Adaption of Pollard's kangaroo algorithm to the FACTOR problem

Mario Romsy
Fakultät für Informatik
Universität der Bundeswehr München
mario.romsy@unibw.de

Abstract

In [BKT11] Baba, Kotyada and Teja introduced the FACTOR problem over non-abelian groups as base of an ElGamal-like cryptosystem. They conjectured that there is no better method than the naive one to solve the FACTOR problem in a general group. Shortly afterwards Stanek published an extension of the baby-step giant-step algorithm disproving this conjecture [Sta11]. Since baby-step giant-step methods are limited in practice because of memory requirements we present a modification of Pollard's kangaroo algorithm that solves the FACTOR problem requiring only negligible memory.

1 Introduction

Let $n \geq 2$ and $(G, \cdot)$ be a non-abelian finite group with identity element e. Suppose $g_1, \ldots, g_n \in G$ with $\langle g_1, \ldots, g_i \rangle \cap \langle g_{i+1} \rangle = \{e\}$ for all $i \leq n-1$ and let $x_1, \ldots, x_n \in \mathbb{Z}$. Given an element

$$h = g_1^{x_1} \cdot \ldots \cdot g_n^{x_n}$$

we wish to determine $g_1^{x_1}, \ldots, g_n^{x_n}$ (we note that the conditions on $g_1, \ldots, g_n$ imply the uniqueness of the solution). This is called generalized FACTOR problem or n-FACTOR problem. In case $n = 2$ we also say FACTOR problem instead of 2-FACTOR problem.

The cryptosystems in [BKT11] are based on the 2-FACTOR problem. Stanek's modification of Shank's baby-step giant-step algorithm solves this problem using time and memory $O(\sqrt{\mathrm{ord}(g_1)\mathrm{ord}(g_2)})$. For practical purposes it is desirable to reduce at least the memory requirements. This can be achieved by a simple modification of Pollard's kangaroo method presented in the next section.

2 Kangaroos solving the FACTOR problem

The connection between the discrete logarithm problem (DLP) and the FACTOR problem was already brought up in [BKT11]. At least after Stanek's work it is natural to look at other generic DLP-algorithms for possible adaptability to the FACTOR problem. Since the iteration function in Pollard's rho method requires the calculation of powers of $h = g_1^{x_1} g_2^{x_2}$ we are stuck because of non-commutativity[1]. But we have better luck with Pollard's kangaroo algorithm. For a detailed description of the original kangaroo algorithm see [Pol78].

We now describe the modified version (where we have no additional information about the exponents x_1 and x_2, see remarks below).

- Phase 0 - initialization:
  Calculate $\mathrm{ord}(g_1)$ and $\mathrm{ord}(g_2)$, fix $s \in \mathbb{N}$ (in practice $s \approx 20$) and define (pseudorandom) partition functions
  $$p_1 : G \to \{1, \ldots, s\}$$
  and
  $$p_2 : G \to \{1, \ldots, s\}$$
  Choose random constants $u_1, \ldots, u_s \approx \sqrt{\mathrm{ord}(g_1)}$ and $v_1, \ldots, v_s \approx \sqrt{\mathrm{ord}(g_2)}$.

- Phase 1 - tame kangaroo:
  Set $T = e = g_1^0 g_2^0$ and $d_1^t = 0, d_2^t = 0$.
  Repeat $\lceil \sqrt{\mathrm{ord}(g_1)\mathrm{ord}(g_2)} \rceil$ times:

  1. $d_1^t \leftarrow d_1^t + u_{p_1(T)} \mod \mathrm{ord}(g_1)$
  2. $d_2^t \leftarrow d_2^t + v_{p_2(T)} \mod \mathrm{ord}(g_2)$
  3. $T \leftarrow g_1^{u_{p_1(T)}} \cdot T \cdot g_2^{v_{p_2(T)}} = g_1^{d_1^t} \cdot g_2^{d_2^t}$

- Phase 2 - wild kangaroo:
  Set $W = h = g_1^{x_1} g_2^{x_2}$ and $d_1^w = 0, d_2^w = 0$.
  While $W \neq T$ do

