JOACHIM VON ZUR GATHEN (2004). Arithmetic Circuits for Discrete Logarithms. In *Proceedings of LATIN 2004*, Buenos Aires, Argentina, MARTIN FARACH-COLTON, editor, volume 2976 of *Lecture Notes in Computer Science*, 557–566. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-21258-4. ISSN 0302-9743 (Print) 1611-3349 (Online). URL `http://dx.doi.org/10.1007/978-3-540-24698-5_58`.

This document is provided as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that these works are posted here electronically. It is understood that all persons copying any of these documents will adhere to the terms and constraints invoked by each copyright holder, and in particular use them only for noncommercial purposes. These works may not be posted elsewhere without the explicit written permission of the copyright holder. (Last update 2016/05/18-14:16.)

# Arithmetic circuits for discrete logarithms

Joachim von zur Gathen
University of Paderborn, Germany
`gathen@upb.de`

**`http://www-math.uni-paderborn.de/~aggathen/`**

**Abstract.** We introduce a new model of "generic discrete log algorithms" based on arithmetic circuits. It is conceptually simpler than previous ones, is actually applicable to the natural representations of the popular groups, and we can derive upper and lower bounds that differ only by a constant factor, namely 10.

**Key words.** discrete logarithm, generic algorithm, arithmetic circuit, cyclic group

## 1 Introduction

Discrete logarithm computations and their presumed difficulty are a central topic in cryptography. Let $G$ be a finite cyclic group of order $d$, $p$ the largest prime divisor of $d$, and $n$ the bit length of $d$ (that is, $n$ is the "private key length"). There are three types of results:

- "Generic" algorithms such as baby-step giant-step, Pollard rho, and Pohlig-Hellman. Together they provide a solution with $O(n\sqrt{p} + n^2)$ group operations.
- Algorithms for special groups, such as the index calculus for the group of units in a finite field, and Weil descent for special elliptic curves.
- Lower bounds $\Omega(\sqrt{p})$ on "generic" algorithms.

This paper proposes a new solution to the last point.

Babai & Szemerédi (1984) first proposed a model in which even a lower bound $\Omega(p)$ holds. Then Nechaev (1994) suggested a deterministic model with an $\Omega(\sqrt{p})$ bound, and Boneh & Lipton (1996) considered finite fields. The most popular model was invented by Shoup (1997). It is probabilistic, has an $\Omega(\sqrt{p})$ lower bound, and also works for the Diffie-Hellman problem. Maurer & Wolf (1998, 1999) continued to work on this, in particular by relating the two questions of discrete logarithms and the Diffie-Hellman task. See also Schnorr & Jakobsson (2000) and Schnorr (2001).

An essential ingredient of Shoup's method is a bit representation of the group elements, and his lower bound holds for a random description of this form. The standard "generic" algorithms consist of two phases: first some group calculations are performed, and in a second phase the resulting lists of group elements are sorted, with the goal of finding a collision. Of course, when one wants to implement such an algorithm, one has to use some bit representation of the group elements in computer memory. But the algorithms will use one "natural" representation, not random ones. Strictly speaking, Shoup's result does not apply to this situation, and thus does not provide a lower bound in the natural setting.

This paper repairs this state of affairs by presenting a new model for "generic" discrete log computations which is both technically simpler and more powerful. It has the following properties:

- the known "generic" algorithms fit in,
- a lower bound of $\Omega(\sqrt{p})$ holds,
- it does not make assumptions about the representation of groups,
- there is a matching upper bound, larger only by a constant factor.

This is basically achieved by ignoring the second phase, where sorting occurs. Then one can do away with the group representation, and describe the first phase in a simple arithmetic model.

It is important to note that the goal here is **not** a way of describing useful discrete log computations. In fact, our computations do not calculate discrete logs, but any "generic" discrete log computation yields one of our type. The asymptotically matching upper and lower bounds are an indication that this may be the "right" level of abstraction.

