SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Quantum attack on RSA by D-Wave Advantage: a first break of 80-bit RSA

Chunlei Hong[1,2], Zhi Pei[1,2], Qidi Wang[1,2], Shuxiao Yang[1,2], Jingjing Yu[1,2] & Chao Wang[1,2*]

[1]*Shanghai University, Shanghai* 200444*, China;*
[2]*Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai* 200444*, China*

Appendix A An example of integer factorization

The hybrid architecture of quantum annealing and classical algorithms can factorize any RSA integer from 4-bit to 80-bit. We will take the 12-bit RSA integer 3959 as an example to explain the process of integer factorization in detail.

■ **Constructing the CVP**

First, a lattice $\mathcal{L}(\mathbf{B_{n,c}}) \in \mathbb{R}^{n+1}$ and the corresponding target vector $\mathbf{T} \in \mathbb{R}^{n+1}$ are constructed. Here, $\mathbf{B_{n,c}} = [\mathbf{b}_1, \cdots, \mathbf{b}_n] \in \mathbb{R}^{n+1,n}$ represents a lattice basis, $c > 0$ is an adjustable parameter, and N is the integer to be factorized.

$$\mathbf{B}_{n,c} = \begin{pmatrix} f(1) & 0 & \cdots & 0 \\ 0 & f(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(n) \\ \lceil m^c \ln p_1 \rfloor & \lceil m^c \ln p_2 \rfloor & \cdots & \lceil m^c \ln p_n \rfloor \end{pmatrix}, \ \mathbf{T} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \lceil m^c \ln N \rfloor \end{pmatrix}.$$

Let $p_i (i = 1, 2, \cdots, n)$ be the smallest n prime numbers, satisfying $-1 = p_0 < p_1 < \cdots < p_n$. Considering the influence of m^c on finding smooth pairs and the efficiency of the LLL algorithm, the value of m is fixed at 5. We define the smooth upper bounds of u, v in $(u, |u - vN|)$ as p_n, and the smooth upper bound for $|u - vN|$ in $(u, |u - vN|)$ as $p_{n'}$, typically setting $p_{n'} = p_{2n}{}^2$ as the upper bound. In order to efficiently identify a large number of smooth pairs, we can continuously adjust the values of c and $[f(1), f(2), f(3), f(4)]$ to change the lattice basis $\mathbf{B_{n,c}}$ and the corresponding target vector $\mathbf{T}$. A lot of experiments suggest that for integers smaller than 30-bit set c between 1 and 4. For integers ranging from 30-bit to 50-bit, the optimal c for searching smooth pairs is between 3 and 6. When dealing with integers between 50-bit and 60-bit, the most effective of c is between 6 and 9, and for integers between 60-bit and 80-bit, it is between 8 and 11. Moreover, the diagonal element $f(i)$ in $\mathbf{B_{n,c}}$ denotes a random permutation $f : [1, \cdots, n] \to [1, \cdots, n]$.

According to reference [1], it can be inferred that the value of the dimension n tends to the following formula:

$$n \approx logN/loglogN. \tag{A1}$$

From the formula (A1), we can determine that the lattice dimension of this example is $n = 4$. Then, we choose the adjustable parameter $c = 3.3$ and the value of $[f(1), f(2), f(3), f(4)]$ is $[2, 1, 3, 4]$ as an example. Thus, the initial lattice basis $\mathbf{B_{4,3.3}} = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4] \in \mathbb{R}^{5,4}$ and the corresponding target vector $\mathbf{T_{4,3.3}} \in \mathbb{R}^5$ are as follows:

$$\mathbf{B}_{4,3.3} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \\ 140 & 223 & 326 & 394 \end{pmatrix}, \mathbf{T}_{4,3.3} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1678 \end{pmatrix}.$$

■ **Quantum annealing to solve the CVP**

Next, we use the LLL algorithm to perform lattice reduction on $\mathbf{B_{4,3.3}}$, obtaining the lattice reduction basis $\mathbf{D_{4,3.3}} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4] \in \mathbb{R}^{5,4}$:

$$\mathbf{D}_{4,3.3} = \begin{pmatrix} 4 & 8 & -2 & -4 \\ -3 & -4 & -2 & -2 \\ 0 & 3 & 9 & 3 \\ 4 & 0 & -4 & 4 \\ 5 & -6 & -2 & -6 \end{pmatrix}.$$

* Corresponding author (email: wangchao@shu.edu.cn)

