Quantum Information and Computation, Vol. 24, No. 7&8 (2024) 0541–0564

BASE OF EXPONENT REPRESENTATION MATTERS - MORE EFFICIENT REDUCTION OF DISCRETE LOGARITHM PROBLEM AND ELLIPTIC CURVE DISCRETE LOGARITHM PROBLEM TO THE QUBO PROBLEM

MICHAŁ WROŃSKI
Department of Cryptology, NASK National Research Institute, Kolska 12 Str. Warsaw, 01-045, Poland

ŁUKASZ DZIERZKOWSKI
Faculty of Cybernetics, Military University of Technology in Warsaw, Kaliskiego 2 Str. Warsaw, 00-908, Poland

Received December 29, 2023
Revised May 10, 2024

This paper presents further improvements in the transformation of the Discrete Logarithm Problem (DLP) and Elliptic Curve Discrete Logarithm Problem (ECDLP) over prime fields to the Quadratic Unconstrained Binary Optimization (QUBO) problem. This is significant from a cryptanalysis standpoint, as QUBO problems may be solved using quantum annealers, and the fewer variables the resulting QUBO problem has, the less time is expected to obtain a solution. The main idea presented in the paper is allowing the representation of the exponent in different bases than the typically used base 2 (binary representation). It is shown that in such cases, the reduction of the discrete logarithm problem over the prime field $\mathbb{F}_p$ to the QUBO problem may be obtained using approximately $1.89n^2$ logical variables for n being the bitlength of prime p, instead of the $2n^2$ which was previously the best-known reduction method. The paper provides a practical example using the given method to solve the discrete logarithm problem over the prime field $\mathbb{F}_{47}$. Similarly, for the elliptic curve discrete logarithm problem over the prime field $\mathbb{F}_p$, allowing the representation of the exponent in different bases than typically used base two results in a lower number of required logical variables for n being the bitlength of prime p, from $3n^3$ to $\frac{6n^3}{\log_2(\frac{3}{4}n)}$ logical variables, in the case of Edwards curves.

Keywords: elliptic curve discrete logarithm problem, D-Wave, quantum annealing, cryptanalysis.

1 Introduction

Quantum computation is an area experiencing significant progress nowadays. Alongside advancements in building quantum computers, there is simultaneous progress in quantum algorithms. This includes all areas of quantum computing, particularly quantum cryptanalysis. In quantum cryptanalysis of classical asymmetric cryptography problems—such as factorization, the discrete logarithm problem (DLP), and the elliptic curve discrete logarithm problem (ECDLP)—Shor's algorithm [11] and its modifications (as in the case of ECDLP in paper [8]) were for years the fastest known algorithms. Nevertheless, there has been very little progress in this area for many years. However, more substantial advancements emerged in 2023. That year, Regev introduced an algorithm with lattice reduction post-processing that reduced the

number of gates from $\tilde{O}(n^2)$ (the original count in Shor's algorithm) to $\tilde{O}(n^{\frac{3}{2}})$ [10], albeit at the cost of increasing the number of logical qubits from $O(n)$ (as in the optimized Shor's algorithm) to $O(n^{\frac{3}{2}})$. Shortly after that, Ragavan and Vaikuntanathan [9] demonstrated a method to reduce the necessary qubits to only $\tilde{O}(n)$ while maintaining the circuit size (depth and a total number of gates) at $\tilde{O}(n^{\frac{3}{2}})$. Following these developments, another paper by Ekerå and Gärtner [6] showed how to adapt the ideas presented in [10] and [9] for discrete logarithm problem computations. These advancements are significant theoretically, but their impact on actual quantum circuit implementations remains to be seen.

Simultaneously with research in quantum cryptanalysis using general-purpose quantum computers, cryptanalysis using quantum annealing has also been analyzed. In this context, interest in applying quantum annealing to the cryptanalysis of classical asymmetric problems began with results in integer factorization [7], where the most significant progress has been made.

The application of quantum annealing to cryptography has yet to be fully explored. Several papers have discussed the application of quantum annealing to the discrete logarithm problem [12] and the elliptic curve discrete logarithm problem [13]. Additionally, there have been investigations into the use of quantum annealing for symmetric cryptography, focusing on ciphers such as AES [4, 2], Speck [3], and the Grain cipher family [14]. These studies suggest that applying quantum annealing to symmetric cryptography might offer greater potential for practical implementation.

Although algorithms for quantum cryptanalysis using quantum annealing and general-purpose quantum computers differ significantly, some high-level ideas from one area inspire the other. One such idea is the method of exponent representation. Ragavan and Vaikuntanathan [9] decreased the necessary logical qubits by using an exponent representation different from the usual binary representation. Specifically, they used Fibonacci numbers rather than powers of two in the exponent representation to avoid modular squaring and instead rely solely on modular multiplication. This idea was also an inspiration for our work. We posed the question: Is binary representation of the exponent the best method for reducing DLP over prime fields to the Quadratic Unconstrained Binary Optimization (QUBO) problem? As detailed below, our results indicate that it is not: ternary representation of the exponent is better than binary! Analytical methods were used to obtain such a result, presented in detail in the subsequent sections. Therefore, this paper introduces a new method for transforming the DLP over prime fields to the QUBO problem. This method allows for converting a discrete logarithm problem over a prime field $\mathbb{F}_p$ to the QUBO problem using approximately $1.89n^2$ logical qubits, where n is the bitlength of p, instead of the previous best result of $2n^2$ [12].

Similar analytical methods have been employed to reduce the necessary logical variables for transforming the ECDLP to the QUBO problem. In the case of Edwards curves, the optimal representation of the exponent is in base $\frac{3n}{4}$. This approach allows for converting a discrete logarithm problem over a prime field $\mathbb{F}_p$ to the QUBO problem using approximately $\frac{6n^3}{\log_2\left(\frac{3}{4}n\right)}$ logical qubits, where n is the bitlength of p, compared to the previous best result of $3n^3$ [13].

Our contribution includes:

- Presenting a more efficient method of reducing the discrete logarithm problem to the

QUBO problem, which requires approximately $1.89n^2$ instead of the previously known $2n^2$ logical qubits for such reduction as presented in [12]. While this improvement may not seem significant, it provides a considerable advantage, especially for small fields where we can compute DLP using quantum annealing.

- Introducing a more efficient method of reducing the elliptic curve discrete logarithm problem to the QUBO problem, requiring approximately $\frac{6n^3}{\log_2\left(\frac{3}{4}n\right)}$ logical qubits instead of the previously known $3n^3$, as shown in [13]. While the difference may not be significant for small values, it is substantial asymptotically.

- Providing a practical example and result of solving DLP over $\mathbb{F}_{47}$ with a generator order of 23 using the D-Wave Advantage QPU. To compute the discrete logarithm in this field, using the base three representation of the exponent, the equivalent QUBO problem requires 36 logical variables and, using the Zephyr topology, 117 physical qubits. We successfully solved this problem using the D-Wave Advantage 2 quantum computer. Conversely, using the base two representation of the exponent, the equivalent QUBO problem requires 57 logical variables and 189 physical qubits. We could not solve the problem using the D-Wave Advantage 2 quantum computer in this case. This marks progress in directly computing DLP using quantum annealing without resorting to brute force (exponential) methods. The previous best result using the direct method was computing discrete logarithm over the field $\mathbb{F}_{11}$.

- Demonstrating a practical example and result of embedding ECDLP over $\mathbb{F}_{13}$ with a generator order of 20 using the D-Wave Advantage QPU. To compute ECDLP in this field, using the base three representation of the exponent, the equivalent QUBO problem requires 218 logical variables and, using the Pegasus topology, the 2654 number of physical qubits. However, using the base two representation of the exponent, the equivalent QUBO problem requires 435 logical variables and may not be even embedded in the D-Wave Advantage QPU. Unfortunately, we were unable to solve neither of these problems using the D-Wave Advantage quantum computer.

2 Methods of Transformation of the Discrete Logarithm Problem to the QUBO Problem

The following description is based on findings in [12]. This section will present an approach to transforming the discrete logarithm problem into the Binary Quadratic Model (BQM) problem. The problem can be easily transformed into the QUBO problem by omitting the constant in the given BQM formulation.

QUBO [5] is a significant problem type with numerous real-world applications. The following optimization problem can express the QUBO model:

$$\min_{x\in\{0,1\}^n} x^T Q x, \tag{1}$$

where Q is an $N \times N$ upper-diagonal matrix of real weights, and x is a vector of binary variables. Additionally, diagonal terms $Q_{i,i}$ are linear coefficients, and the nonzero off-diagonal terms $Q_{i,j}$ are quadratic coefficients.

The QUBO problem can also be viewed as a problem of minimizing the function

$$f(x) = \sum_i Q_{i,i} x_i + \sum_{i<j} Q_{i,j} x_i x_j. \tag{2}$$

It is important to note that the QUBO problem is a special case of the BQM problem, where BQM can be defined as

$$\sum_i a_i v_i + \sum_{i<j} b_{i,j} v_i v_j + c, \tag{3}$$

with a_i and $b_{i,j}$ being real numbers and $v_i \in \{-1, +1\}$ or $\{0, 1\}$. The transformation from the QUBO problem to the BQM problem for $v_i \in \{0, 1\}$ is straightforward; we ignore the constant c appearing in the BQM formulation.

We begin the main part of this section by defining the discrete logarithm problem:

$$g^y = h, \tag{4}$$

in the multiplicative subgroup of the prime field $\mathbb{F}_p$, so $g, h \in \mathbb{F}_p^*$ and $y \in \{1, \ldots, \mathrm{Ord}(g) - 1\}$. This problem is equivalent to:

$$g^y \equiv h \mod p, \tag{5}$$

for integers $g, h \in \{1, \ldots, p-1\}$ and $y \in \{1, \ldots, Ord(g) - 1\}$.

Let m be the bitlength of $Ord(g)$. We begin by making the following transformation. Noting that y can be written using m bits and if $y = 2^{m-1}u_m + \cdots + 2u_2 + u_1$, where $u_1, \ldots, u_m$ are binary variables, then

$$g^y = g^{2^{m-1}u_m + \cdots + 2u_2 + u_1} = g^{2^{m-1}u_m} \cdots g^{2u_2} g^{u_1}. \tag{6}$$

It is worth noting that writing $y = 2^{m-1}u_m + \cdots + 2u_2 + u_1$ allows us to obtain $y > \mathrm{Ord}(g)$, but since we operate in a cyclic group, one can always get the result from $\{0, \ldots, Ord(g) - 1\}$ by computing $y \mod Ord(g)$.

