IEEE Transactions on Quantum Engineering

Received 25 November 2021; revised 31 May 2022; accepted 7 June 2022; date of publication 16 June 2022; date of current version 26 July 2022.

Digital Object Identifier 10.1109/TQE.2022.3183385

The Present and Future of Discrete Logarithm Problems on Noisy Quantum Computers

YOSHINORI AONO[1,5], SITONG LIU[2], TOMOKI TANAKA[3,5], SHUMPEI UNO[4,5], RODNEY VAN METER[2,5] (Senior Member, IEEE), NAOYUKI SHINOHARA[1], AND RYO NOJIMA[1]

[1]National Institute of Information and Communications Technology, Tokyo 184-8795, Japan
[2]Graduate School of Media and Governance, Keio University, Fujisawa 252-0882, Japan
[3]Mitsubishi UFJ Financial Group, Inc., Tokyo 100-8388, Japan
[4]Mizuho Research and Technologies, Ltd., Tokyo 101-8443, Japan
[5]Quantum Computing Center, Keio University, Yokohama 223-8522, Japan

Corresponding author: Yoshinori Aono (e-mail: aono@nict.go.jp).

This work was supported by the MEXT Quantum Leap Flagship Program under Grant JPMXS0118067285 and Grant JPMXS0120319794.

ABSTRACT The discrete logarithm problem (DLP) is the basis for several cryptographic primitives. Since Shor's work, it has been known that the DLP can be solved by combining a polynomial-size quantum circuit and a polynomial-time classical postprocessing algorithm. The theoretical result corresponds the situation where a quantum device working with a medium number of qubits of very small errors can solve the DLP. However, all the quantum devices that we can use have a limited number of noisy qubits, as of the noisy intermediate-scale quantum (NISQ) era. Thus, evaluating the instance size that the latest quantum device can solve and giving a future prediction of the size along the progress of quantum devices are emerging research topics. This article contains two proposals to discuss the performance of quantum devices against the DLP in the NISQ era: 1) a quantitative measure based on the success probability of the postprocessing algorithm to determine whether an experiment on a quantum device (or a classical simulator) succeeded; and 2) a procedure to modify bit strings observed from a Shor's circuit to increase the success probability of a lattice-based postprocessing algorithm. In this article, we conducted our experiments with the `ibm_kawasaki` device and discovered that the simplest circuit (7 qubits) from a 2-bit DLP instance achieves a sufficiently high success probability to proclaim the experiment successful. Experiments on another circuit from a slightly harder 2-bit DLP instance, on the other hand, did not succeed, and we determined that reducing the noise level by half is required to achieve a successful experiment. Finally, we give a near-term prediction based on required noise levels to solve some selected small DLPs and integer factoring instances.

INDEX TERMS Discrete logarithm problem (DLP), IBM quantum, lattice, postprocessing method, Shor's algorithm.

I. INTRODUCTION

Since Shor [1] proved that a reasonably large quantum circuit can solve both the integer factoring problem (IFP) and the discrete logarithm problem (DLP) efficiently, many researchers have been discussing its impact and implementability and have been attempting to reduce the attack's resource costs.

An emerging topic in cryptography is to predict when the progress of quantum computers threatens modern cryptosystems. In order to extrapolate accurately from today, we need to understand the following two factors.

1) The projected progress in quantum hardware, in abstract terms. The quantum hardware industry can be said to still be in its infancy, and the introduction of disruptive new systems remains possible, but several companies have published (and later updated) roadmaps for the evolution of their systems in coming calendar years [2]–[6]. In this article, we use these roadmaps, but our focus is not on assessing or extending them.
2) The relationship between that abstract performance and the ability of the machine to solve specific problem instances. To establish the hardest problem the

TABLE 1. Summary of Parameters for Target DLP Instances

	DLP instance	DLP bits n_F	n_x	n_y	#Q	#CX
O	$2^z = 1 \mod 3$	2	2	2	6	6
I			3	2	7	15
II	$2^z = 2 \mod 3$		3	2	7	32
III			3	3	8	38
IV	$4^z = 2 \mod 7$	3	3	3	9	179
V	$3^z = 4 \mod 7$		4	4	11	255
VI			6	6	15	-

Instance O is used only for comparison of circuit size to the factoring in Section I-B. Instances I–III are used for experiments both in real device and classical simulation. Instances IV–VI are used for classical simulation and future prediction. n_F, n_x, and n_y refer to the number of qubits in the corresponding registers in the circuit in Fig. 1. #Q and #CX are the number of qubits and CNOT gates, respectively.

system can solve, we must also quantify what it means for a quantum computer to "solve" a cryptographic problem.

A. SUMMARY OF OUR CONTRIBUTION

1) QUANTITATIVELY DEFINE A SUCCESSFUL EXPERIMENT

We propose a formalization of success probability, including circuit generation, quantum device execution, and postprocessing. After fixing a problem instance and a quantum circuit, execution on a quantum device outputs a set of bit strings. The postprocessing algorithm, then, takes this set of bit strings as input and returns a set of candidate solutions to the problem instance. The success probability is defined by the probability that the set of candidates contains the desired solution. This success probability can be defined on bit strings from the ideal device (a noiseless device for a quantum circuit, e.g., for small instances, as simulated by a classical computer), a noisy device (real or simulated), and a virtual device that outputs uniformly random bit strings (corresponding to randomly guessed solutions). Note that Shor's algorithm, like many other quantum algorithms, is a probabilistic algorithm even when executed on an ideal device. Thus, by measuring where the quantum device under test lies between the ideal and the uniform devices, we can assess the performance. We propose the following definition:

> A quantum device is said to be able to successfully solve a problem when its success probability is greater than the mean of the success probabilities of an ideal device and a uniformly random device.

2) PRESENT DLP EXPERIMENTS ON AN IBM QUANTUM DEVICE

Table 1 is a summary of the DLP instances and Shor's quantum circuits (see the block diagram in Fig. 1) that we considered in this article. The DLP over a field and the corresponding circuits are outlined in Section II-A and II-C, respectively.

For each DLP instance $g^z = a \pmod p$, the size of the problem n_F is $\lceil \log_2 p \rceil$, i.e., the number of bits to represent an element of the prime field of size p. Each quantum circuit computes the superposition of $|x, y, F(x, y)\rangle$ over all the integers $x = 0, \ldots, 2^{n_x} - 1$ and $y = 0, \ldots, 2^{n_y} - 1$. Here, $F(x, y)$ has periods from which the solution can be recovered. n_x and n_y define the widths of the exponent variables, which are equal to the size of the quantum Fourier transform (QFT) gadgets. $\#Q = n_F + n_x + n_y$ is the number of qubits used in the circuit. #CX is the number of CNOT gates after optimization by the Qiskit `transpile()` command with optimization level 3; because the optimization phase stochastically searches a large solution space, we take the minimum value of 100 calls for each circuit.

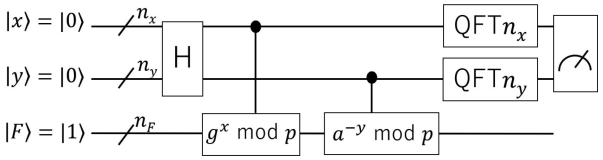


FIGURE 1. Overview of Shor's circuit for solving the DLP comprising Hadamard gates, two controlled modular exponentiations, and two QFTs.

We use the framework described above to furnish some data points for future prediction of the security of the DLP over a finite field against quantum computers. We present results from experiments, in which we used an IBM Quantum device to solve selected 2-bit instances realized by 7- and 8-qubit circuits. More precisely, by using the `ibm_kawasaki` device of quantum volume (QV) = 32, we observed that the simplest DLP instance $2^z \equiv 1 \mod 3$ with the smallest quantum circuit (instance I in Table 1 and Fig. 5) outputs meaningful bit strings and gives success probability higher than the threshold defined above.

We also experimented with two slightly more complicated circuits for the instance $2^z \equiv 2 \mod 3$ (instances II and III). These circuits generate bit strings that give us only a low success probability. To improve the success probability, we propose a simple algorithm for modifying the output bit strings from the devices. Via simulation, we found that the device success probability of the instance III circuit is slightly below the threshold defined above and that reducing the noise by half is required to claim success.

3) PROJECT NEAR-FUTURE DLP SUCCESSES

Via simulation, we also predict how much the noise level needs to be reduced to solve the larger instances $4^z \equiv 2 \mod 7$ and $3^z \equiv 4 \mod 7$ (instances IV and V). We found that advancing from the ability to solve instance III to the ability to solve instance IV requires reducing the noise level by about one decimal order of magnitude. Table 3 summarizes the results. With the several-year trend of reducing averaged CNOT gate errors by a factor of 2 each year [7], instances IV and V are expected to be solved within the next five years. Solving larger instances of the DLP will require better quantum devices and perhaps additional techniques, such as quantum error correction (QEC).

B. RELATED WORK

1) DLP AND ITS APPLICATIONS

The DLP over a finite field is a computational problem believed for a long time to be classically hard [8]. It is used as the basis for a prominent set of digital signature schemes [9]. Researchers from various areas, including quantum computing and classical cryptography, are investigating how resilient the schemes are against quantum computers and when they become compromised.

Besides the DLP over a finite field, the discrete logarithm problem over elliptic curves (ECDLP) [10] has also been used as a security base on some cryptographic systems. Owing to the intractability of the group operations on elliptic curves, no experiments to solve the ECDLP have been reported to the best of our knowledge. Finding the smallest instance of the ECDLP that is executable on quantum devices and experiment is a challenging open problem as of 2022.

2) EXPERIMENTS ON SHOR'S ALGORITHM ON QUANTUM DEVICES

Many experiments that execute Shor's factoring circuits (and its subcircuits) have been performed [11]–[14] by using several quantum devices. The latest record of factoring by a quantum circuit is $21 = 3 \times 7$ by Amico *et al.* [13] by using the `ibmqx5` device, and they also reported that 35 is infeasible. Recently, Skosana and Tame [14] have also reported their experiment on factoring 21.

On the other hand, to the best of our knowledge, no experiments on the DLP have been reported, even though the circuit construction is very similar. We discovered that the DLP is better suited for quantum benchmarking experiments in the noisy intermediate-scale quantum (NISQ) era because some circuits of the DLP can be simpler than the circuit to factor 15, which is the simplest factoring circuit. For instance, the simplest circuit to factor 15 (see Fig. 15 in Appendix F) must have 8 qubits because the registers required to hold both the modulo power and the modulo residue must be larger than $\log_2(15)$ qubits to output a meaningful result for input to the postprocessing stage. On the other hand, the DLP instance $2^z \equiv 1 \pmod 3$ can be achieved in a minimum of 6 qubits (Instance O in Table 1). In addition, the number of CNOT gates after compilation ("transpile" in Qiskit lingo) is only six in the DLP case, while it is about 85 (see Table 4) in the factoring case.

Most of the existing experiments have employed folklore techniques, which are used in logical circuit optimization, to simplify the quantum circuits. Because the latest quantum devices may not be able to execute modular arithmetic with sufficient accuracy [15], our circuit implementations are also simplified from the full implementation. Following this simplifying policy, we carefully designed our circuit so that no information on the solution was used, as it was claimed that some experimental circuits were oversimplified by using problem solution information [16]. Our circuits are made up of shift operations over qubits with no auxiliary bits. Details are explained in Section IV-A.

It is also necessary to follow the policy when we construct a postprocessing algorithm that generates a set of candidate solutions by using bit strings from quantum observations. We also carefully designed our modification algorithm in Section IV-C that transforms an observed bit string to another bit string if it is necessary.

3) QUANTIFICATION OF EXPERIMENTAL RESULTS

It has been a long-standing problem to claim that a quantum computation experiment has succeeded or failed quantitatively. Several quantifier functions have been used to estimate the quality of outputs from quantum devices.

