NOTE

A REDUCTION OF SEMIGROUP DLP TO CLASSIC DLP

MATAN BANIN AND BOAZ TSABAN

Abstract. We present a polynomial-time reduction of the discrete logarithm problem in any periodic (a.k.a. torsion) semigroup (Semigroup DLP) to the classic DLP in a sub*group* of the same semigroup. It follows that Semigroup DLP can be solved in polynomial time by quantum computers, and that Semigroup DLP has subexponential algorithms whenever the classic DLP in the corresponding groups has subexponential algorithms.

1. introduction

For Discrete Logarithm Problem (DLP) based cryptography, it is desirable to find efficiently implementable groups for which sub-exponential algorithms for the DLP are not available. Thus far, the only candidates for such groups seem to be (carefully chosen) groups of points on elliptic curves [4, 6]. Groups of invertible matrices over a finite field, proposed in [8], where proved by Menezes and Wu [5] inadequate for this purpose. In their paper [3], Kahrobaei, Koupparis and Shpilrain propose to use *semigroups* and, in particular, the semigroup of all matrices over a certain finite group-ring as a platform for the Diffie–Hellman protocol.

Let S be a semigroup. The *order* of an element $g \in S$ is the cardinality of the set $\{ g^k : k \in \mathbb{N} \}$. A semigroup S is *periodic* (a.k.a. *torsion*) if each element of S has finite order. The DLP in a periodic semigroup S is the problem of finding, given an element $g \in S$ and a power h of g, a natural number k such that $g^k = h$. *Semigroup DLP* is the general problem of solving the DLP in periodic semigroups.[1]

We will demonstrate that the Semigroup DLP is not harder than the classic DLP in groups. Moreover, the DLP in a semigroup S reduces, in polynomial time, to the DLP in a subgroup G of S. Thus, if there is a subexponential time algorithm for the DLP in subgroups of S (which is the case, for example, when S is a semigroup of matrices over a finite field, by the Menezes–Wu result [5]), then there is one for the DLP in S. In particular, as the DLP in groups is efficiently solvable by quantum computers, it follows that the Semigroup DLP is efficiently solvable by quantum computers.

Related work. In [7], Myasnikov and Ushakov reduce the DLP in the semigroup proposed in [3] to the DLP in a group of invertible matrices over a finite field, and deduce that the DLP in that semigroup can be solved by quantum computers. They achieve this goal by embedding the semigroup in the semigroup of all matrices over a finite field, and then applying Jordan Canonical Form theory to reduce the problem to the DLP in the group of invertible matrices over the same field. Our solution shows, in particular, that specialized methods are not necessary to solve this problem. We have reported our solution, without details, to Myasnikov and Ushakov, and suggested to ask experts whether this was known.

2010 *Mathematics Subject Classification.* 94A60.

Key words and phrases. discrete logarithm problem, quantum algorithms, semigroups.

[1]The nonperiodic case is discussed briefly in Section 3 below.

Following that, Myasnikov and Ushakov consulted Steinwandt, who mentioned the problem to some, including Childs and Ivanyos. Unaware of our solution, Childs and Ivanyos came up with an independent proof that the Semigroup DLP can be solved by quantum computers [2, Sections 1–3]. Their method is different, and uses quantum computers directly. Our solution is slightly more general, since we our reduction uses only classic computational assumptions. This is useful when quantum computers are not available, but subexponential algorithms are available for the DLP in the relevant groups, e.g., in matrix semigroups over finite fields.

2. A solution of the Semigroup DLP using a classic DLP oracle

Let G be a finite group. By *DLP oracle for G* we mean an oracle that, whenever provided with a generator g of G and a power h of g, returns a natural number k, of size polynomial in the relevant parameters (including the length of g and the order of G), such that $h = g^k$. Note that the oracle is not provided with G directly, but via its generator and the definition of multiplication in the group.

We assume that each element in the considered semigroup has a unique representation, or, equivalently for our purposes, a canonical representative that can be computed in polynomial time. We also assume that multiplication in the semigroup can be carried out in polynomial time.

The following lemma should be well known. For completeness, we include a proof.

Lemma 2.1. *Let S be a periodic semigroup, and g be a member of S. Let l, n be minimal with $g^{l+n} = g^l$, and let t be minimal with $tn > l$. Then the set*

$$G := g^l \cdot \{\, g^k : 0 \leq k < n \,\} = \{\, g^{l+k} : 0 \leq k < n \,\}$$

is a cyclic group of order n, with neutral element g^{tn} and generator g^{tn+1}. Moreover, $g^{tn} = g^{sn}$ for all $s \geq t$.

Proof. As $g^{l+n} = g^l$, the set G is closed under products. The element g^{tn} is neutral: As

$$g^{tn} g^l = g^{tn+l} = g^{l+tn} = g^l,$$

we have that $g^{tn} g^l g^k = g^l g^k$ for each element $g^l g^k \in G$. Let $s \geq t$. Then $g^{sn} = g^{tn+(s-t)n} = g^{tn}$.