Let us also note that

$$g^{2^{i-1}u_i} = \begin{cases} 1, & \text{if } u_i = 0, \\ g^{2^{i-1}}, & \text{if } u_i = 1, \end{cases} \tag{7}$$

which is equivalent to

$$x_i = g^{2^{i-1}u_i} = 1 + u_i(g^{2^{i-1}} - 1). \tag{8}$$

Now, we use the observation above to define a transformation approach for the discrete logarithm problem over prime fields to the BQM and, subsequently, the equivalent QUBO problem.

We will transform the discrete logarithm problem into the QUBO problem using a regular binary tree of maximal height for decomposition. It is possible to obtain an equivalent QUBO problem using approximately $2n^2$ logical qubits in such a case, where n is the bitlength of the characteristic of the prime field $\mathbb{F}_p$. The scheme of such a regular binary tree of maximal height for a general number m of leaves used for the decomposition of the general discrete logarithm problem to the QUBO problem is presented in Figure 1. For the discrete logarithm problem over a finite field described above, for each $i = 1, 2, ..., m$, it holds that $X_i = x_i$.

Fig. 1. The scheme of decomposition of the general discrete logarithm problem.

The group operation $\square$ is, in this case, equivalent to multiplication in the finite field $\mathbb{F}_p$. The scheme of such a regular binary tree of maximal height for general m used for decomposing the discrete logarithm problem to the QUBO problem is reiterated in Figure 1.

It is worth noting that other decomposition methods are possible, such as using a binary balanced tree. However, the expected number of logical variables for an equivalent QUBO problem in such a case is approximately $\frac{n^3}{2}$. Therefore, we do not describe this method here.

Firstly, using the problem decomposition scheme presented in Figure 1, in every step, we can create a new variable $v_i = v_{i-1} \cdot x_{i+1}$, which is equivalent to $v_i \equiv v_{i-1} \cdot x_{i+1} \mod p$. It is also easy to show that the total number of new variables v_i will be equal to $m-2$ because $x_1, \ldots, x_m$ are leaves of the binary tree with $m-1$ inner nodes, where each inner node is equivalent to some auxiliary variable. However, the root is not equivalent to any auxiliary variable, but it is equivalent to $v_{m-2} \cdot x_m \equiv h \mod p$, so the number of auxiliary variables v_i is equal to $m-2$.

Furthermore, each equation for $v_1, v_2, \ldots, v_{m-3}, v_{m-2}, v_{m-2}x_m$ must be transformed into an equation over integers:

$$\begin{cases} f_1 = (v_1 - x_1x_2) \mod p - k_1p = 0, \\ f_2 = (v_2 - v_1x_3) \mod p - k_2p = 0, \\ \vdots \\ f_{m-3} = (v_{m-3} - v_{m-4}x_{m-2}) \mod p - k_{m-3}p = 0, \\ f_{m-2} = (v_{m-2} - v_{m-3}x_{m-1}) \mod p - k_{m-2}p = 0, \\ f_{m-1} = (h - v_{m-2}x_m) \mod p - k_{m-1}p = 0. \end{cases} \tag{9}$$

Let us denote the linearized versions of polynomials $f_1, \ldots, f_{m-1}$ as $f_{Lin_1}, \ldots, f_{Lin_{m-1}}$, respectively (the linearization method will be described later in the subsection). Then, the final polynomial F in BQM form is given by

$$F_{Pen} = (f_{Lin_1})^2 + \ldots + (f_{Lin_{m-1}})^2 + Pen, \tag{10}$$

where Pen represents penalties obtained during linearization, and the minimal energy of F_{Pen} is equal to 0.

Therefore, the total number of variables is as follows:

- For $x_1, \ldots, x_m$ - m binary variables are required.
- For $v_1, \ldots, v_{m-2}$ - $(m-2)n$ binary variables are required.
- For $k_1, \ldots, k_{m-1}$ - $(m-1)(\lfloor \log_2(2n) \rfloor + 1)$ binary variables are required.
- For auxiliary variables obtained during the linearization of each polynomial $f_1, \ldots, f_{m-1}$ - n variables each, totaling $(m-1)n$ variables.

Finally, the obtained BQM (and thus equivalent QUBO) problem requires $m + (m-2)n + (m-1)(\lfloor \log_2(2n) \rfloor + 1) + (m-1)n = 2mn + 2m - 3n + (m-1)(\lfloor \log_2(2n) \rfloor + 1)$, which is

approximately equal to $2mn$ variables. Assuming $m \approx n$ (which is true if the given generator is the generator of the multiplicative subgroup of the field $\mathbb{F}_p$), then the total number of variables is approximately $2n^2$.

It is crucial to note that the BQM problem can be derived in two slightly different ways:

1. One can first linearize each equation f_i to obtain the linearized equation f_{Lin_i}, then compute the sum $F_{Pen} = \sum_{i=1}^{m-1} f_{Lin_i}^2 + Pen$, where Pen denotes penalties obtained during linearization. The polynomial F_{Pen} is in such a case in BQM form.

2. One can first compute the sum $F = \sum_{i=1}^{m-1} f_i^2$, and then perform quadratization of the polynomial F to obtain F_{Quadr}, finally deriving $F_{Pen} = F_{Quadr} + Pen$ in BQM form, where Pen denotes penalties obtained during quadratization.

In our methods of reducing the discrete logarithm problem to the BQM problem, the first method allows one to compute the maximum number of required variables in the resulting BQM problem and, thus, in the equivalent QUBO problem. Therefore, this method was employed in the examples presented in Section 5.

Having polynomials $f_1, \ldots, f_{m-1}$ in linear form $f_{Lin_1}, \ldots, f_{Lin_{m-1}}$, we should now transform each modular equation $f_{Lin_i} \equiv h \mod p$, for $i = 1, 2, \ldots, m-1$, into an equation over integers:

$$(f_{Lin_i} - h) \mod p - k_i p = 0, \tag{11}$$

where $k_i \in \mathbb{Z}$ and for every polynomial f_{Lin_i}, the operation $f_{Lin_i} \mod p$ is equivalent to reducing all of the coefficients of the polynomial f_{Lin_i} modulo p. It is important to note that k_i is bounded by the maximum number of monomials appearing in the polynomial $(f_{Lin_i} - h) \mod p$.

When transforming the discrete logarithm problem into the BQM problem, it is necessary to reduce 2-local terms while using approach 1 of obtaining the BQM problem, and 4-local and 3-local terms (for any $w \geq 2$, one can similarly reduce a $(w+1)$-local term to a w-local term) while using approach 2 of obtaining the BQM problem. We will now demonstrate how the resulting 3-local terms may be reduced to 2-local ones. Note that each penalty monomial of the form $a_i a_j a_l$ will be transformed, according to [7], in the following way:

$$a_i a_j a_l \to b_k a_l + 2(a_i a_j - 2b_k(a_i + a_j) + 3b_k). \tag{12}$$

This implies that

$$a_i a_j a_l = b_k a_l = b_k a_l + 2(a_i a_j - 2b_k(a_i + a_j) + 3b_k), \tag{13}$$

if $a_i a_j = b_k$, and

$$a_i a_j a_l < a_l b_k + 2(a_i a_j - 2b_k(a_i + a_j) + 3b_k), \tag{14}$$

if $a_i a_j \neq b_k$.

As a result, the term $a_i a_j a_l$ can be transformed to quadratic form by replacing $a_i a_j$ with b_k plus a constraint, given by a penalty term:

$$\min(a_i a_j a_l) = \min\left(a_l b_k + 2(a_i a_j - 2b_k(a_i + a_j) + 3b_k)\right). \tag{15}$$

3 Which Tree Shape Is the Best? An Analysis

We considered reducing the DLP problem to the QUBO problem in the previous section. To make such a transformation, in each step, a single leaf was added (multiplied) to the inner node, thus forming a binary tree of maximal height. This section will consider an alternative approach to binary tree construction for efficiently transforming DLP to the QUBO problem.

3.1 Grouping Leaves Method

We pose the question: Is it possible to group leaves and multiply them directly? Let us denote x_i as in Equation (20). We will refer to the i-th leaf as

$$X_i = x_{(i-1)k+1} \dots x_{ik}, \tag{16}$$

where $x_i = g^{2^{i-1}u_i} = 1 + u_i(g^{2^{i-1}} - 1)$. X_i as a product of k leaves results in a polynomial of degree k, consisting of 2^k monomials. There are also $2^k - k - 1$ additional variables necessary for the linearization of this product (similar to the brutal approach given in [12]). After linearization, this product will consist of $2^k - 1$ monomials of degree 1. This is crucial when X_i will be multiplied with an inner node v_{i-1}, which consists of n monomials of degree 1.

After multiplying v_{i-1} and X_i, there will be necessary $(2^k - 1)n$ additional variables for linearization. However, it is also noteworthy that the number of inner nodes decreases and will be approximately $m \approx \frac{n}{k}$. The task here is to find k for which the total number of variables will be the smallest.

Let us estimate the number of variables. For each group of leaves, there are $2^k - 1$ variables necessary: k variables to define each single leaf and $2^k - k - 1$ additional variables for the linearization of the product of the group of leaves.

There are also n additional variables necessary to represent each inner node v_i, and there are approximately $\frac{n}{k}$ such nodes, so it gives $\frac{n^2}{k}$. Finally, the linearization of a single product of v_{i-1} and X_i requires $(2^k - 1)n$ variables, but as there are approximately $\frac{n}{k}$ nodes, the total number of additional variables in this context may be estimated as $\frac{n^2}{k}(2^k - 1)$.

Other terms that influence the total number of additional variables have a less significant impact because they will depend at most linearly on n. Therefore, the most significant part of the total number of variables is

$$f(n,k) = \frac{n^2}{k} + (2^k - 1)\frac{n^2}{k} = 2^k \frac{n^2}{k}. \tag{17}$$

Because we aim to find the optimal value of k for which the function $f(n,k) = \frac{n^2}{k}2^k$ is minimized, we can treat this as a function of only the variable k.

