Searching B-smooth numbers using quantum annealing: applications to factorization and discrete logarithm problem*

Olgierd Żołnierczyk[1][0000−0002−5196−3494] and Michał Wroński[1][0000−0002−8679−9399]

Military University of Technology, Kaliskiego Str. 2, Warsaw, Poland
{olgierd.zolnierczyk, michal.wronski}@wat.edu.pl

Abstract. Integer factorization and discrete logarithm problem, two problems of classical public-key cryptography, are vulnerable to quantum attacks, especially polynomial-time Shor's algorithm, which has to be run on the general-purpose quantum computer. On the other hand, one can make quantum computations using quantum annealing, where every problem has to be transformed into an optimization problem, for example, the QUBO problem. Currently, the biggest available quantum annealer, D-Wave advantage, has almost 6,000 physical qubits, and therefore it can solve bigger problems than using general-purpose quantum computers. Even though it is impossible to run Shor's algorithm on a quantum annealer, several methods allow one to transform factorization or discrete logarithm problems into the QUBO problem. Using a D-Wave quantum annealer, the biggest factored integer had 20 bits, and the biggest field, on which it was possible to compute a discrete logarithm problem using any quantum method, had 6 bits. This paper shows how to transform searching for B-smooth numbers, an important part of the quadratic sieve method for factorization and index calculus for solving discrete logarithm problems, to the QUBO problem and then solve it using D-Wave Advantage quantum solver. The linear algebra step for integer factorization and index calculus methods has been solved using classical computations. Using our method, we factorized the 26-bit integer and computed the discrete logarithm problem over the 18-bit prime field. Therefore we broke the current records in factorization using quantum annealing by 6 bits and in discrete logarithm problem, using any quantum method, by 12 bits.

Keywords: Integer factorization · discrete logarithm problem · D-Wave · quantum annealing · cryptanalysis.

* This work was supported by the Military University of Technology's University Research Grant No. 858/2021: "Mathematical methods and models in computer science and physics."

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

1 Introduction

The integer factorization problem (IFP) and discrete logarithm problem (DLP) over finite fields are some of the most widespread problems in modern classical public-key cryptography. Let us recall these notions, denoting the set of prime numbers as $\mathbb{P}$.

Definition 1 (Integer factorisation problem). *The task of finding $p_1, p_2, ..., p_k \in \mathbb{P}$, such that $\prod_{i=1}^{k} p_i = n$, being given n only; is called the integer factorisation problem.*

Definition 2 (Discrete logarithm problem over finite field). *Let $q = p^e, p \in \mathbb{P}, e \in \mathbb{N}_{>0}$ and $\langle g \rangle = G \leq \mathbb{F}_q^*$. The task of finding $y \in \{0, 1, \ldots, ord(g) - 1\}$, being given an element from multiplicative subgroup $h \in G$, such that $g^y = h$; is called discrete logarithm problem over a finite field.*

In general, the most efficient, non-quantum method to solve the integer factorization problem is the general number field sieve (GNFS) method (but in this work, we use a simpler version - the quadratic sieve method). The improved method – number field sieve for discrete logarithm is used to solve DLP.

Since 1994 it has been known, according to Shor [13], that there exists a quantum algorithm that can solve integer factorization and discrete logarithm problem over finite fields in polynomial time. Moreover, in 2003, Proos and Zalka [12] presented how to apply polynomial-time Shor's algorithm to solve the discrete logarithm problem on elliptic curves. Since then, there have been many efforts to implement Shor's algorithm practically on existing quantum computers. Unfortunately, till now, the record of the integer factorization problem, computed using Shor's algorithm, is number 21 [9]. Applications of Shor's algorithm for DLP over finite fields and elliptic curves have not been reported.

On the other hand, many more methods solve these problems. One such method is transforming the given problem into the Ising or QUBO problem. Such problems may be solved using adiabatic quantum computers. Using such a method, it was possible to make factorization of 40-bit length number $N = 1,099,551,473,989$ using superconducting quantum computer [3], 20-bit integer $N = 1,028,171$ using quantum annealing [14]. Using quantum annealing, it was also possible to compute DLP over a 6-bits prime field $\mathbb{F}_{59}$. Secondly, the latest research shows [16] it is possible to solve the integer factorization problem by disposing of approximately $\frac{(\log n)^2}{4}$ logical qubits and DLP over $\mathbb{F}_p$, disposing of $2(\log p)^2$ logical qubits. These methods, however, do not outperform Shor's algorithm. In the case of ECDLP, using a hybrid quantum solver, searching for relations in the index calculus method also has been computed quantumly for 8-bit prime $p = 251$ [15]. The second step of this method, the linear algebra step, has been computed using a classical computer.