After applying the Babai algorithm, the vector $\mathbf{b} = [0, 9, -3, 0, 1681]^T$ that is closest to the target vector $\mathbf{T_{4,3.3}}$ is obtained. Then, we can calculate the Euclidean distance between vectors $\mathbf{b}$ and $\mathbf{T_{4,3.3}}$:

$$dis_{4,3.3} = \|\mathbf{b} - \mathbf{T}_{4,3.3}\|^2 = \left\|[0, 9, -3, 0, 3]^T\right\|^2 = 99.$$

The vector $\mathbf{b}$ can be further optimized using the quantum annealing algorithm to obtain a new vector $\mathbf{b}_{new}$ that is closer to the target vector. Initially, we should construct the Hamiltonian $H_{\mathbf{b}} = \sum_{j=1}^{5} \hat{h}_j$, where:

$$\begin{cases} \hat{h}_1 = (4\hat{y}_1 + 8\hat{y}_2 - 2\hat{y}_3 - 4\hat{y}_4)^2 ; \\ \hat{h}_2 = (-3\hat{y}_1 - 4\hat{y}_2 - 2\hat{y}_3 - 2\hat{y}_4 + 9)^2 ; \\ \hat{h}_3 = (0\hat{y}_1 + 3\hat{y}_2 + 9\hat{y}_3 + 3\hat{y}_4 - 3)^2 ; \\ \hat{h}_4 = (4\hat{y}_1 + 0\hat{y}_2 - 4\hat{y}_3 + 4\hat{y}_4)^2 ; \\ \hat{h}_5 = (5\hat{y}_1 - 6\hat{y}_2 - 2\hat{y}_3 - 6\hat{y}_4 + 3)^2 . \end{cases} \tag{A2}$$

According to the mapping rule, formula (A3) can be mapped to the Pauli matrix, resulting in the corresponding Hamiltonian as follows:

$$H_{\mathbf{b}} = \sum_{j=1}^{5} \hat{h}_j = 143I - 2\sigma_z^1 - 32.5\sigma_z^2 - 22.5\sigma_z^3 - 11.5\sigma_z^4 + 7\sigma_z^1\sigma_z^2 - 14\sigma_z^1\sigma_z^3 - 12\sigma_z^1\sigma_z^4 + 15.5\sigma_z^2\sigma_z^3 + 10.5\sigma_z^2\sigma_z^4 + 17.5\sigma_z^3\sigma_z^4, \tag{A3}$$

where $143I$ is a constant term that does not affect the solution of the lowest energy. According to formula (A3), we can extract the first-order coefficient to form the local field coefficient matrix h^T and extract the second-order coefficient to form the coupling coefficient matrix J.

$$h^T = \begin{pmatrix} \sigma_z^1 & \sigma_z^2 & \sigma_z^3 & \sigma_z^4 \\ -2 & -32.5 & -22.5 & -11.5 \end{pmatrix}, \; J = \begin{pmatrix} & \sigma_z^1 & \sigma_z^2 & \sigma_z^3 & \sigma_z^4 \\ \sigma_z^1 & 0 & 7 & -14 & -12 \\ \sigma_z^2 & 0 & 0 & 15.5 & 10.5 \\ \sigma_z^3 & 0 & 0 & 0 & 17.5 \\ \sigma_z^4 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where the values of h^T and J will affect the success rate of the quantum annealing.

In our experiment, we embedded the local field coefficient matrix h^T and the coupling coefficient matrix J into the Ising model of the D-Wave Advantage. Then, the lowest energy along with five sub-lowest energies and their corresponding solutions were obtained, as shown in Table A1. The column labeled "num" indicates the frequency of each solution across 1000 quantum annealing iterations.

Table A1 Six solutions were obtained using the D-Wave Advantage

S/N	energy	s_1	s_2	s_3	s_4	num
1	-59	-1	$+1$	$+1$	-1	767
2	-53	$+1$	$+1$	$+1$	-1	77
3	-51	-1	$+1$	-1	$+1$	134
4	-45	$+1$	-1	$+1$	$+1$	11
5	**-44**	**+1**	**+1**	**+1**	**+1**	**6**
6	-38	-1	$+1$	-1	-1	4

As shown in Table A1, the bolded solution corresponds to the solution obtained by the Babai algorithm. It is observed that the quantum annealing algorithm can obtain four solutions with lower energy than those obtained by the Babai algorithm.