However, as we are searching for k as an integer, for analysis, we expand the domain of the function f to real positive numbers. In such a case, one can find the global minimum of this function, which occurs at $k = \frac{1}{\ln 2}$ and is equal to $e \ln 2$. As the function does not have any other minima in the desired domain, it implies that the integer k for which the function f obtains its minimum is either $k = 1$ or $k = 2$. Direct evaluation reveals that for both of these values, the function f at these points is the same and equals $f(n,1) = f(n,2) = 2n^2$. This means that although there might be (and in reality, there is) some difference in the number of variables in both cases because here, linear and degree zero monomials were not taken into account, the total number of variables in both cases is asymptotically equal.

From an analytical perspective, the minimal amount of variables that can be obtained using the presented method (there may be other, more efficient methods unknown to us) should not be smaller than $e \ln 2n^2 \approx 1.88n^2$. The question arises: Is there any method to obtain a smaller asymptotic number of variables?

3.2 Representation of the Exponent in Different Number Bases

Let m be the bitlength of $Ord(g)$. We begin by making the following transformation. Note that y can be written using m digits in a base d system, and if $y = d^{m-1}u_m + \cdots + du_2 + u_1$, where $u_1, \ldots, u_m$ are variables from $\{0, \ldots, d-1\}$, then

$$g^y = g^{d^{m-1}u_m} \cdots g^{du_2} g^{u_1}. \tag{18}$$

It is important to note that writing $y = d^{m-1}u_m + \cdots + du_2 + u_1$ allows us to obtain $y > \mathrm{Ord}(g)$, but because we operate in a cyclic group, one can always get the result from $\{0, \ldots, Ord(g)-1\}$ by computing $y \mod Ord(g)$.

Let us also observe that

$$g^{d^{i-1}u_i} = \begin{cases} 1, & \text{if } u_i = 0, \\ g^{d^{i-1}}, & \text{if } u_i = 1, \\ g^{2d^{i-1}}, & \text{if } u_i = 2, \\ \vdots & \\ g^{(d-1)d^{i-1}}, & \text{if } u_i = d-1. \end{cases} \tag{19}$$

The challenge here is that u_i are not binary variables, making the transformation of $g^{d^{i-1}u_i}$ more complex than previously. However, it can be done using $d-1$ binary variables $\mu_{i1}, \ldots, \mu_{i(d-1)}$ as follows:

$$g^{d^{i-1}u_i} = 1 + \mu_{i_1}(g^{d^{i-1}} - 1) + \ldots + \mu_{i_{d-1}}(g^{(d-1)d^{i-1}} - 1). \tag{20}$$

Note that if all binary variables $\mu_{i_1}, \ldots, \mu_{i_{d-1}}$ are equal to 0, then $g^{d^{i-1}u_i} = 1$, and if only μ_{i_l} is equal to 1 and the other binary variables $\mu_{i_1}, \ldots, \mu_{i_{l-1}}, \mu_{i_{l+1}}, \ldots, \mu_{i_{d-1}}$ are equal to 0, then

$$1 + \mu_{i_l}(g^{ld^{i-1}} - 1) = g^{ld^{i-1}}, \tag{21}$$

as intended.

Let us denote

$$X_i = 1 + \mu_{i_1}(g^{d^{i-1}} - 1) + \ldots + \mu_{i_{d-1}}(g^{(d-1)d^{i-1}} - 1), \tag{22}$$

similar to what was done previously.

One might consider that if we operate in a base d system and there are only d different values that $g^{d^{i-1}u_i}$ can take, then perhaps $\lfloor \log_2(d-1) \rfloor + 1$ binary variables should suffice, as any integer number from the interval $\{0, \ldots, d-1\}$ can be represented in binary system using $\lfloor \log_2(d-1) \rfloor + 1$ bits. Unfortunately, we cannot use such a small number of variables in this case. The reason is that the values $g^{d^{i-1}u_i}$ take are not generally consecutive but are distributed uniformly in the $\mathbb{F}_p$ field. Therefore, one has to use $d-1$ binary variables in this context.

Now, let us estimate the total number of binary variables necessary to transform the discrete logarithm problem into the QUBO problem.

As before, we use estimations, not exact computations, to simplify the analysis. The binary tree will have approximately $m = \lceil \frac{n-1}{\log_2 d} \rceil \approx \frac{n}{\log_2 d}$ inner nodes.

Each leaf is a polynomial of degree one, consisting of d terms, of which $d-1$ are degree one monomials and one is a degree 0 monomial. To obtain the product v_i of an inner node v_{i-1} and a leaf x_i, there will be necessary $(d-1)n$ additional variables for linearization. However, note that in this case, the number of inner nodes also decreases and will be approximately $\frac{n}{\log_2 d}$. So, the task here is to find a base d for which the total number of variables will be the smallest.

As before, let us estimate the number of variables. For each leaf, $d-1$ variables are necessary to define each leaf.

There are also n additional variables required for each inner node v_i, and as there are approximately $\frac{n}{\log_2 d}$ such nodes, this gives $\frac{n^2}{\log_2 d}$ additional variables. Finally, the linearization of a single product of v_{i-1} and X_i requires $(d-1)n$ variables, but as there are approximately $m \approx \frac{n}{\log_2 d}$ nodes, the total number of additional variables in this context is $\frac{n^2(d-1)}{\log_2 d}$. Other terms that influence the total number of additional variables have less significant impact because they will depend at most linearly on n. Therefore, the most significant part of the total number of variables is

$$f(n,d) = \frac{n^2}{\log_2 d} + (d-1)\frac{n^2}{\log_2 d} = d\frac{n^2}{\log_2 d}. \tag{23}$$

Since we aim to find the optimal value of d for which the function $f(n,d) = \frac{n^2 d}{\log_2 d}$ reaches a minimum, we can treat this as a function of only the variable d.

The function f in this subsection is conceptually similar to the previous one.

As before, although we are searching for d as an integer, for analysis, we expand the domain of function f to the real positive numbers. In such a case, one can find the global minimum of the function, which occurs at $d = e$ and is equal to $e \ln 2$, the same value as obtained in the previous section. Does this make the analysis worthwhile?

Yes, it does because it allows for a case not permissible in the previous scenario.

Since e is not an integer, we examine the nearest integers to determine for which of them function f will take the smallest value. Taking $d = 2$, we obtain the binary representation of the exponent and therefore $f(n,2) = 2n^2$, as before. So, the next integer to check is $d = 3$. In such a case, $f(n,3) = \frac{3n^2}{\log_2 3} \approx 1.89n^2$, which is smaller than $2n^2$ and very close to the analytically obtained minimal value $e \ln 2 \approx 1.88$ for $d = e$.

To conclude the discussion above, the total number of variables for arbitrarily chosen d and m may be estimated as:

- For $X_1, \ldots, X_m$ - $d-1$ binary variables are required for each, totaling $(d-1)m$ variables.
- For $v_1, \ldots, v_{m-2}$ - a total of $(m-2)n$ binary variables are required.
- For $k_1, \ldots, k_{m-1}$ - up to $\lfloor \log_2 (dn) \rfloor + 1$ binary variables are required for each, totaling at most $(m-1)(\lfloor \log_2 (dn) \rfloor + 1)$ binary variables.

- For auxiliary variables obtained during the linearization of each polynomial $f_1, \ldots, f_{m-1}$, $(d-1)n$ variables are required, totaling $(m-1)(d-1)n$ variables.

The resulting BQM (and thus equivalent QUBO) problem requires $(d-1)m + (m-2)n + (m-1)\left(\lfloor \log_2(dn) \rfloor + 1\right) + (m-1)(d-1)n = dmn + dm - 1 - (d+1)n + (m-1)\left(\lfloor \log_2(dn) \rfloor\right)$. Taking $d = 3$, one obtains that the total number of variables is limited by $\frac{3n^2}{\log_2 3} \approx 1.89n^2$.

Note that the scenario analyzed in the previous section is essentially the same as here. However, in the first scenario, only systems with base 2^k were considered, excluding others (grouping k leaves was equivalent to using a number system with base 2^k).

3.3 Remark on Integer Representation Using Fibonacci Numbers

As presented in [9], representing the exponent using Fibonacci numbers offers a significant improvement in the context of Regev's factorization method. However, for the case presented in this paper, there are more efficient methods than this method of integer representation. It is well-known that each integer l can be represented as $\sum_{i=0}^{j} u_i F_i$, where $u_j \in \{0, 1\}$ and F_j is the j-th Fibonacci number with $F_i < l \leq F_{i+1}$. Given that the asymptotic behavior of Fibonacci numbers can be estimated as $F_j \sim \frac{\phi^j}{\sqrt{5}}$, it can be estimated that $j \sim \frac{\log_\phi l}{\log_\phi 2}$, representing the number of bits necessary to represent the integer l. Unfortunately, it is straightforward to find that even the binary representation of integer l is more efficient because it requires only $\lfloor \log_2 l \rfloor + 1$ bits, which is generally less than $\log_\phi l$.

While it is well-known that base e representation is theoretically the most efficient in the analyzed case, the requirement for operations in finite fields prevents using non-integer bases. Alternatively, other methods of integer representation using Fibonacci-like sequences can be analyzed. However, even in these cases, the ternary representation of integers proves to be more efficient.

4 Methods of Transformation of the Elliptic Curve Discrete Logarithm Problem to the QUBO Problem

The discrete logarithm problem is associated not only with multiplicative subgroups of prime fields but also with subgroups of points on elliptic curves. Given the significance of elliptic curve cryptography, analyzing the impact of quantum computing on current algorithms is important. One possible approach was presented in [13].

The QUBO problem was described in Section 2; therefore, this section will begin by describing the ECDLP.