  1. $d_1^w \leftarrow d_1^w + u_{p_1(W)} \mod \mathrm{ord}(g_1)$
  2. $d_2^w \leftarrow d_2^w + v_{p_2(W)} \mod \mathrm{ord}(g_2)$
  3. $W \leftarrow g_1^{u_{p_1(W)}} \cdot W \cdot g_2^{v_{p_2(W)}} = g_1^{d_1^w + x_1} \cdot g_2^{d_2^w + x_2}$

- Phase 3 - kangaroo collision:
  If $T = W$ we have
  $$g_1^{d_1^t} \cdot g_2^{d_2^t} = T = W = g_1^{d_1^w + x_1} \cdot g_2^{d_2^w + x_2}.$$
  Thus $x_1 = d_1^t - d_1^w$ and $x_2 = d_2^t - d_2^w$, so $g_1^{x_1}$ and $g_2^{x_2}$ are easily calculated.

Remark.
- We note that the kangaroo method is a probabilistic algorithm. If there is no collision after e.g. $3\lceil \sqrt{\mathrm{ord}(g_1)\mathrm{ord}(g_2)} \rceil$ steps in phase 2, the attack should be restarted with different initialization values and/or a different starting value for T.

[1] There is a Pollard-rho-like algorithm of Bisson and Sutherland that can be used to solve the FACTOR problem, see the remark at the end of this section.

- The original kangaroo method solves the discrete logarithm problem $z = y^\beta$ in expected running time $O(\sqrt{\mathrm{ord}(y)})$ or even $O(\sqrt{\gamma - \alpha})$ if it is known that $\beta \in [\alpha, \gamma]$. The analysis of our modified version can be done as in [Pol78] yielding an expected running time of $O(\sqrt{\mathrm{ord}(g_1)\mathrm{ord}(g_2)})$, where the algorithm can be improved in an obvious way if it is known that $x_1 \in [\alpha_1, \gamma_1]$ and/or $x_2 \in [\alpha_2, \gamma_2]$.

- Revealing x_1 and x_2 the algorithm delivers more than we asked for.[2]

Remark. In an earlier version of this paper we claimed that there is no direct way to extend both attacks, Stanek's and ours, to solve the general FACTOR problem. In fact this does not hold for the baby-step giant-step algorithm as the version of Bisson and Sutherland in [BS11] shows. In the very same paper Bisson and Sutherland also present a Pollard-rho-like algorithm for finding short product representations in finite groups. The setting is as follows.

Let $S = (s_1, \ldots, s_t)$ be a (random) sequence of elements of a group G and let $z \in G$. If the sequence S satisfies $t \geq 2\log_2|G|$ the Pollard-rho algorithm of Bisson and Sutherland finds a subsequence $(s_{i_1}, \ldots, s_{i_r})$ of S with $z = s_{i_1} \cdot \ldots \cdot s_{i_r}$ in expected running time $O(\sqrt{|G|}\log_2|G|)$ using negligible memory. This method can be applied to solve the general FACTOR problem if we choose S the following way:

- For all i assemble a sequence S_i from $g_i, g_i^2, \ldots g_i^{2^{b_i}}$ where $b_i = \lceil \log_2(\mathrm{ord}(g_i)) \rceil$ (to make sure that there is a solution) and sufficiently many random powers of g_i .

- For all i permute the elements of S_i in a random way.

- Build S by concatenation of $S_1, \ldots, S_n$.

Acknowledgements

Thanks to G. Bisson and A. Sutherland for pointing me to their work [BS11].

References

[BS11] G. Bisson, A. Sutherland, *A low-memory algorithm for finding short product representations in finite groups*, to appear in Designs, Codes and Cryptography, available at `http://www.springerlink.com/content/4293k3621h3316j7`

[BKT11] S. Baba, S. Kotyada, R. Teja, *A Non-Abelian Factorization Problem and an Associated Cryptosystem*, Cryptology ePrint Archive, Report 2011/048, 2011, `http://eprint.iacr.org/2011/048`.

[Pol78] J. Pollard, *Monte Carlo methods for index computation mod p*, Mathematics of Computation, Vol. 32, 1978.

[Sta11] M. Stanek, *Extending Baby-step Giant-step algorithm for FACTOR problem*, Cryptology ePrint Archive, Report 2011/059, 2011, `http://eprint.iacr.org/2011/059`.

[2] In fact the author is not aware of any algorithm that solves the FACTOR problem without giving x_1 and x_2.