The most natural way of saying that we "only want to use group operations" is by using arithmetic circuits (a.k.a. straight-line programs) with group operations. This model was introduced in great generality by Strassen (1972). However, a circuit computes only group elements and not discrete logs, which live in the "exponent group". Success in the usual algorithms is signalled by a collision, where the same group element is calculated in two different ways. The basic idea is to declare a circuit as successful if it produces such a collision. One has to be a bit careful: it is easy to produce trivial collisions, say by calculating the group element 1 in two different ways. This leads to our notion of a collision "respecting" a divisor $q$ of the group order: it is not trivial in the "exponent group modulo $q$".

In Section 2, we set up the required notions. Section 3 starts with the usual "nonzero preservation" result modulo a prime power; it is somewhat simplified in comparison with other generic models by considering only linear polynomials. Technically, this Lemma 7 is the main overlap with Shoup's method. Then we prove the main result, a lower bound of $\Omega(\sqrt{p})$ in Theorem 8. The model is sufficiently powerful (or weak, as you have it) that essentially matching upper and lower bounds hold; they differ only by a constant factor, namely 10 (Corollary 10).

The model so far is deterministic; Section 4 extends it to probabilistic computations. The same lower bound holds. This is no surprise, since randomized algorithms such as Pollard's rho method do not reduce the computing time. This method is important because it reduces the required memory to a constant number of group elements, but we do not consider this resource.

## 2 Arithmetic circuits for discrete logarithms

We fix the following notation:

$$\begin{aligned} &G = \langle g \rangle \text{ is a finite cyclic group}, d = \#G, \qquad (1)\\ &p \text{ is the largest prime divisor of } d, \text{ and } n \text{ is the binary length of } d. \end{aligned}$$

We consider algorithms that use only the group operations, starting with three special group elements: 1, the generator $g$, and $x$. From these we build further group elements by multiplication and inversion.

**Example 2.** Here is a formulation of the baby-step giant-step algorithm for $d = 20$:

| instruction | trace | trace exponent |
|---|---|---|
| $y_{-2} \longleftarrow 1$ | $1$ | $0$ |
| $y_{-1} \longleftarrow g$ | $g$ | $1$ |
| $y_0 \longleftarrow x$ | $x$ | $t$ |
| $y_1 \longleftarrow y_0 \cdot y_{-1}$ | $xg$ | $t+1$ |
| $y_2 \longleftarrow y_1 \cdot y_{-1}$ | $xg^2$ | $t+2$ |
| $y_3 \longleftarrow y_2 \cdot y_{-1}$ | $xg^3$ | $t+3$ |
| $y_4 \longleftarrow y_3 \cdot y_{-1}$ | $xg^4$ | $t+4$ |
| $y_5 \longleftarrow y_4 \cdot y_{-1}$ | $xg^5$ | $t+5$ |
| $y_6 \longleftarrow y_5 \cdot y_0^{-1}$ | $g^5$ | $5$ |
| $y_7 \longleftarrow y_6 \cdot y_6$ | $g^{10}$ | $10$ |
| $y_8 \longleftarrow y_7 \cdot y_6$ | $g^{15}$ | $15$ |
| $y_9 \longleftarrow y_8 \cdot y_6$ | $g^{20}$ | $20$ |
| $y_{10} \longleftarrow y_9 \cdot y_6$ | $g^{25}$ | $25$ |

The "trace" gives the group element computed in each step. The "trace exponent" is explained below. The algorithm is in its simplest form, ignoring shortcuts like $g^{20} = 1$.

If $\log_g x = 5b + c$, with $0 \leq b, c < 5$, then $x = g^{5b+c}$, hence $xg^{5-c} = g^{5(b+1)}$, and both elements appear in the computation. If we take $G = \mathbb{Z}_{25}^{\times} = \langle 2 \rangle$, a group of order 20, and $x = 19 = 2^{18}$, then we have $18 = 5{\cdot}3{+}3$ and $y_2 = xg^2 = g^{20} = y_9$. $\Diamond$

How do we express that the algorithm successfully computes $\log_z x$? We are very generous: we say that the algorithm is **successful** if a "collision" $u = v$ occurs for two previously computed results $u$ and $v$ for which "$u = v$ is not trivial". If we computed $y_1 = y_{-1} \cdot y_{-1}^{-1}$, $y_2 = y_0 \cdot y_0^{-1}$, then $y_1 = y_2$ would be trivial. We will make this precise in a minute.

