Electronic Colloquium on Computational Complexity, Report No. 108 (2016)

Discrete Logarithm and Minimum Circuit Size

M. Rudow[1]

Abstract

This paper shows that the Discrete Logarithm Problem is in $\mathsf{ZPP}^{\mathsf{MCSP}}$ (where MCSP is the Minimum Circuit Size Problem). This result improves the previous bound that the Discrete Logarithm Problem is in $\mathsf{BPP}^{\mathsf{MCSP}}$ Allender et al. (2006). In doing so, this paper helps classify the relative difficulty of the Minimum Circuit Size Problem.

Keywords: Computational Complexity, Minimum Circuit Size Problem, Discrete Logarithm Problem

1. Introduction

The Minimum Circuit Size Problem (MCSP) is a well known problem which is suspected to be NP-intermediate. MCSP has been a problem of interest for many years; for example, it was a focus of study with respect to Brute Force Search in the 1950's in the Soviet UnionTrakhtenbrot (1984). Despite its long history, the exact complexity of MCSP remains a mystery. Thus MCSP has been puzzling computer scientists for decades. Recently, several results reducing other problems to MCSP have been shown. For example, Allender and Das proved that $\mathsf{SZK} \subseteq$ $\mathsf{Promise\text{-}BPP}^{\mathsf{MCSP}}$ and $\mathsf{GI} \in \mathsf{RP}^{\mathsf{MCSP}}$ where Statistical Zero Knowledge and Graph Isomorphism are denoted as SZK and GI Allender and Das (2014).

Furthermore, there is some interest in improving $\mathsf{BPP}^{\mathsf{MCSP}}$ (and likewise $\mathsf{RP}^{\mathsf{MCSP}}$) reductions to $\mathsf{ZPP}^{\mathsf{MCSP}}$ reductions. For instance, Allender and Das list determining whether $\mathsf{GI} \in \mathsf{ZPP}^{\mathsf{MCSP}}$ as an open problemAllender and Das (2014). Additionally, Allender, Grochow, and Moore prove that Graph Automorphism (GA) is in $\mathsf{ZPP}^{\mathsf{MKTP}}$Allender et al. (2015), where MKTP is a time-bounded Kolmogorov complexity problem that is similar to MCSP and often studied in tandem with MCSP.

The Discrete Logarithm Problem (DLP) is another famous candidate of suspected NP-intermediate status. Used widely in cryptography, DLP is an important complexity problem with many useful applications. Allender, Buhrman, Koucký, Van Melkebeek, and Ronneburger proved that Factoring is in $\mathsf{ZPP}^{\mathsf{MCSP}}$ and DLP is in $\mathsf{BPP}^{\mathsf{MCSP}}$Allender et al. (2006). This paper improves Allender *et al.*'s result by showing that DLP is in $\mathsf{ZPP}^{\mathsf{MCSP}}$ by modifying the authors' method used in Allender et al. (2006).

Email address: mrudow@seas.upenn.edu (M. Rudow)

[1]DIMACS, Rutgers University, Piscataway, NJ, USA. Candidate for Bachelor of Science in Engineering, University of Pennsylvania, Philadelphia, PA, USA.

ISSN 1433-8092

Our proof uses Allender *et al.*'s construction Allender et al. (2006) for computing the prime factorization in $\mathsf{ZPP}^{\mathsf{MCSP}}$ in order to compute the prime factorization of $p-1$, thus enabling efficient computation of a generator of the group. We then use the fact (previously established by Allender et al. (2006)) that DLP can be solved in $\mathsf{ZPP}^{\mathsf{MCSP}}$ when g is a generator of the multiplicative group mod p. Using such a construction to compute the values of the inputs as powers of a generator, one can efficiently both determine if a valid discrete logarithm exists and, provided it exists, compute it quickly.

2. Preliminaries

This section will serve to provide definitions of the computational problems that we study.

