w}}) \le \rho + O(\log^2 k).$$

□

Using the logarithm circuit described above, we make its efficiency dependent on the existence of a "smooth" factorization[4] of $2^k - 1$ into a product of coprime factors. The smaller the maximum of the factors, the more efficient the factorization.

For interpolation, a field containing at least p elements is required. In practice, among several fields of dimensions $k \ge \lceil \log p \rceil$, it is necessary to choose a field with the smoothest order of the multiplicative group.

For example, $GF(2^9)$ is less smooth than $GF(2^{10})$, since $2^9 - 1 = 7 \cdot 73$, and $2^{10} - 1 = 3 \cdot 11 \cdot 31$. But $GF(2^{12})$ is even smoother, since $2^{12} - 1 = 5 \cdot 7 \cdot 9 \cdot 13$.

To evaluate the efficiency of the logarithm operation, let us consider several ways to choose a smooth field for an arbitrary p.

The first of these takes the smallest even of the suitable values of k, $k = 2l \ge \lceil \log p \rceil$, and exploits the factorization of $2^{2l} - 1$ into the always coprime factors $2^l - 1$ and $2^l + 1$. Note that one of these numbers is divisible by 3, whence we have

$$2^{2l} - 1 = \begin{cases} 3^d \frac{2^l - 1}{3^d}(2^l + 1), & l \text{ — even;} \\ 3^d \frac{2^l + 1}{3^d}(2^l - 1), & l \text{ — odd,} \end{cases}$$

where 3^d is the largest power of 3 that divides $2^{2l} - 1$. By Theorem 3, we obtain the following estimates for a circuit computing discrete logarithm in the field $GF(2^{2l})$: the order of complexity is $(8 + o(1))2^l \lesssim (16 + o(1))\sqrt{p}$, the depth is $l + o(l) \lesssim (1/2)\log p$.

A better circuit is obtained by choosing the field $GF(2^k)$, where $k = 6l \ge \lceil \log p \rceil$ (the smallest possible value of k is chosen). The coprime factors $2^{3l} - 1$ and $2^{3l} + 1$ admit a further factorization: $2^{3l} \pm 1 = (2^l \pm 1)(2^{2l} \mp 2^l + 1)$. Since $2^{2l} \pm 2^l + 1 = 3 \bmod (2^l \mp 1)$, then 3 is the only common divisor that the factors in the given factorization can have.

$$2^{6l} - 1 = \begin{cases} 3^{d+1} \frac{2^l - 1}{3^d}(2^l + 1)\frac{2^{2l} + 2^l + 1}{3}(2^{2l} - 2^l + 1), & l \text{ — even;} \\ 3^{d+1} \frac{2^l + 1}{3^d}(2^l - 1)\frac{2^{2l} - 2^l + 1}{3}(2^{2l} + 2^l + 1), & l \text{ — odd.} \end{cases}$$

The complexity of the logarithm circuit for the field $GF(2^{6l})$ is estimated as $(20/3 + o(1))2^{2l} \lesssim (80/3 + o(1))\sqrt[3]{p}$, the depth is $2l + o(l) \lesssim (1/3)\log p$. For $p > 2^8$ (recall

[4]A factorization is "smooth" if all its factors are small. A number is also called smooth if it admits a smooth factorization (see, e.g., [22]).

the example of the field $GF(2^{12})$), the estimates obtained by the second method are better than in the first case.

Further development of the idea of decomposing polynomials of the form $x^k - 1$ into irreducible polynomials over $\mathbb{Z}$ allows us to construct a circuit computing the discrete logarithm with complexity $C(\varepsilon)O(p^\varepsilon)$ and depth $\varepsilon \log p + o(\log p)$, $\varepsilon > 0$. This is the subject of the next section.

4.4 Asymptotic estimate of the efficiency of computing logarithms

Consider the field $GF(2^{k_v l})$, where $k_v = p_1 p_2 \cdot \ldots \cdot p_v$, and $\{p_i\}$ is an increasing sequence of prime numbers.

Theorem 7. *Let $l \in \mathbb{N}$. Then the number $2^{k_v l} - 1$ can be represented as a product of pairwise coprime factors $r_1, r_2, \ldots, r_s$ with*

$$\max_i r_i \le 2^{\varphi(k_v)l} e^{\varphi(k_v)/2^l},$$

where $\varphi(k)$ is the Euler function.

We preface the proof of the theorem with several auxiliary statements. First, we turn to the theory of cyclotomic polynomials.

Let $d \in \mathbb{N}$. A monic polynomial $F_d \in \mathbb{C}[x]$ of minimal possible degree such that its roots are all primitive roots[5] of order d is called the *d-th cyclotomic polynomial.*

The following properties of cyclotomic polynomials are well known (for more details on cyclotomic polynomials, see [23]):

(1) $F_d \in \mathbb{Z}[x]$;
(2) $\deg F_d = \varphi(d)$;
(3) Polynomials $\{F_d\}$ are pairwise coprime;
(4) $x^h - 1 = \prod_{d|h} F_d(x)$.

Lemma 6. *Let $x \ge 1$. Then $F_d(x) \le x^{\varphi(d)} e^{\varphi(d)/x}$.*

Proof. The roots of the polynomial $F_d(x)$ are equal to 1 in absolute value, we denote them by ξ_i. Then

$$F_d(x) = \prod_{i=1}^{\varphi(d)} (x - \xi_i) < \prod_{i=1}^{\varphi(d)} (x + |\xi_i|) = (x+1)^{\varphi(d)} = x^{\varphi(d)} \left(1 + \frac{1}{x}\right)^{\varphi(d)} \le x^{\varphi(d)} e^{\varphi(d)/x}.$$

$\square$

Note that if $x \to \infty$, then $F_d(x) = O(x^{\varphi(d)})$. We will need another lemma on divisibility of numbers (it can be found in [31, problem 8]).

Lemma 7. *Let q be a prime number, and $a \in \mathbb{Z}$. Then*

$$\mathrm{GCD}(a-1, \frac{a^q - 1}{a-1}) = \mathrm{GCD}((a-1)^2, \frac{a^q - 1}{a-1}) = \mathrm{GCD}(a-1, q).$$

Proof. Divide the polynomial $x^{q-1} + \ldots + 1 = \frac{x^q - 1}{x-1}$ by $x - 1$ twice with remainder:

$$x^{q-1} + x^{q-2} + \ldots + 1 =$$
$$= (x-1)^2 \left(x^{q-3} + 3x^{q-4} + \ldots + \frac{(q-1)(q-2)}{2}\right) + (x-1)\frac{q(q-1)}{2} + q.$$

[5]A root of unity of some order is primitive, if it's not a root of any lower order.

Substitute a for x. The subsequent verification of divisibility relations is not difficult. □

P r o o f o f t h e m a i n T h e o r e m 7. The polynomial $x^{k_v}-1$ can be factored into a product of cyclotomic polynomials (property (4))

$$x^{k_v}-1=\prod_{d\mid k_v}F_d(x).$$

The number of factors in this product is 2^v, they correspond to the divisors of k_v.

Under assignment $x=2^l$, we obtain the factorization

$$2^{k_v l}-1=\prod_{d\mid k_v}F_d(2^l) \qquad (*)$$

of the number $2^{k_v l}-1$ into factors not exceeding $2^{\varphi(k_v)l}e^{\varphi(k_v)/2^l}$, which follows from Lemma 6, since $\varphi(k_v)$ is the maximum of the degrees of the polynomials F_d. Let us further investigate the possibility of transforming this decomposition into a product of *pairwise coprime* factors not exceeding $2^{\varphi(k_v)l}e^{\varphi(k_v)/2^l}$.