The type of computation shown in the table above could be called an "arithmetic group circuit with inputs 1, $g$, and $x$". We abbreviate the assignment $y_k \longleftarrow y_i \cdot y_j^{\pm 1}$ as $(i, j, \pm 1)$, and also trace the exponents of $g$ and $x$ in the circuit. Then we arrive at the following notion.

**Definition 3.** *(i) An* **arithmetic circuit** *is a finite sequence $\mathcal{C} = (I_1, \ldots, I_\ell)$ of instructions $I_k = (i, j, \varepsilon)$, with $-2 \leq i, j < k$ and $\varepsilon \in \{1, -1\}$. The* **size** *of $\mathcal{C}$ is $\ell$. Note that $\mathcal{C}$ is not connected to any particular group.*

*(ii) If $\mathcal{C} = (I_1, \ldots, I_\ell)$ is an arithmetic circuit, $G$ a group and $g, x \in G$, then the* **trace** *of $\mathcal{C}$ on input $(g, x)$ is the following sequence $z_{-2}, z_{-1}, \ldots, z_\ell$ of elements $z_k$ of $G$:*

$$z_{-2} = 1, z_{-1} = g, z_0 = x, z_k = z_i \cdot z_j^\varepsilon \text{ for } k \geq 1 \text{ and } I_k = (i, j, \varepsilon).$$

*(iii) For an arithmetic circuit $\mathcal{C} = (I_1, \ldots, I_\ell)$, the* **trace exponents** *consist of the following sequence $\tau_{-2}, \tau_{-1}, \ldots, \tau_\ell$ of linear polynomials $\tau_k$ in $\mathbb{Z}[t]$:*

$$\tau_{-2} = 0, \tau_{-1} = 1, \tau_0 = t, \tau_k = \tau_i + \varepsilon \cdot \tau_j \text{ for } k \geq 1 \text{ and } I_k = (i, j, \varepsilon).$$

*We think of $g$ as fixed, and also write $z_k(x)$ for the trace elements $z_k$ in (ii).*

The connection between the trace and the trace exponents is clear: if $x = g^a$ and $\tau_k = c \cdot t + b$, then

$$z_k(x) = g^b x^c = g^b \cdot g^{ac} = g^{\tau_k(a)}.$$

Recall that in the exponents, we may calculate modulo the group order $d$, once we consider a fixed group.

**Example 4.** Here are two more examples of trivial collisions.

(i) We take $g, x = g^a$ in a group of order $d$, and an arithmetic circuit which computes $y_m = g^d$ with an addition chain of some length $m$, and also $y_{2m} = x^d$. Then $\tau_m = d$ and $\tau_{2m} = dt$, $y_m = g^d = 1 = x^d = y_{2m}$, and we take the congruence $\tau_m - \tau_{2m} \equiv 0 \bmod d$ as an indicator for the triviality of this collision.

(ii) Now let $q$ be an arbitrary prime divisor of $d$, maybe a small one, and assume that $d \neq q$. Again we calculate some $y_m = g^{d/q}$ and $y_{2m} = x^{d/q}$. Now both results lie in the subgroup $H = \langle g^{d/q} \rangle$ of order $q$, and we can find a collision with a further $q$ (or even $O(\sqrt{q})$) steps. But we have only calculated a discrete logarithm in $H$, not in $G$. If, say, $q = 2$, then $y_m = g^{d/2} \neq 1$ and $y_{2m}$ is either $y_m$ or 1. Thus we have a collision, either $y_{-2} = y_{2m}$ or $y_m = y_{2m}$. $\Diamond$

How do we express that "$u = v$ is trivial"? We certainly want to say that "the collision $y_i = y_j$ is trivial" if $\tau_i = \tau_j$, or even if $\tau_i \equiv \tau_j \bmod d$, but this is not quite enough. We have to rule out unpleasant cases like the one at the end of Example 4, where a collision occurs but the discrete logarithm is not really computed.