It should be noted that a proposal for solving the IFP using the Schnorr method and QAOA has also emerged [17]. However, determining its effectiveness (similarly to the aforementioned examples) requires a deeper analysis.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

Let us define the problem whose solution we can obtain using the quantum annealing machine. The problem can be presented as a multivariate binary polynomial $f(x_i, \dots x_n)$ of degree less or equal to 2 with real coefficients. The task is to find the values of variables for which the polynomial takes the minimal value (in particular, one of such vectors).

Having constrained the degree of the polynomial, we could denote coefficients as follows: a_i –terms of degree one, $b_{i,j}$ – terms of degree two. Such a problem is called the QUBO problem – Quadratic Unconstrained Binary Optimization and can be expressed as follows:

$$f(x_1, \dots, x_n) = \sum_i a_i x_i + \sum_{i<j} b_{i,j} x_i x_j. \tag{1}$$

In this paper, we use the modified factorization procedure, presented by [5], as a subroutine to check if an integer, randomly chosen in a quadratic sieve or index calculus method, has all factors less than or equal to the given bound B.

It is worth noting that quadratic sieve and index calculus methods have limited applications. The complexities of these methods in the classical case are subexponential. In the quantum case, it is also clear that the quadratic sieve and index calculus method for finite fields will not outperform Shor's algorithm. Nevertheless, the methods presented below may show some promise compared to classical methods, depending on a more thorough investigation of the complexity of quantum annealing.

Even if the presented by us method has some limits in its applications, we factorized the 26-bits integer and computed the DLP over the 18-bits prime field using this method. Therefore we broke the current records in IFP using quantum annealing by 6 bits and in DLP, using any quantum method, by 12 bits. As reported later, a hybrid classical-quantum application of a general number field sieve for both IFP and DLP should allow finding solutions in about 50-bit cases without increasing the number of logic variables in given QUBO problems. These researches require, however, much more effort.

2 Classical methods for integer factorization and discrete logarithm

2.1 Quadratic sieve method

One of the most crucial integer factorization method is the quadratic sieve method, proposed by Carl Pomerance in 1981. Up to an approximately 340-bit length of factored number, the QS is the most efficient method since, for larger inputs, the general number field sieve method is more efficient.

An observation underlying this method is the following fact. For a given factorization problem (as defined by Definition 1), having a pair of numbers for which holds:

$$a^2 \equiv b^2 \pmod{n} \tag{2}$$

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

and

$$a \not\equiv \pm b \pmod{n}, \tag{3}$$

implies the possibility of computing $d \,|\, n$, (it is because $a^2 - b^2 \equiv 0 \pmod{n} \vdash n \,|\, (a+b)(a-b) \vdash gcd(a \pm b, n)$ – non-trivial divisor of n, where $\vdash$ denotes a conditional assertion, so $\mathcal{P} \vdash \mathcal{Q}$ means that from $\mathcal{P}$, we know that $\mathcal{Q}$.

The naive way of searching the above relation (a, b) is replaced by splitting this task into more steps. Namely, we explore many weakened relations of the form $a_i^2 \equiv b_i$, where all prime factors of b_i are less or equal a bound B – it means b_i is B-smooth. Finally, we multiply the congruences corresponding to the selected pair (a, b), creating primary relation (Equation 2), which we need. We choose the base of prime factors for this purpose:

$$\mathcal{B} = \big\{ -1, p_2, p_3, \ldots, p_k \mid \mathbb{P} \ni p_i < B \big\}, \tag{4}$$

fixing a well adjusted bound B and next, denoting $\#\mathcal{B} = k$.

Using the following polynomial:

$$T(c) = (m+c)^2 - n \equiv (m+c)^2 \pmod{n}, \tag{5}$$

where $m = \lfloor \sqrt{n} \rfloor$, we can generate the set of weakened relations: $T(c) = b_c \equiv a_c^2$. We select only these, where b_c is B-smooth, and we vary $-M < c < M$, to obtain suitable cardinality of set $\mathcal{A} \ni b_i$. If we establish each prime factorization $\mathcal{A} \ni b_i = \prod_{j=1}^{k} p_j^{e_j}$, we are able to obtain a non-trivial solution of the following system of equations:

$$\begin{cases} e_{11}x_{11} + e_{21}x_{21} + \cdots + e_{k1}x_{k1} & = 0, \\ e_{12}x_{12} + e_{22}x_{22} + \cdots + e_{k2}x_{k2} & = 0, \\ \vdots & \\ e_{1A}x_{1A} + e_{2A}x_{2A} + \cdots + e_{kA}x_{kA} & = 0, \end{cases} \tag{6}$$

over a prime field $\mathbb{F}_2$. The simplest way for this linear algebra step is Gauss elimination, based on transforming the matrix to row echelon form by adding multiplication (in the general case over $\mathbb{F}_p$) of a row to picked rows. Although the cost of the Gauss algorithm is too high ($O(B^3)$), in practice, we exploit the fact that we work with a sparse matrix in the quadratic sieve method. As a result, the application of an efficient general procedure as Lanczos's and Wiedemann's algorithms [4], achieve complexity $O\left(B^{2+o(1)}\right)$, equaling the time bound for sieving.