Consider the following decompositions of the polynomial $x^{k_v}-1$ into a product of two factors (for brevity, we introduce the notation $y_i=x^{k_v/p_i}$):

$$x^{k_v}-1=y_i^{p_i}-1=(y_i-1)(y_i^{p_i-1}+\ldots+1). \qquad (i)$$

It follows from property (4) of cyclotomic polynomials that

$$y_i-1=\prod_{d\mid k_v,\,p_i\nmid d}F_d(x), \qquad y_i^{p_i-1}+\ldots+1=\prod_{d\mid k_v,\,p_i\mid d}F_d(x),$$

therefore, any polynomial $F_d(x)$, $d\mid k_v$, divides some of the two factors on the right-hand side of every expression (i).

Let us show that the only common divisors of the values of two cyclotomic polynomials (under substitution of 2^l) can be prime numbers p_i, $i=2,\ldots,v$.

Consider an arbitrary pair $F_{d_1}(2^l)$ and $F_{d_2}(2^l)$, where $d_1,d_2\mid k_v$, and assume $d_1<d_2$. Then there necessarily exists a number p_i such that $p_i\mid d_2$ and $p_i\nmid d_1$. Consider the factorization (i). The polynomial $F_{d_1}(x)$ divides the first of the factors, and $F_{d_2}(x)$ divides the second. From Lemma 7 it follows that

$$\mathrm{GCD}(\,(y_i-1)\mid_{x=2^l},\ (y_i^{p_i-1}+\ldots+1)\mid_{x=2^l}\,)\in\{1,p_i\}.$$

Hence,

$$\mathrm{GCD}(F_{d_1}(2^l),\ F_{d_2}(2^l))\in\{1,p_i\}.$$

By extracting $p_i^{c_i}$, $i=2,\ldots,v$, into separate factors, where c_i is the multiplicity of p_i in the product, we obtain a factorization into pairwise coprime factors. It remains to show that the newly formed factors $p_i^{c_i}$ satisfy $2^{\varphi(k_v)l}e^{\varphi(k_v)/2^l}$.

If p_i divides only one of the factors of the original decomposition $(*)$, then there is nothing to prove. Consider the case when p_i divides two factors of the decomposition $(*)$ $F_{d_1}(2^l)$ and $F_{d_2}(2^l)$, $d_1\neq d_2$. It follows from Lemma 7 that in any decomposition (j), where $j\neq i$, the polynomials $F_{d_1}(x)$ and $F_{d_2}(x)$ divide the same factor on the right-hand side (otherwise p_i cannot be a common divisor). We will show that in the decomposition (i) they divide different factors.

Note that an arbitrary divisor d of k_v is uniquely determined if it is known for any number p_s, $s=1,\ldots,v$, whether it divides d or not. Therefore, the polynomial $F_d(x)$ is uniquely determined by its belonging to one of the factors in each decomposition (s).

It follows from this remark that if in the decomposition (i) (as in all the others) the polynomials $F_{d_1}(x)$ and $F_{d_2}(x)$ divided the same factor, they would coincide, which would contradict the condition $d_1 \neq d_2$. Thus, in the decomposition (i) they divide different factors.

Now suppose that another factor $F_{d_3}(2^l)$ of the original factorization $(*)$, different from the two indicated, is divisible by p_i. Reasoning similarly, we conclude that the polynomial $F_{d_3}(x)$ in all decompositions except the (i)-th divides the same factors as the pair $F_{d_1}(x)$, $F_{d_2}(x)$. But then, depending on which of the factors of expansion (i) it divides, it coincides either with $F_{d_1}(x)$ or with $F_{d_2}(x)$, which contradicts the assumption.

Thus, p_i can divide at most two numbers from the set $F_d(2^l)$, $d \mid k_v$, simultaneously. Moreover, as follows from Lemma 7, if p_i divides exactly two factors in the expansion $(*)$, then one of the factors (namely, the one that corresponds to the polynomial dividing $y_i - 1$ in the decomposition (i)) is divisible by $p_i^{c_i-1}$. Consequently, a factor $p_i^{c_i}$ can, at most, be p_i times greater than any of the factors $F_d(2^l)$ of the initial decomposition for which $d \mid k_v$ and $p_i \nmid d$ hold. In this case $\varphi(d) \le \varphi(k_v)/(p_i - 1)$. By Lemma 6, we have

$$p_i^{c_i} \le p_i \cdot \max_{d \mid k_v,\, p_i \nmid d} F_d(2^l) \le p_i 2^{\varphi(k_v)l/(p_i-1)} e^{\varphi(k_v)/(2^l(p_i-1))}.$$

Let us show that

$$p_i 2^{\varphi(k_v)l/(p_i-1)} e^{\varphi(k_v)/(2^l(p_i-1))} < 2^{\varphi(k_v)l} e^{\varphi(k_v)/2^l},$$

where $1 < i \le v$, $l \ge 1$. First, consider the case $i = v = 2$, $l = 1$ (i.e. $p_i = 3$, $k_v = 6$). After substituting the parameters into the inequality, it takes the form $6\sqrt{e} < 4e$, which is true. The inequality especially remains true with increasing parameters v and (or) l. If $i > 2$, then $p_i < 2^{\varphi(k_i)/2}$, which can be verified, for example, by induction as follows. For $i = 3$, the inequality holds due to $5 < 2^4$. If $p_{i-1} < 2^{\varphi(k_{i-1})/2}$, then

$$p_i < 2p_{i-1} < p_{i-1}^{p_i-1} < 2^{\varphi(k_{i-1})(p_i-1)/2} = 2^{\varphi(k_i)/2}.$$

Here we applied the well-known inequality $p_i < 2p_{i-1}$ (Bertrand's postulate). From the proved intermediate inequality we deduce

$$p_i 2^{\varphi(k_v)l/(p_i-1)} e^{\varphi(k_v)/(2^l(p_i-1))} < 2^{\varphi(k_i)/2+\varphi(k_v)l/(p_i-1)} e^{\varphi(k_v)/2^l} <$$
$$< 2^{\varphi(k_v)l(1/2+1/(p_i-1))} e^{\varphi(k_v)/2^l} < 2^{\varphi(k_v)l} e^{\varphi(k_v)/2^l}.$$

Thus, the theorem is completely proven. □

Note that the number of factors in the constructed factorization does not exceed $2^v + v - 1$, where 2^v is the number of factors in the initial factorization $(*)$, plus no more than $v - 1$ factors are additionally factored out.