**Definition 5.** *Let $\mathcal{C}$ be an arithmetic circuit of size $\ell$, $G = \langle g \rangle$, $q$ an arbitrary divisor of the group order $d = \#G$, and $i, j \leq \ell$.*

*(i) Then $(i, j)$ is said to* respect $q$ *if and only if $\tau_i - \tau_j \not\equiv 0 \bmod q$.*

*(ii) If on input some $g, x \in G$, a collision $y_i = y_j$ occurs, then this collision* respects $q$ *if and only if $(i, j)$ respects $q$.*

Thus we have the linear polynomial $\tau_i - \tau_j \in \mathbb{Z}[t]$ which is nonzero modulo $q$, hence modulo $d$, and if a collision occurs for $x = g^a$, then $g^{\tau_i(a)} = z_i(x) = z_j(x) = g^{\tau_j(a)}$, so that $(\tau_i - \tau_j)(a) \equiv 0 \bmod d$.

If $q_1 \mid q_2 \mid d$, and $(i, j)$ respects $q_1$, then it also respects $q_2$.

**Example 4 continued.** (ii) For $q = 2$, we have $\tau_m = d/2$, $\tau_{2m} = dt/2$, and $\tau_m - \tau_{2m} \equiv d/2 \cdot (1 - t) \bmod d$. We assume that $d$ is not a power of 2, and take a prime divisor $q \neq 2$ of $d$. Then $q$ divides $d/2$, and $\tau_m - \tau_{2m} \equiv 0 \bmod q$. Thus $(m, 2m)$ does not respect $q$, and if on some input $x$ from some group $G$, the collision $g^{d/2} = z_m(x) = z_{2m}(x) = x^{d/2}$ occurs, then this does not respect $q$, either. $\Diamond$

**Definition 6.** *Let $G = \langle g \rangle$ be a finite cyclic group, $\mathcal{C}$ an arithmetic circuit, and $q$ an arbitrary divisor of the group order $d = \#G$. Then the* success rate $\sigma_{\mathcal{C},q}$ *of $\mathcal{C}$ over $G$* respecting $q$ *is the fraction of group elements for which a collision respecting $q$ occurs:*

$$\sigma_{\mathcal{C},q} = d^{-1} \cdot \#\{x \in G: \text{ on input } x, \text{ a collision respecting } q \text{ occurs in } \mathcal{C}\}.$$

Thus $0 \leq \sigma_{\mathcal{C},q} \leq 1$, and a circuit, for which a collision respecting $q$ occurs for every input $x$, has $\sigma_{\mathcal{C},q} = 1$. If $q_1 \mid q_2 \mid d$, then $\sigma_{\mathcal{C},q_1} \leq \sigma_{\mathcal{C},q_2}$. Example 2 indicates that the baby-step giant-step algorithm gives a circuit of size $O(\sqrt{d})$, where $d = \#G$ and $\sigma_{\mathcal{C},d} = 1$. For simplicity, our notation does not reflect the dependence of the success rate on the group.

Also, the Pohlig–Hellman algorithm is a generic algorithm. But index calculus in $G = \mathbb{F}_p^\times$ is not generic; it makes essential use of the representation of the elements of $G$ as integers less than $p$, and the ability to compute with these integers, say to check whether they factor over the factor base.

## 3 The deterministic lower bound

"Nonzero preservation" is a generally useful tool. It says that the value of a nonzero polynomial at a random point is likely to be nonzero. It is well-known over integral domains; we need a slight generalization here. See Shoup (1997) for a more general version.

**Lemma 7.** *Let $d \geq 2$ be an integer, $p^e$ a prime power divisor of $d$, where $p$ is a prime, and $\tau = c_1 t + c_0 \in \mathbb{Z}[t]$ a linear polynomial with $\tau \not\equiv 0 \bmod p^e$. Then*