Thus, the solution will indicate the set of indices S, such that $\prod_{i \in S} b_i = b^2$. Additionally, due to the form of the polynomial, we have $\prod_{i \in S} a_i = a^2$ and $a^2 \equiv b^2 \pmod{n}$. Eventually, we obtain primary relation (Equation 2). It is worth noting, because of the preceding, that we neglect the condition from Equation 3; however, the possibility of gaining a non-trivial divisor is relatively high (> 0.5).

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

The quadratic sieve method has a few vital improvements: one big prime (two big primes), a multi-polynomial variant, and self-initialization. The complexity of the method, according to [2], is $O\left(\exp\left(\sqrt{\ln n \ln\ln n}\right)\right)$.

2.2 Index calculus method

An analogous method for DLP, corresponding to QS, is the index calculus method, proposed by Leonard Adleman in 1979. It has similar proprieties, the same main idea based on searching B-smooth number, and the same complexity. This is the simplest variant of the method, presented in two stages.

First stage. Let $\langle g \rangle = \mathbb{F}_p^*$ for some prime p and our goal is to find $y : g^y \equiv h \pmod{p}$. The factor base

$$\mathcal{B} = \left\{p_1, p_2, \ldots, p_k \mid \mathbb{P} \ni p_i < B\right\} \tag{7}$$

consists of k small primes. For random $x \in \{1, 2, \ldots, p-2\}$ we try to obtain relations of the form:

$$\mathbb{F}_p^* \ni g^x = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}, \tag{8}$$

by finding g^x smooth over $\mathcal{B}$. Each relation obtained implies, that for certain indexes $i_{p_j} : \mathbb{F}_p \ni g^{i_{p_j}} = p_j$, holds

$$e_1 i_{p_1} + e_2 i_{p_2} \ldots e_k i_{p_k} \equiv x \pmod{p-1}. \tag{9}$$

Collecting enough relations (coefficient matrix to be of full column rank), we will be able to set and solve the system of equations modulo $p-1$:

$$\begin{cases} e_{11} i_{p_1} + e_{12} i_{p_2} + \cdots + e_{1k} i_{p_k} & \equiv x_1 \pmod{p-1} \\ e_{21} i_{p_1} + e_{22} i_{p_2} + \cdots + e_{2k} i_{p_k} & \equiv x_2 \pmod{p-1} \\ \vdots & \\ e_{a1} i_{p_1} + e_{a2} i_{p_2} + \cdots + e_{ak} i_{p_k} & \equiv x_a \pmod{p-1}. \end{cases} \tag{10}$$

If so, we uniquely get the indices i_{p_j}. Detailed computation involves factoring $p-1$, solving the system modulo each prime power separately, and combining all solutions using the Chinese Remainder Theory to obtain the unique one. We use the same techniques announced in subsection 2.1. This fulfills the first stage.

Second stage. The second stage is possible due to collected values $i_{p_1}, i_{p_2}, \ldots, i_{p_k}$. Namely, we keep picking $z \in \{1, 2, \ldots, p-1\}$ randomly to find one such that:

$$\mathbb{F}_p^* \ni h g^z = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}, \tag{11}$$

so until hg^z will be B-smooth. As a result, we can compute searched y, indeed

$$y \equiv a_1 i_{p_1} + a_2 i_{p_2} + \cdots + a_k i_{p_k} - z \pmod{p-1}. \tag{12}$$

Therefore the goal has been achieved.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

3 Hybrid methods

It is well known that the relation-collection phase determines the running time of the methods above and all algorithms based on searching B-smooth numbers. Given the quantum factorization technique with limited input, one can apply it to check smoothness in one of these methods, making them a hybrid with the quantum smoothness searching stage.

3.1 Known results and previous work

In 2017 Daniel Bernstein et al. proposed a low-resource quantum factoring algorithm [1]. This algorithm uses improved Shor's [13] procedure for integer factorization as a search criterion function in Grover's searching algorithm [6] to create a quantum circuit for detecting B-smooth numbers. Eventually, this circuit is a subroutine in the general number field sieve. Due to some improvements, Shor's algorithm can take as an input superposition of many sieved numbers. Thus, Bernstein's method is a trade-off between quantum and classical resources. As a result, this very efficient approach has classical complexity $O(L^{\sqrt[3]{\frac{3}{8}}+o(1)})$, where $L = \exp\left((\ln n^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}})\right)$ (n is a number to be factored) and uses $O\left(\log n^{\frac{2}{3}}\right)$ qubits. In terms of quantum computing development, this method can be used with fewer quantum resources than Shors's algorithm to factorize numbers greater than using the general number field sieve method.