■ **Collecting smooth pairs**

Based on the six sets of solutions obtained through quantum annealing, we can derive six distinct vectors $\mathbf{b}_{new}$. Using these vectors, we can calculate the corresponding values of u, v, and $|u - vN|$. A smooth boundary of p_4 was set for u, v during the factorization of the integer 3959, which implies that the prime factors of u, v are less than or equal to 7. The smooth boundary for $|u - vN|$ is p_{16}, meaning that the prime numbers factorized by $|u - vN|$ are less than or equal to 53. Table A2 shows the corresponding energy, quantum states, the values of $u, v, |u - vN|$, and the smoothness of $(u, |u - vN|)$ for these vectors.

Table A2 Comparison of smoothness between the quantum annealing algorithm and the Babai algorithm

levels	energy	state	u	v	$\lvert u - vN\rvert$	smooth
1	-59	1001	3^*7^2	1	$2 * 5$	yes
2	-53	0001	$3^7 * 7$	$2 * 2$	$17 * 31$	yes
3	-51	1010	$3 * 3^4 * 5^2$	1	$7 * 13$	yes
4	-45	0100	$2^4 * 3^5$	1	71	no
5	**-44**	**0000**	$\mathbf{3^9}$	**5**	$\mathbf{2^4*7}$	**yes**
6	-38	0010	$3^2 * 5^3 * 7$	2	43	yes

Table A3 Eighteen different smooth pairs

sn	u	v	$\lvert u-vN\rvert$
1	2^2*3^5*5	1	$17*53$
2	$2*3^4*5^2$	1	$7*13$
3	3^6*7	1	$2^3*11*13$
4	3^4*7^2	1	$2*5$
5	3^7*7	2^2	$17*31$
6	3^9	5	2^4*7
7	3^2*5^3*7	2	43
8	5^4*7	1	2^5*13
9	$2*3*5^4$	1	$11*19$
10	2^2*5^3*7	1	3^3*17
11	$5*7^4$	3	2^7
12	$2^2*5^2*7^3$	3^2	11^3
13	2^4*5*7^2	1	$3*13$
14	$2*5*7^3$	1	23^2
15	2^8*7^2	3	$23*29$
16	2^5*7^3	3	$17*53$
17	$3*7^4$	2	$5*11*13$
18	2^8*3*5	1	$7*17$

The bolded solution in Table A2 corresponds to the solution obtained by the Babai algorithm. Thus, it is evident that the four new vectors that outperform the Babai algorithm contain three new smooth pairs. By changing the initial lattice basis $\mathbf{B_{4,3.3}}$ and target vector $\mathbf{T_{4,3.3}}$, more different smooth pairs $(u, |u-vN|)$ can be obtained until it reaches 18 different smooth pairs. All smooth pairs are shown in Table A3.

■ **Solving linear equations**

We can construct a set of Boolean index vectors from the 18 smooth pairs $(u, |u-vN|)$ in Table A3, as shown in Table A4. The first column shows the serial number of the smooth pair that corresponds to each Boolean index. The second column shows whether the value of $u-vN$ is negative. The value is recorded as 0 if positive; otherwise, it is recorded as 1. For each $s_i (1 \leqslant i \leqslant 16)$, a Boolean value of 0 is assigned if its power index is even; otherwise, the corresponding Boolean value is 1.

Table A4 Boolean index table corresponding to the smooth pairs

sn	sg	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1
2	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
4	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0
6	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
8	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
9	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
10	1	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
11	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
13	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
14	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
16	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
17	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
18	1	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0

According to Table A4, we can construct the linear equations. By using the Gaussian elimination, one of the solutions obtained is $(0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0)$. This solution implies that the 4th and 11th vectors are linearly dependent.

■ **The result of integer factorization**

Taking the linearly dependent vectors obtained in the previous part, i.e., the 4th and 11th vectors are linearly dependent. It can be calculated that $X^2 = 3087^2, Y^2 = 16^2$. Finally, we can obtain two prime factors of the integer 3959 as: $P = gcd(3087+16, 3959) = 107$, $Q = gcd(3087-16, 3959) = 37$. Thus, the result of integer factorization is $3959 = 37 \times 107$.

Appendix B Analysis of experimental results

We conducted all experiments using the D-Wave Advantage_system4.1, which is equipped with a Pegasus topology that includes 5760 physical qubits. Additionally, we utilized the default annealing schedule of [[0,0],[20,1]], which represents a linear progression of the control parameter. This parameter starts at a minimum value of 0 and linearly increases to a maximum value of 1 throughout a period of $20\mu s$.