$$\#\{a \in \mathbb{Z}_d \colon \tau(a) \equiv 0 \bmod p^e\} \leq d/p.$$

*Proof.* Let $i \geq 0$ be the largest exponent with $\tau \equiv 0 \bmod p^i$. Thus $i < e$, and we can write $\tau = p^i \cdot (c_1' t + c_0')$, with $c_0', c_1' \in \mathbb{Z}_{d/p^i}$ and at least one of them nonzero modulo $p^{e-i}$. If $c_1' \equiv 0 \bmod p$, then there is no $a \in \mathbb{Z}_d$ with $\tau(a) \equiv 0 \bmod p^{i+1}$, let alone modulo $p^e$. Otherwise there is exactly one $a_0 \in \mathbb{Z}_p$ with $c_1' a_0 + c_0' \equiv 0 \bmod p$, namely $a_0 \equiv -c_0' \cdot c_1'^{-1} \bmod p$. The residue class mapping $\mathbb{Z}_d \longrightarrow \mathbb{Z}_p$ maps any $a \in \mathbb{Z}_d$ to $a \bmod p$. Exactly $d/p$ elements of $\mathbb{Z}_d$ are mapped to the same element of $\mathbb{Z}_p$. Now if $p^i(c_1' a + c_0') = \tau(a) \equiv 0 \bmod p^e$, then $c_1' a + c_0' \equiv 0 \bmod p$, and hence $a \bmod p = a_0$. There are exactly $d/p$ such $a$, and the claim follows. □

**Theorem 8.** *Let $G = \langle g \rangle$ be a finite cyclic group, $q = p^e$ a prime power divisor of the group order $d = \#G$, $\mathcal{C}$ an arithmetic circuit over $G$ of size $\ell$, and $\sigma_{\mathcal{C},q}$ its success rate respecting $q$. Then*

$$\ell \geq \sqrt{2\sigma_{\mathcal{C},q}p} - 3.$$

*When $\sigma_{\mathcal{C},q}$ is a positive constant, then $\ell \in \Omega(\sqrt{p})$.*

*Proof.* On some input $x$, a collision in $\mathcal{C}$ is of the form $y_i(x) = y_j(x)$ with $-2 \leq i < j \leq \ell$. There are $(\ell+2)(\ell+3)/2$ such $(i, j)$. Any $(i, j)$ which respects $q$ leads to a collision for at most $d/p$ values of $x$, by Lemma 7, since the exponents $a \in \mathbb{Z}_d$ correspond bijectively to the group elements $x = g^a$. Thus the total number of possible collisions respecting $q$ is at most $(\ell + 2)(\ell + 3)/2 \cdot d/p$, and hence

$$\sigma_{\mathcal{C},q} \leq (\ell + 2)(\ell + 3)/2p,$$
$$(\ell + 3)^2 \geq (\ell + 2)(\ell + 3) \geq 2\sigma_{\mathcal{C},q}p.$$ □

The various well-known algorithms yield an $O(n\sqrt{p} + n^2)$ upper bound for discrete logarithm computations, and we now have a lower bound $\Omega(\sqrt{p})$ where $p$ is the largest prime divisor of $d$. In what follows, we derive upper and lower bounds that differ only by a constant factor. We start with a lower bound different from Theorem 8, namely $\Omega(n)$. This is not of direct cryptographic interest, since $n \approx \log_2 d$ is roughly the "key length" or "input length"—in contrast to $\sqrt{p}$ which will usually be chosen so that it is exponentially large in $n$. The interest is a desire to understand the complexity of discrete logarithms as well as possible.

**Theorem 9.** *Let $\mathcal{C}$ be an arithmetic circuit of size $\ell$, $G = \langle g \rangle$ a cyclic group of order $d \geq 3$, with $\sigma_{\mathcal{C},d} = 1$, and let $n = \lfloor \log_2 d \rfloor + 1$ be the binary length of $d$. Then*

$$\ell \geq \frac{n}{2} - 2,$$

*and hence $\ell \in \Omega(n)$.*

*Proof.* Any element $a$ of $\mathbb{Z}_d$ has exactly one *balanced representative* $b \in \mathbb{Z}$ with

$$a = (b \bmod d), -d/2 < b \leq d/2.$$

For $-2 \leq k \leq \ell$, we write the trace exponent $\tau_k \in \mathbb{Z}_d[t]$ as $\tau_k = (c_k \bmod d) \cdot t + (b_k \bmod d)$, where $c_k, b_k \in \mathbb{Z}$ are balanced representatives. By induction on $k$ it follows that $|b_k|, |c_k| \leq 2^k$ for $0 \leq k \leq \ell$ (and $|b_k|, |c_k| \leq 1$ for $k = -2, -1$). Now let $a_0 = \lfloor \sqrt{d} \rfloor$, $a = (a_0 \bmod d) \in \mathbb{Z}_d$ and $x = g^a \in G$. The assumption $\sigma_{\mathcal{C},d} = 1$ implies that there are $i, j \leq \ell$ with $\tau_i - \tau_j \not\equiv 0 \bmod d$ and $(\tau_i - \tau_j)(a) \equiv 0 \bmod d$. We let

$$u = (c_i - c_j) \cdot a_0 + (b_i - b_j) \in \mathbb{Z}.$$

The above implies that $u \equiv 0 \bmod d$.