Satoh *et al.* [17] used the Kullback–Leibler (KL) divergence. Cross *et al.* [18] used the probability that the output bit strings are in the heavy output set. The linear cross-entropy benchmark was used to claim the demonstration of quantum supremacy by Arute *et al.* [19], although Barak *et al.* [20] later claimed that their result can be approximated by a polynomial time classical computer.

These works share a spirit that compares distances between the ideal outputs, device outputs, and uniform random using their chosen distance functions. The device output probability distribution P_{Dev} will be the same as the ideal output distribution P_{Ideal} if the device is free of any kind of noise and will be the same as the uniform output distribution P_{Unif} if the device output is completely dominated by random noise. On the abstract level, P_{Dev} should lie between P_{Ideal} and P_{Unif}, and a suitable distance function to measure the difference between two probability distributions can define the position of P_{Dev} in a quantitative way.

In other words, we can score the device output by a real number, usually between 0 and 1, via an appropriate distance function and a scaling. The existing works use the following criteria to claim their success by using a score to quantify the device's outputs.

First, fix the quantum circuit and some function d to quantify the difference between two probability distributions. Here, d need not be a distance metric, but it should satisfy conditions typically called premetric: $d(x, x) = 0$ and $d(x, y) \geq 0$ for legitimate inputs x and y. Then, with the function, define a score for the device output by

$$s_1 := 1 - \frac{d\left(P_{\text{Dev}}, P_{\text{Ideal}}\right)}{d\left(P_{\text{Unif}}, P_{\text{Ideal}}\right)} \tag{1}$$

which captures Satoh *et al.*'s [17] KL-divergence-based measure. The score is close to 1 if the device output is good.

Assuming that the triangle inequality $d(P_{\text{Unif}}, P_{\text{Ideal}}) \leq d(P_{\text{Unif}}, P_{\text{Dev}}) + d(P_{\text{Dev}}, P_{\text{Ideal}})$ holds, we have the following relation:

$$s_2 := \frac{d\left(P_{\text{Unif}}, P_{\text{Dev}}\right)}{d\left(P_{\text{Unif}}, P_{\text{Dev}}\right) + d\left(P_{\text{Dev}}, P_{\text{Ideal}}\right)} \leq s_1. \tag{2}$$

Thus, for any threshold t, the condition $s_2 \geq t$ is stronger than the condition $s_1 \geq t$. Therefore, we should use s_2 to claim that an experiment is successful.

However, it is useless to conduct experiments with a slightly large number of qubits. Since about $2^{\#Q}$ samples of bit strings are needed to obtain a good approximation of P_X, the number of samples grows quickly even with a modest increase in $\#Q$. This motivates us to employ a distance function $d(P_X, P_Y) := |p_X - p_Y|, X, Y \in \{\text{Ideal, Dev, Unif}\}$, capturing the difference between some probabilities. Typically, on a set of bit strings, p_X is defined by the probability that the output of a device X satisfies the condition.

Assuming $p_{\text{Unif}} \leq p_{\text{Device}} \leq p_{\text{Ideal}}$, we can show that

$$s_2 = \frac{p_{\text{Dev}} - p_{\text{Unif}}}{p_{\text{Ideal}} - p_{\text{Unif}}}$$

and

$$s_2 \geq 0.5 \Leftrightarrow p_{\text{Dev}} \geq \frac{p_{\text{Ideal}} + p_{\text{Unif}}}{2}. \tag{3}$$

The condition with $s_2 = 0.5$ means that if the probability of the device is larger than the median of the probabilities of the ideal and the uniform, then one can claim success. We call this the "median principle." The condition is used to measure the QV by Cross *et al.* [18, (6)]. In addition, this s_2 is equivalent to the fidelity of the cross-entropy benchmark introduced by Arute *et al.* [19].

In the spirit of this previous work, we define our threshold for success, explicitly given in (8) in Section III-B, by following the form of (2). However, when we simply applied the existing framework to Shor's DLP algorithm, we discovered the following issue.

A typical condition for success on DLP computation should be defined over multiple vectors since the postprocessing algorithm must take at least two bit strings from a device. However, the existing success criteria are defined by using a distribution over a single bit string. This is another reason that we chose the success probability p_{Dev} based on the outputs from the postprocessing algorithm rather than some distance between probability distributions. Our criteria as of now are a basic version, and we expect that other researchers will update them according to their own needs.

Another way for quantification has been proposed. The square of the statistical overlap was introduced by Monz *et al.* [12] to measure the similarity between the ideal and device outputs. It is also used by Amico *et al.* [13] to claim their advantage from previous works.

C. ARTICLE ORGANIZATION

In Section II, we give a theoretical introduction to DLP, lattices, and an overview of Shor's algorithm for solving the DLP and a computational problem to recover the solution. In Section III, we define our discussion framework, including circuit generation, device execution, and postprocessing. In addition, we define a quantitative method for determining whether an experiment has succeeded or failed. Section IV introduces the modular-exponentiation gadgets used in our experiments and our lattice-based postprocessing algorithm. (Appendix A contains background theory on this postprocessing algorithm.) Section V gives our experimental results on IBM Quantum. Section VI gives simulation results of noisy quantum devices and comparison with the real device. Finally, Section VII concludes this article.

II. PRELIMINARIES

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ are the set of natural numbers, integers, rational numbers, and real numbers, respectively. For a prime number p, $\mathbb{Z}_p = \{0, \ldots, p-1\}$ is the field under modulo p. $[n]$ denotes the set $\{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. $\text{Ball}_K(\boldsymbol{x}, \rho)$ is the Euclidean ball of radius $\rho > 0$ with center $\boldsymbol{x} \in \mathbb{R}^K$, and $V_K(\rho)$ denotes its volume.

The KL divergence defined over two discrete probability distributions P and Q is

$$D_{\text{KL}}(P||Q) := \sum_{i=1}^{N} P(x_i) \log \frac{P(x_i)}{Q(x_i)}$$

where $P(x_i)$ and $Q(x_i)$ are the probability densities at x_i. It is used to measure a "distance" from P to Q, though it is asymmetric and the triangle inequality does not hold. Note that $D_{\text{KL}}(P||Q) = 0$, if and only if $P = Q$.

A. DLP OVER A FIELD

The DLP considered in this article is the version defined over a prime field $\mathbb{Z}_p$. An instance of the DLP is given by a tuple $(g, a, p) \in \mathbb{N}^3$ that represents the equation

$$g^z \equiv a \pmod{p} \tag{4}$$

where the problem is to find $z \in \mathbb{Z}_p$. Here, g is assumed to be a generator under modulo p, that is, it satisfies $g^n = 1 \pmod{p}$ for $n = p-1$ and $\neq 1$ for any $n \in [p-2]$.

A variant of the DLP where one has extra information, e.g., the upper bound of z, has been considered in a cryptographic context, and it significantly reduces the classical complexity to solve the problem [21], [22]. In this article, we assume that no information is provided except for the DLP instance.

B. LATTICES

We provide a brief overview of the lattices used in the analysis of Shor's algorithm on the DLP. Bremner's textbook [23] provides a gentle introduction. The use of lattices in postprocessing to solve the DLP by Shor's algorithm is also discussed in [22] and [24].

For a sequence of (not necessarily independent) vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K \in \mathbb{Q}^m$, the lattice spanned by them is defined by the set

$$L(B) := \left\{ \sum_{i=1}^{K} a_i \boldsymbol{b}_i : \forall i, a_i \in \mathbb{Z} \right\}$$

where $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K$ are called the basis vectors. A vector in $L(B)$ is represented by row vectors. We use the matrix $B :=$

IEEE Transactions on Quantum Engineering

$[\boldsymbol{b}_1^T, \ldots, \boldsymbol{b}_K^T]^T$ to represent the basis in this article. In particular, we say that the lattice is full rank if $K = m$ and the vectors are all independent.

A point $\boldsymbol{x} \in \mathbb{Q}^m$ is called a lattice point if $\boldsymbol{x} \in L(B)$. Many useful lattice algorithms take a matrix representation of a lattice basis as input. The majority of them assume that an input basis is a set of independent vectors, though nonindependent bases are occasionally used in applications. There is an efficient algorithm for converting a nonindependent basis to an independent basis that spans the same lattice (see [23, Sec. 6] or [25, Sec. 2.6.4]). We assume the existence of such an algorithm in the postprocessing of Shor's algorithm.

For a given basis, the fundamental region is defined by

$$P(B) = \left\{ \sum_{i=1}^{K} \alpha_i \boldsymbol{b}_i : \forall i, \alpha_i \in [0, 1) \right\}$$

and the covolume covol(L) of a lattice is defined by the volume of the region. Both the lattice and its fundamental region are subsets of

$$\mathrm{span}(B) := \left\{ \sum_{i=1}^{K} \alpha_i \boldsymbol{b}_i : \forall i, \alpha_i \in \mathbb{R} \right\}.$$

Note that covol(L) is efficiently computable from the basis if the basis vectors are independent, since it is equal to $\prod_{i=1}^{K} \|\boldsymbol{b}_i^*\|$, where $\boldsymbol{b}_i^*$ is the ith vector in the Gram–Schmidt basis.

The Gaussian heuristic of lattices in the context of this article claims that for an n-dimensional full-rank lattice L and a ball Y, the number of lattice points in Y is expected to be $\mathrm{vol}(Y)/\mathrm{covol}(L)$. We should note that the original version of the Gaussian heuristic [26] argued for a probabilistic distribution over random lattices, whereas the lattices discussed in the postprocessing of Shor's algorithm may not be close enough to random. Appendix C contains experimental evidence in small dimensions.

For a lattice L, its dual lattice $L^\times$ is defined by the set

$$L^\times := \{\boldsymbol{x} \in \mathrm{span}(L) : \forall \boldsymbol{v} \in L, \langle \boldsymbol{x}, \boldsymbol{v} \rangle \in \mathbb{Z}\}$$

where $\langle \boldsymbol{x}, \boldsymbol{v} \rangle$ denotes the standard Euclidean inner product. A basis of $L^\times$ is explicitly given as $B(B^T B)^{-1}$, where B is a basis of the primal lattice L. It is simplified by $(B^T)^{-1}$ if B is a square matrix.

For a given lattice basis B in the m-dimensional space and a target vector $\boldsymbol{y} \in \mathbb{Q}^m$, the closest vector problem (CVP) is the computational problem to find a lattice point $\boldsymbol{x} \in L$ that minimizes $\|\boldsymbol{x} - \boldsymbol{y}\|$. Any lattice point is allowed if there are many solutions to minimize. A similar problem is the bounded distance decoding (BDD) problem. For a problem instance $(B, \boldsymbol{y})$ and a bound $\rho > 0$, the goal is to find all the lattice points $\boldsymbol{x} \in L$ such that $\|\boldsymbol{x} - \boldsymbol{y}\| \leq \rho$. If there is no vector in the ball, $\mathrm{Ball}_m(\boldsymbol{y}, \rho)$, the empty symbol $\perp$ is returned. We should point out that the version of the BDD presented above differs significantly from the standard one. In the typical situation of the BDD, it is assumed that a unique solution $\boldsymbol{x}$ exists.

C. OVERVIEW OF SHOR'S ALGORITHM IN OUR FRAMEWORK

Fix a DLP instance (4) represented by $I = (g, a, p) \in \mathbb{N}^3$. The parameters that specify the circuit size are $\mathtt{param} = (n_x, n_y)$. They are used to define the size of the two QFTs. Let $N_x := 2^{n_x}$ and $N_y := 2^{n_y}$. n_F is the size of the DLP instance in bits, i.e., we set $n_F = \lceil \log_2 p \rceil$. We assume that $n_x, n_y \geq n_F$. Note that $n_x, n_y > 2n_F$ is necessary in theory; however, the computing device we used cannot operate the necessary number of qubits because the fidelity of quantum operations is not high enough to get meaningful results. Thus, we will use smaller numbers $n_x, n_y \approx n_F$ as in Table 1.