Let E be an elliptic curve over the prime field $\mathbb{F}_p$, and let P, Q be points on this curve. The elliptic curve discrete logarithm problem (ECDLP) is defined as finding an integer $y \in \{1, \ldots, Ord(P) - 1\}$ such that for the two given points P and Q, the following property holds:

$$[y]P = \underbrace{P + P + \cdots + P}_{\text{y addends}} = Q. \tag{24}$$

Number y can be represented as $y = 2^{m-1}u_m + \cdots + 2u_2 + u_1$, where m is the bitlength of $Ord(P)$ and u_i are binary variables. It follows that:

$$Q = [y]P = [2^{m-1}u_m + \cdots + 2u_2 + u_1]P = [2^{m-1}u_m]P + \cdots + [2u_2]P + [u_1]P = P_m + \cdots + P_2 + P_1. \tag{25}$$

Every summand from the above equation may be presented as:

$$[u_i]([2^{i-1}]P) = \begin{cases} \mathcal{O}, & \text{if } u_i = 0, \\ [2^{i-1}]P, & \text{if } u_i = 1, \end{cases} \tag{26}$$

which can be transformed to:

$$[u_i]([2^{i-1}]P) = \mathcal{O} + u_i \left([2^{i-1}]P - \mathcal{O}\right). \tag{27}$$

Therefore, every point $P_i = [u_i]([2^{i-1}]P)$ can be represented in affine coordinates as:

$$\begin{cases} P_{i,x} = \mathcal{O}_x + u_i \left([2^{i-1}]P_x - \mathcal{O}_x\right), \\ P_{i,y} = \mathcal{O}_y + u_i \left([2^{i-1}]P_y - \mathcal{O}_y\right). \end{cases} \tag{28}$$

Solving the problem given by Equation (25) is feasible with decomposition into a regular binary tree, as shown in Fig. 1. The differences are:

- Nodes X_i should be substituted with elliptic curve points P_i.
- Operations $\square$ should be replaced with elliptic curve point additions.

It is important to note that if E is an elliptic curve with complete arithmetic and if every point, being a multiple of the point P, can be presented in affine coordinates, then for any three points $Q = (Q_x, Q_y)$, $P_i = (P_{i,x}, P_{i,y})$, $P_j = (P_{j,x}, P_{j,y})$ from the curve E, where $Q = P_i + P_j$, it holds that:

$$\begin{cases} Q_x = \frac{\phi(P_i,P_j)}{\psi(P_i,P_j)} = \frac{\phi(P_{i,x},P_{i,y},P_{j,x},P_{j,y})}{\psi(P_{i,x},P_{i,y},P_{j,x},P_{j,y})}, \\ Q_y = \frac{\xi(P_i,P_j)}{\psi(P_i,P_j)} = \frac{\xi(P_{i,x},P_{i,y},P_{j,x},P_{j,y})}{\psi(P_{i,x},P_{i,y},P_{j,x},P_{j,y})}, \end{cases} \tag{29}$$

where ϕ, ψ, and ξ are polynomials.

4.1 Edwards Curves

In this subsection, Edwards curves, an elliptic curve model described in [1], will be discussed.

Definition 1 *An Edwards curve E_{Ed} over a field $\mathbb{F}_p$ is given by the equation*

$$E_{Ed}/\mathbb{F}_p : x^2 + y^2 = 1 + dx^2y^2, \tag{30}$$

where the coefficient $d \notin \{0, 1\}$.

The addition of points $P, Q \in E_{Ed}(\mathbb{F}_p)$ on this model is given by:

$$P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3) = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2}\right). \tag{31}$$

If d is not a square in the field $\mathbb{F}_p$, then the above formula is complete.

The neutral element is $\mathcal{O} = (0, 1)$, so for Edwards curves, the formulas corresponding to System (28) are

$$\begin{cases} P_{i,x} = u_i \left([2^{i-1}]P_x\right), \\ P_{i,y} = 1 + u_i \left([2^{i-1}]P_y - 1\right). \end{cases} \tag{32}$$

Consider a case where the Edwards curve is defined over a prime field $\mathbb{F}_p$ and let:

- n be the bitlength of the field characteristic p,
- m be the bitlength of the order of $\langle P \rangle$ - a group generated by P.

Then, using formulas for point addition (31) and the idea to convert ECDLP into the QUBO with a system of equations (15) from [13], we obtain a system of equations (19) in [13], which allows transforming ECDLP on Edwards curves into the QUBO problem. However, for small finite fields, and thus a small number of required binary variables, it may be unnecessary to decompose the two equations for x_3, y_3 from Eq. (31) into nine equations from System (33) for every point addition.

$$\begin{cases}
f_{1,1} = (A_1 - P_{1,y}P_{2,y}) \bmod p - k_{1,1}p = 0, \\
f_{1,2} = (B_1 - P_{1,x}P_{2,x}) \bmod p - k_{1,2}p = 0, \\
f_{1,3} = (C_1 - P_{1,x}P_{2,y}) \bmod p - k_{1,3}p = 0, \\
f_{1,4} = (D_1 - P_{1,y}P_{2,x}) \bmod p - k_{1,4}p = 0, \\
f_{1,5} = (E_1 - dC_1D_1) \bmod p - k_{1,5}p = 0, \\
f_{1,6} = (F_1 - R_{1,x}E_1) \bmod p - k_{1,6}p = 0, \\
f_{1,7} = (G_1 - R_{1,y}E_1) \bmod p - k_{1,7}p = 0, \\
f_{1,8} = (C_1 + D_1 - R_{1,x} - F_1) \bmod p - k_{1,8}p = 0, \\
f_{1,9} = (A_1 - B_1 - R_{1,y} + G_1) \bmod p - k_{1,9}p = 0, \\
\dots \\
f_{i,1} = (A_i - P_{i+1,y}R_{i-1,y}) \bmod p - k_{i,1}p = 0, \\
f_{i,2} = (B_i - P_{i+1,x}R_{i-1,x}) \bmod p - k_{i,2}p = 0, \\
f_{i,3} = (C_i - P_{i+1,x}R_{i-1,y}) \bmod p - k_{i,3}p = 0, \\
f_{i,4} = (D_i - P_{i+1,y}R_{i-1,x}) \bmod p - k_{i,4}p = 0, \\
f_{i,5} = (E_i - C_iD_i) \bmod p - k_{i,5}p = 0, \\
f_{i,6} = (F_i - R_{i,x}E_i) \bmod p - k_{i,6}p = 0, \\
f_{i,7} = (G_i - R_{i,y}E_i) \bmod p - k_{i,7}p = 0, \\
f_{i,8} = (C_i + D_i - R_{i,x} - F_i) \bmod p - k_{i,8}p = 0, \\
f_{i,9} = (A_i - B_i - R_{i,y} + G_i) \bmod p - k_{i,9}p = 0, \\
\dots \\
f_{m-2,1} = (A_{m-2} - P_{m,y}R_{m-2,y}) \bmod p - k_{m-2,1}p = 0, \\
f_{m-2,2} = (B_{m-2} - P_{m,x}R_{m-2,x}) \bmod p - k_{m-2,2}p = 0, \\
f_{m-2,3} = (C_{m-2} - P_{m,x}R_{m-2,y}) \bmod p - k_{m-2,3}p = 0, \\
f_{m-2,4} = (D_{m-2} - P_{m,y}R_{m-2,x}) \bmod p - k_{m-2,4}p = 0, \\
f_{m-2,5} = (E_{m-2} - C_{m-2}D_{m-2}) \bmod p - k_{m-2,5}p = 0, \\
f_{m-2,6} = (F_{m-2} - Q_xE_{m-2}) \bmod p - k_{m-2,6}p = 0, \\
f_{m-2,7} = (G_{m-2} - Q_yE_{m-2}) \bmod p - k_{m-2,7}p = 0, \\
f_{m-2,8} = (C_{m-2} + D_{m-2} - Q_x - F_{m-2}) \bmod p - k_{m-2,8}p = 0, \\
f_{m-2,9} = (A_{m-2} - B_{m-2} - Q_y + G_{m-2}) \bmod p - k_{m-2,9}p = 0.
\end{cases} \tag{33}$$

4.2 A Case with Edwards Curves and Bases of Exponent Representation

In [13], the number of variables necessary for transforming the ECDLP to the QUBO problem for Edwards curves and binary representation of the point decomposition was calculated to be approximately $3n^3$, with n being the bitlength of the field characteristic p. Now, we can consider options with other number systems.

Assuming that we use a number system with base d and $m = \lfloor \log_d \mathrm{Ord}(P) \rfloor + 1$, the variable y can be expressed as $y = d^{m-1}u_m + \dots + du_2 + u_1$, where $u_m, \dots, u_1 \in \{0, \dots, d-1\}$. We can adapt Equation (25) to the case with the new base:

$$\begin{aligned} Q = [y]P &= [d^{m-1}u_m + \dots + du_2 + u_1]P = [d^{m-1}u_m]P + \dots + [du_2]P + [u_1]P \\ &= [u_m]([d^{m-1}]P) + \dots + [u_2]([d]P) + [u_1]P = P_m + \dots + P_2 + P_1. \end{aligned} \tag{34}$$

Each term from the above equation can be expressed as:

$$[u_i]([d^{i-1}]P) = \begin{cases} \mathcal{O}, & u_i = 0, \\ [d^{i-1}]P, & u_i = 1, \\ [2 \cdot d^{i-1}]P, & u_i = 2, \\ \quad \vdots & \quad \vdots \\ [(d-1) \cdot d^{i-1}]P, & u_i = d-1, \end{cases} \tag{35}$$

which can be converted into:

$$[u_i]([d^{i-1}]P) = \mathcal{O} + \mu_{i_1}([d^{i-1}]P - \mathcal{O}) + \mu_{i_2}([2d^{i-1}]P - \mathcal{O}) + \ldots + \mu_{i_{d-1}}([(d-1)d^{i-1}]P - \mathcal{O}). \tag{36}$$

Therefore, every point $P_i = [u_i]([d^{i-1}]P)$ can be represented in affine coordinates as:

$$\begin{cases} P_{i,x} = \mathcal{O}_x + \sum\limits_{j=1}^{d-1} \mu_{i_j} \left([j \cdot d^{i-1}]P_x - \mathcal{O}_x \right), \\ P_{i,y} = \mathcal{O}_y + \sum\limits_{j=1}^{d-1} \mu_{i_j} \left([j \cdot d^{i-1}]P_y - \mathcal{O}_y \right), \end{cases} \tag{37}$$

which, for Edwards curves where $\mathcal{O} = (0, 1)$, simplifies to:

$$\begin{cases} P_{i,x} = \sum\limits_{j=1}^{d-1} \mu_{i_j} \left([j \cdot d^{i-1}]P \right)_x, \\ P_{i,y} = 1 + \sum\limits_{j=1}^{d-1} \mu_{i_j} \left(\left([j \cdot d^{i-1}]P \right)_y - 1 \right). \end{cases} \tag{38}$$

We can now calculate the number of necessary variables for conducting computations from Equation (33) for Edwards curves, as in [13], but now using a base d system.