For illustration, consider the example $v = 3$, $k_3 = 30$.

$$x^{30} - 1 = F_1F_2F_3F_5F_6F_{10}F_{15}F_{30};$$
$$F_1(x) = x - 1, \quad F_2(x) = x + 1, \quad F_3(x) = x^2 + x + 1,$$
$$F_5(x) = x^4 + x^3 + x^2 + x + 1, \quad F_6(x) = x^2 - x + 1,$$
$$F_{10}(x) = x^4 - x^3 + x^2 - x + 1, \quad F_{15}(x) = x^8 - x^7 + x^5 - x^4 + x^3 - x + 1,$$
$$F_{30}(x) = x^8 + x^7 - x^5 - x^4 - x^3 + x + 1.$$

Let $l = 1$. Substituting $x = 2$ yields a factorization of $2^{30} - 1$ (the order of the factors is preserved)

$$2^{30} - 1 = 1 \cdot 3 \cdot 7 \cdot 31 \cdot 3 \cdot 11 \cdot 151 \cdot 331.$$

We separate into single factors the occurrences of the first two odd primes in the product: 3 (occurs twice, $F_2(2) = F_6(2) = 3$) and 5 (never). Finally, as guaranteed by the theorem, we obtain a satisfying all requirements decomposition

$$2^{30} - 1 = 7 \cdot 9 \cdot 11 \cdot 31 \cdot 151 \cdot 331,$$

which in this case coincides with the canonical factorization of $2^{30} - 1$ into prime factors.

Using Theorem 7, it can be shown that in the field $GF(2^{30l})$ the logarithm can be computed with complexity $O(2^{8l})$, which for $2^{30l} > p \geq 2^{30(l-1)}$ corresponds to the estimate $O(p^{4/15})$. However, the multiplicative constant in this bound is too large.

Further we will exploit the following fact:

$$\frac{c_1}{\log(v+1)} < \frac{\varphi(k_v)}{k_v} = \prod_{i=1}^{v} \frac{p_i - 1}{p_i} < \frac{c_2}{\log v} \to 0, \quad \text{for } v \to \infty.$$

It is easy to show that $c_1 > e^{-5/2}$, and $c_2 < \ln 2 = 0.693\ldots$ More precise estimates are provided in [28][6].

Theorem 8. *Let $v \in \mathbb{N}$, $p \geq 2^{k_v}$. Then there exists a field of characteristic 2 containing at least p elements in which the complexity of taking the logarithm asymptotically (as $p \to \infty$) does not exceed*

$$\log(v+1) 2^{\varphi(k_v)+v+4} e^{\varphi(k_v) p^{-1/k_v}} p^{\varphi(k_v)/k_v},$$

and the depth does not exceed

$$(\varphi(k_v)/k_v) \log p + 2\varphi(k_v) + O(\log^2 \log p).$$

Proof. For a given p, consider the field $GF(2^{k_v l})$, where l satisfies $2^{k_v(l-1)} < p \leq 2^{k_v l}$. According to Theorem 7, the number $2^{k_v l} - 1$ can be factored into a product of coprime factors not exceeding $2^{\varphi(k_v) l} e^{\varphi(k_v)/2^l}$.

Using Theorem 6, we estimate the circuit depth of the logarithm in the field $GF(2^{k_v l})$ as

$$\begin{aligned} D(\boldsymbol{\Lambda}_{\boldsymbol{k_v l}}) &\leq \log((2^l e^{1/2^l})^{\varphi(k_v)}) + O(\log^2(k_v l)) < \\ &< \varphi(k_v)(l+1) + O(\log^2 \log p) < \varphi(k_v)(2 + (\log p)/k_v) + O(\log^2 \log p) < \\ &< (\varphi(k_v)/k_v) \log p + 2\varphi(k_v) + O(\log^2 \log p). \end{aligned}$$

The same theorem 6 allows to estimate the order of complexity of the circuit,

$$L(\boldsymbol{\Lambda}_{\boldsymbol{k_v l}}) \lesssim (2^v + v)(2 + k_v/\log r_{max}) r_{max} \leq (2^v + v)(2 + k_v/\varphi(k_v)) 2^{\varphi(k_v) l} e^{\varphi(k_v)/2^l}.$$

[6] According to Mertens' theorem, this expression has an asymptotic behavior of $\frac{e^\gamma}{\ln v}$, where γ is Euler's constant (see also [28]).

The following inequality holds:

$$(2^v + v)(2 + k_v/\varphi(k_v)) < 2^{v+4}\log(v+1),$$

which can be verified for $v = 1, \dots, 4$ by direct calculation, and for $v \ge 5$ one can apply the inequality $k_v/\varphi(k_v) < e^{5/2}\log(v+1)$, then

$$(2^v + v)(2 + k_v/\varphi(k_v)) < \log(v+1)2^v \left[\left(1 + \frac{v}{2^v}\right)\left(e^{5/2} + \frac{2}{\log(v+1)}\right)\right].$$

The expression in square brackets is a monotonically decreasing function of v, and from the fact that for $v = 5$ its value is less than 16, the stated inequality follows.

Further,

$$2^{\varphi(k_v)l} = 2^{\varphi(k_v)}(2^{k_v(l-1)})^{\varphi(k_v)/k_v} < 2^{\varphi(k_v)}p^{\varphi(k_v)/k_v}.$$

Finally, since $2^l \ge p^{-1/k_v}$, we deduce

$$e^{\varphi(k_v)/2^l} \le e^{\varphi(k_v)p^{-1/k_v}}.$$

Combining all the above inequalities, we obtain the desired result. $\square$

Corollary 3. *Let $\varepsilon > 0$. Then there exists a field of characteristic 2 containing at least p elements in which taking logarithms is performed with complexity (as $p \to \infty$) not exceeding $C_1(\varepsilon)O(p^\varepsilon)$ and depth $\varepsilon\log p + C_2(\varepsilon) + O(\log^2\log p)$.*

Proof. We choose v such that $\frac{\varphi(k_v)}{k_v} \le \varepsilon$ (for example, $v = \lceil 2^{0.7/\varepsilon}\rceil$ will do), and apply the proven theorem. $\square$

5 Refining the complexity bound

Consider another way to calculate x^{-1}. Represent a number $n-1$ as $n_1n_2 + n_3$, where $0 \le n_i \le \lceil\sqrt{n-1}\,\rceil$. From this representation we derive the representation of the number $2^n - 2$ as $N_1N_2 + N_3$, where

$$N_1 = 2 + 2^2 + \ldots + 2^{n_1}, \quad N_2 = 1 + 2^{n_1} + 2^{2n_1} + \ldots + 2^{(n_2-1)n_1},$$
$$N_3 = 2^{n_1n_2+1} + 2^{n_1n_2+2} + \ldots + 2^{n_1n_2+n_3}.$$

The weight of each of the numbers N_i is n_i, $i = 1, 2, 3$.

We use the formula

$$x^{-1} = (x^{N_1})^{N_2}x^{N_3},$$

which shows that inversion reduces to three relatively small-weight exponentiations and one multiplication in the field $GF(2^n)$. More formally,

$$\boldsymbol{I_n}(x) = \boldsymbol{M_n}(\boldsymbol{E_{n,n_2}} \cdot \boldsymbol{E_{n,n_1}}(x), \boldsymbol{E_{n,n_3}}(x)).$$

Implementing the operations $\boldsymbol{E_{n,n_i}}$ by the algorithm from §3, we can estimate the complexity of such a circuit as

$$L(\boldsymbol{I_n}) \le L(\boldsymbol{M_n}) + \sum_{i=1}^{3} L(\boldsymbol{E_{n,n_i}}) \lesssim \sum_{i=1}^{3} \frac{\log(n_in)}{\log(n_i^2n)}n_i^2n^2 \le \frac{9}{4}n^3.$$

The depth of the circuit is estimated as

$$D(\boldsymbol{I_n}) \le D(\boldsymbol{M_n}) + \max\{D(\boldsymbol{E_{n,n_2}}) + D(\boldsymbol{E_{n,n_1}}), D(\boldsymbol{E_{n,n_3}})\} \lesssim \\ \lesssim 2\log n + (4+\varepsilon)\log n + 4.44\log n \lesssim 10.44\log n.$$

This technique can be generalized as follows.