An overview of the quantum circuit for solving the DLP is shown in Fig. 1. We do not consider the serializing implementations [13], [27] that can reduce the number of qubits in the exponent part $n_x + n_y$ to one, because our experimental environment (see Section 5-A) does not fully support such an implementation.

It is easy to see that the bivariate function $F(x, y) := g^x a^{-y} \bmod p$ has two periods $(p, 0)$ and $(z, 1)$, where z is the desired DLP solution. Thus, finding the periods will reveal the solution. The circuit is designed to compute the superposition of $\sum_{x,y} |x, y, F(x, y)\rangle$ over the box $[0, N_x - 1] \times [0, N_y - 1]$ and apply the QFT for finding the periods.

The ideal state that assumes no quantum error, which corresponds to the distribution P_{ideal} in our framework, is computed as follows. The initial state $|x, y\rangle|F\rangle = |0, 0\rangle|1\rangle$ is spread by the Hadamard gates

$$|0, 0\rangle|1\rangle \to \sum_{x=0,y=0}^{N_x-1,N_y-1} |x, y\rangle|1\rangle \quad \text{(ignoring normalization)}.$$

Then, the modular-exponentiation circuit block sets the superposition

$$\to \sum_{x=0,y=0}^{N_x-1,N_y-1} |x, y\rangle|g^x a^{-y} \bmod p\rangle.$$

Finally, by applying the QFT circuit block over $|x\rangle$ and $|y\rangle$, the state to be measured is

$$\longrightarrow \sum_{x=0,y=0}^{N_x-1,N_y-1} \sum_{k,\ell}^{N_x-1,N_y-1} e^{2\pi i \cdot \left(\frac{kx}{N_x} + \frac{\ell y}{N_y}\right)} |k, \ell\rangle|g^x a^{-y} \bmod p\rangle. \tag{5}$$

An observed bit string $\boldsymbol{s}$ is represented as a pair (k, ℓ) of integers within the box $[0, 0] \times [N_x - 1, N_y - 1]$. It is also interpreted as the point in $[0, 1) \times [0, 1)$

$$\boldsymbol{p} := (p, r) = \left(\frac{k}{N_x}, \frac{\ell}{N_y}\right). \tag{6}$$

1) COMPUTATIONAL PROBLEM IN POSTPROCESSING

A postprocessing algorithm tries to generate a set of candidates $z_1, \ldots, z_J$ as proposed solutions to the instance I. There could be several strategies for recovering the candidates.

Formally, the computational problem we have to consider after the quantum observation is as follows. The detailed derivation will be provided in Appendix A.

Problem 1: Let L be the 2-D integer lattice spanned by $(p-1, 0)$ and $(z, 1)$, where p is known and z is unknown. $\left\{\left(\frac{k_i}{N_x}, \frac{\ell_i}{N_y}\right)\right\}_{i=1,\ldots,K}$ are the points from the observations, which are (noisy) approximations of lattice points in $L^{\times}$. Then, find (a suitable approximation of) z.

We remark that it is enough to assume the existence of an algorithm that finds an approximation of z, say that the algorithm recovers z' s.t. $|z - z'| < z/4$, where the notation $|\cdot|$ is considered under modulo p. With the approximated solution, consider the new DLP instance $(g, a \cdot g^{-z'}, p)$, which has a smaller solution $z - z'$. The approximation algorithm returns z'' s.t. $|z - z' - z''| < |z - z'|/4 < z/16$. Repeating this process, we can recover the desired z.

An observed point is shifted from a lattice point by two factors. The first is due to the finiteness of the QFT, which is represented by a closed but difficult to analyze formula precisely described by (14) in Appendix A. The other is caused by noise in quantum devices. If the quantum circuit works without any quantum noise, we know that the observed points are close to lattice points with a high probability [24], [28].

III. GENERIC FRAMEWORK AND SUCCESS PROBABILITY

We revisit the standard procedures of computation by using quantum devices and propose our definition of the success of experiments and success probability. Note that our framework can be applied for any NP problem besides the IFP and the DLP, because it is based on a check for whether the given solution candidates satisfy the problem instance.

A. THREE-STEP FRAMEWORK

Our standard procedure to solve a computational problem using a quantum computer has the following three steps.

1) Generate a quantum circuit from a given problem instance.
2) Execute the circuit on a quantum device.
3) Recover solution candidates and check.

In the NISQ era, steps 1 and 3 are assumed to be classical (probabilistic polynomial) algorithms, which we may call preprocessing and postprocessing, respectively. If the performance of quantum devices improves, they can be replaced by quantum algorithms. However, we think considering this change is pointless for the time being, and we have left it as an open problem for the future.

Furthermore, we assume that the quantum circuit generated in step 1 is optimized to solve the given problem instance rather than being designed to solve generic instances of a fixed size. Generally speaking, the classical input values for quantum algorithms are expressed via classical choice of quantum gates to execute, rather than being encoded in quantum registers. This approach has the potential to simplify the circuit significantly and reduce resource consumption compared to a circuit designed for generic instances.

More formally, we write the algorithms using the notation shown below.

1) CircuitGen(I, `param`) $\to QC$: It is a probabilistic algorithm that for given a problem instance I and auxiliary information `param`, such as the qubit size of the circuit, outputs a quantum circuit QC. The output can be different each time.
2) Device(QC) $\to \boldsymbol{s}$: Execute the circuit QC by a quantum device and get a bit string $\boldsymbol{s}$. Here, a device is considered to be either a real quantum device or a quantum simulator on a classical computer.
3) PostProcess($\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$; I, `param`) $\to Z$: It is a probabilistic algorithm that for given bit strings, problem instance I, and `param`, outputs a set of solution candidates $Z = \{z_1, \ldots, z_J\}$ to I. Here, the numbers J, K of solution candidates and bit strings are not fixed but are assumed to be a polynomial of the instance size.

Let us add some remarks on the postprocessing step. The step could be further divided into two steps: 1) modification of bit strings; and 2) recovering a solution. The latter will be discussed as a lattice-based algorithm in Section IV-B. The former attempts to transform a bit string into a better one (a simple method will be presented in Section IV-C) or to modify a probabilistic distribution. One well-known technique for improving probability distributions is error mitigation [29], [30], which recovers a probabilistic distribution from an approximated distribution of bit strings derived from many shots and some information on the error distributions derived from additional experiments. We do not consider the modification of the probabilistic distribution because our preliminary experiments using error mitigation do not significantly change the success probability. In addition, in the situation where we want to solve a large DLP instance, execution costs mean that it will be feasible to execute only a few shots so that we cannot estimate the distribution. Therefore, in the cryptographic context, we only consider the modification of bit strings.

B. DEFINING SUCCESS PROBABILITIES

Within the framework in the above section, we can define "success" for the experiments using real devices. We start the discussion by introducing the devices that we will compare.

1) *Ideal:* This outputs the bit strings from the noiseless quantum circuit.
2) *Dev:* This is a real quantum device to be tested.
3) *Unif:* This outputs the random bit strings uniformly.

IEEE Transactions on Quantum Engineering

Definition 1: Fix a postprocessing algorithm; in particular, the number of input bit strings K is fixed. For a device $X \in \{\text{Ideal, Dev, Unif}\}$, we denote by P_X the probability distribution over the bit strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$ from K executions of the circuit $QC_i = \text{CircuitGen}(I, \texttt{param})$ on the device X. The subscript i denotes the ith execution, where we can usually reuse the same circuit for each execution. The probability includes the circuit generator's random coins and superposition in the device.

The success probability on the computational problem is defined by the probability that the set of output candidates contains the desired solution to I

$$p_X := \Pr\left[z_{\text{sol}} \in \text{PostProcess}\left(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K; I, \texttt{param}\right)\right]. \quad (7)$$

Fixing the generator and postprocessing algorithm with the number of input bit strings, the success probabilities p_{Ideal}, p_{Dev}, and p_{Unif} are fixed.

If a program can be executed without any noise on a device, the output distribution and the success probability are expected to be the same as the ideal. Increasing the quantum noise of execution, p_{Dev} drops. In addition, if the bit strings approach uniform noise, p_{Dev} approaches to p_{Unif}. Thus, we can expect that $p_{\text{Unif}} < p_{\text{Dev}} < p_{\text{Ideal}}$ and the scaled value $s_2 = (p_{\text{Dev}} - p_{\text{Unif}})/(p_{\text{Ideal}} - p_{\text{Unif}}) \in (0, 1)$ in (2) can be used to measure the performance of devices.

Following the common strategy of the existing works that we named the median principle (described in Section I-B), we say that the device succeeded in solving the problem if $s_2 \geq 0.5$. In other words, we can say that the device experiment to solve a problem instance is a success if

$$p_{\text{Dev}} > \frac{p_{\text{Ideal}} + p_{\text{Unif}}}{2}. \quad (8)$$

We observe that the above definition of success, which we may call "the success of device experiments as a computing algorithm," is somewhat disconnected from the cryptographic context in the real world. In the cryptographic area, an attacker can claim success if there exists a trial such that $z_{\text{sol}} \in \text{PostProcess}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K; I, \texttt{param})$ even though superpolynomial time was wasted.

For small instances considered in the NISQ era, classical simulations to compute the accurate value of p_{Ideal} and p_{Unif} are possible, and the median principle is a good criterion for checking whether the device performance is adequate. For large quantum circuits, we may assume $p_{\text{Unif}} \approx 0$ since the number of candidates is polynomial despite the search space being exponential. For the DLP [28, Sec. 4], we can assume the success probability $p_{\text{Ideal}} \approx 1$ for sufficiently large instances. The above assumptions deduce the threshold 0.5. It turns out that some lazy version of the median principle for a large-scale device is smoothly connected with the exact version.

C. SUCCESS PROBABILITY AND KL DIVERGENCE

We give an experimental motivation to use the success probabilities to measure the difference between P_X and P_Y rather than the KL divergence. We simulated the probability density functions of ideal and noisy execution for the quantum circuit instance VI, which has the exponent widths $n_x = n_y = 6$. Thus, each simulation of a shot generates $n_x + n_y = 12$ bits.

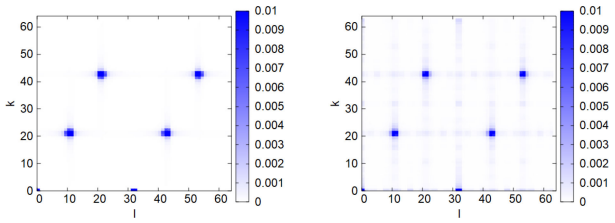


FIGURE 2. Density function from the exact simulation P_{Ideal} and the noisy simulation P_{Noisy} with parameters $p_1 = 0.0003$ and $p_2 = 0.003$ that output similar data. Although we might think the device works well on first impression, the asymmetry of the KL divergence drives us to question the results.

In this case, we set the noise parameters to have a 2-bit gate depolarizing error $p_2 = 0.003$ and a single-bit gate depolarizing error $p_1 = 0.1 \cdot p_2$. Fig. 2 depicts a comparison of the ideal distribution and the noisy distribution. The horizontal and vertical axes represent the position of (ℓ, k) as determined by 12 bits from a simulation of one shot.

Although we might think the virtual noisy device works well on first impression, the following KL divergence metrics show that it is questionable. The concrete values are $D_{\text{KL}}(P_{\text{Ideal}}, P_{\text{Noisy}}) = 1.022$ and $D_{\text{KL}}(P_{\text{Noisy}}, P_{\text{Unif}}) = 2.881$. Thus, P_{Noisy} is closer to the ideal than the uniformly random, and we can conclude that the device output is good.