Equations from System (38) show a representation of a point P_i's coordinates. The first, $P_{i,x}$, can be written as a sum of $d-1$ monomials of degree 1. The second, $P_{i,y}$, is represented as a sum of a monomial of degree 0 and $d-1$ monomials of degree 1. Coordinates $R_{i,x}$ and $R_{i,y}$ of the sum R_i, along with auxiliary variables A_i, B_i, etc., must belong to $\mathbb{F}_p$, meaning n binary variables are necessary to represent each of them. As every equation from System (33) equals 0, the value of a product $k_{i,j}p$ must be the same as the value of the polynomial, whose coefficients are reduced modulo p. Thus, the product $k_{i,j}p$ must be less than or equal to the maximum value of this polynomial. From this, the bitlength of $k_{i,j}$ can be calculated. Linearization is necessary as equations from System (33) involve multiplications. An additional variable must be introduced for every monomial of degree greater than 1.

With the above information, we can calculate the variables required to transform the ECDLP on Edwards curves to the QUBO problem using a number system with base d to represent the desired multiplicity of a generator P.

Starting with equation $f_{i,1}$, for variable A_i we need n bits. As a result of multiplying $P_{i+1,y}$ and $R_{i-1,y}$, dn monomials will appear, n of degree 1 and $(d-1)n$ of degree 2. The maximum value of the polynomial occurs with the sum of 1 variable A_i from the set $\{0, \ldots, p-1\}$ and dn monomials with coefficients from the set $\{0, \ldots, p-1\}$, so at most it can be $(dn+1)(p-1)$. This will impact the bitlength of $k_{i,1}$, because $k_{i,1}p \leq (dn+1)(p-1)$. We derive that $k_{i,1} \leq dn$ and the bitlength of $k_{i,1}$ equals $\lfloor \log_2(dn) \rfloor + 1$ at most. Summing up, equation $f_{i,1}$ requires:

- n boolean variables for variable A_i,
- $(d-1)n$ variables for the linearization of $(d-1)n$ monomials,
- $\lfloor \log_2(dn) \rfloor + 1$ boolean variables for $k_{i,1}$,

giving a total of $dn + \lfloor \log_2(dn) \rfloor + 1$ additional variables.

Continuing with the equation $f_{i,2}$, for variable B_i we need n bits. As a result of the multiplication of $P_{i+1,x}$ and $R_{i-1,x}$, $(d-1)n$ monomials of degree 2 will appear. The polynomial may achieve the maximum value with the sum of 1 variable B_i from the set $\{0, \ldots, p-1\}$ and $(d-1)n$ monomials with coefficients from the set $\{0, \ldots, p-1\}$, so at most it can be equal to $((d-1)n+1)(p-1)$. Analogously to the previous case, we can calculate the maximum bitlength of $k_{i,2}$, which equals $\lfloor \log_2((d-1)n) \rfloor + 1$ at most. To summarize, the equation $f_{i,2}$ requires:

- n boolean variables for variable B_i,
- $(d-1)n$ variables for the linearization of $(d-1)n$ monomials of degree 2,
- $\lfloor \log_2((d-1)n) \rfloor + 1$ boolean variables for $k_{i,2}$,

which gives a total of $dn + \lfloor \log_2((d-1)n) \rfloor + 1$ additional variables.

The next part is the equation $f_{i,3}$. Because the forms of $R_{i-1,x}$ and $R_{i-1,y}$ are similar, the case of $f_{i,3}$ is the same as $f_{i,2}$. That means it requires $dn + \lfloor \log_2((d-1)n) \rfloor + 1$ additional variables.

The subsequent piece is the equation $f_{i,4}$. Because the forms of $R_{i-1,x}$ and $R_{i-1,y}$ are similar, the case of $f_{i,4}$ is the same as $f_{i,1}$, therefore it requires $dn + \lfloor \log_2(dn) \rfloor + 1$ additional variables.

The continuation will be an analysis of the equation $f_{i,5}$. For variable E_i, we need n bits. As a result of the multiplication of C_i and D_i (both consisting of n boolean variables), n^2 monomials of degree 2 will appear. The polynomial may achieve the maximum value with the sum of 1 variable E_i from the set $\{0, \ldots, p-1\}$ and n^2 monomials with coefficients from the set $\{0, \ldots, p-1\}$, so it can be equal to $(n^2+1)(p-1)$. Analogously to the previous cases, we can calculate the maximum bitlength of $k_{i,5}$, which equals $\lfloor \log_2(n^2) \rfloor + 1$ at most. The constant d does not affect the number of monomials nor the maximum value of the polynomial (as every coefficient is reduced *mod* p), so it can be omitted. To summarize, equation $f_{i,5}$ requires:

- n boolean variables for variable E_i,
- n^2 variables for the linearization of n^2 monomials of degree 2,
- $\lfloor \log_2(n^2) \rfloor + 1$ boolean variables for $k_{i,5}$,

which gives a total of $n^2 + n + \lfloor \log_2(n^2) \rfloor + 1$ additional variables.

The situation for the equation $f_{i,6}$ is similar to the previous cases. The only difference is that we need additional n binary variables to represent $R_{i,x}$. That means it requires $n^2 + 2n + \lfloor \log_2(n^2) \rfloor + 1$ extra variables.

As the forms of $R_{i,x}$ and $R_{i,y}$ are similar, the case of the equation $f_{i,7}$ is the same as $f_{i,6}$ and requires $n^2 + 2n + \lfloor \log_2(n^2) \rfloor + 1$ additional variables.

In the case of equation $f_{i,8}$, we do not have any new variables, so we only need to count the bitlength of $k_{i,8}$. The result of $(-R_{i,x} - F_i) \mod p$ will consist of $2n$ monomials of degree 1. The maximum value of the polynomial occurs with the sum of 2 variables C_i and D_i from the set $\{0, \ldots, p-1\}$ and $2n$ monomials with coefficients from the set $\{0, \ldots, p-1\}$, so it can be equal to $(2n+2)(p-1)$. As with previous cases, we can calculate the maximum bitlength of $k_{i,8}$, which equals $\lfloor \log_2(2n+1) \rfloor + 1$ at most.

The last equation, $f_{i,9}$, has a similar form to $f_{i,8}$, requiring $\lfloor \log_2(2n+1) \rfloor + 1$ additional variables.

To sum up, for one of the systems of equations, there are

$$\begin{aligned} &2 \cdot (dn + \lfloor \log_2(dn) \rfloor + 1) + 2 \cdot (dn + \lfloor \log_2((d-1)n) \rfloor + 1) + (n^2 + n + \lfloor \log_2(n^2) \rfloor + 1) \\ &+ 2 \cdot (n^2 + 2n + \lfloor \log_2(n^2) \rfloor + 1) + 2 \cdot (\lfloor \log_2(2n+1) \rfloor + 1) \\ &\approx 3n^2 + (4d+5)n + 4\log_2 d + 12\log_2 n + 11 = 3n^2 + (4d+5)n + O(\log_2 n) \end{aligned}$$

necessary logical variables.

Eventually, the number of necessary boolean variables for the whole System (33) can be calculated. As mentioned, using a number system with base d instead of 2 reduces the number of nodes in the binary tree for $m \approx n$, from m to approximately $\frac{m}{\log_2 d} \approx \frac{n}{\log_2 d}$. This makes it $\frac{n}{\log_2 d}(3n^2 + (4d+5)n + O(\log_2 n))$. The final number of required boolean variables is approximately $\frac{3n^3 + (4d+5)n^2}{\log_2 d}$.

Our goal is to find such d that the function $f(n,d) = \frac{3n^3+(4d+5)n^2}{\log_2 d} = \frac{3n^3}{\log_2 d} + \frac{(4d+5)n^2}{\log_2 d}$ reaches a minimum. Given that for increasing variable d, the coefficient at n^3 decreases and the coefficient at n^2 increases, the function f has a minimum for both terms being equal. This leads to solving the equation:

$$\frac{3n^3}{\log_2 d} = \frac{(4d+5)n^2}{\log_2 d}, \tag{39}$$

and after omitting the constant 5, which is negligible in asymptotic comparison with term $4d$, it gives us $d \approx \frac{3n}{4}$. Finally, after substituting the calculated value into the formula for counting the designated number of variables $f(n,d)$, we obtain:

$$f\left(n, \frac{3n}{4}\right) = \frac{3n^3}{\log_2\left(\frac{3n}{4}\right)} + \frac{3n^2 n}{\log_2\left(\frac{3n}{4}\right)} = \frac{6n^3}{\log_2\left(\frac{3n}{4}\right)}, \tag{40}$$

which means that $f(n, \frac{3n}{4}) = O\left(\frac{n^3}{\log_2 n}\right)$, thus $f(n,d) = O(n^3)$ as well. This result is asymptotically better than the $3n^3$ obtained for a binary number system.

5 Experiments

Our goal was to solve the discrete logarithm problem (DLP) and the elliptic curve discrete logarithm problem (ECDLP) over prime fields using the D-Wave Advantage Quantum Processing Unit (QPU).

In the case of the DLP, our choice of the field was determined by selecting the largest value d that requires three trits (3 digits in base three representation) and for which $p = 2d + 1$ is a prime number. This method yielded our experiment's prime field $\mathbb{F}_p$. We also chose generators with an order equal to d. The most notable result of solving the discrete logarithm problem over the prime field using the D-Wave Advantage QPU was solving the problem over the 6-bit prime field $\mathbb{F}_{47}$ using the new method.

For the ECDLP, our field choice was started with finding the largest prime value p that requires 4 bits in binary representation, resulting in the prime field $\mathbb{F}_p$ for our experiment. Given that the characteristic p of the field $\mathbb{F}_{13}$ is 4 bits long, we compared the binary point's multiplicity representation with the one obtained using the formula computed from Equation (39). Let n be equal to the bitlength of p, then d should be equal to $d = \frac{3}{4} \cdot 4 = 3$, meaning the second representation was ternary. Knowing that the ECDLP is more complex than the DLP, our primary goal was to demonstrate that our method could reduce the number of required logical qubits.