Inversion: Given $g^{l+k} \in G$, let d be such that $l + k + d = tn - l$ modulo n. Then

$$g^{l+k} g^{l+d} = g^l g^{l+k+d} = g^l g^{tn-l} = g^{tn}.$$

Generator: As g^{tn} is neutral, for each element g^a in G we have that

$$g^a = (g^{tn})^a g^a = g^{tna+a} = (g^{tn+1})^a. \quad \square$$

Let S be a semigroup and let g be an element of S. Let l, n, and t be the numbers defined in Lemma 2.1.

Reduction 2.2. *Finding n, using a DLP oracle for G.*

Procedure. Fix a number N with $N \gg l + n$. This can be done by beginning with a fixed number N, and doubling it until the following procedure works. Choose random $k \in \{\lceil N/2 \rceil, \ldots, N\}$, and compute $h := g^k$. The distribution of h is statistically indistinguishable from the uniform distribution on G. As $\varphi(n)/n$ is greater than $1/(e^\gamma \log\log n + 3/\log\log n)$

for $n > 2$ [1, Theorem 8.8.7] (and is at least $1/2$ for $n = 1$ or 2), we assume, for a while, that h is a generator of G.

It is known that the order of a group G can be computed given a generator h and a DLP oracle for that group. Briefly, this can be done, using our notation, as follows. Choose a random $k \in \{1, \ldots, N\}$, and compute $k' := \log_h(h^k)$. It may be that calling the oracle twice with the same input, we obtain different values of k'. However, the distribution of k' depends only on $k \bmod n$ and not on k itself. Thus, taking the greatest common divisor of $O(1)$ differences $k - k'$ of this kind, we will obtain n.

Now, in any case, h generates a subgroup of G, whose order is found by the above-mentioned algorithm. This order divides the order of G. Repeating this procedure for $O(\log \log n)$ elements h, the maximum (or least common multiple) of the obtained orders will be the order of G. □

We now find t, using our knowledge of n. The neutral element of a group is its unique element e satisfying $e = e^2$, an idempotent element. By Lemma 2.1, we need to find the minimal t such that g^{tn} is an idempotent. Given that $g^{sn} = g^{tn}$ for all $s \geq t$ and $g^{sn} \neq g^{tn}$ for $s < t$, this can be done by binary search, as in the following algorithm.

Algorithm *Finding the minimal t such that g^{tn} is the neutral element of G*

1. $b \leftarrow 1$
2. **while** $g^{nb} \neq (g^{nb})^2$
3. $\quad b \leftarrow 2b$
4. $e \leftarrow g^{nb}$ (this is the neutral element of G)
5. $a \leftarrow \frac{b}{2}$
6. **repeat**
7. $\quad c \leftarrow \frac{a+b}{2}$
8. $\quad$ **if** $g^{nc} \neq e$
9. $\quad\quad$ **then**
10. $\quad\quad\quad a \leftarrow c$
11. $\quad\quad$ **else**
12. $\quad\quad\quad b \leftarrow c$
13. **until** $b - a = 1$
14. **return** b

Remark 2.3. One can use a variation of the above algorithm, that precomputes a logarithmic number of powers g^{2^i}, and replaces each power computation by one multiplication. This applies to all algorithms in this paper.

Let g^x be given. There are two cases to consider. First, assume that $g^x \in G$. Find t and n as above. Compute tn. Let $r = g^{tn+1}$, a generator of G. Using the oracle, we obtain a number x' such that $r^{x'} = g^x$. Then $g^x = r^{x'} = g^{x'(tn+1)}$. Take $k = x'(tn+1)$. Then $g^k = g^x$, and we are done.

The following immediate fact shows that membership in G can be tested efficiently.

Lemma 2.4. *For each $x \in \mathbb{N}$, we have that $g^x \in G$ if and only if $g^n g^x = g^x$.* □

If $g^x \notin G$, we use binary search to find the minimal b such that $g^{bn} g^x \in G$, as follows.

Algorithm *Finding the minimal b such that $g^{bn} g^x \in G$*

1. $b \leftarrow 1$
2. **while** $g^{bn}g^x \notin G$
3. $\quad b \leftarrow 2b$
4. $a \leftarrow \frac{b}{2}$
5. **repeat**
6. $\quad c \leftarrow \frac{a+b}{2}$
7. $\quad$ **if** $g^{cn}g^x \notin G$
8. $\quad\quad$ **then**
9. $\quad\quad\quad a \leftarrow c$
10. $\quad\quad$ **else**
11. $\quad\quad\quad b \leftarrow c$
12. **until** $b - a = 1$
13. **return** b

Similarly, if $g^k \in G$ and k is known, we can use binary search to find the maximal c such that $k - cn > 0$ and $g^{k-cn} \in G$.