On the other hand, the values of the reverse KL divergences are $D_{\text{KL}}(P_{\text{Unif}}, P_{\text{Noisy}}) = 1.745$ and $D_{\text{KL}}(P_{\text{Noisy}}, P_{\text{Ideal}}) = 3.305$, which are evidence that the device output is closer to the uniform than the ideal distribution. Hence, we can obtain contradicting results to each other.

From the result, we think that a naïve comparison of KL divergence is not suitable to decide the success of experiments. Furthermore, our goal is not only to benchmark quantum devices but also to measure the device performance as an accelerator for solving the DLP. This is why we proposed using the overall success probability to determine whether an experiment is succeed, as in (8).

IV. OUR IMPLEMENTATION

In Section II-C, we omitted the details of our modular-exponentiation arithmetic gadgets and postprocessing algorithm to generate solution candidates to Problem 1. Many implementations have been proposed for their building blocks. In this section, we give the details of our version of the implementation used in our experiments.

A. OUR QUANTUM MODULAR-EXPONENTIATION CIRCUIT

The modular-exponentiation gadgets are the most complicated part of Shor's circuit. They are typically implemented by a sequence of modular-multiplication operations.

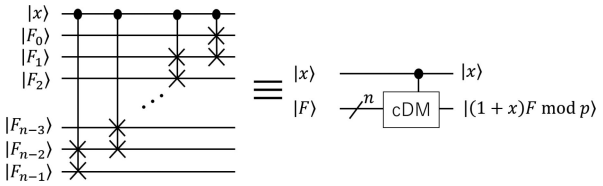


FIGURE 3. ***n*-bit double then modular quantum circuit without any auxiliary qubits.**

Based on our understanding of the latest hardware, the general modular-multiplication circuit, even for problems involving only a few qubits, is too demanding for the latest generation of quantum machines [15]. As a result, we must consider specific cases and simplify the modular-multiplication circuit as much as possible by the following existing reports.

We used the standard binary representation of nonnegative integers. For a state $|F_{n-1}\cdots F_0\rangle$ comprising n qubits, we regard it as the integer $F = F_{n-1}2^{n-1} + \cdots + F_0$ and denote it as $|F\rangle$. To simplify the circuits, we exploited the fact that the left rotate operation $|F_{n-1}\cdots F_0\rangle \to |F_{n-2}\cdots F_0F_{n-1}\rangle$ computes the function $F \mapsto 2F \bmod (2^n - 1)$ [12] without using any auxiliary qubits. A stack of these rotation circuits realizes the modulus power function $F \mapsto F \cdot 2^k \bmod (2^n - 1)$ for any k.

With this gadget, the controlled version of double-then-modular operation is expressed as

$$|x\rangle|F\rangle \to |x\rangle|(1+x)F \bmod (2^n-1)\rangle = \begin{cases} |F_{n-1}\cdots F_0\rangle & (x=0) \\ |F_{n-2}\cdots F_0F_{n-1}\rangle & (x=1) \end{cases}$$

whose circuit is shown in Fig. 3.

In the modular exponentiation, we considered quantum circuits that use only NOT and CNOT gates and the above gadget. In particular, such simplified circuits meet the following requirements: 1) p is a number of the form $2^n - 1$ (p should be prime in a cryptographic context); and 2) both $g^{2^j} \bmod p$ for $j \geq 1$ and $(a^{-1})^{2^k} \bmod p$ for $k \geq 0$ are represented by a power of 2.

Thus, the modular-exponentiation part that computes $|x, y, 1\rangle \to |x, y, g^x a^{-y} \bmod p\rangle$ comprises of $|x, y, 1\rangle \to |x, y, g \cdot x_0\rangle$ (implemented by NOT and CNOT gates, where x_0 is the least significant bit of x), and the controlled double-then-modular gadgets.

An illustrative example is, for instance, $3^z \equiv 4 \pmod 7$ with $n_x = n_y = 4$ (instance V in Table 1). By the relations $3^2 \equiv 3^8 \equiv 2 \pmod 7$, $3^4 \equiv 4 \pmod 7$, $4^{-1} \equiv 4^{-4} \equiv 2 \pmod 7$, and $4^{-2} \equiv 4^{-8} \equiv 4 \pmod 7$, the following calculation is derived:

$$g^x a^{-y} \bmod p = 3^{x_0} \cdot 2^{x_1} \cdot 4^{x_2} \cdot 2^{x_3} \cdot 2^{y_0} \cdot 4^{y_1} \cdot 2^{y_2} \cdot 4^{y_3} \bmod 7.$$

Thus, the computing circuit is started by $|x, 1\rangle \to |x, 3^{x_0}\rangle$, which is implemented by two CNOT gates, and the other parts are double-then-modular gadgets.

Algorithm 1: Our Postprocessing Algorithm for DLP.

Input: $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$: nonzero bit strings from a device.
Output: $Z = \{z_1, \ldots, z_J\}$: a set of solution candidates for the DLP instance

1: Convert $\boldsymbol{s}_i$ to (p_i, r_i) for $i = 1, \ldots, K$
2: Construct the BDD instance $(B, \boldsymbol{y}, \rho_K)$ in (9)–(11)
3: $BDD(B, \boldsymbol{y}, \rho_K) \to V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_J\}$
4: **if** $V = \phi$ **then**
5: $\quad CVP(B, \boldsymbol{y}) \to V = \{\boldsymbol{v}_1\}$
6: **end if**
7: Recover solution candidates z_i from $\boldsymbol{v}_i$ for $i = 1, \ldots, J$
8: **return** $z_1, \ldots, z_J$

B. OUR POSTPROCESSING

We present our version of a classical algorithm for recovering candidates of z via dual lattices, whose naïve extension would be useful in many situations involving Shor-type algorithms. See, for example, [22], [24], and [31] for details on the algorithm that employs primal lattices. In theoretical analysis, we assume, as in many previous works, that there is no gate or measurement noise during algorithm execution.

We outline the algorithm in Algorithm 1. For the input of a sequence of bit strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$, the algorithm outputs a set of solution candidates $z_1, \ldots, z_J$ to the DLP instance. As we described in Appendix A, the bit strings correspond to the points $\{(p_i, r_i)\}_{i=1,\ldots,K}$, which are approximations of points in $L^\times \cap [0, 1)^2$, where $L^\times$ is the dual lattice of L spanned by $(p - 1, 0)$ and $(z, 1)$ with the DLP solution z.

In addition, the lattice defined by the $(K + 1) \times K$ matrix

$$B := \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \vdots \\ \boldsymbol{b}_{K+1} \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & \cdots & p_K \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{9}$$

has a lattice point $\boldsymbol{x} = \sum_{i=1}^{K+1} a_i \boldsymbol{b}_i$ close to the target vector

$$\boldsymbol{y} = (r_1, r_2, \ldots, r_K). \tag{10}$$

Finding $\boldsymbol{x}$, we can recover the DLP candidate solution z since the combination coefficient a_1 is $-z$. The details of the postprocessing are given in Appendix A.

Therefore, for a given point set $\{(p_i, r_i)\}_{i=1,\ldots,K}$, executing a BDD subroutine $\mathrm{BDD}(B, \boldsymbol{y}, \rho_K)$ generates a list of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_J$, and the corresponding combination coefficients derive the candidate set $z_1, \ldots, z_J$. The parameters in the BDD subroutine are B and $\boldsymbol{y}$, which are defined above. The remainder of this section is concerned with the selection of the searching radius ρ_K.

We need to set the radius so that the overall success probability is sufficiently high while keeping the computing time feasible. In other words, we need to keep the number of lattice points in $\mathrm{Ball}_K(\boldsymbol{y}, \rho_K)$ small, namely, $O(K)$.

IEEE Transactions on Quantum Engineering

To bound the number, the Gaussian heuristic assumption provides us a good estimate in general. The assumption claims that the number of lattice points in $\mathrm{Ball}_K(\boldsymbol{y}, \rho_K)$ is average about one if we set

$$\rho_K = V_K(1)^{\frac{1}{K}} \mathrm{covol}(B)^{\frac{1}{K}} = 2^{-n_x/K} \Gamma(K/2+1)^{1/K} \cdot \pi^{-\frac{1}{2}}. \tag{11}$$

For details of the derivation, see Appendix C. Although we basically followed Ekerå's theory [28] to implement our postprocessing, we have found two issues to modify whose details are postponed to Appendixes B and C.

The first issue is the determinant computation of the lattice B. In the lattice application, a lattice basis is usually given by a triangular matrix for the simplicity of determinant analysis. In our situation, the lattice is represented by a $(K+1) \times K$ matrix as in (9). An auxiliary column $\boldsymbol{c} = (\tau, 0, \ldots, 0)^T$ on the left of B is commonly used to analyze such bases, where τ is a parameter to be optimized. Because the dimension has been changed, it may cause some theoretical issues. In Theorem 1 of Appendix B, we provide a theoretical analysis of $\mathrm{covol}(B)$. This is appropriate for our situation, in which experiments to solve the DLP are carried out using small-dimensional lattices.

The second issue concerns the number of lattice points within $\mathrm{Ball}(\boldsymbol{y}, \rho_K)$. We experimentally find that an exponential number of lattice points is contained in the ball (see Appendix C). This result shows that the DLP lattices are not close to the level of randomness required to use the Gaussian heuristic. Thus, we should work in low dimensions, such as $K+1 \leq 10$, to limit the number of found vectors. Unfortunately, to the best of our knowledge, a method for setting the radius in a large dimension is unknown.

In small dimensions, the number of lattice points within the ball has a nonnegligible variance. We experimentally discovered that there is a nonnegligible number of trials, in which there is no lattice point within $\mathrm{Ball}_K(\boldsymbol{y}, \rho_K)$. In this case, we add the execution of the CVP oracle that finds the closest lattice point to $\boldsymbol{y}$ and recovers a solution candidate from it. Algorithm 1 describes the postprocessing algorithm. We can easily check the candidates by computing $g^{z_i} \pmod p$. We declare the success of the experiment when one of the candidates passes the check. Otherwise, the experiment is declared failed and we try with a new set of inputs.

C. SELECTION AND MODIFICATION OF BIT STRINGS

In our preliminary experiments, we found that a naïve execution of Algorithm 1 sometimes fails to find the desired DLP solution from device outputs. To increase the success probability, we propose two methods to select and modify the bit strings from the devices.

The first method is to remove the zeros from the set. The zero vector $\boldsymbol{p}_i = (0, 0)$, derived from the zero bit string, is useless because the zero vector is always a point in the dual lattice $L^\times$. As a result, it provides no information, so we can remove vectors representing points near (0, 0) from the postprocessing algorithm's inputs. Because the numbers considered in our experiments are very small, we removed only the zero vectors, whereas other vectors near the origin should also be removed in the case of large DLP instances.

Algorithm 2: Our Bit String Modification Algorithm for the DLP.

Input: $\boldsymbol{s}$: a bit string from a device
Output: $(p,' r')$: modified vector
1: Convert $\boldsymbol{s}$ to $\boldsymbol{p} = (p, r)$ as in (6)
2: **if** $\boldsymbol{p}$ is sufficiently close to a point in S_p in (12) **then**
3: **return** $\boldsymbol{p}$
4: **end if**
5: **for all** $\boldsymbol{p}_i = (p,' r')$ is from the ith bit flip of $\boldsymbol{s}$ **do**
6: **if** $\boldsymbol{p}_i$ is sufficiently close to a point in S_p **then**
7: $C \leftarrow C \cup \{\boldsymbol{p}_i\}$
8: **end if**
9: **end for**
10: **if** $C = \phi$ **then**
11: **return** $\perp$
12: **else**
13: Choose randomly $\boldsymbol{p}$ from C
14: **return** $\boldsymbol{p}$
15: **end if**

The second method is using the properties of distributions after applying the QFT. Since $L^\times$ is spanned by two vectors in the matrix D (see Appendix A for details), any points in $L^\times \cap [0, 1)^2$ from any DLP instance considered under modulo p must be in the set

$$S_p = \{(0,0)\} \cup \left\{ \frac{1}{p-1}(c_1, c_2) : \begin{matrix} c_1 = 1, \ldots, p-2 \\ c_2 = 0, \ldots, p-2 \end{matrix} \right\}. \tag{12}$$

Thus, if the converted point $\boldsymbol{p}' = (p,' r')$ is not very close to a point in S_p, errors during the quantum computation or measurement are expected to be occurred. We simply assume that the errors are bit flips at the measurement and can modify an output by checking that all the candidates of bit strings that are small fraction of bits are flipped.