Similar hybrid ideas exist for quantum annealing. Michele Mosca et al. presented a significant conceptual result in 2019 [10]. Mosca's method also uses GNFS and assumes searching of smoothness quantumly. However, in this method, the quantum stage consists of the transformation elliptic curve method for factorization (ECM) to the QUBO problem (in the original paper, more general to the SAT problem). ECM complexity depends on the least divisor of factorized number. Thus, the algorithm is useful for splitting probably B-smooth number. The QUBO problem, equivalent to the smoothness problem, is made from a sequence of ECM blocks connected input-output. This way is realized by extracting all smooth divisors. Mosca's method achieves better complexity than classical GNFS if only quantum annealing reaches non-trivial speedup compared to classical solvers. Under maximal optimistic conditions, this method has computational complexity asymptotically equal to Bernstein's method.

We present a similar concept below, using QS and index calculus method.

3.2 Our result – factorization by quantum annealing as a subroutine

We present two hybrid methods based on the quadratic sieve and the index calculus algorithm. We use the modified quantum annealing factorization technique in both to detect B-smooth numbers. The complexity of these methods cannot be better than subexponential (due to the number of quantum subroutines required), so they do not significantly outperform known results. However,

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

the proposed methods allow for examining the efficiency of one of the quantum-classical solutions in practice and break factorization and discrete logarithm records with quantum annealing. Furthermore, their more detailed comparison is currently unknown due to the lack of a deeper analysis of the complexity of quantum annealing.

The difference between the direct and presented quantum annealing factorization method is the following. Based on the idea of [11], we try to find a lot of small factors instead of finding the factorization of semiprime. Therefore, we introduce a different definition of the input problem. The second biggest difference is the nested way we apply the factorization using quantum annealing. So presented quantum annealing factorization technique below is a subroutine of both hybrid methods. As a result, even though the direct method has better complexity, applying the sieve methods allows solving bigger instances of IFP and DLP than direct methods (with quantum annealing power available today). Our idea is also some intermediate step between classical sieve methods and the propositions of Bernstein et al. [1] and Moska et al. [10].

Checking smoothness in presented hybrid methods is performed by splitting the candidate to be smooth until all the divisors are less or equal to B, or factorization will return a false divisor (not smooth). In the factorization subroutine, we try to find a divisor of the factorized number less or equal to some bound B. We achieve this by setting suitable bit lengths of both searched divisors, so the smaller one is $\leq B$. However, if factorization fails, we treat obtained outputs as primes greater than B. The essential idea of the quantum annealing factorization technique is to present the factorization problem as a QUBO problem. It will be done in the following main steps.

Establishing multiplication table. Let n be the number to be factored (the candidate for being smooth). First, the procedure requires constant bit lengths of factors b, d, where $d < b$, denoted as l_d, l_b (we will generally indicate the bit length in this way). The expected maximal length depends on the least and the largest primes in the base: $l_b = l_{\lfloor n/p_1 \rfloor}, l_d = l_{p_t}$.

(assuming p_1 is the smallest prime in the base and p_t is the biggest one). In practice, the base used in quantum annealing can also be reduced by applying initial naive division on a classic machine.

Then we follow the steps discussed in [11], so division into column blocks should be established in the multiplication table. The width of each column block, denoted by the W_i for an i-th block, can be 2 or 3 (except the first block width, which always equals 1). We consider bits d_i, b_i, n_i of, respectively: d, b, n and carry bits c_i. Preceding the quantum annealing phase by splitting powers of 2 from the input number, we get the following variables and column blocks, represented on Table 1.

Counting carry bits. We aim to express the factorization problem as a cost function. We split the problem into smaller parts by treating each block separately. This technique reduces the values of bias coefficients and the number of logical

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

							d_{l_d}	.	.	.	d_3	d_2	1
							b_{l_b}	.	.	.	b_3	b_2	1
					c_1	d_{ld}	.	.	.	d_3	d_2	1	
			c_2	$d_{l_d}b_1$	.	.	.	d_3b_2	d_2b_2	b_2			
		$. d_{l_d}b_3$		.	.	.	d_3b_3	d_2b_3	b_3				
	$. .$							.					
	$. .$						.						
c_k	.					.							
$d_{l_d}b_{l_b}$	$. . .$	$d_3b_{l_b}$	$d_2b_{l_b}$	b_{l_b}									
n_{l_n}		.		.	.						n_3	n_2	1
		.		.	.								
s-th block											2. block		1. block

Table 1: Multiplication table.

qubits needed to solve this problem. Thus, we sum terms with reduced column weights: from 2^0 to 2^{W_i-1}. The notation is the following:

- N_i – the number read from bits of n narrowed to an i-th block with shifted weights
  2^0 – 2^{W_i-1} (called target value),
- $F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b})$ – multiplication result polynomial for an i-th block,
- $C_i(c_1, \dots c_k)$ – carry variables polynomial for an i-th block, the sum of variables c from i-th block with shifted weights 2^0 – 2^{W_i-1}.