Reduction 2.5. *Computing a discrete logarithm of g^x, using a DLP oracle for G.*

Procedure. It remains to consider the case where $g^x \notin G$. Let $r = g^{tn+1}$ be the generator of G. Use the above algorithm to find the minimal b such that $g^{bn}g^x \in G$. As n is the order of G, for each a with $g^{bn+x} = g^a$, we have that $bn + x \leq a$.

Using the oracle, compute $x' := \log_r g^{bn+x}$. Then $g^{bn+x} = g^{x'(tn+1)}$, and thus $bn + x \leq x'(tn+1)$. Note that the number $x'(tn+1)$ is known. Using binary search, find the maximal c such that $g^{x'(tn+1)-cn} \in G$. Then $bn+x = x'(tn+1)-cn$, and thus $x = x'(tn+1)-cn-bn$ is found. □

3. A comment on nonperiodic semigroups

Our assumption on the given semigroups are natural, but it may be still interesting to consider the case where the element g of S has *infinite* order. Indeed, this case was proposed, by Vladimir Shpilrain, at the conference *Algebraic Methods in Cryptography*, Ruhr Universität Bochum, Germany, 2005. In this case, the semigroup $\langle g \rangle$ generated by g is isomorphic to the additive semigroup $\mathbb{N}$, but the isomorphism may be infeasible to compute. However, as the second named author commented in that conference, there is likely to be a strong correlation between the bitlength of g^k and the power k (for each fixed coding of the semigroup elements). The following algorithm should be able to recover k from g^k in many cases of interest. In this algorithm, the function $\mathrm{len}(g)$ may be the bitlength of the presentation of g as a bitstring, or any other reasonable length function that tends to get larger for larger powers of g. Line 6 of the following algorithm should be implemented by binary search.

Algorithm *Find k given $h := g^k$, for g of infinite order.*

1. Choose large P and m, polynomial in the relevant parameters.
2. **for** i from 1 to m
3. $\quad$ Choose a random element $r \in \{\lceil P/2 \rceil, \ldots, P\}$.
4. $\quad \tilde{g} \leftarrow g^r$
5. $\quad \tilde{h} \leftarrow h^r$

6. $k \leftarrow \max\{\, k : \mathrm{len}(\tilde{g}^k) \leq \mathrm{len}(\tilde{h}) \,\}$
7. **if** $g^k = h$,
8. **return** k

The rational of this proposal is that a small number of bits can only code a limited number of semigroup elements, and thus a limited number of powers of g. Thus, on average, the higher the power of g is, the more bits are needed to code this power.

We have tested this algorithm in the case where G is the braid group, m (the number of tries) is 1 (!), P is 16, and $\mathrm{len}(g)$ is the number of generators in the canonical form of g.[2] For several parameter settings tested, the algorithm never failed. The algorithm did fail, occasionally, when we took P to be very small, so the contribution of the random power seems important.

It may be possible to fool this algorithm if the coding is chosen in a malicious way. The question whether there is, for each prescribed (black-box) infinite cyclic semigroup, an efficient solution to the the DLP remains, at present, open.

References

[1] E. Bach, J. Shallit, **Algorithmic Number Theory**, Vol I: Efficient Algorithms, MIT Press Series in the Foundations of Computing, Cambridge, MA, 1996.

[2] A. Childs, G. Ivanyos, *Quantum computation of discrete logarithms in semigroups*, arXiv eprint 1310.6238, October 2013.

[3] D. Kahrobaei, C. Koupparis, V. Shpilrain, *Public key exchange using matrices over group rings*, Groups, Complexity, and Cryptology 5 (2013), 97–115.

[4] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of computation 48.177 (1987), 203–209.

[5] A. Menezes, Y. Wu, *The discrete logarithm problem in $GL(n, q)$*, Ars Combinatoria 47 (1997), 23–32.

[6] V. Miller, *Use of elliptic curves in cryptography*, Advances in Cryptology–CRYPTO'85 Proceedings, Springer–Berlin–Heidelberg, 1986.

[7] A. Myasnikov, A. Ushakov, *Quantum algorithm for discrete logarithm problem for matrices over finite group rings*, Journal of Symbolic Computation, to appear.

[8] R. Odoni, V. Varadharajan, P. Sanders, *Public key distribution in matrix rings*, Electronics Letters 20.9 (1984), 386–387.

E-mail address, Matan Banin: baninmmm@gmail.com

E-mail address, Boaz Tsaban: tsaban@math.biu.ac.il
URL, Boaz Tsaban: http://www.cs.biu.ac.il/~tsaban

Department of Mathematics, Bar-Ilan University, Ramat Gan 5290002, Israel

[2] There are much better length functions for this group. We wanted to make life hard for our algorithm.