We emphasize that determining whether a vector is close to a point in S_p can be done quickly and without using any DLP solution information. Furthermore, unlike error mitigation techniques, it does not make use of any information from the probability distribution.

For a bit string containing errors, we can try all the 1-bit flips to ensure that the corresponding vectors are on the correct points. If all of the trials fail, we reject the bit string and try all the 2-bit, 3-bit, and so on flips if necessary. In our experiments, we try every single 1-bit flip.

Therefore, we modify the bit strings from devices as in Algorithm 2. Note that we use the criteria $\boldsymbol{p} \in S_p$ and $\boldsymbol{p}_i \in S_p$ exactly to check the conditions in steps 2 and 6, respectively, in our experiments.

Using the above two modification methods in Step 1 in Algorithm 1, each input bit string is translated to a modified

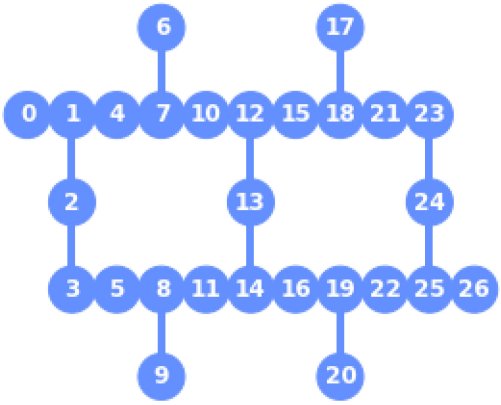


FIGURE 4. Connectivity of `ibm_kawasaki`.

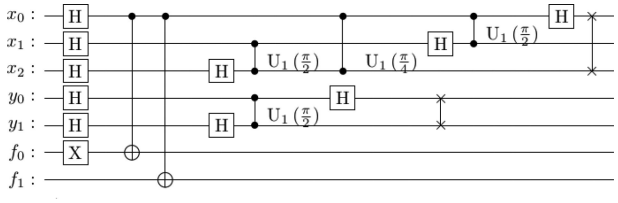


FIGURE 5. Our abstract circuit of instance I.

point (p_i, r_i) or rejected sample $\perp$. We replace $\perp$ with a new modified sample from a device and move to step 2 after all the points have been processed.

V. EXPERIMENTS IN IBM QUANTUM

This section provides experimental results on a real superconducting quantum computer. For Shor's quantum circuits to solve selected instances of DLP, we measured probability distributions and success probabilities.

A. EXPERIMENTAL ENVIRONMENT

First, we explain the target DLP instances and the conditions of the experiments. The experiments were performed on the IBM Quantum device `ibm_kawasaki`, which has the connectivity of qubits shown in Fig. 4.

The circuit of instance I assuming full connectivity is shown in Fig. 5 as an illustrative example. On the other hand, current superconducting quantum computers do not have full connectivity. Thus, we have to embed the circuit by fitting it the topology of the target quantum device. The performance loss should be as small as possible.

There are many methods to measure the loss. We tried to reduce the total number of CNOT (`cx`) gates, which is expected to be equal to increase the total performance because the fidelity of CNOT gate is lower than single gate in IBM Quantum devices. Such circuit-to-circuit translation is performed by the `transpile` command in Qiskit [32]. This command returns a variety of feasible circuits since it uses a random number in its optimizing subroutine. We set the seed of random number generator as the input of the `transpile` command, so that we can generate many circuits and take the circuit whose number of CNOT gates is minimum. We also set the options so that the output circuit consists of the gate set ["`cx`," "`id`," "`rz`," "`sx`," "`x`"] and optimization level 3. Fig. 6 shows the circuit of instance I used in our experiments.

The circuits of instances II and III, which are used in our experiments, are shown in Fig. E1 in Appendix E. As we explained in the next section, experiments using both the instances are not succeeded because their success probabilities do not meet the level of threshold values. Instances IV–VI are not executed because they are clearly more complicated than instances II and III, and the success probabilities are very lower than the thresholds.

The summary of the experimental environment is shown in Table 2. One execution consists of 8192 shots and measurements, and one experiment consists of 100 repeats of execution to evaluate the statistical variance. Totally, we have 819 200 bit strings for each instance.

TABLE 2. Experiments Conducted on `ibm_kawasaki`

	qubits indices	Date of experiment
I	$\{1, 3, 2, 24, 25, 0, 4\}$	Jul 21st, 2021
II	$\{1, 0, 4, 5, 8, 2, 3\}$	Jul 21st, 2021
III	$\{1, 0, 4, 5, 8, 11, 2, 3\}$	Jul 21st, 2021

B. RESULTS FROM REAL QUANTUM DEVICES

We compare the output distribution by which we denote P_{Kawasaki} and the ideal distribution P_{Ideal}. Fig. 7 shows the comparison among the probability distributions of instances I–III.

From the graph, clearly, instance I is solved. In fact, using our postprocessing algorithm (see Section IV-B) on instance I, the success probability is always higher than 99.9% for $K = 2, \ldots, 20$. On the other hand, in instances II and III, although the exact peaks of P_{Kawasaki} and P_{Ideal} are close to each other, there are some other peaks that are not expected, such as 00100. Therefore, in the next section, we discuss the effects of these peaks in postprocessing.

C. SUCCESS PROBABILITY RESULTS

To facilitate a more quantitative comparison, we checked the success probability of the postprocessing algorithm. Fig. 8 shows the success probability of p_{Ideal} and p_{Unif} and the experimental success probability p_{Kawasaki} (i.e., the probability that the return of Algorithm 1 includes the correct solution) of instances II and III, using 819 200 samples from the device. We compared the probabilities without bit string modification in both the cases, as described in Section IV-C.

Fig. 8 depicts a summary of the results. The success probability from the ideal distribution is almost 1, and the threshold values are computed by $(1 + p_{\text{Unif}})/2$. Unfortunately, the success probabilities of the device outputs are lower than that of the uniform. This can be expressed as follows. In instances II and III, the DLP has the solution $z = 1$ and the vectors that span the dual lattice are (0,1) and $(1/2, 1/2)$. Thus, $(p, r) = (0, 0)$ and $(1/2, 1/2)$ from the bit string $s_0 = 00000$ and

IEEE Transactions on Quantum Engineering

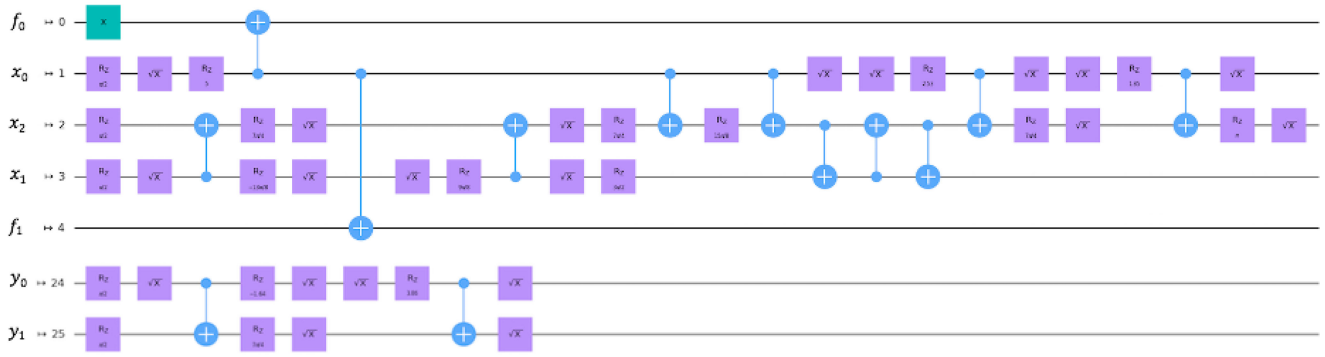


FIGURE 6. Compiled circuit of instance I after optimization by the `transpile` command with optimization level 3.

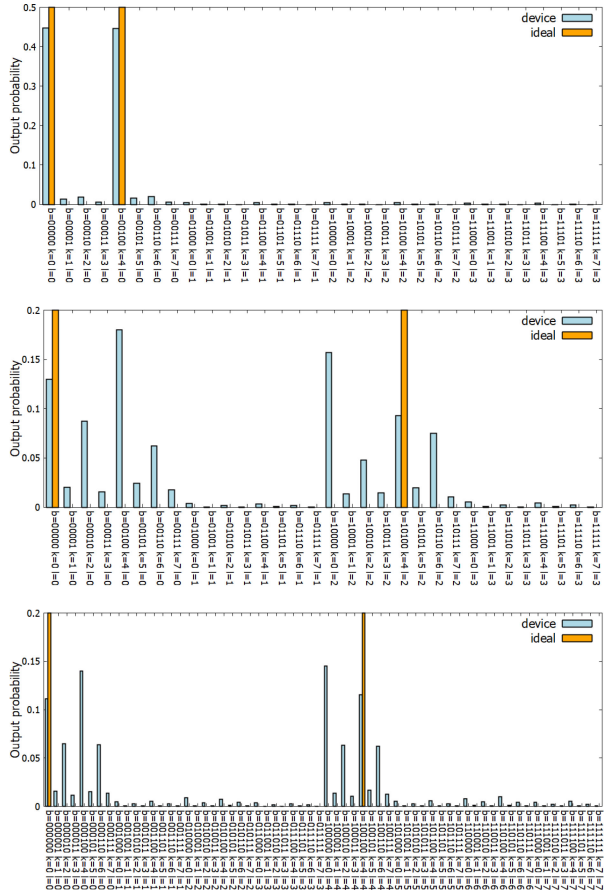


FIGURE 7. Probability distribution of ideal output and IBM Quantum real device output. From the top, the experimenting circuits are I, II, and III in Table 1. In instances II and III, the values are cut at 0.2 and the ideal probabilities are all 0.5.

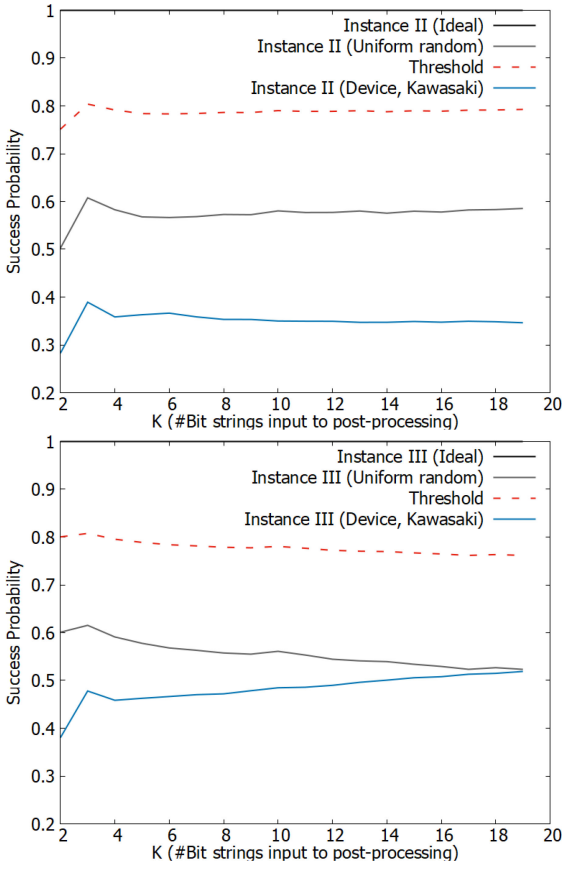


FIGURE 8. Experiment success probabilities using our postprocessing algorithm (see Algorithm 1). The top and bottom are on instances II and III, respectively. Note that this result is before using our bit modification algorithm, whose detail and result are explained in Sections V-D and IV-C. Here, K is the number of used bit strings sampled from the set of bit strings generated by the IBM Quantum device.