Appendix B.1 Comparative analysis of quantum algorithms for integer factorization

Recent reference [2] uses the quantum approximate optimization algorithm (QAOA) to factorize three RSA integers of 11-bit, 26-bit, and 48-bit. Based on the Hamiltonian in reference [2], Narendra *et al.* [3] used the digital counter-diabatic quantum factorization (DCQF) algorithm to factorize the same 26-bit and 48-bit integers. We also choose the same integers to conduct integer factorization experiments in order to compare the factorization effects of the quantum annealing (QA), QAOA and DCQF, as shown in Table B1.

Table B1 Compare the success rate of quantum algorithms for solving CVP corresponding to integer factorization.

The scale of the integer	Logical qubits	Physical qubits	QA	QAOA	DCQF
26-bit (48567227)	5	6	80.4%	$<30\%$(P=3)	49.3%
48-bit (261980999226229)	10	16	64.5%	2%(P=3),$<4\%$(P=4)	9.7%

According to Table B1, it can be observed that the success rate of solving the CVP corresponding to integer factorization using the QA far exceeds that of using DCQF and QAOA. Take the CVP for a 48-bit integer factorization as an example: when P=4 (P is the quantum circuit depth), the maximum success rate using the QAOA algorithm does not exceed 4%. In an ideal state, Narendra used DCQF to factorize the same CVP with a success rate of 11.5%, which has increased to some extent compared to the reference [2]. However, factorizing the same CVP using a QA on the D-Wave Advantage has a success rate as high as 64.5%.

By comparing the success rates of 48-bit RSA integer factorization corresponding to CVP, it can be inferred that the success rates of QAOA and DCQF algorithms will be extremely low when solving larger problems, especially CVPs corresponding to 80-bit RSA integers. Although the success rate of our method in solving the CVP for the 80-bit RSA integer is only about 10%, it can undoubtedly factorize the 80-bit integer after multiple iterations of solving the CVP problem. As a result, our integer factorization scale is far beyond the current scale of QAOA and DCQF.

Appendix B.2 Comparative analysis of QA and classical algorithms

In this subsection, we will explore the efficiency of integer factorization using the QA algorithm and the SA algorithm. We will use the factorization of a 64-bit integer $N = 11769431398084725929$ as the example to compare and analyze these methods. The CVP with a lattice dimension of 22 is utilized, where the initial lattice basis diagonal is $Diagonal$ =[19, 21, 22, 11, 13, 3, 12, 6, 18, 2, 1, 7, 5, 10, 15, 16, 17, 20, 4, 8, 9, 14]. The adjustable parameter c is 9.1, and the smoothness bound is 968, aiming to identify at least 970 smooth pairs for integer factorization.

■ QA algorithm

We set up the QA process to run 3000 iterations using 63 physical qubits on the D-Wave Advantage_system4.1. To address the problem, we used an annealing schedule of [[0, 0], [20, 1]]. Although not every iteration successfully solves the ground state, acquiring a set of superior suboptimal solutions is beneficial for the search for smooth pairs. Consequently, we do not present the success rate for each CVP; instead, we primarily focus on comparing the quality and time consumption of the solution set.

Quality of the solution: We obtain an optimal minimum and about 550 suboptimal solutions, where the lowest energy is -6246. The 61st suboptimal solution identified a smooth pair for integer factorization.

Time consumption: The time for D-Wave to solve the CVP using QPU was $0.51s$, and verifying the smoothness of these solutions took $1.16s$. Total_time = QPU solve time + Verify smoothness time= $0.51s + 1.16s = 1.67s$.

■ SA algorithm

We set the initial temperature for the SA process at 1000, with the final temperature at 0.01 and a cooling coefficient of 0.99. We configured the SA to execute 3000 iterations and preserved 600 solutions for this problem, consisting of an optimal minimum and 599 suboptimal solutions.

Quality of the solution: The optimal minimum obtained through SA was significantly inferior to that of QA, ranking about the 536th suboptimal solution in QA, and failed to identify a smooth pair for these 600 solutions.

Time consumption: The SA takes $263.75s$ to solve the CVP and $1.04s$ to verify the smoothness of these solutions. Total_time = SA solve time + Verify smoothness time= $263.75s + 1.04s = 264.79s$.

It is observed that the QA successfully obtains a smooth pair in $1.67s$. In contrast, the SA algorithm fails to find any smooth pairs within $264.79s$.