Definition 1 (The Discrete Logarithm Problem (DLP))**.** *On input (g, z, p) where p is a prime, compute x such that $g^x \equiv z \mod p$ if such an x exists, and otherwise return* 0*.*

Definition 2 (The Factoring Problem)**.** *On input N return the prime factorization of N.*

Definition 3 (The Minimum Circuit Size Problem ($MCSP$))**.** *On input (T, s) where T is a truth table of some function $f : \{0,1\}^n \to \{0,1\}$ of size 2^n, determine if f can be represented by a boolean circuit of size $\leq s$ Kabanets and Cai (2000).*

3. Main Result

Theorem 1. $\mathsf{DLP} \in \mathsf{ZPP}^{\mathsf{MCSP}}$.

Proof. Start with input (g, z, p) where p is a prime, $0 < g < p$, $0 < z < p$. We terminate with $x \in \{0, \cdots, p-1\}$ such that $x > 0$, $g^x \equiv z \mod p$ if such an x exists and $x = 0$ otherwise.

Apply Lemma 1 to compute h such that h is a generator of $\mathbb{Z}_p^\times$ in expected polynomial time. Then apply Lemma 2 to compute a, b such that $h^a \equiv g \mod p$ and $h^b \equiv z \mod p$ in expected polynomial time. Use Lemma 3 to determine if $\exists x \in \{1, \cdots, p-1\}$ such that $a \cdot x \equiv b \mod p-1$. If so, the application of Lemma 3 produces such a x. Thus $g^x \equiv (h^a)^x \equiv h^{a \cdot x} \equiv h^b \equiv z \mod p$. So return x. If no such x exists, then there is no x such that $g^x \equiv z \mod p$ and we return 0. The total runtime is polynomial in $\log p$ in expectation and the x returned is always correct.

□

Lemma 1. *Finding a generator g of $\mathbb{Z}_p^\times$, the multiplicative group* mod p*, is in* $\mathsf{ZPP}^{\mathsf{MCSP}}$*.*

Proof. We start by computing the prime factorization of $p-1$ in expected polynomial time. This is possible because $FACTORING \in \mathsf{ZPP}^{\mathsf{MCSP}}$. Let L denote the list of unique prime factors of $p-1$. Define $f : \{0, \cdots, p-1\} \to \{0,1\}$ as

$$f(a) = \begin{cases} 1 & \forall q \in L \mid a^{\frac{p-1}{q}} \not\equiv 1 \mod p \\ 0 & \exists q \in L \mid a^{\frac{p-1}{q}} \equiv 1 \mod p \end{cases}$$

We know a is a generator if and only if $\forall q \in L \mid a^{\frac{p-1}{q}} \not\equiv 1 \mod p$ (Stein, 2008, p. 44). Thus $f(a) = 1$ if and only if a is a generator. Furthermore, $|L| \leq \log(p-1) < \log p$, and $\forall w \in \{1, \cdots, p-1\}$ we know $a^w \mod p$ can be computed in polynomial time with fast modular exponentiation through repeated squaring. Thus f is computable in time $O(\log^k p)$ for some $k \in \mathbb{N}$.

We know that $\exists h$ that generates $\mathbb{Z}_p^\times$. Thus $\mathbb{Z}_p^\times = \{h^i \mid 1 \leq i \leq p-1\}$. Clearly, $p-1$ can have at most $\log(p-1)$ distinct prime factors. Furthermore, the prime number theorem tells us that there are $\Omega(\frac{p-1}{\log(p-1)})$ prime numbers in $\{1, \cdots, (p-1)-1\}$. Thus after we eliminate the at most $\log(p-1)$ of these possibilities which correspond to prime factors of $p-1$, we are left with at least

$$c \cdot \frac{p-1}{\log(p-1)} - \log(p-1) > \frac{p-1}{\log(p-1)^2} - \log(p-1) = \frac{p-1-\log(p-1)^3}{\log(p-1)^2} > \frac{p-1}{2\log(p-1)^2}$$

primes that do not divide $p-1$.

Let d be any such prime and let $g = h^d$. Then let $|g|$ denote the order of g (which is $\leq p-1$). Then $g^{|g|} \equiv 1 \equiv h^{d|g|} \mod p$ which means $d|g| \mid p-1$. Thus $|g| = p-1$ and $g = h^d$ is a generator of $\mathbb{Z}_p^\times$. Thus for every prime number in $\{1, \cdots, (p-1)-1\}$ that does not divide $p-1$ there is a corresponding generator of the group. Hence there are at least $\frac{p-1}{2\log(p-1)^2}$ generators, and a random element of the group has at least a $\frac{1}{2\cdot\log(p-1)^2}$ chance of being a generator.