$s_1 = 10100$ in instance II are the points observed under ideal conditions. On the other hand, noise bit strings $s_2 = 00100$ and $s_3 = 10000$ are frequently observed in experiments. In particular s_2 corresponds to $(p, r) = (1/2, 0)$ and the solution $z = 0$ that the postprocessing algorithm makes mistake. We think the reason why the error bit sequence s_2 is higher than the correct solutions ($s_0 = 00000$ and s_1) is a concentration of error since the Hamming distance relation $d_H(s_0, s_2) = d_H(s_1, s_2) = 1$.

D. RESULTS AFTER 1-BIT MODIFICATION

To obtain better results, we apply our procedure to modify the bits introduced in Section IV-C. We also apply the modification for the uniform output for a fair comparison. The

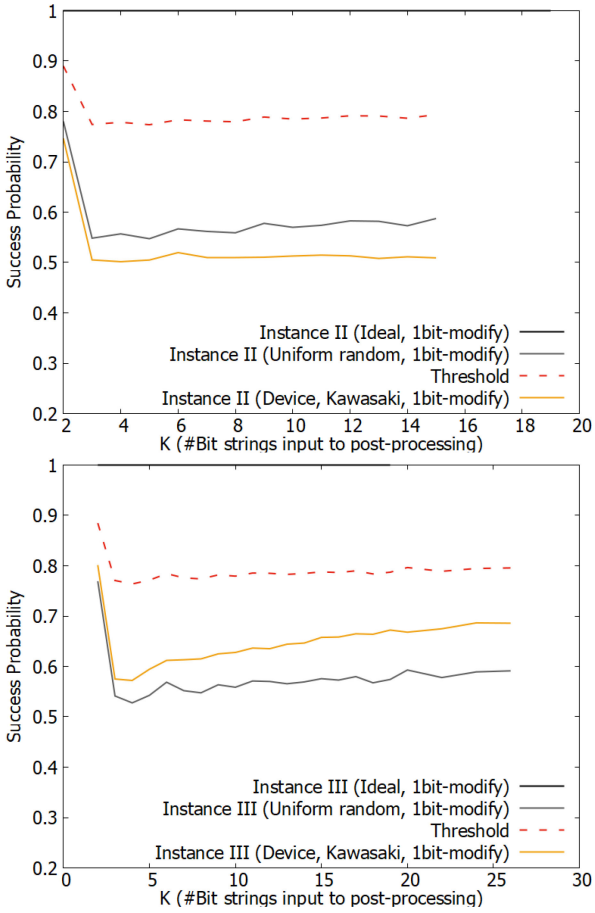


FIGURE 9. Experimented success probabilities of our postprocessing algorithm with modified bit string.

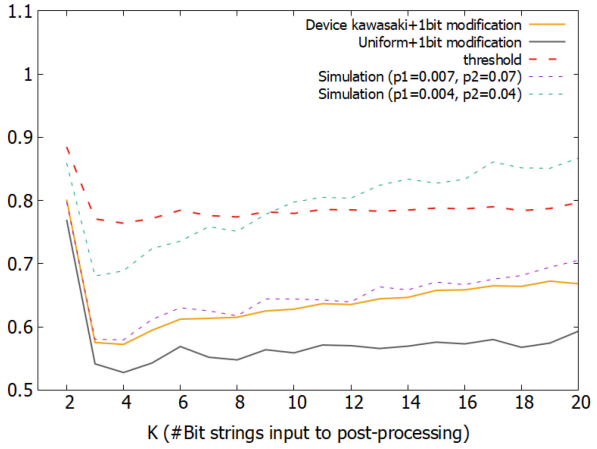


FIGURE 10. Comparison of success probabilities. The yellow line is the probability of random bits with 1-bit modification. Because the success probability of the ideal output is 1, the threshold (red dotted line) is computed by $(p_{\mathrm{Unif}}+1)/2$. The orange line is the probability of the real Kawasaki device, which matches the purple dot line of simulation with $p_2 = 0.07$. To claim success, it is necessary to reduce the noise level to $p_2 = 0.04$.

TABLE 3. Summary of Necessary Noise Level (as of p_2 in Simulation) for the Instances Under Consideration

	#cx	p_{mod}	p_{nomod}
I	15	0.6	0.35
II	32	0.04	0.025
III	38	0.04	0.025
IV	179	0.004	0.004
V	255	0.0055	0.0055

The #cx row is the same as the #cx min row in Table I. p_{mod} and p_{nomod} are necessary p_2 to claim success when 1-bit modification is used and unused, respectively.

probability distribution of the device output before and after modification is supplied in Appendix D.

Fig. 9 shows the comparison of success probabilities. We can see that the device probabilities have increased, as have the thresholds. The success probabilities of device outputs are still lower than the thresholds. We emphasize that in instance III, the probabilities of the device become higher than the uniform. As a result, it appears that if noise levels are reduced, there is a chance of success. In the following section, we will discuss how much noise we need to reduce through simulation.

VI. IMPLICATIONS FOR REQUIRED NOISE REDUCTION

As we can see in the above section, the problem level that the latest quantum devices can solve is between instances I and II. This section provides the results of our simulation of noisy devices to discuss the near future of the DLP and the IFP against quantum devices.

Noise in the real quantum devices is represented by many parameters. To simplify the discussion, we represent the noise by one real number p_2 that indicates the 2-bit gate depolarizing error.

We simulated the circuit of instance III, setting 1-bit and 2-bit gate depolarizing errors to $0.1 \cdot p_2$ and p_2, respectively. Fig. 10 shows the success probabilities. Since the success probability of random bits with 1-bit modification is about 0.55, the threshold (red dot line) is about 0.8. On the other hand, the success probability of Kawasaki output with 1-bit modification is about 0.6, which does not meet the threshold. The success probability graph is comparable to that of our simulation with $p_2 = 0.07$. It is necessary to achieve $p_2 = 0.04$ to obtain a success experiment. We remark that $p_2 = 0.07$ does not match the claimed CNOT gate error rate reported for `ibm_kawasaki` [33], because we tried to represent the complex effect of diverse errors in the real device by using a single value. From the ratio of p_2, we think that it is necessary to halve the noise to claim success for experiments on instance III.

We also simulated the circuits in Table 1 to find the noise levels we need to achieve to claim success. The results are summarized in Table 3 and Fig. 11. p_{mod} and p_{nomod} are the maximum of p_2 to claim success when 1-bit modification is used and unused, respectively. Here, we declare success if one of the success probabilities of postprocessing for $K = 2, \ldots, 10$ is higher than the threshold.

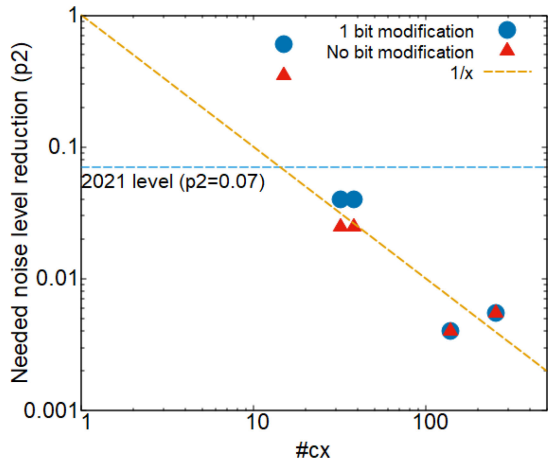


FIGURE 11. Comparison between #cx and the necessary noise level p_2. The blue circles and red triangles are the results with bit string after and before modification, respectively. The points from instances II–V all fit on the line 1/#cx. The horizontal dotted line is $p_2 = 0.07$, the noise level as of 2021.

The values on instances II and III look the same, whereas the details are different. The size of the QFTs is one of the differences. Based on the experimental results, we discovered that increasing the QFT size results in increased noise resiliency. In these cases, it appears that the noise increase caused by the number of gates and the resiliency from the QFT size are balanced.

For instances IV and V, the values p_{mod} and p_{nomod} are the same because the modification algorithm (see Algorithm 2) has very small effects on the inputs. Since both the instances are from the DLP instance with $p = 6$, the set of legitimate points S_p defined by (12) contains 31 points and the number of corresponding legitimate bit strings is inherently greater than 31. On the other hand, the possible number of output bit strings is $2^{n_x+n_y} = 64$ and 256 for instances IV and V, respectively. Thus, most outputs are regarded as legitimate such that the true condition is satisfied in step 2 in Algorithm 2.

Fig. 11 shows the log plot of #cx and the probability. We can see that the values of instances II–V are on the line $1/x$. This explains the weakness of the circuits against noise. If any one of the CNOT gates has an error, the whole computation fails.

Future prediction: IBM provides the history of averaged CNOT gate error rates [7] over the past five years. The rate continues to decrease, falling by roughly half every year. We can predict when the DLP instances will be solved if the total noise level of real quantum devices continues to decrease at the same rate. As previously stated, the IBM Quantum (`ibm_kawasaki`) released in 2021 corresponds to the noise level $p_2 = 0.07$, which we use to simulate CNOT and single-gate errors. To solve instances II and III, $p_2 = 0.04$ is required. If the current trend continues, we would expect these instances to be solved by a quantum device released in 2022. In addition, in 2025, the noise level will reach about $p_2 = 0.07 \cdot 2^{-4} = 0.004375$ and instances IV and V are expected to be solved.

We also give a prediction on Shor's factoring algorithm based on the number of CNOT gates and a slightly speculative assumption

$$p_2 \approx 1/\#\text{cx} \tag{13}$$

because we have never discussed enough lattice-based postprocessing algorithms and bit string modification for integer factoring.

TABLE 4. Summary of Quantum Circuits for the Proof of Concept of Shor's Factoring Algorithm

	Factoring instance	Bits of N	QFT size	a	#Q	#cx avg	#cx min
15A	15	4	4	7	9	131.22	100
15B				2		94.42	86
21A	21	5	3	2	8	163.99	126
21B			4		9	448.28	365

Table 4 summarizes the factoring quantum circuits that were considered. Similar to the DLP circuits, we implement them using `Qiskit` and compile them using the `transpile()` function with 100 different seeds for the random portions of compilation. The columns #cx min and #cx avg represent the minimum and average of #cx of transpiled circuits.

The factoring circuits considered here are textbook proof-of-concept circuits. That is, they are simplified by using the properties of $a^x \bmod N$. For example, the second line instance 15B is simplified by using the fact $2^x \bmod 15 \in \{1, 2, 4, 8\}$ and $2^4 \bmod 15 = 1$. The third line, instance 21A, is from the circuit by Amico *et al.* [13]. The details of the factoring circuits considered are described in Appendix F.

From the viewpoint of #cx, 15A, 15B, and 21A are two to three times harder than the DLP instances II and III. Under our assumption, when $p_2 = 0.07 \times 2^{-3} = 0.00875$ is achieved in 2024, 15A and 15B may be solved since $1/\#\text{cx} = 0.01$ and 0.0116, respectively. The instance 21A has a chance to be solved since $1/\#\text{cx} = 0.00794$. On the other hand, to solve the larger instance 21B, we have to wait for the machines to progress until 2026.