Theorem 9. *Let $r \in \mathbb{N}$ and $q = \lceil \sqrt[r]{n}\,\rceil$. Then the circuit complexity and depth of the inversion in the field $GF(2^n)$ are estimated as*

$$L(\boldsymbol{I_n}) \le (2r-1)L(\boldsymbol{E_{n,q}}) + (r-1)L(\boldsymbol{M_n}),$$
$$D(\boldsymbol{I_n}) \le 2D(\boldsymbol{E_{n,q}}) + D(\boldsymbol{M_n}) + (r-2)\max\{D(\boldsymbol{E_{n,q}}), D(\boldsymbol{M_n})\}.$$

Proof. Let $n-1 = [m_r, m_{r-1}, \ldots, m_1]$ in the number system with base q, i.e.,

$$n-1 = q^{r-1}m_r + q^{r-2}m_{r-1} + \ldots + qm_2 + m_1,$$

where all m_i do not exceed $\sqrt[r]{n}$.

Set

$$N_i = 1 + 2^{q^{i-1}} + 2^{2q^{i-1}} + \ldots + 2^{(q-1)q^{i-1}} = \frac{2^{q^i}-1}{2^{q^{i-1}}-1},$$

Then

$$N_1N_2\cdot\ldots\cdot N_i = 2^{q^i}-1.$$

Let $M_i = 0$ in the case $m_i = 0$, and

$$M_i = 2^{[m_r,\ldots,m_{i+1},0,\ldots,0]+1}(1 + 2^{q^{i-1}} + \ldots + 2^{(m_i-1)q^{i-1}})$$

otherwise. Then,

$$(2^{q^{i-1}}-1)M_i = 2(2^{[m_r,\ldots,m_i,0,\ldots,0]} - 2^{[m_r,\ldots,m_{i+1},0,\ldots,0]}),$$

where the square brackets contain an r-digit number in the q-ary number system. Summing these equalities over $i = 1,\ldots,r$, we obtain

$$2^n - 2 = N_1\cdot\ldots\cdot N_{r-1}M_r + N_1\cdot\ldots\cdot N_{r-2}M_{r-1} + \ldots + N_1M_2 + M_1,$$

where the weight of each number N_i is q, and the weight of M_i is m_i.

We will construct the inversion circuit based on the formula

$$x^{-1} = x^{2^n-2} = \Big(\big(\ldots((x^{N_1})^{N_2})\ldots\big)^{N_{r-1}}\Big)^{M_r} \times \\ \times \Big(\big(\ldots((x^{N_1})^{N_2})\ldots\big)^{N_{r-2}}\Big)^{M_{r-1}} \cdot\ldots\cdot (x^{N_1})^{M_2}x^{M_1}.$$

Let us write out the sequence of calculations step by step.

0.	x;		
1.	$x_1 = x^{N_1}$,	$y_1 = x^{M_1}$;	
2.	$x_2 = x_1^{N_2}$,	$y_2 = x_1^{M_2}$,	$z_2 = y_1$;
$i = 3\ldots r-1$.	$x_i = x_{i-1}^{N_i}$,	$y_i = x_{i-1}^{M_i}$,	$z_i = z_{i-1}\cdot y_{i-1}$;
r.		$y_r = x_{r-1}^{M_r}$,	$z_r = z_{r-1}\cdot y_{r-1}$;
$r+1$.			$x^{-1} = z_r\cdot y_r$.

The computation circuit includes no more than $2r-1$ subcircuits implementing exponentiations with weights at most q and no more than $r-1$ subcircuits implementing field multiplications. The depth of layers 1 and 2 of the circuit can be estimated as $D(\boldsymbol{E_{n,q}})$, of layers $3,\ldots,r$ — as $\max\{D(\boldsymbol{E_{n,q}}), D(\boldsymbol{M_n})\}$, and of layer $r+1$ — as $D(\boldsymbol{M_n})$, from which the depth estimate stated by the theorem follows. □

Corollary 4. *Let $r \in \mathbb{N}$ and $q = \lceil \sqrt[r]{n}\, \rceil$. Then the circuit complexity and depth of the division in the field $GF(2^n)$ are estimated as*

$$L(\boldsymbol{\Delta_n}) \le (2r-1)L(\boldsymbol{E_{n,q}}) + rL(\boldsymbol{M_n}),$$
$$D(\boldsymbol{\Delta_n}) \le D(\boldsymbol{E_{n,q}}) + D(\boldsymbol{M_n}) + (r-1)\max\{D(\boldsymbol{E_{n,q}}), D(\boldsymbol{M_n})\}.$$

P r o o f. To calculate y/x, it is sufficient to embed multiplication by y into the circuit from Theorem 9 computing x^{-1}:

$$\begin{array}{llll}
0. & x,\quad y; & & \\
1. & x_1 = x^{N_1}, & y_1 = x^{M_1}, & z_1 = y; \\
i = 2 \dots r-1. & x_i = x_{i-1}^{N_i}, & y_i = x_{i-1}^{M_i}, & z_i = z_{i-1} \cdot y_{i-1}; \\
r. & & y_r = x_{r-1}^{M_r}, & z_r = z_{r-1} \cdot y_{r-1}; \\
r+1. & & & y/x = z_r \cdot y_r.
\end{array}$$

□

Under the natural assumption $D(\boldsymbol{E_{n,q}}) \ge D(\boldsymbol{M_n})$, the depth of the constructed inversion and division circuits is estimated as

$$D(\boldsymbol{I_n}), D(\boldsymbol{\Delta_n}) \le rD(\boldsymbol{E_{n,q}}) + D(\boldsymbol{M_n}).$$

Substituting the estimates of Theorem 2 into Theorem 9 and Corollary 4, and employing the estimates $L(\boldsymbol{M_n}) = O(n^2)$, $D(\boldsymbol{M_n}) \le (2 + o(1)) \log n$, we obtain

Corollary 5. *Let $r \in \mathbb{N}$. Then for inversion and division in the field $GF(2^n)$ one can construct circuits with complexity and depth* (*as $n \to \infty$*)

$$L(\boldsymbol{I_n}), L(\boldsymbol{\Delta_n}) \le \left(2r - 3 + \frac{5}{r+2} + o(1)\right) n^{2+\frac{2}{r}},$$
$$D(\boldsymbol{I_n}), D(\boldsymbol{\Delta_n}) \le (2r + 6.44 + o(1)) \log n.$$

Thus, an inversion circuit with logarithmic depth and almost quadratic complexity is constructed. In the next section, it will be shown that it is possible to construct circuits with depth $O(\log n)$ and complexity $o(n^2)$.

6 A subquadratic complexity algorithm

Here is a brief description of the modified algorithm for raising an element $x \in GF(2^n)$ to a power $M = \sum_i 2^{e_i}$ of weight m.

1. Compute all $x^{2^{e_i}} = f_i(t)$, $i = 1, \dots, m$. Let $f(t) = f_1(t) \cdot \ldots \cdot f_m(t)$. Set $p = m(n-1)+1$, choose a field $GF(2^k)$ containing at least p elements; in it, choose a set of elements $\alpha_1, \dots, \alpha_p$.