5.1 Solving the Discrete Logarithm Problem Over $\mathbb{F}_{47}$ Using an Efficient Approach

In this experiment, we solved the discrete logarithm problem over the field $\mathbb{F}_{47}$, which is a 6-bit prime field. The given generator was 2, and the multiplicative order of 2 in $\mathbb{F}_{47}$ is 23. We demonstrated how to transform the following discrete logarithm problem into the QUBO problem:

$$2^y \equiv 36 \mod 47, \tag{41}$$

using the classical binary representation of the exponent.

5.1.1 Transformation of the DLP problem to the QUBO problem for the field $\mathbb{F}_{47}$ using binary exponent representation

Let's consider a finite field $\mathbb{F}_{47}$. The order of the multiplicative subgroup in this field equals $46 = 2 \cdot 23$. The element 2 is a generator with a multiplicative order equal to the largest prime divisor of 46, which is 23. We consider the DLP with $h = 36 = 2^y$. We aim to solve this DLP by finding the appropriate y. First, we demonstrate how to transform this problem into the QUBO problem.

Using binary variables $u_1, u_2, \ldots, u_5 \in \{0, 1\}$, we can write y as $y = u_1 + 2u_2 + 4u_3 + 8u_4 + 16u_5$ and therefore:

$$\begin{aligned} 2^y &= 2^{u_1+2u_2+4u_3+8u_4+16u_5} = 2^{u_1} \cdot 2^{2u_2} \cdot 2^{4u_3} \cdot 2^{8u_4} \cdot 2^{16u_5} \\ &= 2^{u_1} \cdot \left(2^2\right)^{u_2} \cdot \left(2^4\right)^{u_3} \cdot \left(2^8\right)^{u_4} \cdot \left(2^{16}\right)^{u_5} = 2^{u_1} \cdot 4^{u_2} \cdot 16^{u_3} \cdot 21^{u_4} \cdot 18^{u_5}. \end{aligned} \tag{42}$$

Note that all values are elements from $\mathbb{F}_{47}$ and thus $2^8 = 21$ and $2^{16} = 18$.

Next, according to Equation (22), we can represent $x_1 = 2^{u_1}$, $x_2 = 4^{u_2}$, $x_3 = 16^{u_3}$, $x_4 = 21^{u_4}$, $x_5 = 18^{u_5}$, $x_6 = u_6 + 2u_7 + 4u_8 + 8u_9 + 16u_{10} + 15u_{11} = x_1 \cdot x_2$, $x_7 = u_{17} + 2u_{18} + 4u_{19} + 8u_{20} + 16u_{21} + 15u_{22} = x_6 \cdot x_3$, $x_8 = u_{34} + 2u_{35} + 4u_{36} + 8u_{37} + 16u_{38} + 15u_{39} =$

$x_7 \cdot x_4,\ x_9 = 36 = x_8 \cdot x_5$ and then

$$\begin{cases} x_1 = 2^{u_1} = u_1 + 1, \\ x_2 = 4^{u_2} = 3u_2 + 1, \\ x_3 = 16^{u_3} = 15u_3 + 1, \\ x_4 = 21^{u_4} = 20u_4 + 1, \\ x_5 = 18^{u_5} = 17u_5 + 1, \\ x_6 = u_6 + 2u_7 + 4u_8 + 8u_9 + 16u_{10} + 15u_{11}, \\ x_7 = u_{17} + 2u_{18} + 4u_{19} + 8u_{20} + 16u_{21} + 15u_{22}, \\ x_8 = u_{34} + 2u_{35} + 4u_{36} + 8u_{37} + 16u_{38} + 15u_{39}, \\ x_5 = 36. \end{cases} \tag{43}$$

Let us note that variables which are necessary to transform in later steps equation from finite field $\mathbb{F}_{47}$, to the pseudo-boolean function (functions with binary variables and integer coefficients), are equal to $k_1 = u_{12}+2u_{13}+4u_{14}+u_{15},\ k_2 = u_{22}+2u_{23}+4u_{24}+8u_{25}+u_{26},\ k_3 = u_{38} + 2u_{39} + 4u_{40} + 8u_{41} + u_{42},\ k_4 = u_{48} + 2u_{49} + 4u_{50} + 4u_{51}$.

Now one obtains that because $x_6 = x_1 \cdot x_2$, the following equation holds:

$$F_1 = u_6 + 2u_7 + 4u_8 + 8u_9 + 16u_{10} + 15u_{11} - (u_1 + 1) \cdot (3u_2 + 1) = 0. \tag{44}$$

Similarly, because $x_7 = x_6 \cdot x_3$, then holds following equation

$$\begin{aligned} F_2 &= u_{17} + 2u_{18} + 4u_{19} + 8u_{20} + 16u_{21} + 15u_{22} \\ &- (u_6 + 2u_7 + 4u_8 + 8u_9 + 16u_{10} + 15u_{11}) \cdot (15u_3 + 1) = 0, \end{aligned} \tag{45}$$

because $x_8 = x_7 \cdot x_4$, then holds following equation

$$\begin{aligned} F_3 &= u_{34} + 2u_{35} + 4u_{36} + 8u_{37} + 16u_{38} + 15u_{39} + 15u_{23} \\ &- (u_{17} + 2u_{18} + 4u_{19} + 8u_{20} + 16u_{21} + 15u_{22}) \cdot (20u_4 + 1) = 0, \end{aligned} \tag{46}$$

and finally, because $x_9 = 36 = x_8 \cdot x_5$

$$F_2 = 36 - (u_{34} + 2u_{35} + 4u_{36} + 8u_{37} + 16u_{38} + 15u_{39}) \cdot (17u_5 + 1) = 0. \tag{47}$$

The equations above will then be equal to

$$\begin{cases} F_1 = 44u_1u_2 + 46u_1 + 44u_2 + u_6 + 2u_7 + 4u_8 + 8u_9 + 16u_{10} + 15u_{11} - 47u_{12} - 94u_{13} \\ -188u_{14} - 47u_{15} + 46, \\ F_2 = 32u_3u_6 + 17u_3u_7 + 34u_3u_8 + 21u_3u_9 + 42u_3u_{10} + 10u_3u_{11} + 46u_6 + 45u_7 + 43u_8 \\ +39u_9 + 31u_{10} + 32u_{11} + u_{17} + 2u_{18} + 4u_{19} + 8u_{20} + 16u_{21} + 15u_{22} - 47u_{23} - 94u_{24} \\ -188u_{25} - 376u_{26} - 47u_{27}, \\ F_3 = 27u_4u_{17} + 7u_4u_{18} + 14u_4u_{19} + 28u_4u_{20} + 9u_4u_{21} + 29u_4u_{22} + 46u_{17} + 45u_{18} + 43u_{19} \\ +39u_{20} + 31u_{21} + 32u_{22} + u_{34} + 2u_{35} + 4u_{36} + 8u_{37} + 16u_{38} + 15u_{39} - 47u_{40} - 94u_{41} \\ -188u_{42} - 376u_{43} - 47u_{44}, \\ F_4 = 30u_5u_{34} + 13u_5u_{35} + 26u_5u_{36} + 5u_5u_{37} + 10u_5u_{38} + 27u_5u_{39} + 46u_{34} \\ +45u_{35} + 43u_{36} + 39u_{37} + 31u_{38} + 32u_{39} - 47u_{51} - 94u_{52} - 188u_{53} - 188u_{54} + 36. \end{cases} \tag{48}$$

The process begins with reducing squares using properties of binary variables; specifically, for any binary variable u, it holds that $u^k = u$, for integer $k \geq 1$. Next, each equation from the pseudo-boolean function over $\mathbb{F}_{47}$ must be transformed into a pseudo-boolean function over integers. For more details on this transformation process, see, for example, [12]. Following this, the square of each equation is computed, and then the sum of all squared equations. After this step, quadratization is performed followed by the addition of penalties.

An alternative method is first to linearize each of the equations and then square each, compute their sum, and add penalties. This approach is detailed in, for example, [4]. The second method allows for easier computation of the necessary number of variables, which is why it was used to analyze the number of necessary variables in Section 2.

5.1.2 Transformation of the DLP problem to the QUBO problem for the field $\mathbb{F}_{47}$ using ternary exponent representation

Consider the same discrete logarithm problem, as given in Subsection 5.1, given by the Equation (41). The difference here is that a ternary representation of the exponent will be used.

Using ternary variables $u_1, u_2, u_3 \in \{0, 1, 2\}$ we can write y as $y = u_1 + 3u_2 + 9u_3$ and therefore:

$$2^y = 2^{u_1+3u_2+9u_3} = 2^{u_1} \cdot 2^{3u_2} \cdot 2^{9u_3} = 2^{u_1} \cdot \left(2^3\right)^{u_2} \cdot \left(2^9\right)^{u_3} = 2^{u_1} \cdot 8^{u_2} \cdot 42^{u_3}. \tag{49}$$

Let us note that all values are elements from $\mathbb{F}_{47}$ and therefore $2^9 = 42$.

Now let's note, that according to Equation (22) we can write $x_1 = 2^{u_1}$, $x_2 = 8^{u_2}$, $x_3 = 42^{u_3}$, $x_4 = 2^{u_1} \cdot 8^{u_2} = x_1 \cdot x_2$, $x_5 = 36 = x_3 \cdot x_4$ and then

$$\begin{cases} x_1 = 2^{u_1} = \mu_1 + 3\mu_2 + 1, \\ x_2 = 8^{u_2} = 7\mu_3 + 16\mu_4 + 1, \\ x_3 = 42^{u_3} = 41\mu_5 + 24\mu_6 + 1, \\ x_4 = \mu_7 + 2\mu_8 + 4\mu_9 + 8\mu_{10} + 16\mu_{11} + 15\mu_{12}, \\ x_5 = 36. \end{cases} \tag{50}$$

Let us note that the operations above are still performed in the $\mathbb{F}_{47}$ field.