If $c_i = c_j$, then $b_i \equiv b_j \bmod d$ and $\tau_i - \tau_j \equiv 0 \bmod d$, which is ruled out. Thus $c_i \neq c_j$. If $u = 0$, then

$$\sqrt{d} - 1 \leq |a_0| = \frac{|b_i - b_j|}{|c_i - c_j|} \leq |b_i - b_j| \leq |b_i| + |b_j| \leq 2^{\ell+1}.$$

If $u \neq 0$, then $|u| \geq d$, and

$$\begin{aligned} 2^{\ell+1}(\sqrt{d}+1) = 2^{\ell+1}\sqrt{d} + 2^{\ell+1} &\geq |c_i - c_j| a_0 + |b_i - b_j| \\ &\geq |(c_i - c_j)a_0 + (b_i - b_j)| = |u| \geq d, \\ 2^{\ell+1} &\geq \frac{d}{\sqrt{d}+1} \geq \sqrt{d} - 1. \end{aligned}$$

Thus $\ell \geq \log(\sqrt{d} - 1) - 1$ in both cases. The claim now follows from

$$\log(\sqrt{d} - 1) \geq \frac{1}{2}\log d - \frac{1}{2} \geq \frac{1}{2}\lfloor \log d \rfloor - \frac{1}{2} = \frac{n}{2} - 1$$

for $d \geq 12$. (One checks the cases $3 \leq d \leq 11$ separately.) □

For an upper bound in our model, we just compute $g^{d/p}$ and $x^{d/p}$, and then perform a baby-step giant-step search in the subgroup $\langle g^{d/p} \rangle$ of $p$ elements. The total cost is $2(n + \sqrt{p})$, and we have the lower bounds of $n/2$ and $\sqrt{2p}$, approximately. Thus the gap is a factor of about 4 or $\sqrt{2}$, depending on whether $n$ or $\sqrt{p}$ is larger. We can obtain a specific estimate as follows.

**Corollary 10.** *Let $G$ be a cyclic group with $d$ elements, $n = \lfloor \log_2 d \rfloor + 1$ the binary length of $d$, $p$ the largest prime divisor of $d$, $e$ the multiplicity of $p$ in $d$,*

$$m = \max\{\sqrt{2p} - 3, n/2 - 2\},$$

*and assume that $m \geq 37$. Then there exists an arithmetic circuit $\mathcal{C}$ with success rate $\sigma_{\mathcal{C},p^e} = 1$ over $G$ and size at most $10m$. Any circuit $\mathcal{C}$ with $\sigma_{\mathcal{C},p^e} = 1$ has size at least $m$.*

*Proof.* The last claim follows from Theorems 8 and 9. For $\mathcal{C}$ we take the circuit described above. Then $\sigma_{\mathcal{C},p^e} = 1$, and its size $\ell$ is at most $2 \cdot 2\log(d/p) + 2\sqrt{p}$. Thus

$$\begin{aligned}\ell \leq 4\log d + 2\sqrt{p} &\leq 8(n/2-2) + \sqrt{2}\cdot(\sqrt{2p}-3) + 17 + 3\sqrt{2}\\ &\leq (8+\sqrt{2})m + 17 + 3\sqrt{2} \leq 10m. \qquad \square\end{aligned}$$

In usual models of computation, upper bounds come from algorithms—the real thing—and lower bounds impose barriers on improving these. But here, the lower bound is the real thing, and the upper bound a barrier on deriving better ones. As stated before, the above circuit cannot claim to actually compute discrete logarithms in $G$.

## 4 Probabilistic arithmetic circuits

We now have a model for discrete log computations with essentially matching upper and lower bounds. However, Pollard's rho method works in any group, but does not fit into our model because it makes random choices. The method does not make progress over the baby-step giant-step method in terms of time (= size of circuit), but cuts the space dramatically down to a constant. Space is not accounted for in our model, but we now adapt it to allow probabilistic choice. Once the model is appropriately set up within our framework, it is easy to obtain the same lower bound as before. Thus random choices do not help, in this specific sense.