Next, we will present the time consumption of each algorithm for the adjustable parameter c ranging from 9.0 to 10.0, as demonstrated in Table B2. Furthermore, the comparative analysis of solution quality between the QA and SA algorithms will be presented in Table B3, using the optimal minimum from SA as a reference for comparing its position relative to the solutions obtained through QA.

Table B2 Time comparison.

n	c	SA(s)	QA(s)
22	9.0	269.07	1.35
22	9.1	264.79s	1.67
22	9.2	276.18	3.17
22	9.3	264.28	1.66
22	9.4	266.31	1.34
22	9.5	271.24	1.53
22	9.6	274.56	2.62
22	9.7	269.17	1.61
22	9.8	265.23	1.80
22	9.9	269.15	1.59

Table B3 Solution quality comparison.

n	c	SA	QA
22	9.0	1	343
22	9.1	1	96
22	9.2	1	599
22	9.3	1	138
22	9.4	1	419
22	9.5	1	304
22	9.6	1	451
22	9.7	1	252
22	9.8	1	250
22	9.9	1	103

The QA successfully identified two smooth pairs across these CVPs, while the SA algorithm found none. The data presented in the preceding tables clearly illustrate that the SA algorithm significantly underperforms compared to QA in both efficiency and solution quality for addressing suboptimal solutions in these CVPs of this scale.

It is essential to identify at least 970 smooth pairs for effective factorization of this 64-bit integer. We then use the average time required to solve the ten different CVPs previously described as a foundation for estimating the time necessary to factorize the 64-bit integer using the QA and the SA.

■ **QA algorithm**

Due to repeated occurrences of the same smooth pairs during the solution process, our experiments showed that QA solved 4800 distinct CVPs and successfully identified 970 smooth pairs. From the data gathered on the ten different CVPs, we estimate that QA would require approximately $4800 \times 1.83s = 8784s$. However, the actual time consumed was about 12 hours, due to additional factors such as internet upload times and other delays.

■ **SA algorithm**

Disregarding the quality of the solutions provided by the SA, it would need to solve at least 4800 CVPs as QA. Consequently, the time required for SA would be $4800 \times 269s = 1291200s$, or approximately 15 days. Unfortunately, SA struggles to effectively find a set of high-quality suboptimal solutions. Therefore, when applied to 4800 different CVPs, it faces difficulties in identifying 970 smooth pairs, necessitating the resolution of additional CVPs to achieve 970 smooth pairs. It is estimated that the QA would take more than 15 days to factorize this 64-bit integer effectively.

Based on the above, it is evident that the QA algorithm is more efficient than the SA method for the factorization of 64-bit integers. Moreover, the SA method is highly challenging and time-consuming, which highlights its limitations in factorizing larger RSA integers efficiently. Consequently, it can be deduced that the SA method faces extreme difficulties when applied to the factorization of 80-bit RSA integers.

Appendix B.3 Experiment results

In this subsection, we present the factorization data of randomly selected RSA integers ranging from 4-bit to 80-bit, the number of logical and physical qubits, and the success rate of the QA for solving the CVP corresponding to the integer factorization. The detailed results are shown in the following Table B4.

Table B4 Experiment results

Scale	Data	Logical qubits	Physical qubits	Succes rate
4	$15 = 3 \times 5$	2	2	100
5	$21 = 3 \times 7$	2	2	100
6	$35 = 5 \times 7$	3	3	100
7	$77 = 7 \times 11$	3	3	100
8	$143 = 11 \times 13$	3	3	100
9	$451 = 11 \times 41$	3	3	100
10	$899 = 29 \times 31$	3	3	100
11	$1769 = 29 \times 61$	3	3	100
12	$4009 = 19 \times 211$	4	4	86.4
13	$5723 = 59 \times 97$	4	4	88.7
14	$15853 = 83 \times 191$	4	4	87.3
15	$21733 = 103 \times 211$	4	4	86.3
16	$49583 = 179 \times 277$	4	4	84.4
17	$94037 = 271 \times 347$	4	4	87.4
18	$159317 = 313 \times 509$	4	4	84.9
19	$374599 = 521 \times 719$	4	4	85.8
20	$881851 = 727 \times 1213$	4	4	84.2
21	$1794953 = 907 \times 1979$	4	4	85.4