Eventually, we formulate the following expressions from the multiplication table:

$$\begin{aligned} f_i &= F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b}) + C_i(c_1, \dots c_k) - 2^{W_i} C_{i+1}(c_1, \dots c_k) - N_i \\ f &= f_2^2 + f_3^2 \dots f_s^2 \end{aligned} \tag{13}$$

(we assume $C_{s+1} = 0$), receiving the formula $f(d_2, \dots, d_{l_d}, b_2, \dots, b_{l_b}, c_1, \dots c_k) \to 0$ as an equivalent to IFP.

While numbers l_d, l_b, l_n are known, we need to determine the number of carry bits k, especially the number of carry bits in i-th block K_i (surely, there is no gap between c_i within one block). Let us express the maximal value of the i-th block $F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b})$ as Max_i. Obviously, a congruence $F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b}) \equiv N_i \pmod{2^{W_i}}$ must hold. So, we define Max_i as:

$$Max_i = max\{F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b}) | F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b}) \equiv N_i \ (mod\ 2^{W_i})\}. \tag{14}$$

Therefore we want to know if $F_i(d_2, \dots d_{l_d}, b_2 \dots, b_{l_b})$ takes any value between minimum and maximum. The blocks with the exact position of columns signed by d_{l_d}, b_{l_b}, d_2, within this block, and exact width, turn out to have the same properties - similarly to common properties shown in [11]. For example, all blocks in type $\{left, 1, right\}$ with width 3 have the same properties. Thus, the set of taken values for F_i could be precomputed for all types of blocks in practice, and the exact value of K_i can be computed taking into consideration C_i, which can take any values between minimum and maximum.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

Linearisation. We must reduce the polynomial to quadratic form to transform the cost function to the QUBO. Therefore, if in the polynomial f_i there are monomials of degree ≥ 2, then we need to linearise them so that they will be quadratic after the polynomial is squared.

Let us consider the monomial of binary variables $\gamma x_1 x_2 x_3$ with real coefficient γ. We aim to reduce the degree of this monomial by one (finally, to degree equals 1). We create an auxiliary variable $a \in \{0, 1\}$ to replace $x_1 x_2$. Adding a new expression, the so-called penalty

$$P_a = 2(x_1 x_2 - 2a(x_1 + x_2) + 3a), \tag{15}$$

to monomial, we get the equivalence of the following transformation

$$\gamma x_1 x_2 x_3 \longrightarrow \gamma a x_3 + P_a. \tag{16}$$

To be more precise, the minimal value of both expressions is the same, and the set of point for which the left expression take minimal value is equal to the set of point for which the right expression take minimal value.

We can prove this by considering all possible values of these expressions.

x_1	x_2	x_3	$\gamma x_1 x_2 x_3$	a	$\gamma a x_3 + P_a$	
0	0	0	0	0	0	$x_1 x_2 = a$
0	0	1	0	0	0	
0	1	0	0	0	0	
0	1	1	0	0	0	
1	0	0	0	0	0	
1	0	1	0	0	0	
1	1	0	0	1	0	
1	1	1	γ	1	γ	
0	0	0	0	1	6	$x_1 x_2 \neq a$
0	0	1	0	1	$\gamma + 6$	
0	1	0	0	1	2	
0	1	1	0	1	$\gamma + 2$	
1	0	0	0	1	2	
1	0	1	0	1	$\gamma + 2$	
1	1	0	0	0	2	
1	1	1	γ	0	2	

Table 2: Table of all values taken by considered expression $\gamma a x_2 + P_a$. The minimal values are taken only when $x_1 x_2 = a$, so $P_a = 0$.

It is worth noting that our expression takes minimal value only when $P_a = 0$. However, since we are interested in minimal values of a squared polynomial, we should consider the absolute value of the sum of such monomials. We observe that the absolute value of the sum of many of these monomials has a minimum equal to zero and is minimal only when $P_a = 0$ also. Moreover, excluding P_a from the absolute value brackets $x_3 = 1$ does not change these properties. So, in particular, in the cases when $x_3 = 1$. In other words, we obtain equivalence in the following transformation $f^2 \to f_{lin}^2 + P$, where f_{lin} is the linear form of the polynomial f, obtained from many substitutions $x_i x_j \to a_l$ performed on f.

The conclusions for the linearisation procedure follow. We can transform a cost function f_i to the QUBO form by keeping on substitution $x_i x_j \to a_l$ and

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

summing up all penalties separately, as long as f has a degree greater than 1. Finally, we square f_{lin}, add the sum of penalties, and $deg(f_{lin}^2 + P) = 2$.