2. Calculate all possible $f_i(\alpha_j) \in GF(2^k)$, where $i = 1, \dots, m$, $j = 1, \dots, p$.

3. For all j, calculate the products $f_1(\alpha_j) \cdot \ldots \cdot f_m(\alpha_j) = f(\alpha_j)$.

4. Given the values $f(\alpha_j)$, $j = 1, \dots, p$, reconstruct a polynomial $f(t) \bmod m_n(t)$, $\deg f < p$, representing x^M.

In this algorithm, the general computation scheme of the original algorithm from §3 is completely preserved, only the implementation method changes. To implement linear steps 1, 2, the so-called matrix acceleration is used. Also, we consider an alternative implementation of step 4 (steps 4, 5 of the original algorithm). The implementation of step 3 remains unchanged. Recall that, according to Corollary 3, step 3 is performed by a circuit of complexity $O(m^{2+\varepsilon} n^{1+\varepsilon})$ and depth $(\varepsilon + o(1)) \log(mn)$, where $\varepsilon = const > 0$.

6.1 Generalized modular composition

As is known (see, e.g., [25]), the $O(ns/\log n)$ complexity upper bound for a linear mapping of dimension $n \times s$ cannot be improved in the general case. However, it can be reduced for mappings satisfying some extra conditions. Consider *homomorphic* mappings that preserve both the addition and multiplication operations.

The following lemma is a simple generalization of a result of Brent and Kung [6] on the complexity of modular composition (the composition of two polynomials modulo a third). By $\boldsymbol{T_{q,r,s}}$ we denote the operation of multiplication of binary matrices of size $q \times r$ and $r \times s$.

Lemma 8. *Let $\boldsymbol{G}$ be a homomorphism from $GF(2)[t]$ to a set V that has the structure of a vector space of dimension s over $GF(2)$ with the operation of multiplication. Let $\boldsymbol{G_n}$ denote the restriction of $\boldsymbol{G}$ to the set of polynomials of degree at most $n-1$, and let $rq \geq n$. Then*

$$L(\boldsymbol{G_n}) \leq L(\boldsymbol{T_{q,r,s}}) + (q-1)(L(\boldsymbol{M_V}) + s), \quad D(\boldsymbol{G_n}) \leq D(\boldsymbol{T_{q,r,s}}) + D(\boldsymbol{M_V}) + \lceil \log q \rceil,$$

where $\boldsymbol{M_V}$ is the multiplication operator in V.

Proof. Let $f(t) \in GF(2)[t]$, $\deg f \leq n-1$. Write

$$f(t) = f_0(t) + f_1(t)t^r + \ldots + f_{q-1}(t)t^{(q-1)r},$$

where $\deg f_i < r$. By assumption,

$$\boldsymbol{G_n}(f) = \boldsymbol{G_r}(f_0) + \boldsymbol{G_r}(f_1)\boldsymbol{G_n}(t^r) + \ldots + \boldsymbol{G_r}(f_{q-1})\boldsymbol{G_n}(t^{(q-1)r}),$$

where $\boldsymbol{G_r}$ is the restriction of $\boldsymbol{G}$ to the set of polynomials of degree less than r. In particular, $\boldsymbol{G_r}$ is a linear mapping of dimension $r \times s$. Calculation of all $\boldsymbol{G_r}(f_i)$, $i = 0, \ldots, q-1$, corresponds to multiplying the $q \times r$-matrix of coefficients of the polynomials $f_i(t)$ by the $r \times s$-matrix of coefficients $\boldsymbol{G_r}(t^j) \in V$.

Assuming that all $\boldsymbol{G_n}(t^{ir})$, $i = 1, \ldots, q-1$, are precomputed (this is the case in the circuit implementation), to compute $\boldsymbol{G_n}(f)$ it remains to perform $q-1$ multiplications and sum q vectors from V. □

Corollary 6 (Brent, Kung, 1978). *Let*

$$\boldsymbol{C_{g,h}}(f) = f(g(t)) \bmod h(t),$$

where $g(t)$, $h(t)$ are fixed polynomials of degree $n-1$ and n respectively, let also $rq \geq n$. Then

$$L(\boldsymbol{C_{g,h}}) \leq L(\boldsymbol{T_{q,r,n}}) + (q-1)(L(\boldsymbol{M_n}) + n),$$
$$D(\boldsymbol{C_{g,h}}) \leq D(\boldsymbol{T_{q,r,n}}) + D(\boldsymbol{M_n}) + \lceil \log q \rceil.$$

A special case of modular composition is the Frobenius operation: raising an element $x \in GF(2^n)$ to a power of the form 2^l. Indeed, let $x = f(t)$ in the polynomial representation, then

$$f^{2^l}(t) \bmod m_n(t) = f(t^{2^l}) \bmod m_n(t) =$$
$$= f(t^{2^l} \bmod m_n(t)) \bmod m_n(t) = f(\xi_l(t)) \bmod m_n(t),$$

where $\xi_l(t) = t^{2^l} \bmod m_n(t)$. The Frobenius operation (denoted by $\boldsymbol{S_{n,l}}$) is an automorphism of the field $GF(2^n)$.

Lemma 9. *The Frobenius operation in the field $GF(2^n)$ is implemented with complexity and depth*

$$L(\boldsymbol{S_{n,l}}) = O(n^{1.667}), \qquad D(\boldsymbol{S_{n,l}}) = O(\log n).$$

Proof. In the estimates of Corollary 6 we substitute $q, r \sim \sqrt{n}$. From the Schönhage multiplication algorithm [29] we have $L(\boldsymbol{M_n}) = O(n \log n \log\log n)$ and $D(\boldsymbol{M_n}) = O(\log n)$. The operation $\boldsymbol{T_{q,r,n}}$, i.e. the multiplication of a matrix of size $\sqrt{n} \times \sqrt{n}$ by a matrix of size $\sqrt{n} \times n$, is performed, as shown in [16], with complexity $O(n^{1.667})$. In addition, it is known (see, e.g., [3, Section 4.3]) that any method of matrix multiplication admits implementation by a circuit of logarithmic depth with an increase in the order of complexity by n^{ε}. To complete the proof, we choose ε within the rounding error of the constant from [16] to 1.667. $\square$

It follows from Lemma 9 that the complexity of implementing step 1 of the algorithm is $O(mn^{1.667})$.

The value of a polynomial at a fixed point is also obtained by the action of a homomorphic transform. Let, as before, $p = m(n-1)+1$, $m < n$, and let some set $\{\alpha_1, \dots, \alpha_p\} \subset GF(2^k)$ be chosen. By $\boldsymbol{C^m_{k,n}}$ denote the operation of evaluating of m polynomials of degree at most $n-1$ at the points α_i.

Lemma 10. *The operation $\boldsymbol{C^m_{k,n}}$ is implemented by a circuit of complexity and depth*

$$L(\boldsymbol{C^m_{k,n}}) = O((mn)^{1.667}k) + O((mn)^{1.5})L(\boldsymbol{M_k}), \qquad D(\boldsymbol{C^m_{k,n}}) = O(\log(nk)).$$

Proof. The mapping $\boldsymbol{C^m_{k,n}}$ can be viewed as a union of m operators $\boldsymbol{C_{k,n,\alpha_i}}$ in the notation adopted in §2. Apply Lemma 8, setting $\boldsymbol{G_n} = \boldsymbol{C^m_{k,n}}$ and $V = GF(2^k)^p$. Note that instead of m independent matrix multiplications of type $\boldsymbol{T_{q,r,pk}}$, it suffices to perform a single multiplication of type $\boldsymbol{T_{qm,r,pk}}$, since the coefficient matrix $\boldsymbol{G_r}(t^j)$ is the same for all matrix products.