Now one obtains that because $x_4 = x_1 \cdot x_2$, the following equation holds:

$$F_1 = \mu_7 + 2\mu_8 + 4\mu_9 + 8\mu_{10} + 16\mu_{11} + 15\mu_{12} - (\mu_1 + 3\mu_2 + 1) \cdot (7\mu_3 + 16\mu_4 + 1) = 0. \tag{51}$$

Similarly, because $x_5 = x_3 \cdot x_4$, then holds following equation

$$F_2 = 36 - (41\mu_5 + 24\mu_6 + 1) \cdot (\mu_7 + 2\mu_8 + 4\mu_9 + 8\mu_{10} + 16\mu_{11} + 15\mu_{12}) = 0. \tag{52}$$

The equations above will then be equal to

$$\begin{cases} F_1 = 40\mu_1\mu_3 + 31\mu_1\mu_4 + 26\mu_2\mu_3 + 46\mu_2\mu_4 + 46\mu_1 + 44\mu_2 + 40\mu_3 + 31\mu_4 + \mu_7 + 2\mu_8 \\ +4\mu_9 + 8\mu_{10} + 16\mu_{11} + 15\mu_{12} + 46, \\ F_2 = 6\mu_5\mu_7 + 12\mu_5\mu_8 + 24\mu_5\mu_9 + \mu_5\mu_{10} + 2\mu_5\mu_{11} + 43\mu_5\mu_{12} + 23\mu_6\mu_7 + 46\mu_6\mu_8 + 45\mu_6\mu_9 \\ +43\mu_6\mu_{10} + 39\mu_6\mu_{11} + 16\mu_6\mu_{12} + 46\mu_7 + 45\mu_8 + 43\mu_9 + 39\mu_{10} + 31\mu_{11} + 32\mu_{12} + 36. \end{cases} \tag{53}$$

The method of obtaining the final problem is then the same as presented in Subsection 5.1.1.

Contrary to the example from Subsection 5.1.1, we solved the problem using the D-Wave Advantage2 prototype this time.

5.2 Solving elliptic curve discrete logarithm problem over $\mathbb{F}_{13}$ using efficient approach

This experiment considers the elliptic curve discrete logarithm problem over a field $\mathbb{F}_{13}$, a 4-bit prime field. The given elliptic curve was $E_{Ed} : x^2 + y^2 = 1 + 7x^2y^2$, the generator was a point $P = (4, 2)$, and the order of P on the chosen curve and in the chosen field is 20. We show how to transform to the QUBO problem the following elliptic curve discrete logarithm problem on $E_{Ed}/\mathbb{F}_{13}$:

$$Q = [y]P = [y](4, 2) = (5, 7), \tag{54}$$

using the classical binary representation of the multiplicity of a point.

5.2.1 Transformation of the ECDLP problem to the QUBO problem for the field $\mathbb{F}_{13}$ using binary multiplicity representation

Consider a finite field $\mathbb{F}_{13}$. The order of the additive subgroup in this field is equal to 20. Point $P = (4, 2)$ is the generator with additive order equal to 20. Consider ECDLP with $Q = (5, 7) = [y](4, 2)$. Our target is to break this ECDLP by finding proper y. We will first show how to transform this problem into the QUBO problem.

Using binary variables $u_1, \ldots, u_5 \in \{0, 1\}$ we can write $y = u_1 + 2u_2 + 4u_3 + 8u_4 + 16u_5$ and thus:

$$\begin{aligned}
[y](4, 2) &= [u_1 + 2u_2 + 4u_3 + 8u_4 + 16u_5](4, 2) \\
&= [u_1](4, 2) + [2u_2](4, 2) + [4u_3](4, 2) + [8u_4](4, 2) + [16u_5](4, 2) \\
&= [u_1](4, 2) + [u_2]([2](4, 2)) + [u_3]([4](4, 2)) + [u_4]([8](4, 2)) + [u_5]([16](4, 2)) \\
&= [u_1](4, 2) + [u_2](6, 5) + [u_3](11, 9) + [u_4](6, 8) + [u_5](2, 9).
\end{aligned} \tag{55}$$

what gives us $P_1 = [1]P = (4, 2), P_2 = [2]P = (6, 5), P_3 = [4]P = (11, 9), P_4 = [8]P = (6, 8), P_5 = [16]P = (2, 9)$ and $Q = [y]P = [u_1]P_1 + [u_2]P_2 + [u_3]P_3 + [u_4]P_4 + [u_5]P_5$.

Now, we can use the idea of a problem decomposition from Figure 1. We obtain

$$\begin{cases}
(x_1, y_1) = [u_1]P_1, \\
(x_2, y_2) = [u_2]P_2, \\
(x_3, y_3) = R_1 = [u_1]P_1 + [u_2]P_2, \\
(x_4, y_4) = [u_3]P_3, \\
(x_5, y_5) = R_2 = R_1 + [u_3]P_3, \\
(x_6, y_6) = [u_4]P_4, \\
(x_7, y_7) = R_3 = R_2 + [u_4]P_4, \\
(x_8, y_8) = [u_5]P_5, \\
(x_9, y_9) = R_3 + [u_5]P_5 = Q,
\end{cases} \tag{56}$$

what, according to the System (32), is equal to

$$\begin{cases} (x_1, y_1) = (4u_1, 1 + u_1) \\ (x_2, y_2) = (6u_2, 1 + 4u_2), \\ (x_3, y_3) = (u_6 + 2u_7 + 4u_8 + 8u_9, u_{10} + 2u_{11} + 4u_{12} + 8u_{13}), \\ (x_4, y_4) = (11u_3, 1 + 8u_3), \\ (x_5, y_5) = (u_{14} + 2u_{15} + 4u_{16} + 8u_{17}, u_{18} + 2u_{19} + 4u_{20} + 8u_{21}), \\ (x_6, y_6) = (6u_4, 1 + 7u_4), \\ (x_7, y_7) = (u_{22} + 2u_{23} + 4u_{24} + 8u_{25}, u_{26} + 2u_{27} + 4u_{28} + 8u_{29}), \\ (x_8, y_8) = (2u_5, 1 + 8u_5), \\ (x_9, y_9) = (5, 7). \end{cases} \tag{57}$$

With the knowledge about a points addition arithmetic from the Equation (31) and that $-1 \equiv 12 \mod 13$, we can transform the above equations for R_1, R_2, R_3, Q to the form

$$\begin{cases} F_1 = (1 + dx_1x_2y_1y_2)x_3 + 12(x_1y_2 + y_1x_2) = 0, \\ F_2 = (1 - dx_1x_2y_1y_2)y_3 + 12(y_1y_2 - x_1x_2) = 0, \\ F_3 = (1 + dx_3x_4y_3y_4)x_5 + 12(x_3y_4 + y_3x_4) = 0, \\ F_4 = (1 - dx_3x_4y_3y_4)y_5 + 12(y_3y_4 - x_3x_4) = 0, \\ F_5 = (1 + dx_5x_6y_5y_6)x_7 + 12(x_5y_6 + y_5x_6) = 0, \\ F_6 = (1 - dx_5x_6y_5y_6)y_7 + 12(y_5y_6 - x_5x_6) = 0, \\ F_7 = (1 + dx_7x_8y_7y_8)x_9 + 12(x_7y_8 + y_7x_8) = 0, \\ F_8 = (1 - dx_7x_8y_7y_8)y_9 + 12(y_7y_8 - x_7x_8) = 0. \end{cases} \tag{58}$$

As mentioned in the subsection 4.1, for small finite fields, one may use the whole formula for a point coordinate instead of decomposing each of them into nine shorter ones as in the System (33). Thanks to this, we have here eight equations instead of 36, and because we use the field $\mathbb{F}_{13}$, conducting the above multiplications should not cause any trouble.

Replacing variables x_i, y_i with the proper values from the System (57), and taking $d = 2$, ends with

$$\begin{cases} F_1 = 9u_0^2u_1^2u_5 + 5u_0^2u_1^2u_6 + 10u_0^2u_1^2u_7 + \ldots + 2u_6 + 4u_7 + 8u_8, \\ F_2 = 4u_0^2u_1^2u_9 + 8u_0^2u_1^2u_{10} + 3u_0^2u_1^2u_{11} + \ldots + 4u_{11} + 8u_{12} + 12, \\ F_3 = 5u_2^2u_5u_9u_{13} + 10u_2^2u_6u_9u_{13} + 7u_2^2u_7u_9u_{13} + \ldots + 2u_{14} + 4u_{15} + 8u_{16}, \\ F_4 = 8u_2^2u_5u_9u_{17} + 3u_2^2u_6u_9u_{17} + 6u_2^2u_7u_9u_{17} + \ldots + 2u_{18} + 4u_{19} + 8u_{20}, \\ F_5 = 8u_3^2u_{13}u_{17}u_{21} + 3u_3^2u_{14}u_{17}u_{21} + 6u_3^2u_{15}u_{17}u_{21} + \ldots + 2u_{22} + 4u_{23} + 8u_{24}, \\ F_6 = 5u_3^2u_{13}u_{17}u_{25} + 10u_3^2u_{14}u_{17}u_{25} + 7u_3^2u_{15}u_{17}u_{25} + \ldots + 2u_{26} + 4u_{27} + 8u_{28}, \\ F_7 = u_4^2u_{21}u_{25} + 2u_4^2u_{22}u_{25} + 4u_4^2u_{23}u_{25} + \ldots + 9u_{23} + 5u_{24} + 5, \\ F_8 = 9u_4^2u_{21}u_{25} + 5u_4^2u_{22}u_{25} + 10u_4^2u_{23}u_{25} + \ldots + 9u_{27} + 5u_{28} + 7. \end{cases} \tag{59}$$

The method of obtaining the final problem is then the same as presented in Subsection 5.1.1. The only difference is that in this experiment, we transform each equation from the pseudo-boolean function over $\mathbb{F}_{13}$ to the pseudo-boolean function over integers.

The obtained QUBO problem consists of 435 logical qubits. We were unable to solve this problem quantumly using the D-Wave Advantage prototype.

5.2.2 Transformation of the ECDLP problem to the QUBO problem for the field $\mathbb{F}_{13}$ using ternary multiplicity representation

Consider the same elliptic curve discrete logarithm problem, as given in Subsection 5.2, given by the Equation (54). The difference here is that a ternary representation of the multiplicity of point P will be used.

Using ternary variables $u_1, u_2, u_3 \in \{0, 1, 2\}$ we can write y as $y = u_1 + 3u_2 + 9u_3$ and therefore:

$$\begin{aligned}[y](4,2) &= [u_1 + 3u_2 + 9u_3](4,2) \\ &= [u_1](4,2) + [3u_2](4,2) + [9u_3](4,2) \\ &= [u_1](4,2) + [u_2]([3](4,2)) + [u_3]([9](4,2)) \\ &= [u_1](4,2) + [u_2](8,7) + [u_3](4,11).\end{aligned} \tag{60}$$

what gives us $P_1 = [1]P = (4,2), P_2 = [3]P = (8,7), P_3 = [9]P = (4,11)$ and $Q = [y]P = [u_1]P_1 + [u_2]P_2 + [u_3]P_3$.