Formulating QUBO problem. To summarise, formulating QUBO to find divisors of n consists of the following steps, preceded by precomputation of lists of values taken by the function F_i from each type of block. One can also assume that n is nondivisible by a few small primes, for example, $\{2, 3, 5, 7\}$.

1. Establish the multiplication table (see Table 1) by initializing an array of subsequent indices of the first columns of each block.
2. Initialize $f = 0, P = 0$.
3. For second block fix $C_2 = 0$
4. For each non-empty, subsequent block, do:
   - Fix the carry bits number K_{i+1}, exiting i-th block, knowing target value N_i and list of values taken by function F_i (see section 3.2). Initialise corresponding polynomial C_{i+1} (knowing K_{i+1}).
   - Substitute d_2 by $n_2 \oplus b_2$ in F_i.
   - Linearise polynomial F_i, saving all triples (x_1, x_2, a) and adding penalties to P (see section 3.2).
   - Fix the polynomial $f_i^2 = \left(F_i + C_i - N_i - 2^{W_i} C_{i+1}\right)^2$ for this block.
   - Add $f \leftarrow f + f_i^2$
5. Add $f \leftarrow f + Pen$

For simplicity, we do not stand out the last step, which one can reduce to the substitution of d_1, linearisation, fixing f_i^2 and adding $f + f^2$.

Requesting quantum computer. The Quantum annealing sampler can be used now to find the solution to the QUBO problem, determining divisors d, b due to returned low-energy states of the objective function.

3.3 Quantum annealing stage – summary

With suitably adjusted l_d and B, we no longer have to split d to check smoothness, but it is generally sufficient to compare d to B. The nested applications of the factorization subroutine above lead us to examine if all prime factors of n belong to $\mathcal{B}$ or not. Potentially failure is simply detected by verifying if $d \mid n$ is considered as a negative response in checking smoothness. Thus we obtain smoothness checking by the quantum annealing stage.

4 Experiments

The experiments investigated the maximal size of solvable factorization and discrete logarithm problems by applying the above method on the most potent quantum annealing computer available today, D-Wave. We have established current size records of solved discrete logarithm problem: 18-bits, using any quantum method, integer factorization problem: 26-bits, using quantum annealing.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

All quantum computations have been made involving direct QPU solver *Advantage_system4.1* from D-Wave. The quantum computer was requested using Ocean SDK and *D-Wave.system* library. Sampling has been carried out with the number of reads equaling 10,000, and requested problems were formulated as QUBO problems, as described above. Coefficients in QUBO were autoscaled by solver API to hardware ranges to meet the requirements of the QPU. The used quantum machine has the following main properties: the number of working qubits: 5627, weight (qubits coefficients) range: $[-4, 4]$, strengths (couplers coefficients) range: $[-1, 1]$, annealing time range: $[0.5\mu s, 2ms]$, default annealing time: 20 μs.

4.1 Results for integer factorization

We apply our hybrid method for integer factorization based on the variant of the quadratic sieve method described in subsection 2.1. As shown above, the hybrid method involved modified factoring by annealing procedure used as a subroutine in the sieving stage from the quadratic sieve.

We have adopted the following methodology to select sample input data for each input size. Firstly, we used a parameter β in bounding arguments for generating polynomial from subsection 2.2. Secondly, trying different β, B, n values, each complete performance of the method was initially realized on the classical computer as many times as needed to find an instance with sieved numbers size according to asymptotic constrain. Then, for fixed β, the candidates for being smooth were generated via this polynomial with random arguments from range $[0, \beta]$ as far as a full rank matrix of relations has been completed. In some cases, the procedure was repeated several times. We have excluded the possibility of simultaneously finding the whole relation of two squares. The best approach is to use the official RSA challenge numbers, but it is impossible due to the current state of development of quantum annealing computers available today.

The final results are presented in two tables. In table 3 are listed specifications of problems, so the quantities from the classical part of the method, while in table 4 are listed values describing the usage of quantum machine for every problem.

n	problem size [bit]	β	sieved numbers size [bit]	matrix dimensions	base
$874219 = 1013 \times 863$	20	4	4-14	2 x 5	$[-1, 2, 3, 5, 11]$
$3812491 = 2029 \times 1879$	22	4	5-15	2 x 6	$[-1, 2, 3, 5, 7, 11]$
$8732021 = 2953 \times 2957$	24	5	8-15	2 x 6	$[-1, 2, 3, 5, 7, 11]$
$42273409 = 6709 \times 6301$	26	4	7-16	3 x 6	$[-1, 2, 3, 5, 13, 17]$

Table 3: Results of experiments with solving integer factorization problem by hybrid quadratic sieve: classical part.