Table B5 Experiment results

Scale	Data	Logical qubits	Physical qubits	Succes rate
22	3819047 = 1873 × 2039	4	4	86.1
23	6187267 = 2003 × 3089	5	6	82.9
24	16668371 = 3637 × 4583	5	6	81.4
25	17140121 = 3803 × 4507	5	6	82.2
26	40072889 = 5279 × 7591	5	6	80.4
27	104250163 = 9733 × 10711	5	6	80.2
29	293222843 = 13883 × 21121	5	6	80.9
30	911938673 = 29483 × 30931	6	8	77.3
31	1328094533 = 31607 × 42019	6	8	78.3
32	2956694171 = 50683 × 58337	6	8	79.3
33	5893535383 = 72551 × 81233	6	8	78.5
34	10080582527 = 99787 × 101021	6	8	79.9
35	32508933983 = 116279 × 279577	6	8	78.4
36	66847008023 = 257093 × 260011	7	10	76.9
37	77608649207 = 275207 × 282001	7	10	79.4
38	146640072011 = 355361 × 412651	7	10	76.6
39	290430014011 = 521869 × 556519	7	10	72.5
40	673988410211 = 806051 × 836161	8	12	75.7
41	1458932014483 = 23333 × 62526551	8	12	69.9
42	4286501650531 = 68399 × 62669069	8	12	73
43	8441487583957 = 125789 × 67108313	8	12	72
44	9055216757189 = 133051 × 68058239	9	14	72.3
45	18313358001143 = 257077 × 71236859	9	14	72.9
46	45106724258927 = 5387951 × 8371777	9	14	61.4
47	73986123192697 = 1183277 × 62526461	9	14	69.9
48	220190119273699 = 3191303 × 68996933	10	16	64.5
49	464525190752281 = 6731651 × 69006131	10	16	54.3
50	587912113935781 = 25164673 × 23362597	11	20	52.9
51	1188470172067261 = 63308191 × 18772771	12	24	47.8
52	2397353600295959 = 23973869 × 99998611	13	27	45.6
53	4771078001207339 = 47711291 × 99998929	14	32	42.6
54	10404327652314191 = 99991819 × 104051789	15	34	41.1
55	25548168982321613 = 159519929 × 160156597	16	40	39.1
56	47849463811209517 = 204257069 × 234260993	17	42	37.9
57	95970560348802499 = 236242459 × 406237561	17	42	36.1
58	224139446127173377 = 188583061 × 1188544957	18	47	33.6
59	356722702338263059 = 288557369 × 1236228011	18	47	32.5
60	719399297416802543 = 673780379 × 1067705917	19	52	30.6
61	1568108874908342021 = 836883731 × 1873747591	19	52	30.1
62	2434987204800059537 = 1073773889 × 2267690833	20	54	28.6
63	5953607126904735149 = 8511091 × 699511628639	21	60	27.1
64	11769431398084725929 = 3247483669 × 3624169541	22	64	25.2
65	18446787952707673019 = 536872169 × 34359739651	22	64	24.1
66	39202751625508001520 = 977340276 × 40111671020	22	64	23.8
67	130702184608413767897 = 956460047 × 136652006551	23	77	20.1
68	198945304514186667527 = 192377231 × 1034141636617	23	77	19.8
69	344984029271193203537 = 107459773 × 3210355090469	23	77	18.6
70	660405945185390371301 = 7021574327 × 94053828163	23	77	18.2
71	1976827751084769231491 = 7465547413 × 7465547413	24	82	16.6
72	2575206468538622817667 = 5893260847 × 436974798061	24	82	16.4
73	5203485688790766095689 = 26967937511 × 192950821199	24	82	15.8
74	11993367236676401330641 = 120741372161 × 99331049681	24	82	15.1
75	20130328299259895126191 = 85947645103 × 234216170497	25	88	14.3

Table B6 Experiment results

Scale	Data	Logical qubits	Physical qubits	Succes rate
76	$46608139720297949704307 = 134140447121 \times 347457763267$	25	88	13.8
77	$83661429386261157394423 = 204140136817 \times 409823519719$	25	88	12.4
78	$195723581313196083887639 = 206267979703 \times 948880100513$	25	88	11.7
79	$416177807286551819979989 = 194984774269 \times 2134411821881$	25	88	10.9
80	$1034879359475633166138643 = 1001721172891 \times 1033101213673$	26	94	10.3

References

1 Schnorr CP. Fast factoring integers by SVP algorithms, corrected. Cryptology ePrint Archive, 2021.
2 Yan B, Tan Z, Wei S, et al. Factoring integers with sublinear resources on a superconducting quantum processor. arXiv preprint, 2022, 2212-12372.
3 Hegade N, Solano E. Digitized counter-diabatic quantum factorization. arXiv preprint, 2023, 2301. 11005.