So, we obtain estimates

$$L(\boldsymbol{C^m_{k,n}}) \le L(\boldsymbol{T_{qm,r,pk}}) + m(q-1)(L(\boldsymbol{M_V}) + pk),$$

$$D(\boldsymbol{C^m_{k,n}}) \le D(\boldsymbol{T_{qm,r,pk}}) + D(\boldsymbol{M_V}) + \lceil \log q \rceil,$$

where $\boldsymbol{M_V}$ is the componentwise multiplication over $GF(2^k)$. Thus, $L(\boldsymbol{M_V}) \le pL(\boldsymbol{M_k})$, $D(\boldsymbol{M_V}) = D(\boldsymbol{M_k})$. Let $r \sim \sqrt{p}$, $q \sim n/r$. Regarding $\boldsymbol{T_{qm,r,pk}}$ as performing k multiplications of matrices of size $\sqrt{mn} \times \sqrt{mn}$ by matrices of size $\sqrt{mn} \times mn$, we finally obtain the required estimates. $\square$

From Lemma 10 follows the complexity estimate $O((mn)^{1.667}k)$ of step 2 of the algorithm.

6.2 Modular interpolation

Turn to step 4. In the previously introduced notations, step 4 implements the operator $\boldsymbol{B_{n,p}} \cdot \boldsymbol{F^{-1}_{k,p}}$, where $\boldsymbol{F^{-1}_{k,p}}$ reconstructs the coefficients of a polynomial over $GF(2)$ of degree no greater than $p-1$ from its values on a set of p elements from $GF(2^k)$, and $\boldsymbol{B_{n,p}}$ reduces a polynomial of degree $p-1$ modulo $m_n(t)$. The construction presented below is actually a modification of the algorithm [1, par. 8.7]

Lemma 11. *Let $r = \lceil p/q \rceil$, $s\,q \le n$. Then*

$$L(\boldsymbol{B_{n,p}} \cdot \boldsymbol{F^{-1}_{k,p}}) \le O\left(\frac{k^2mnq}{\log q}\right) + O(rs)L(\boldsymbol{M_{q,k}}) + O\left(\frac{rn}{s^2q}\right) L(\boldsymbol{M_{sq,k}}) + 2L(\boldsymbol{M_n}),$$

$$D(\boldsymbol{B_{n,p}} \cdot \boldsymbol{F^{-1}_{k,p}}) \le O(\log(kn)) + D(\boldsymbol{M_{q,k}}) + D(\boldsymbol{M_{sq,k}}),$$

where $\boldsymbol{M_{q,k}}$ is the operation of multiplication of polynomials of degree $q-1$ over $GF(2^k)$.

Proof. According to the Lagrange interpolation formula,

$$\boldsymbol{F^{-1}_{k,p}}(f(\alpha_1), \ldots, f(\alpha_p)) = f(t) = \sum_{i=1}^{p} f(\alpha_i) l_i(t),$$

where $l_i(t)$ are fundamental Lagrange polynomials whose coefficients depend only on constants $\alpha_1, \ldots, \alpha_p$:

$$l_i(t) = \prod_{j \ne i} \frac{t - \alpha_j}{(\alpha_i - \alpha_j)}.$$

Split the set $\{\alpha_1, \ldots, \alpha_p\}$ into subsets $A_1, \ldots, A_r$, with q items in each (except, perhaps, the last one — but further for convenience we will assume that $|A_r| = q$). Represent $f(t)$ as

$$f(t) = \sum_{i=1}^{r} \varphi_i(t)\lambda_i(t),$$

where

$$\varphi_i(t) = \sum_{\alpha_l \in A_i} f(\alpha_l) \frac{\prod_{j \ne l,\, \alpha_j \in A_i}(t - \alpha_j)}{\prod_{j \ne l}(\alpha_l - \alpha_j)}, \qquad \lambda_i(t) = \prod_{\alpha_j \notin A_i} (t - \alpha_j).$$

Note that the coefficients of any polynomial $\varphi_i(t)$ are linear combinations with respect to $\{f(\beta) \mid \beta \in A_i\}$, and $\deg \varphi_i \le q - 1$. The polynomials $\lambda_i(t)$ are fixed.

Let $v = \lceil r/s \rceil$ (but for convenience assume $r = sv$). Set

$$\Lambda_j(t) = \mathrm{GCD}(\,\lambda_{js+1}(t),\, \lambda_{js+2}(t), \ldots, \lambda_{(j+1)s}(t)\,), \qquad j = 0, \ldots, v-1.$$

Denote $\mu_{js+l}(t) = \lambda_{js+l}(t)/\Lambda_j(t)$, for all $l = 1, \ldots, s$ and $j = 0, \ldots, v-1$. Obviously, $\deg \mu_i = (s-1)q$. So we have

$$f(t) = \sum_{j=0}^{v-1} \Lambda_j(t) \sum_{l=1}^{s} \varphi_{js+l}(t)\mu_{js+l}(t). \qquad (\triangle)$$

Finally,

$$f(t) \bmod m_n(t) = \sum_{j=0}^{v-1} (\Lambda_j(t) \bmod m_n(t)) \sum_{l=1}^{s} \varphi_{js+l}(t)\mu_{js+l}(t) \bmod m_n(t).$$

Consider the following sequence of computations.

4.1. Compute all $\varphi_i(t)$, $i = 1, \ldots, r$, via linear operators of dimension $kq \times kq$.

4.2. Compute the products $\varphi_i(t)\mu_i(t)$, each of which (by splitting the polynomial $\mu_i(t)$ of higher degree into blocks) can be performed using $s-1$ multiplications of polynomials of degree $q-1$ (operations $\boldsymbol{M_{q,k}}$) followed by reduction of similar terms.

4.3. Multiply the polynomials $\Lambda_j(t) \bmod m_n(t)$ by the corresponding sums $\sum_l \varphi_{js+l}(t)\mu_{js+l}(t)$ of products obtained in the previous step, $j = 0, \dots, v-1$. Each of the multiplications is performed via $\lceil n/sq \rceil$ operations $\boldsymbol{M_{sq,k}}$ followed by reduction of similar terms.

4.4. Summing all the polynomials computed in the previous step, we obtain a polynomial of degree at most $2n-1$ with coefficients from $GF(2)$, which is guaranteed by the conditions of the lemma. Its reduction modulo $m_n(t)$ is performed via two multiplications and addition of polynomials of degree $n-1$ (see §2.3).

The terms in the complexity estimate stated by the lemma correspond to the steps of this algorithm in the same order. These terms also absorb the complexity of the addition operations performed at different stages. The same is true for the depth (the depth of step 4.4 is taken into account in the first term). □

The choice of parameters q, r, s depends on the algorithm for multiplying polynomials over $GF(2^k)$. In relation to the construction of an exponentiation circuit of logarithmic depth, a restriction on the depth of the algorithm should be imposed, $D(\boldsymbol{M_{n,k}}) = O(\log(kn))$ (then the total depth of the algorithm of Lemma 11 will be $O(\log(kn))$).

Let us discuss known algorithms for multiplying polynomials of degree $n-1$ with depth $O(\log n)$ over the field of coefficients.