This near-term prediction also explains why the existing reports on successful integer factoring over the past 20 years (see[14]–[34]) use oversimplified circuits (and subcircuits) and yet only factor 15 and 21. Current error rates are not good enough to execute the canonical proof-of-concept circuits for factoring 15, which require hundreds of CNOT gates. Achieving this first step will take several years. We predict that larger instances will be solved constantly after the first report of complete execution of factoring 15 or 21 including postprocessing because the growth rate of the number of gates is polynomial in $\log N$ of the number N being factored. We expect that new reports of factoring numbers greater than 35 will be published frequently after 2025.

VII. DISCUSSION

We reported our experiments on the DLP executed on an IBM Quantum device. The entire performance of the latest quantum device against the DLP, including the postprocessing algorithm, is very limited. Such a device can solve the smaller 2-bit instance $2^z \equiv 1 \pmod 3$ (instance I in Table 1) in the sense of the median principle on the success probability. On the other hand, it might require a reduction of the noise by half to solve the larger instance $2^z \equiv 2 \pmod 3$ (instances II and III in Table 1).

We have predicted that the 3-bit DLPs considered in Table 1 and the factoring instances in Table 4 will be solved around 2025. On the other hand, solving 4-bit or larger instances seem to necessitate the use of a device with QEC. The importance of noise reduction has been known since the early days of Shor's algorithm [35]. Here, we provide quantitative support by (13) for that consensus.

Threatening RSA-2048: The growing speed of device performance under the assumption from the history of device noises and (13) is notable. Assuming that the executable number of CNOT gates with negligible errors doubles every year, it can be expected that a quantum circuit with millions of gates can be executed in the coming decades. The estimate of Gidney and Ekerå [36, Table 1] claims that about 2.7×10^9 logical Toffoli gates are required to factor RSA-2048. Assuming that one Toffoli gate is implemented using five CNOT gates and several single qubit gates, about $1.35 \times 10^{10} \approx 2^{34}$ CNOT gates are necessary to implement the circuit. As a result, it can be expected that a quantum device that threatens integer factoring-based cryptosystems will be developed in the next 30–40 years.

This prediction backs up previous reports predicting that the time of compromise RSA-2048 will also arrive in the coming decades. Sevilla and Riedel [37] make a prediction based on the assumption of exponential progress of physical qubits and gate fidelity, claiming that this is an optimistic scenario. Their prediction is based on their quantifier of quantum devices that they named generalized logical qubits. They predicted that a superconducting quantum device capable of solving RSA-2048 (using 4100 qubits) would be available in the early 2050s, rather than before 2039. This is more optimistic than expert opinions [38], [39] published in 2019 and updated in 2020. Mosca and Piani [38], [39] say that 90% of experts predict that there is 50% or greater chance of a quantum device that can break RSA-2048 in 24 h being released in the next 20 years.

In addition, our method also shares the difficulty of early prediction of emerging technologies. That is, the predictions on RSA in the above assumed the exponential growth of some performance values at a glance from historical values. Owing to the small number of data points, the evidence supporting our assumptions is not very strong, and we need to refine the predictions by taking account into the progress of quantum devices and experiments over the next few years.

We also remark that our prediction method is slightly different from the typical methods in cryptographic research. To predict threats to RSA and DSA in future, cryptographers typically have used "bit lengths of cryptosystems," represented by the bit lengths of numbers to be factored in RSA and the modulus in DSA, to measure the security of cryptosystems. However, this strategy cannot be applied simply to the situation of quantum computing. Despite the advances in technology, the known claimed successful factoring experiments using Shor's circuit on real quantum computers have only been for the values 15 and 21 over a span of 20 years [14]–[34]. Thus, it is difficult to predict the time of compromise RSA-2048 from only previous experiments. Therefore, another predicting method was needed, leading us to the present analysis.

Technical future work: Our experiments in this article employ the standard approach to constructing the circuits. To increase the success probability, several techniques can be applied.

In addition to the simplification of modular-exponentiation circuits, an approximate QFT can also be used [40], [41]. The approximation, or omitting rotation gates of very small angles in particular, can reduce the number of gates and required total noise levels, at the expense of accuracy. Balancing the hardware noise and its inaccuracy is a problem that we hope to address in the near-future.

QEC is under continuing development. Once implemented, it can also significantly reduce the noise level. However, with short code distances and limited distillation capabilities for the magic states needed for the fault-tolerant implementation of Toffoli gates, early fault-tolerant systems will still have residual (post-QEC, post-FT) error rates that must be handled using techniques similar to those described in this article, rather than assuming that logical gates are perfect. It will be interesting to see how the noise level trend changes as QEC and fault-tolerant techniques are applied to actual algorithm execution [4], [42].

The postprocessing algorithm is also a work in progress, with room for continued improvement. Based on the set S_p of valid bit strings, we proposed a simple bit modification algorithm. Other methods should be considered. Outputs with low levels of noise, in particular, can aid in the recovery of the correct solution in some security-related problems [43].

APPENDIX A
BACKGROUND THEORY OF OUR POSTPROCESSING

This section provides a framework for deriving the computational problem (see Problem 1), which is implicitly described in several previous works [22], [31], [44]. We begin by explaining why the noiseless execution of Shor's circuit (see Fig. 1) produces a point close to a lattice point.

We regroup (5) by the value of $F = g^x a^{-y} \bmod p$ as

$$\sum_{k,\ell}^{N_x-1,N_y-1} \sum_{F=0}^{p-1} \sum_{\substack{x=0,y=0 \\ g^x a^{-y} \bmod p = F}}^{N_x-1,N_y-1} e^{2\pi i \cdot \left(\frac{kx}{N_x}+\frac{\ell y}{N_y}\right)} |k, \ell\rangle |F\rangle.$$

IEEE Transactions on Quantum Engineering

Thus, the probability that we observe $|k, \ell\rangle$ is

$$\sum_{F=0}^{p-1} \left| \sum_{\substack{x=0,y=0 \\ g^x a^{-y} \bmod p = F}}^{N_x-1,N_y-1} e^{2\pi i \cdot \left(\frac{kx}{N_x} + \frac{\ell y}{N_y}\right)} \right|^2 \tag{14}$$

up to the normalization constant. To the best of our knowledge, there is no closed simple formula to compute or approximate this.

However, if there exists a pair (k, ℓ) and a constant $c_{k,\ell,F}$ satisfying the following conditions: 1) the constant does not depend on x and y and 2) for any (x, y) satisfying $g^x a^{-y} \bmod p = F$, the constant satisfies

$$\frac{kx}{N_x} + \frac{\ell y}{N_y} - c_{k,\ell,F} \bmod 1 \approx 0.$$

Then, the probability factor $|\cdot|^2$ in (14) is high for the pair. Here, mod 1 transforms a real number to a number within the range $(-0.5, 0.5]$. In other words, the above (k, ℓ) satisfies that for any pairs (x, y), $(x,' y')$ satisfying $g^x a^{-y} \bmod p = g^{x'} a^{-y'} \bmod p$, we have

$$\begin{aligned} &\left(\frac{kx}{N_x} + \frac{\ell y}{N_y}\right) - \left(\frac{kx'}{N_x} + \frac{\ell y'}{N_y}\right) \bmod 1 \\ &= \left(\frac{k}{N_x}, \frac{\ell}{N_y}\right) \cdot (x - x,' y - y') \bmod 1 \approx 0. \end{aligned}$$

This condition can be interpreted using the terminology of lattices. Since the function $g^x a^{-y} \bmod p$ have two periods $(z, 1)$ and $(p, 0)$, the above $(\frac{k}{N_x}, \frac{\ell}{N_y})$ satisfies

$$\left(\frac{k}{N_x}, \frac{\ell}{N_y}\right) \cdot (x, y) \approx \mathbb{Z}$$

for all the lattice point $(x, y) \in L \cap [0, N_x - 1] \times [0, N_y - 1]$, where the lattice $L(B)$ is spanned by $(z, 1)$ and $(p, 0)$. Thus, we can expect that $(\frac{k}{N_x}, \frac{\ell}{N_y})$ is an approximation of a point in $L^\times \cap [0, 1)^2$ if the observed pair has no quantum errors.

We explain the derivation of an instance $(B, \boldsymbol{y}, \rho)$ of the BDD problem and how the solution implies the desired DLP solution z. The matrix representation of B and its dual D are given as

$$B = \begin{bmatrix} p-1 & 0 \\ z & 1 \end{bmatrix} \quad \text{and} \quad D = \frac{1}{p-1} \begin{bmatrix} 1 & -z \\ 0 & p-1 \end{bmatrix}.$$

Suppose that we carry out K shots on a quantum device and have bit strings $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K$. In addition, we let the ith point interpreted from the ith vector as

$$\boldsymbol{p}_i := (p_i, r_i) = \left(\frac{k_i}{N_x}, \frac{\ell_i}{N_y}\right). \tag{15}$$

There exists a short error vector (ε_i, η_i) so that $(p_i + \varepsilon_i, r_i + \eta_i) \in L^\times$. Decomposing into coordinates, for each measurement, there exists integers a_i and b_i, and they satisfy

$$p_i = \frac{a_i}{p-1} - \varepsilon_i \quad \text{and} \quad r_i = \frac{-a_i z}{p-1} + b_i + \eta_i.$$

Therefore, erasing a_i, we obtain the fundamental relation

$$z \cdot p_i + r_i - b_i = -z \cdot \varepsilon_i + \eta_i. \tag{16}$$

Revealing z from given sequence of p_i and r_i is the computational problem (see Problem 1) that we have to solve. One straightforward way is a procedure that calls a BDD oracle. Consider the BDD instance given by the $(K+1) \times K$ matrix

$$B := \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \vdots \\ \boldsymbol{b}_{K+1} \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & \cdots & p_K \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{17}$$

the target vector

$$\boldsymbol{y} = (r_1, r_2, \ldots, r_K), \tag{18}$$

and the radius ρ to be optimized. To find suitable ρ, a refined determinant analysis is necessary, which we will describe in Appendix B.

We can see that there exists $\boldsymbol{x} \in L$ close to $\boldsymbol{y}$ satisfying

$$\boldsymbol{y} - \boldsymbol{x} = (-z\varepsilon_1 + \eta_1, -z\varepsilon_2 + \eta_2, \ldots, -z\varepsilon_K + \eta_K). \tag{19}$$

We explain how to recover z. We first remark that finding the combination coefficient a_i of the lattice vector $\boldsymbol{x} = \sum_{i=1}^{K+1} a_i \boldsymbol{b}_i$, z is easily recovered since $z = -a_1$. Our implementation via lattice algorithms is slightly technical as follows.

The first operation is to transform the lattice basis (9) to its full-rank form since a BDD subroutine typically takes a full-rank matrix basis as its input. For instance, there is an efficient algorithm (see, e.g., [23, Sec. 6] or [25, Sec. 2.6.4]) that outputs a pair of a matrix U and a square basis matrix B' so that satisfies

$$\begin{aligned} B' &:= \begin{bmatrix} \boldsymbol{b}'_1 \\ \boldsymbol{b}'_2 \\ \boldsymbol{b}'_3 \\ \vdots \\ \boldsymbol{b}'_K \end{bmatrix} \\ &= UB = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,K+1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{K,1} & u_{K,2} & \cdots & u_{K,K+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \vdots \\ \boldsymbol{b}_{K+1} \end{bmatrix}. \end{aligned}$$

In particular, each $\boldsymbol{b}'_j$ is expressed as $\sum_{i=1}^{K+1} u_{j,i} \boldsymbol{b}_i$.