Columns of table 3 contain the following values, respectively, from left: 1. input number n, to be factored, 2. the bit length of input number n, 3. parameter β fixed during selecting examples used for bound random range for candidates to being smooths, 4. ranges of bit length of numbers, from every nested factorization subroutine step, sieved in smoothness checking stage, 5. dimension of the matrix of coefficients from the system of congruences, 6. base $\mathcal{B}$ used to sieving.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

The above values show that with currently available quantum annealing resources, we can factor up to 26-bit semiprime by the procedure of quantumly splitting many 16-bit numbers, applied in a nested way in the quadratic sieve.

problem size [bit]	number of logical qubits	number of physical qubits	QPU total time [ms]
20	9-45	14-261	182
22	9-49	14-326	796
24	29-41	60-367	678
26	22-77	67-772	8705

Table 4: Results of experiments with solving integer factorization problem by hybrid quadratic sieve: quantum part.

Columns of table 4 contain the following values, respectively, from left: 1. the bit length of input number n denoting the same problem from Table 3, 2. number of variables from the QUBO problem, from every nested factorization subroutine step, 3. number of qubits used by QPU to make embedding of the requested problem from every nested factorization subroutine step, 4. total working time of QPU in milliseconds. Table 4 shows that for 26-bit semiprime, 772 qubits of quantum resources have been used. Despite the total working qubits numbers being much greater, attempts to solve bigger problems have failed because of chain lengths (the number of physical qubits required to represent one logical qubit).

4.2 Results for discrete logarithm problem over prime field

The second application of quantum annealing smoothness checking was the discrete logarithm problem over a prime field. In this case, we have used the basic index calculus method (see subsection 2.2). As in the case of IFP, modified factoring by annealing procedure was used as a subroutine in the sieving stage from the index calculus method.

The whole method was performed on the classical computer many times, and, unlike in the case of the integer factorization, all randomly picked candidates to be B-smooth were saved from each trial. We established the maximum number of trials: 150, and we have chosen the shortest one from these tries and have realized them on the quantum computer, in some cases, several times to succeed.

Each problem has been solved in the whole multiplication subgroup of the prime field (it means $\langle g \rangle = \mathbb{F}_p^*$), and each of p is cryptographic prime ($p - 1 = 2q, q \in \mathbb{P}$).

The final results were divided into two tables also. In Table 5, the problems are characterized, while in Table 6 are listed similar to previous problem values, presenting performing of quantum computation for every problem.

p	problem size [bit]	$p-1$	$g^x = y$	sieved numbers size [bit]	base
8543	14	2×4271	$186^x = 7986$	4-14	$[2, 3, 5, 7, 11]$
23399	15	2×11699	$17856^x = 2525$	4-15	$[2, 3, 5, 7, 11]$
33623	16	2×16811	$25065^x = 25932$	4-15	$[2, 3, 5, 7]$
79943	17	2×39971	$16657^x = 9503$	4-17	$[2, 3, 5, 7, 11, 13]$
147047	18	2×73523	$8962^x = 38492$	3-18	$[2, 3, 5, 7, 11, 13]$

Table 5: Problems chosen for experiments with solving discrete logarithm problem over a prime field by hybrid index calculus.

Columns of table Table 5 contain the following values, respectively, from left: 1. characteristic p of prime field $\mathbb{F}_p$, 2. the bit length of characteristic p, 3. factorization of multiplicative subgroup order $p - 1$, 4. the value of discrete logarithm, 5. ranges

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

of bit length of numbers, from every nested factorization subroutine step, sieved in smoothness checking stage, 6. the base used in sieving.

These values mean that with currently available quantum annealing resources, we can solve DLP up to an 18-bit prime field by the procedure of quantumly splitting many 18-bit numbers, applied in a nested way in the index calculus method.

problem size [bit]	number of logical qubits	number of physical qubits	QPU total time [ms]
14	10-54	12-203	903
15	10-58	14-229	1427
16	6-48	8-172	2292
17	10-68	12-273	6297
18	9-71	11-292	17331

Table 6: Results of experiments with solving discrete logarithm problem by hybrid quadratic sieve: quantum part.

Columns of table Table 6 contain the following values, respectively, from left: 1. the bit length of input number n denoting the same problem from Table 5, 2. number of variables from the QUBO problem, from every nested factorization subroutine step, 3. number of qubits used by QPU to make embedding of the requested problem from every nested factorization subroutine step, 4. total working time of QPU in milliseconds.

Table 6 shows that for 18-bit DLP, 292 qubits of quantum resources were used. It is much less than in the case of IFP because of the size of the sieving numbers. Similarly, the length of the chains did not allow to solve a bigger problem.

5 Summary

This paper aimed to show how one can apply classical integer factorization methods and discrete logarithm problem computation using quantum annealing. Both for quadratic sieve and index calculus methods, searching for B-smooth numbers has been performed using quantum annealing and modified factorization method presented by [5] [8], applied as a subroutine. The second part of both algorithms, the linear algebra step, has been computed using classical methods. It is worth noting that the linear algebra step could also be implemented using quantum annealing, especially in the case of quadratic sieve, where we operate on the matrix defined over $\mathbb{F}_2$.