Obviously, the standard algorithm has depth $O(\log kn)$, since all multiplications in it are performed at one level (and addition operations in the field $GF(2^k)$ are performed with depth one, as in $GF(2)$). The same is true for Karatsuba's method [18], as well as for the more general Toom's method with complexity estimate $O(n^{\log_{l+1}(2l+1)})$ operations over $GF(2^k)$, $l \in \mathbb{N}$ (see [33]).

Note that already using Karatsuba's method, we can obtain an acceptable complexity estimate for step 4. Indeed, let $L(\boldsymbol{M_{n,k}}) = O(n^{\log 3}k^2)$ (for multiplication in $GF(2^k)$ the standard algorithm is applied). Choose the parameters to provide $q \sim sq^{\log 3 - 1} \sim n(sq)^{\log 3 - 2}$. So we obtain

$$q \sim n^{1/(1+(2-\log 3)(3-\log 3))} \sim n^{0.63}, \qquad s \sim n^{(2-\log 3)/(1+(2-\log 3)(3-\log 3))} \sim n^{0.26},$$

from which it follows that step 4 can be implemented with complexity $O(mk^2n^{1.631})$.

Let us, however, derive a stronger estimate using the method of Schönhage [29]. This method is based on the Fourier transform, due to which all multiplications are also performed at one level, and the depth satisfies the estimate $O(\log kn)$.

In the estimates of Lemma 10, substitute $\boldsymbol{M_{n,k}} = O(nk^2 \log n \log\log n)$. From the condition $q \sim s \sim n/qs$ we find that $q, s \sim \sqrt[3]{n}$. Therefore, we obtain the following result.

Lemma 12. *The operator $\boldsymbol{B_{n,p}} \cdot \boldsymbol{F_{k,p}^{-1}}$ is implemented with complexity and depth*

$$L(\boldsymbol{B_{n,p}} \cdot \boldsymbol{F_{k,p}^{-1}}) \le O(mn^{4/3}k^2 \log n \log\log n), \qquad D(\boldsymbol{B_{n,p}} \cdot \boldsymbol{F_{k,p}^{-1}}) = O(\log(kn)).$$

Remark. By choosing the parameters more carefully, the complexity estimate may be obtained in the form

$$O\left(m(\log\log n + \log k \log\log k)\sqrt[3]{k^4n^4 \log n / \log\log n}\right).$$

Note also that the decomposition of formula (△) can be iterated by grouping the polynomials $\Lambda_j(t)$, extracting their common factor, etc. Thus, for any natural number $d \geq 3$, one can construct a circuit of complexity $O(mn^{1+1/d}k^2)$ and depth $O(d\log(kn))$. The described construction is actually obtained by applying the method [15], which deals with a similar numerical operation, to polynomials.

The choice of $k = \log(mn) + O(1)$ and the application of Lemma 12 together with the previously proven Corollary 3 and Lemmas 9 and 10 to the algorithm given at the beginning of this section leads to the following result.

Theorem 10. *The operation of raising to a power of weight m in the field $GF(2^n)$ is implemented by a circuit of complexity and depth*

$$L(\boldsymbol{E_{n,m}}) = O((mn)^{w+\delta}\log n + m^{2+\delta}n^{1+\delta}), \qquad D(\boldsymbol{E_{n,m}}) = O(\log n),$$

where w is the exponent of the multiplication of matrices of size $\sqrt{n} \times \sqrt{n}$ and $\sqrt{n} \times n$, and δ is an arbitrary positive constant. In particular, $L(\boldsymbol{E_{n,m}}) = O((mn)^{1.667})$.

The application of this algorithm in combination with the method of Theorem 9 allows us to state the main result of this section.

Theorem 11. *Let $r = const \in \mathbb{N}$. Then the inversion and division operations in the field $GF(2^n)$ are implemented by circuits with complexity and depth*

$$L(\boldsymbol{I_n}), L(\boldsymbol{\Delta_n}) = O(rn^{w+(w+1)/r}\log n + n^{1+3/r}), \qquad D(\boldsymbol{I_n}), D(\boldsymbol{\Delta_n}) = O(r\log n).$$

In particular,

$$L(\boldsymbol{I_n}), L(\boldsymbol{\Delta_n}) = O(n^{1.667}), \qquad D(\boldsymbol{I_n}), D(\boldsymbol{\Delta_n}) = O(\log n).$$

The multiplicative constants hidden under "O" symbol that can be specified for the latter estimate are very large (at least tens of thousands), so the proposed algorithm has no practical value. However, by multiplying matrices by Strassen's method [32] (see also [21, par. 4.6.4]) with an exponent 1.904 in the complexity estimate and applying Theorem 9 with the parameter $r = 20$, we can derive a circuit of complexity $O(n^{1.999})$ and depth $O(\log n)$ with multiplicative constants under "O" of the order of several hundred. However, such an algorithm is also ineffective in fields of practical importance.

7 Remarks

7.1 On the Litow—Davida and von zur Gathen methods

Inversion by the method [24], based on the matrix representation of the field elements, involves reconstructing the coefficients of a polynomial of degree n from its roots, which are encoded by $O(n^2)$ bits, which means calculating elementary symmetric functions of n numbers, each containing $O(n^2)$ digits. In particular, the product of these n numbers is computed.

Method [10], which also exploits matrix representation, is intended for general finite fields $GF(q^n)$. In the case $q = 2$, an elementary inversion algorithm can be directly derived from [7]. As above, write

$$x^{-1} = x^{2^n-2} = x^2x^{2^2} \cdot \ldots \cdot x^{2^{n-1}}.$$

The product of the polynomials $f_i(t)$ corresponding to the elements x^{2^i} may be reduced, according to [7], to computing an integer product

$$f_1(2^L)f_2(2^L)\cdot\ldots\cdot f_{n-1}(2^L),$$

where L is approximately $n\log n$. Thus, it is required to multiply $n-1$ numbers containing about $n^2\log n$ digits.

Both [24] and [7] suggest to use the result from [2]. The complexity of a circuit that multiplies n numbers, encoded by n bits, with logarithmic depth is estimated in [2] as $O(n^5\log^2 n)$. The multiplicative coefficient in the depth estimate is not given, but elementary analysis shows that it is not less than 15.

7.2 On inversion in normal bases

The roots of an irreducible binary polynomial of degree n in the field $GF(2^n)$ form a normal system $\{\alpha,\alpha^2,\ldots,\alpha^{2^{n-1}}\}$, which, if its elements are linearly independent, is a basis of the field. Such bases are called *normal.* It is known that there are quite a lot of normal bases: at least one such basis exists in any field (see [5, 17, 23]).

Normal bases were studied as early as the 19th century, but practical interest in them arose only at the end of the 20th century, with the development of finite field cryptography. The use of normal bases is supported by the exceptional simplicity of the implementation of Frobenius operations, which do not require any circuit resources at all, since they are reduced to a cyclic shift of coefficients (this is clear from the definition). However, the situation with multiplication is more complicated. The standard Massey—Omura algorithm (see, e.g., [5, 17]) has a theoretical complexity $O(n^3)$, which, even in the case of an optimal choice of basis from the point of view of this algorithm, is of order n^2. For some specific types of bases, the complexity estimate may be reduced to $O(n\log n\log\log n)$ (see [8]), but such bases do not exist in every field.