The following algorithm finds a generator in expected polynomial time: pick a random element e of the group. If $f(e) = 1$ then return e as the generator. Otherwise repeat the algorithm.

Each application of the algorithm takes polynomial time, and we need only run it at most $2 \cdot \log(p-1)^2 < \log(p)^3$ times in expectation to succeed. Whenever it terminates, $f(e) = 1$ thus e is truly a generator. Hence it terminates with the correct output in expected polynomial time. □

Lemma 2. *Given a valid input to* DLP, (g, z, p) *where g is a generator of the multiplicative group mod p, $\mathbb{Z}_p^\times$, computing x such that $g^x \equiv z \mod p$ can be done in* $\mathsf{ZPP}^{\mathsf{MCSP}}$.

Proof. Note, Lemma 2 was already observed by Allender et al. (2006) and the proof is included for completeness.

First let us state (Allender et al., 2006, Theorem 45): *Let L be a language of polynomial density such that for some $\epsilon > 0$, for every $x \in L$, $KT(x) \geq |x|^\epsilon$. Let f(y,x) be computable uniformly in time polynomial in $|x|$. There exists a polynomial-time probabilistic oracle Turing machine N and a polynomial q such that for any n and y $Pr_{|x|=n,s}[f(y, N^L(y, f(y,x), s)) = f(y,x)] \geq \frac{1}{q(n)}$, where x is chosen uniformly at random and s denotes the internal coin flips of N.* This theorem uses a construction from Håstad et al. (1999). In this theorem, $KT(x)$ represents the time-bounded Kolmogorov complexity of x.

Let $y = (g, p)$ and denote $f_y(x) = g^x \mod p$. Allender *et al.* observe that there is an L in $\mathsf{P}^{\mathsf{MCSP}}$ that satisfies the hypothesis Allender et al. (2006). Thus we apply (Allender et al., 2006, Theorem 45) with this $L \in \mathsf{P}^{\mathsf{MCSP}}$ to conclude that there is a a polynomial-time probabilistic oracle Turing machine N and a polynomial r satisfying $Pr_{z,s}[f_y(N^L(y,z,s)) \equiv z \mod p] \geq \frac{1}{r(n)}$ for random input bits s and $z \in_R \{1, \cdots, p-1\}$.

Repeat the following trial until success:

Pick v from $\{1, \cdots, p-1\}$ randomly and pick a random s.

Let $w = N^L((g,p), z \cdot g^v, s)$.

Report success if $g^w \equiv z \cdot g^v \mod p$.

Note that because g is a generator $\forall v \exists w \mid z \cdot g^v = g^w$. Furthermore because $\exists x \mid g^x \equiv z \mod p$ we know that $z \cdot g^v \equiv g^{v+x} \mod p$. Thus $z \cdot g^v$ is a random power of g and is therefore uniformly distributed in the codomain. Thus we need only invert a single element randomly chosen from the codomain to succeed. In expectation, we must repeat the trial $r(\log p) = \text{poly}(\log p)$ times to succeed. Hence the total runtime is polynomial in expectation.

Given $g^w \equiv z \cdot g^v \mod p$ we know $g^w \cdot g^{p-1-v} \equiv z \cdot g^v \cdot g^{p-1-v} \equiv z \cdot g^{p-1} \equiv z \mod p$. Thus $g^{p-1+w-v} \equiv z \mod p$. We have therefore computed $x = p-1+w-v$ in expected polynomial time. $\square$

Lemma 3. *On input $(a, b, p-1)$ for $0 \leq a, b < p-1$, determining if $\exists x$ such that $a \cdot x \equiv b \mod p-1$ and computing such an x if it exists can be done in $\mathsf{ZPP}^{\mathsf{MCSP}}$.*

Proof. Begin by factoring $p - 1 = \Pi_{i=1}^{k} p_i^{e_i}$. Let $q(x) = (x \mod p_1^{e_1}, \cdots, x \mod p_k^{e_k})$. By the Chinese Remainder Theorem, q is a bijection with $q^{-1}((y_1, \cdots, y_k))$ computed in polynomial time defined as follows: let $M_i = \frac{p-1}{p_i^{e_i}}$ and let $u_i = M_i^{-1} \mod p_i^{e_i}$ (under multiplication). Then $q^{-1}((x_1, \cdots, x_k)) = \sum_{i=1}^{k} x_i \cdot u_i \cdot M_i \mod p-1$ (Ding et al., 1996, p. 23).