We allow two types of random choices in our algorithms: random group elements, and random exponents. For the first, we might allow a new instruction

$$y_k \longleftarrow \operatorname{rand}(G)$$

which assigns an element of $G$ to $y_k$. On executing the circuit, this element is chosen uniformly at random, independent of other executions of the circuit. Actually, this feature is not used in any discrete log algorithm that we are aware of. For the corresponding trace exponent, we take new variables $t_1, \ldots, t_s$ if $s$ instructions $\operatorname{rand}(G)$ occur. Thus $\tau_k = t_i$ if $t_1, \ldots, t_{i-1}$ have been used so far. But actually this feature is not required, because the next one subsumes it.

We also want to allow random exponents, that is, an element $y^e$ with random $e$ and previously computed $y$. When $y = g$, this may be thought of as a random element of $G$ with known discrete logarithm. So, as a new feature we allow our circuits to use a string

$$b = (b_1, \ldots, b_r) \in \{0,1\}^r$$

of random bits, via assignments

$$y_k \longleftarrow y_i^{b_j}$$

with $i < k$ and $1 \leq j \leq r$. The corresponding trace exponent is

$$\tau_k = b_j \cdot \tau_i.$$

**Example 11.** In a probabilistic version of Pollard's rho method, the next element $y_{k+1}$ is calculated as one of $y_k \cdot g$, $y_k^2$, or $y_k \cdot x$, each with probability $1/3$. This is easy to simulate, using two random bits $b$ and $c$. If we set

$$y_{k+1} = g^{(1-b)(1-c)} y_k^{(1-b)c+1} x^{b(1-c)}, \tag{12}$$

then $y_{k+1}$ will take one of the three required values for $(b, c) = (0, 0), (0, 1), (1, 0)$, respectively. We set the probability of $(b, c) = (1, 1)$ to 0. The formula can be implemented with an arithmetic circuit of size 11. In another version of Pollard's rho method, one divides the group into three parts and makes the three-fold choice according to where $y_k$ has landed. This does not fit into our model. $\Diamond$

**Definition 13.** *(i) A* ***probabilistic arithmetic circuit*** *is a pair $\mathcal{C} = (\mathcal{C}_r, u)$ consisting of a probability distribution $u$ on $\{0, 1\}^r$ for some nonnegative integer $r$ and an arithmetic circuit $\mathcal{C}_r$ as in Definition 3 except that in addition the following type of assignment is allowed:*

$$y_k \longleftarrow y_i^{b_j} \tag{14}$$

*with $-2 \leq i < k$ and $1 \leq j \leq r$.*

*(ii) The size $\ell$ of $\mathcal{C}$ is the number of group operations performed. Operations of the type (14) are not counted. (Formally, we might give appropriate rational indices $k$ to their results.)*

*(iii) If $b \in \{0, 1\}^r$ is provided, then we obtain a circuit $\mathcal{C}(b)$ of size $\ell$ as follows. If $y_k$ is given by (14), then we replace all references to $y_k$ by a reference to $y_{-2}\,(= 1)$ if $b_j = 0$, and by a reference to $y_i$ if $b_j = 1$. This replacement is performed recursively starting at the beginning of the instruction list until no more references to an assignment of type (14) exist. The new instructions are denoted as $y_k(b) \longleftarrow y_i(b) \cdot y_j(b)^{\varepsilon}$.*

*(iv) For a divisor $q$ of $d$, the success rate of $\mathcal{C}$ with respect to $q$ is*

$$\sigma_{\mathcal{C},q} = \sum_{b \in \{0,1\}^r} u(b) \cdot \sigma_{\mathcal{C}(b),q}.$$

Thus $\sigma_{\mathcal{C},q}$ is the average success rate of the $\mathcal{C}(b)$ for random $b$. Recall that

$$\sigma_{\mathcal{C}(b),q} = d^{-1} \cdot \#\{x \in G : \text{there is a collision } y_i(b)(x) = y_j(b)(x) \text{ respecting } q\},$$

and such a collision respects $q$ if and only if $\tau_i(b) - \tau_j(b) \not\equiv 0 \bmod q$, with the usual trace exponent $\tau_i(b)$, $\tau_j(b) \in \mathbb{Z}[t]$. These are defined only when some $b \in \{0, 1\}^r$ is fixed, not for $\mathcal{C}$ itself.