Using our method, we factorized the 26-bits integer and computed the discrete logarithm problem over the 18-bits prime field. Therefore we broke the current records in factorization using quantum annealing by 6 bits and in discrete logarithm problem, using any quantum method, by 12 bits. One can easily estimate the number of qubits needed to run presented methods for larger-scale consideration. Our modified factorization subroutine uses the same number $(\frac{\log^2 n}{4})$ of qubits, as in [8]. Thus, sieving numbers $\sim \sqrt{n}$ in IFP and $\sim p$ in DLP, we need, respectively, $\frac{\log^2 n}{16}$ and $\frac{\log^2 p}{4}$ logical qubits.

Further works should include the general number field sieve algorithm applications for integer factorization and discrete logarithm problem computation. Our estimations show that it should be possible to factorize a 50-bit integer or compute a discrete logarithm problem over a 50-bit prime field in such a case.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

For this achievement, we do not need more resources. The needed memory is about 700-800 physical qubits. Moreover, the number of logical qubits needed to solve IFP and DLP by hybrid methods with GNFS is about $\frac{\sqrt[3]{\log^5 n \log\log n}}{4}$, this is also because of the size of sieved numbers: $\sim n^{\frac{1}{d}}$, where $d \sim \sqrt[3]{\frac{3\log n}{\log\log n}}$ (see [7]). For example, a 400-bit problem (both IFP and DLP) requires 1.500 logical qubits versus 40.000 logical qubits in the case of the direct method.

It is an open question if applying quadratic sieve, index calculus method, or general number field sieve may give better performance on quantum computers than on the classical one using for B-smooth numbers searching different algorithm than Shor's.

References

1. Bernstein, D.J., Biasse, J.F., Mosca, M.: A low-resource quantum factoring algorithm. Cryptology ePrint Archive, Paper 2017/352 (2017), https://eprint.iacr.org/2017/352, https://eprint.iacr.org/2017/352
2. Crandall, R., Pomerance, C.: Prime Numbers. A Computational Perspective. Springer (2005)
3. Crane, L.: Quantum computer sets new record for finding prime number factors. New Scientist (2020)
4. Das, A.: Computational number theory. CRC Press Taylor and Francis Group (2013)
5. Dattani, N.S., Bryans, N.: Quantum factorization of 56153 with only 4 qubits (2014). https://doi.org/10.48550/ARXIV.1411.6758, https://arxiv.org/abs/1411.6758
6. Grover, L.K.: A fast quantum mechanical algorithm for database search (1996). https://doi.org/10.48550/ARXIV.QUANT-PH/9605043, https://arxiv.org/abs/quant-ph/9605043
7. Jarvis, F.: Algebraic number theory. Springer (2014)
8. Jiang, S., Britt, K.A., McCaskey, A.J., Humble, T.S., Kais, S.: Quantum annealing for prime factorization. Scientific reports **8**(1), 1–9 (2018)
9. Martin-Lopez, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.Q., O'brien, J.L.: Experimental realization of shor's quantum factoring algorithm using qubit recycling. Nature photonics **6**(11), 773–776 (2012)
10. Mosca, M., Vensi Basso, J.M., Verschoor, S.R.: On speeding up factoring with quantum sat solvers (2019). https://doi.org/10.48550/ARXIV.1910.09592, https://arxiv.org/abs/1910.09592
11. Peng, W., Wang, B., Hu, F., Wang, Y., Fang, X., Chen, X., Wang, C.: Factoring larger integers with fewer qubits via quantum annealing with optimized parameters. SCIENCE CHINA Physics, Mechanics & Astronomy **62**(6), 60311 (2019)
12. Proos, J., Zalka, C.: Shor's discrete logarithm quantum algorithm for elliptic curves. arXiv preprint quant-ph/0301141 (2003)
13. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
14. Wang, B., Hu, F., Yao, H., Wang, C.: Prime factorization algorithm based on parameter optimization of ising model. Scientific reports **10**(1), 1–10 (2020)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1

15. Wroński, M.: Index calculus method for solving elliptic curve discrete logarithm problem using quantum annealing. In: International Conference on Computational Science. pp. 149–155. Springer (2021)
16. Wroński, M.: Practical solving of discrete logarithm problem over prime fields using quantum annealing. In: Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2022. pp. 93–106. Springer International Publishing, Cham (2022)
17. Yan, B., Tan, Z., Wei, S., Jiang, H., Wang, W., Wang, H., Luo, L., Duan, Q., Liu, Y., Shi, W., Fei, Y., Meng, X., Han, Y., Shan, Z., Chen, J., Zhu, X., Zhang, C., Jin, F., Li, H., Song, C., Wang, Z., Ma, Z., Wang, H., Long, G.L.: Factoring integers with sublinear resources on a superconducting quantum processor (2022)

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36030-5_1