Therefore, the problem of transition to a standard basis in which multiplication is performed comparatively simply — theoretically no more difficult than in $O(n\log n\log\log n)$ operations, as follows from [29], is of practical interest. The transition between bases is a linear transform of coordinates and can be performed by O. B. Lupanov's method with complexity $O(n^2/\log n)$, which automatically leads to a decrease in the estimate of the complexity of multiplication in normal bases. It turns out that the transition is performed especially simply for the same bases for which the Massey—Omura algorithm works well. And using the arithmetic of standard bases, the complexity of multiplication in them turns out to be almost linear (see [4, 5]). For Gaussian normal bases, the transition to a standard basis in the field extension is applied (see, e.g., [5, 8]).

Since the inversion algorithm from §3 performs linear transforms at the initial and final stages, and the dimension of these transforms does not change when composed with the transition from one basis to another, the estimates of Theorems 3 and 4 are valid for computations in any basis, not just a normal or standard basis. The estimate of Theorem 11 should be adjusted for the implementation of the transition, then the estimates of the complexity and depth of inversion and division in a normal basis will take the form $O(n^2/\log n)$ and $O(\log n)$.

7.3 On raising to an arbitrary power

In recent works [8, 12] devoted to exponentiation in finite fields, upper bounds on the complexity of raising to an arbitrary power in the field $GF(2^n)$ are obtained in the form $O(n^2 \log\log n)$ for standard bases and normal bases of linear complexity (for a normal basis, the same estimate generally holds, since the transition to a polynomial basis and back is always performed in no more than $O(n^2/\log n)$ operations). The depth of all the listed methods is not less than $O(\log^2 n)$. In [10], a logarithmic depth circuit is constructed, but the complexity of such a circuit is rather large. The question of implementing exponentiation with at least $O(n^2)$ complexity apparently remains open.

The author is grateful to his scientific advisor S. B. Gashkov for posing the problem, numerous ideas, discussions and support in solving it.

References

[1] Aho A. V., Hopcroft J. E., Ullman J. D. The design and analysis of computer algorithms. — Reading: Addison–Wesley, 1976.

[2] Beame P. W., Cook S. A., Hoover H. J. Log depth circuits for division and related problems // SIAM J. Comput. — 1986. — V. 15, n. 4. — P. 994–1003.

[3] Bini D., Pan V. Y. Polynomial and matrix computations. Vol. 1. — Boston: Birkhäuser, 1994.

[4] Bolotov A. A., Gashkov S. B. On fast multiplication in normal bases of finite fields // Discrete Math. Appl. — 2001. — V. 11, n. 4. — P. 327–356.

[5] Bolotov A. A., Gashkov S. B., Frolov A. B., Chasovskikh A. A. Elementary introduction to elliptic cryptography: algebraic and algorithmic foundations. — Moscow: KomKniga, 2006. (in Russian)

[6] Brent R. P., Kung H. T. Fast algorithms for manipulating formal power series // J. ACM. — 1978. — V. 25, n. 4 — P. 581–595.

[7] Eberly W. Very fast parallel polynomial arithmetic // SIAM J. Comput. — 1989. — V. 18, n. 5. — P. 955–976.

[8] Gao S., von zur Gathen J., Panario D., Shoup V. Algorithm for exponentiation in finite field // J. Symb. Comput. — 2000. — V. 29. — P. 879–889.

[9] Gashkov S. B., Khokhlov R. A. On the depth of logic circuits for operations in the fields $GF(2^n)$ // Chebyshevskii Sbornik. — 2003. — V. 4, n. 4(8). — P. 59–71. (in Russian)

[10] von zur Gathen J. Inversion in finite fields using logarithmic depth // J. Symb. Comput. — 1990. — V. 9. — P. 175–183.

[11] von zur Gathen J., Gerhard J. Modern computer algebra. — Cambridge University Press, 1999.

[12] von zur Gathen J., Nöcker M. Polynomial and normal bases for finite fields // J. Crypt. — 2005. — V. 18. — P. 337–355.

[13] Grinchuk M. I., Bolotov A. A. Process for designing comparators and adders of small depth // US patent application. — 2006. — no. 7020865.

[14] Grove E. Proofs with potential. — Ph.D. thesis, U.C. Berkeley, 1993.

[15] Hastad J., Leighton T. Division in $O(\log n)$ depth using $O(n^{1+\varepsilon})$ processors. Unpublished manuscript. — 1986. — www.nada.kth.se/˜yohanh/paraldivision.ps.

[16] Huang X., Pan V. Y. Fast rectangular matrix multiplication and applications // J. Complexity. — 1998. — V. 14. — P. 257–299.

[17] Jungnickel D. Finite fields: structure and arithmetics. — Mannheim: Wissenschaftsverlag, 1995.

[18] Karatsuba A. A., Ofman Yu. P. Multiplication of multidigit numbers on automata // Soviet Physics Doklady. — 1963. — V. 7. — P. 595–596.

[19] Khokhlov R. A. Implementation of multiplication and inversion operations in finite fields of characteristic 2 by logic circuits. — Ph.D. thesis, Moscow State Univ., 2005. (in Russian)

[20] Khrapchenko V. M. Asymptotic estimation of addition time of a parallel adder // in: Problemy Kibernetiki. Vol. 19. — Moscow: Nauka, 1967. — P. 107–120. (in Russian)

[21] K n u t h D. E. The art of computer programming. Vol. 2. Seminumerical algorithms. — Reading: Addison–Wesley, 1997.

[22] K o b l i t z N. A course in number theory and cryptography. — New York: Springer-Verlag, 1994.

[23] L i d l R., N i e d e r r e i t e r H. Finite fields. — Cambridge: Cambridge Univ. Press, 1983.

[24] L i t o w B. E., D a v i d a G. I. $O(\log n)$ parallel time finite field inversion // Proc. Aegean Workshop on Computing, Lecture Notes in Computer Science 319. — Berlin, 1988. — P. 74–80.

[25] L u p a n o v O. B. Asymptotic estimates of the complexity of control systems. — Moscow: Izd. MGU, 1984. (in Russian)

[26] L u p a n o v O. B. On rectifier and switching-and-rectifier schemes // Doklady AN SSSR. — 1956. — V. 111, n. 6. — P. 1171–1174. (in Russian)

[27] P a t e r s o n M. S., P i p p e n g e r N., Z w i c k U. Optimal carry save networks // LMS Lecture Notes Series. — V. 169. Boolean function Complexity. — Cambridge University Press, 1992. — P. 174–201.

[28] R o s s e r J. B., S c h o e n f e l d L. Approximate formulas for some functions of prime numbers // Ill. J. Math. — 1962. — V. 6 — P. 64–94.

[29] S c h ö n h a g e A. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2 // Acta Inf. — 1977. — V. 7. — P. 395–398.

[30] S e r g e e v I. S. Inversion in finite fields of characteristic 2 using logarithmic depth // Moscow Univ. Math. Bull. — 2007. — V. 62, n. 1. — P. 29–33.

[31] S i e r p i ń s k i W. 250 problems in elementary number theory. — Warszawa: PWN, 1970.

[32] S t r a s s e n V. Gaussian elimination is not optimal // Numer. Math. — 1969. — B. 13, n. 4. — P. 354–356.

[33] T o o m A. L. The complexity of a scheme of functional elements simulating the multiplication of integers. Soviet Math. Doklady. — 1963. — V. 3. — P. 714–716.

Submitted to editors 09 XI 2005