**Theorem 15.** *Let $G = \langle g \rangle$ be a finite cyclic group, $p$ a prime divisor of the group order $d = \#G$, $\mathcal{C}$ a probabilistic arithmetic circuit of size $\ell$, and $\sigma_{\mathcal{C},p}$ it success rate respecting $p$. Then*

$$\ell \geq \sqrt{2\sigma_{\mathcal{C},p}\,p} - 3.$$

*When $\sigma_{\mathcal{C},p}$ is a constant, then $\ell \in \Omega(\sqrt{p})$.*

*Proof.* The probabilistic circuit $\mathcal{C} = (\mathcal{C}_r, u)$ and each (deterministic) circuit $\mathcal{C}(b)$ have size $\ell$. From the proof of Theorem 8, we have

$$\frac{(\ell+3)^2}{2p} \geq \sigma_{\mathcal{C}(b),p}$$

for each $b \in \{0,1\}^r$. Hence

$$\frac{(\ell+3)^2}{2p} = \frac{(\ell+3)^2}{2p} \sum_{b\in\{0,1\}^r} u(b) \geq \sum_{b\in\{0,1\}^r} u(b)\sigma_{\mathcal{C}(b),p} = \sigma_{\mathcal{C},p}. \qquad \square$$

## References

1. László Babai & Endre Szemerédi (1984). On the complexity of matrix group problems I. In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science,* Singer Island FL, 229–240. IEEE Computer Society Press. ISBN 0-8186-0591-X. ISSN 0272-5428.
2. Dan Boneh & Richard J. Lipton (1996). Algorithms for Black-Box Fields and their Application to Cryptography. In *Advances in Cryptology: Proceedings of CRYPTO '96,* Santa Barbara CA, Neal Koblitz, editor, number 1109 in Lecture Notes in Computer Science, 283–297. Springer-Verlag. ISSN 0302-9743.
3. Ueli Maurer & Stefan Wolf (1998). Lower Bounds on Generic Algorithms in Groups. In *Advances in Cryptology: Proceedings of EUROCRYPT 1998,* Santa Barbara, CA, Kaisa Nyberg, editor, number 1403 in Lecture Notes in Computer Science, 72–84. Springer-Verlag. ISSN 0302-9743. URL `http://link.springer.de/link/service/series/0558/bibs/1403/14030072.htm`.
4. Ueli M. Maurer & Stefan Wolf (1999). The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing* **28**(5), 1689–1721.
5. V. I. Nechaev (1994). К вопросу о сложности детерминированного алгоритма для дискретного логарифма. Российская Академия Наук. Математические Заметки **55**(2), 91–101, 189. ISSN 0025-567X. Complexity of a determinate algorithm for the discrete logarithm, *Mathematical Notes* **55**(2) (1994), 165-172.
6. Claus Peter Schnorr (2001). Security of DL-encryption and signatures against generic attacks-a survey. In *Public-Key Cryptography and Computational Number Theory Conference 2000*, 257–282. URL `http://www.mi.informatik.uni-frankfurt.de/research/papers.html`.
7. Claus Peter Schnorr & Markus Jakobsson (2000). Security Of Discrete Log Cryptosystems in the Random Oracle and the Generic Model. Technical report, Universität Frankfurt/Main and Bell Laboratories, Murray Hill, New Jersey. URL `http://www.mi.informatik.uni-frankfurt.de/research/papers.html`. The Mathematics of Public-Key Cryptography, The Fields Institute, Toronto.
8. Victor Shoup (1997). Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology: Proceedings of EUROCRYPT 1997,* Konstanz, Germany, Springer-Verlag, editor, number 1233 in Lecture Notes in Computer Science, 256–266. ISSN 0302-9743. URL `http://www.shoup.net/papers/`.
9. V. Strassen (1972). Berechnung und Programm. I. *Acta Informatica* **1**, 320–335.