Now, we can use the idea of a problem decomposition from Figure 1. We obtain

$$\begin{cases}(x_1, y_1) = [u_1]P_1, \\ (x_2, y_2) = [u_2]P_2, \\ (x_3, y_3) = R_1 = [u_1]P_1 + [u_2]P_2, \\ (x_4, y_4) = [u_3]P_3, \\ (x_5, y_5) = R_1 + [u_3]P_3 = Q.\end{cases} \tag{61}$$

what, according to the System (38), is equal to

$$\begin{cases}(x_1, y_1) = (4\mu_1 + 6\mu_2, 1 + \mu_1 + 4\mu_2), \\ (x_2, y_2) = (8\mu_3 + 11\mu_4, 1 + 6\mu_3 + 3\mu_4), \\ (x_3, y_3) = (\mu_5 + 2\mu_6 + 4\mu_7 + 8\mu_8, \mu_9 + 2\mu_{10} + 4\mu_{11} + 8\mu_{12}), \\ (x_4, y_4) = (4\mu_{13} + 7\mu_{14}, 1 + 10\mu_{13} + 4\mu_{14}), \\ (x_5, y_5) = (5, 7).\end{cases} \tag{62}$$

With the knowledge about a points addition arithmetic from the Equation (31) and that $-1 \equiv 12 \mod 13$, we can transform the above equations for R_1, Q to the form

$$\begin{cases}F_1 = (1 + dx_1x_2y_1y_2)x_3 + 12(x_1y_2 + y_1x_2) = 0, \\ F_2 = (1 - dx_1x_2y_1y_2)y_3 + 12(y_1y_2 - x_1x_2) = 0, \\ F_3 = (1 + dx_3x_4y_3y_4)x_5 + 12(x_3y_4 + y_3x_4) = 0, \\ F_4 = (1 - dx_3x_4y_3y_4)y_5 + 12(y_3y_4 - x_3x_4) = 0.\end{cases} \tag{63}$$

Because we need to conduct only 2 points additions, we use the shorter path, as in the Experiment 5.2.1.

Replacing variables x_i, y_i and d with the proper values from the System (62) ends with

$$\begin{cases}F_1 = 5u_0^2u_2^2u_4 + 8u_0u_1u_2^2u_4 + 4u_1^2u_2^2u_4 + \ldots + 2u_5 + 4u_6 + 8u_7 \\ F_2 = 8u_0^2u_2^2u_8 + 5u_0u_1u_2^2u_8 + 9u_1^2u_2^2u_8 + \ldots + 4u_{10} + 8u_{11} + 12, \\ F_3 = 9u_4u_8u_{12}^2 + 5u_5u_8u_{12}^2 + 10u_6u_8u_{12}^2 + \ldots + 9u_6 + 5u_7 + 5, \\ F_4 = 3u_4u_8u_{12}^2 + 6u_5u_8u_{12}^2 + 12u_6u_8u_{12}^2 + \ldots + 9u_{10} + 5u_{11} + 7.\end{cases} \tag{64}$$

The method of obtaining the final problem is then the same as presented in Subsection 5.2.1.

The obtained QUBO problem consists of 218 logical qubits. We were unable to solve this problem quantumly using the D-Wave Advantage prototype.

5.3 Experiments summary

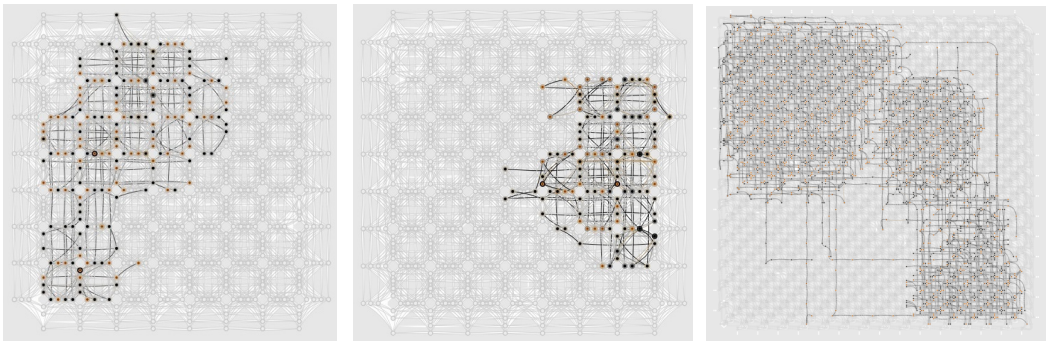
Unfortunately, we could not solve discrete logarithm problems over prime fields with a bitlength greater than 6 using a quantum solver and the D-Wave Advantage QPU.

Solving elliptic curve discrete logarithm problems was also beyond our capabilities due to the number of variables required. However, utilizing different points' multiplicity representations provided fewer source variables, allowing us to embed the problem in the D-Wave Advantage system. This would have been impossible using a binary representation.

Parameter	DLP over $\mathbb{F}_{47}$ (base 2)	DLP over $\mathbb{F}_{47}$ (base 3)	ECDLP over $\mathbb{F}_{13}$ (base 2)	ECDLP over $\mathbb{F}_{13}$ (base 3)
Name (chip ID)	Advantage2 prototype1.1	Advantage2 prototype1.1	Advantage system6.3	Advantage system6.3
Qubits	576	576	5760	5760
Topology	Zephyr	Zephyr	Pegasus	Pegasus
Number of source variables	57	36	435	218
Number of target variables	189	117	none	2654
Max chain length	6	5	none	35
Chain strength	32,848.03	37,256.21	none	1,398,369.72
QPU access time (μs)	491,197	555,092	none	672,923
QPU programming time (μs)	6,846.81	6,842	none	15,923.57
QPU sampling time (μs)	484,350	548,250	none	657,000
Total post processing time (μs)	2,948	3,464	none	18,214
Post processing overhead time (μs)	2,948	3,464	none	3706
Solved	No	Yes	No	No

Table 1. Parameters used in solving QUBO problem equivalent to the DLP over $\mathbb{F}_{47}$ and ECDLP over $\mathbb{F}_{13}$ with exponents/multiplicities represented in different bases.

Figure 3 shows how different QUBO problems were embedded on the D-Wave Advantage computer.



(a) (b) (c)

Fig. 3. Embedding of QUBO problems equivalent to discrete logarithm problems over the prime field $\mathbb{F}_{47}$ using binary exponent representation in (a) and ternary representation in (b), as well as elliptic curve discrete logarithm problem over the prime field $\mathbb{F}_{13}$ using ternary multiplicity representation in (c). 2c.

6 Conclusion

This paper presents a novel method for transforming the discrete logarithm problem and the elliptic curve discrete logarithm problem over prime fields. The method leverages a representation system for the exponent. Using the presented method, we were able to achieve a reduction of the DLP over prime field $\mathbb{F}_p$ to the QUBO problem, asymptotically equal to $1.89n^2$ logical variables, where n is the bitlength of the prime number p. Similarly, for the ECDLP, the reduction of this problem defined over a prime field $\mathbb{F}_p$ to the QUBO problem requires asymptotically $\frac{6n^3}{\log_2(\frac{3}{4}n)}$ logical variables. Both of these results are an improvement over previously known methods for reducing these problems. For the DLP, such a reduction previously required $2n^2$ logical variables, and for the ECDLP, it required $3n^3$ logical variables.

It is also noteworthy that using the presented method of reduction of the DLP to the QUBO problem, we were able to solve an example of the DLP defined over $\mathbb{F}_{47}$, using a ternary representation of the exponent. As shown, this reduction requires significantly fewer variables than the reduction using a binary representation of the exponent.

References

1. Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, pages 29–50, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
2. Elżbieta Burek, Krzysztof Mańk, and Michał Wroński. Searching for an Efficient System of Equations Defining the AES Sbox for the QUBO Problem. *Journal of Telecommunications and Information Technology*, 4:30–37, 2023.
3. Elżbieta Burek and Michał Wroński. Quantum Annealing and Algebraic Attack on Speck Cipher. In *International Conference on Computational Science*, pages 143–149. Springer, 2022.
4. Elżbieta Burek, Michał Wroński, Krzysztof Mańk, and Michał Misztal. Algebraic attacks on block ciphers using quantum annealing. *IEEE Transactions on Emerging Topics in Computing*, 10(2):678–689, 2022.
5. The Quantum Computing Company D-WAVE. Getting started with the d-wave system. *User manual*, 2020.

6. Martin Ekerå and Joel Gärtner. Extending regev's factoring algorithm to compute discrete logarithms. In *International Conference on Post-Quantum Cryptography*, pages 211–242. Springer, 2024.
7. Shuxian Jiang, Keith A Britt, Alexander J McCaskey, Travis S Humble, and Sabre Kais. Quantum annealing for prime factorization. *Scientific reports*, 8(1):1–9, 2018.
8. John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *arXiv preprint quant-ph/0301141*, 2003.
9. Seyoon Ragavan and Vinod Vaikuntanathan. Optimizing space in regev's factoring algorithm. *arXiv preprint arXiv:2310.00899*, 2023.
10. Oded Regev. An efficient quantum factoring algorithm. *arXiv preprint arXiv:2308.06572*, 2023.
11. Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.
12. Michał Wroński. Practical solving of discrete logarithm problem over prime fields using quantum annealing. In Derek Groen, Clélia de Mulatier, Maciej Paszyński, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2022*, pages 93–106, Cham, 2022. Springer International Publishing.
13. Michał Wroński, Elżbieta Burek, Łukasz Dzierzkowski, and Olgierd Żołnierczyk. Transformation of Elliptic Curve Discrete Logarithm Problem to QUBO Using Direct Method in Quantum Annealing Applications. *Journal of Telecommunications and Information Technology*, (1):75–82, 2024.
14. Michał Wroński, Elżbieta Burek, and Mateusz Leśniak. (In)security of stream ciphers against quantum annealing attacks on the example of the Grain 128 and Grain 128a ciphers. *Cryptology ePrint Archive*, 2023.