Denote $q(a) = (a_1, \cdots, a_k)$ and $q(b) = (b_1, \cdots, b_k)$. By the Chinese Remainder Theorem, we know $\exists x \in \{0, \cdots, (p-1)-1\}$ such that $a \cdot x \equiv b \mod p$ if and only if $\exists x \mid q(a \cdot x) = b$. We either construct $(x_1, \cdots, x_k)$ such that $(a_1 \cdot x_1, \cdots, a_k \cdot x_k) = (b_1, \cdots, b_k)$ and return $q^{-1}((x_1, \cdots, x_k))$ or show that no such $(x_1, \cdots, x_k)$ exists. If we return $x = q^{-1}((x_1, \cdots, x_k))$, it is correct. The reason is that there is a unique solution $q^{-1}(b_1, \cdots, b_k)$ and $q(a \cdot x) = (a \cdot x \mod p_1^{e_1}, \cdots, a \cdot x \mod p_k^{e_k}) = ((a \mod p_1^{e_1}) \cdot (x \mod p_1^{e_1}), \cdots, (a \mod p_k^{e_k}) \cdot (x \mod p_k^{e_k})) = (a_1 \cdot x_1, \cdots, a_k \cdot x_k) = (b_1, \cdots, b_k)$. Thus $q(a \cdot x) = q(b)$ and so $q^{-1}(q(a \cdot x)) \equiv q^{-1}(q(b))$. Thus $a \cdot x \equiv b \mod p-1$.

The following algorithm constructs each x_i for $i \in \{1, \cdots, k\}$. The algorithm runs in polynomial time, thus repeating it k times to construct the entirety of $(x_1, \cdots, x_k)$ or disproving its existence can be done in polynomial time.

Case 1: $b_i = 0$ then let $x_i = 0$. Thus $a_i \cdot x_i = a_i \cdot 0 = 0 = b_i$.

Case 2: $b_i \neq 0$ and $a_i = 0$. Then return no such x exists because $\forall x_i$ we know $a_i \cdot x_i = 0 \neq b_i$.

Denote $d = gcd(p_i^{e_i}, a_i)$.

Case 3: $d = 1$. Use the Euclidean Algorithm in polynomial time to compute k, l such that $p_i^{e_i} \cdot k + a_i \cdot l = 1$ thus $a_i \cdot l \equiv 1 \mod p_i^{e_i}$. Let $x_i = l \cdot b_i$, thus $a_i \cdot x_i \equiv (a_i \cdot l) \cdot b_i \equiv 1 \cdot b_i \equiv b_i \mod p_i^{e_i}$.

Case 4: $d \neq 1, d \mid b_i$. Denote $b_i = z \cdot d$, then use the Euclidean Algorithm in polynomial time to compute i, j such that $p_i^{e_i} \cdot i + a_i \cdot j = d$ thus $a_i \cdot j \equiv d \mod p_i^{e_i}$. Let $x_i = j \cdot z$ thus $a_i \cdot x_i \equiv (a_i \cdot j) \cdot z \equiv d \cdot z \equiv b_i \mod p_i^{e_i}$.

Case 5: $d \neq 1, d \not| \; b_i$. Return no such x exists. The proof of this fact goes as follows: suppose towards contradiction that $\exists x_i \mid a_i \cdot x_i \equiv b_i \mod p_i^{e_i}$. By the definition of modulo, we know that $a_i \cdot x_i - w \cdot p_i^{e_i} = b$ for some $w \in \mathbb{Z}^+ \cup \{0\}$. Then since $b_i \neq 0$ we know $a_i \cdot x_i - w \cdot p_i^{e_i} \neq 0$. Hence $b_i = d \cdot (\frac{a_i}{d} \cdot x_i - w \cdot \frac{p_i^{e_i}}{d}) = d \cdot m$ for appropriately defined integer $m \in \{1, \cdots, p_i^{e_i}\}$. Thus $d \mid b_i$ which contradicts the assumption. Thus the original assumption is false, and hence no such x exists.

In all cases we either produce the appropriate x_i or show that no such x_i exists. Thus in polynomial time, we either compute $(x_1, \cdots, x_k)$ such that $(a_1 \cdot x_1, \cdots, a_k \cdot x_k) = (b_1, \cdots, b_k)$ or show no such $(x_1, \cdots, x_k)$ exists. Hence we either show no valid x exists, or return $x = q^{-1}((x_1, \cdots, x_k))$ in expected polynomial time. For the sake of consistency of notation, if x exists and $x = 0$ then return instead $x = p - 1$. $\square$

4. Conclusion and Future Directions

This paper's proof can be extended to other versions of DLP. Bach shows that computing the Discrete Logarithm modulo a composite N can be done by computing its prime factorization and calculating the discrete logarithms modulo each prime in its prime factorizationBach (1984). The Discrete Logarithm Problem with a composite is the same as DLP except that the input is (g, z, n) for composite n rather than (g, z, p) for prime p. Furthermore, it is possible to compute the Discrete Logarithm modulo a prime power p^e by simply computing the discrete logarithm $\mod p$ Bach (1984). Consequently, this paper's proof that $\mathsf{DLP} \in \mathsf{ZPP}^{\mathsf{MCSP}}$ can be extended to inputs of the form (g, z, p) where p need not be prime.

Several results have shown problems such as Graph Automorphism, Graph Isomorphism, Factoring, and DLP reduce to MCSP Allender et al. (2015); Allender and Das (2014); Allender et al. (2006). Theorem 1 improves one such reduction; many other reductions are also candidates for such progress. Allender *et al.* show that the Shortest Independent Vector Problem, Shortest Basis Problem, Length of Shortest Vector Problem, Unique Shortest Vector Problem, Closest Vector Problem, and Covering Radius Problem are all in $\mathsf{BPP}^{\mathsf{MCSP}}$ Allender et al. (2006). One open question is whether any of those results can be improved to $\mathsf{ZPP}^{\mathsf{MCSP}}$.

Acknowledgements The author acknowledges that this work was carried out while the he was a participant in the 2016 DIMACS REU program at Rutgers University, supported by NSF grant CCF-1559855. He also acknowledges the invaluable advice of Professor Allender. The author also acknowledges Karel Král, Martin Böhm, Martin Töpfer, Veronika Slívová, Mark Karpilovskij, and Stanislav Kučera for helpful discussions on the subject.

Bibliography

Allender, E., Buhrman, H., Koucký, M., van Melkebeek, D., Ronneburger, D., 2006. Power from random strings. SIAM Journal on Computing 35 (6), 1467–1493. URL http://dx.doi.org/10.1137/050628994

Allender, E., Das, B., 2014. Zero knowledge and circuit minimization. In: Mathematical Foundations of Computer Science (MFCS). Vol. 8635. Springer, pp. 25–32.

Allender, E., Grochow, J. A., Moore, C., 2015. Graph isomorphism and circuit size. CoRR abs/1511.08189. URL http://arxiv.org/abs/1511.08189

Bach, E., Jun 1984. Discrete logarithms and factoring. Tech. Rep. UCB/CSD-84-186, EECS Department, University of California, Berkeley. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1984/5973.html

Ding, C., Pei, D., Salomaa, A., 1996. Chinese remainder theorem: applications in computing, coding, cryptography. World Scientific.

Håstad, J., Impagliazzo, R., Levin, L., Luby, M., 1999. A pseudorandom generator from any one-way function 28, 1364–1396.

Kabanets, V., Cai, J.-Y., 2000. Circuit minimization problem. pp. 73–79.

Stein, W., 2008. Elementary Number Theory: Primes, Congruences, and Secrets. Springer-Verlag New York.

Trakhtenbrot, B. A., 1984. A survey of Russian approaches to perebor (brute-force searches) algorithms. IEEE Annals of the History of Computing 6 (4), 384–400.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de