Then, a BDD subroutine with the input $(B,' \boldsymbol{y}, \rho)$ returns a set of lattice vectors. Let $\boldsymbol{x} = \sum_{j=1}^{K} c_j \boldsymbol{b}'_j$ be a vector in the set. Here, a typical BDD subroutine returns its combination coefficients c_j. If not, we can easily compute them by simple matrix multiplication. By the relation

$$\sum_{j=1}^{K} c_j \boldsymbol{b}'_j = \sum_{j=1}^{K} c_j \sum_{i=1}^{K+1} u_{j,i} \boldsymbol{b}_i$$

the coefficient of $\boldsymbol{b}_1$ is $\sum_{j=1}^{K} c_j u_{j,1}$ and it is $-z$, the negative of the desired DLP solution candidate.

APPENDIX B
REFINED DETERMINANT ANALYSIS

This section gives our refined theoretical analysis of the covolume of the lattice defined by (9), which is used in our BDD implementation in relatively small dimensions. The technique is generally called the point-counting method, and we apply it to the dual lattice of B.

Theorem 1: Let B be the matrix of the form (9). In particular, assume that all p_i are fractions of the form $a_i/2^{m_i}$, where a_i is odd integer or zero. For $p_i = 0$, we set $a_i = m_i = 0$. Then, the covolume of the lattice spanned by the rows of B is

$$\mathrm{covol}(B) = 2^{-m_j}$$

where m_j is the maximum of all m_is.

Proof: Let $\boldsymbol{w} = (w_1, \ldots, w_K)$ be a vector in the dual lattice and consider the condition the vector must satisfy. By the definition of dual lattice, $\langle \boldsymbol{w}, \boldsymbol{b}_i \rangle \in \mathbb{Z}$ for all $i = 1, \ldots, K+1$. Thus, $\boldsymbol{w}$ must be an integer vector by the condition for $i = 2, \ldots, K+1$. Consider the condition that for $i = 1$, we have

$$\sum_{\ell=1, \ell \neq j}^{K} w_\ell \cdot \frac{a_\ell}{2^{m_\ell}} + w_j \cdot \frac{a_j}{2^{m_j}} \in \mathbb{Z}.$$

After fixing all w_ℓ except for w_j, there exists just one w_j in 2^{m_j} integers, so that the inner product is an integer. Thus, we have $\mathrm{vol}(D) = 2^{m_j}$ and $\mathrm{vol}(B) = \mathrm{vol}(D)^{-1} = 2^{-m_j}$. ■

In our situation, $\max m_i = n_x$ (QFT size) holds with high probability since $p_i = a_i/2^{n_x}$ are considered as random numbers from the observations.

APPENDIX C
SELECTING RADII IN BDD

Computing the lattice covolume, one can try to set the searching radius ρ in the BDD subroutine so that the computational time of postprocessing is feasible. The number of lattice points in the ball, $\mathrm{Ball}_K(\boldsymbol{y}, \rho)$, should be small, for this purpose, i.e., bounded by some constant or subpolynomial function of the lattice dimension. As a result, determining a relationship between the number of lattice points and the radius is crucial. In typical situations, the Gaussian heuristic plays an important role. However, we discover that the heuristic does not hold in our DLP situation, posing a new unsolved problem.

The Gaussian heuristic says that there exists approximately $\mathrm{vol}(\mathrm{Ball}_K)/\mathrm{covol}(L)$ lattice points in the intersection $\mathrm{Ball}_K \cap L$ for a K-dimensional full-rank lattice. In our situation, the number can be computed as

$$\frac{\mathrm{vol}\left(\mathrm{Ball}_K(\boldsymbol{y}, \rho)\right)}{\mathrm{covol}(L)} = 2^{n_x} \cdot \frac{\pi^{K/2} \cdot \rho^K}{\Gamma(K/2+1)}.$$

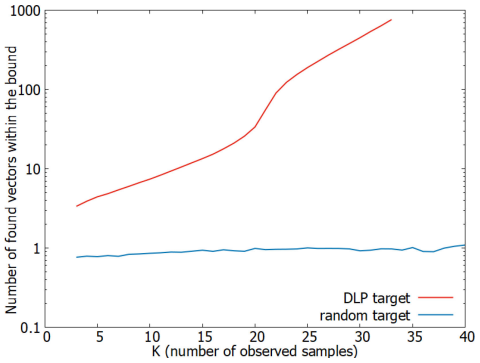


FIGURE 12. Averaged numbers of vectors within the ball, $\mathrm{Ball}_K(\boldsymbol{y}, \rho)$, where $\boldsymbol{y}$ is set from the DLP (red) and set randomly (blue). The radius ρ is set by (20) so that the average number is expected to be 1.

Thus, setting the radius $\rho'_K = N^{1/K} \cdot \rho_K$, where

$$\rho_K = 2^{-n_x/K} \Gamma(K/2+1)^{1/K} \pi^{-\frac{1}{2}} \tag{20}$$

the expected number of lattice points is about N.

Ekerå's [28, Sec. 5.2.1] estimation compares the above radius and an asymptotic upper bound Δ_K to $\|\boldsymbol{y} - \boldsymbol{x}\|$ in the K-dimensional space. Then, estimate how many samples K is required so that the lattice-based postprocessing can be completed in a reasonable amount of time. However, based on our preliminary experiments, we discovered that the strategy needs to be modified.

We simulated noiseless quantum computation and sample many bit strings for the DLP instance V. We generate 10 000 sets of BDD instances defined by (9) and (10) for each dimension K and count the number of lattice points within $\mathrm{Ball}_K(\boldsymbol{y}, \rho)$, where ρ is defined by (20), corresponding to $N = 1$. For the control, we also count the number of lattice points when the target point $\boldsymbol{y}$ is randomly generated from $[0, 1)^K$. The result is depicted in Fig. 12. We can see that the number in the ball grows exponentially in the DLP case, while the random case keeps about 1.

Hence, we think that we cannot use the Gaussian heuristic simply in the postprocessing of Shor's algorithm for the DLP. Either lattices or target vectors are not random. The reason can be explained as follows. Suppose that there exists a lattice vector $\boldsymbol{x}$ in the ball. Then, the neighborhood lattice vectors $\boldsymbol{x} \pm \boldsymbol{e}_i \pm \boldsymbol{e}_j \pm \cdots$ are very likely in the ball where $\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the unit vector. As a result, we believe that using the Gaussian heuristic to set the BDD radius is not a good strategy in this situation. How to set the searching radius or searching space for large K should be an open problem. This is why, in the main section, we propose using postprocessing with small K.

We remark that another setting method by using the covering radius could be considered, but may not be very useful. The covering radius of a lattice L is defined by the minimum radius c of the ball so that the set $\cap_{\boldsymbol{x} \in L} \mathrm{Ball}_K(\boldsymbol{x}, c) = \mathrm{span}(L)$. In other words, the BDD instance $(B, \boldsymbol{y}, c)$ has always a

IEEE Transactions on Quantum Engineering

TABLE 5. Summary of 1 Bit Modification (Algorithm 2) on Instance II

Input	Output Candidates
00000	00000
10000	00000, 10100
01000	00000
00100	00100
00010	00000
00001	00000
10100	10100
01100	00100
00110	00100
00101	00100
11100	10100
10110	10100
10101	10100
Others	$\perp$

solution for any $\boldsymbol{y} \in \mathrm{span}(L)$ if c is greater than the radius. Thus, setting ρ_K by some upper bound of the covering radius, the return of BDD subroutine is always not empty. However, there could find an exponential number of lattice points even for $K = 2$. Let us consider the following extreme situation with $K = 2$, $p_{1,1} = N_x^{-1}$, and $p_{2,k} = 0$. The covering radius of the lattice is $c = \frac{1}{2}\sqrt{1 + N_x^{-2}}$, which is achieved by the target vector $\boldsymbol{y} = (0, 0.5)$. On the other hand, for the target vector $\boldsymbol{y} = (0, 0)$, the ball of radius c contains an exponential number of lattice points. Thus, we decided to keep using the simple radius from the Gaussian heuristic and use an extra CVP subroutine if the BDD subroutine returns the empty set.

APPENDIX D
DEVICE PROBABILITIES AFTER 1-BIT MODIFICATION

This section gives supporting materials of Section V-D. We give a detailed example of our modification algorithm and probability distribution results of the `ibm_kawasaki` device and the distribution after the 1-bit modification on instances II and III.

Since both instances II and III consider the DLP instance of $p = 3$, the set of legitimate points is

$$S_3 = \{(0, 0), (0.5, 0), (0.5, 0.5)\}.$$

The corresponding bit strings are `00000`, `00100`, and `10100` in instance II and `000000`, `000100`, and `100100` in instance III, respectively. We should explain the bit string $b_0b_1b_2b_3b_4$ from the Qiskit output is interpreted as $(k, \ell) = (4b_2 + 2b_3 + b_4, 2b_0 + b_1)$ in instance II. In addition, the bit string $b_0b_1b_2b_3b_4b_5$ in instance III is interpreted as $(k, \ell) = (4b_3 + 2b_4 + b_5, 4b_0 + 2b_1 + b_2)$.

Table 5 shows the summary of 1-bit modification for an instance with $p = 3$. For instance II, its ideal distribution takes (0,0) and (0.5,0.5) with probabilities 0.5, which correspond to the bit strings `00000` and `10100`. The device frequently outputs bit strings that are modified to incorrect bit string `00100`, and we can see that the probabilities of `10100` (correct) and `00100` are very close to each other. This is one reason why the experiment failed.

Fig. 13 shows the probability distribution before and after modification of the `ibm_kawasaki` device outputs.

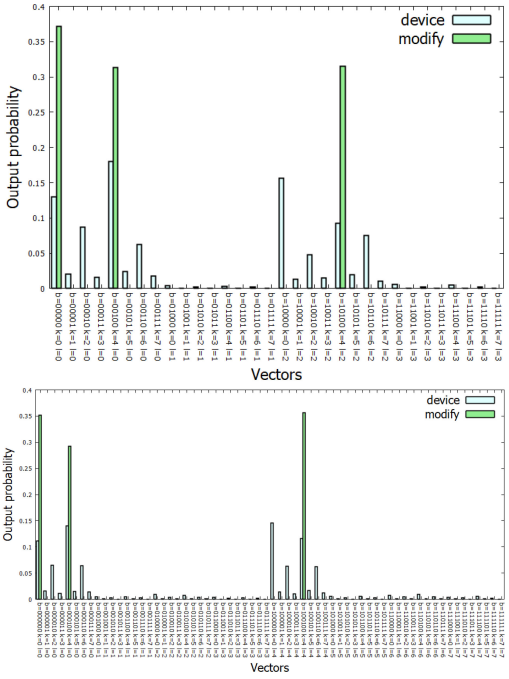


FIGURE 13. Comparison between the probability distribution of the `ibm_kawasaki` outputs and after 1-bit modification.

After the modification, we can see that instance III has better results because the probability of the correct bit string (`100100`) is higher than the other (`000100`), and this explains the difference in success probabilities (see Figs. 8 and 9).

APPENDIX E
CONSTRUCTION OF OUR DLP CIRCUITS

This section provides the detailed construction of the modular-exponentiation part of the DLP circuit experimented in Section V. Fig. 14 shows transpiled circuits of instances II and III to solve the DLP instance $2^z \equiv 2 \pmod 3$.

APPENDIX F
CONSTRUCTION OF OUR FACTORING CIRCUITS

This section provides the detailed construction of the modular-exponentiation part of Shor's factoring circuit considered in Table 4 for the experimental reproducibility.

We first remark that the four circuits that we considered are proof-of-concept versions. That is, they are oversimplified by techniques that are not scalable.

Construction of 15A: Factoring circuit with $(a, N) = (7, 15)$ is traditionally discussed in [34], and recently, Monz *et al.* [12] and Duan *et al.* [45] reported experiments with using semiclassical QFT. Our circuit (see Fig. 15) follows [34]. It computes $Y = (y_3 2^3 + y_2 2^2 + y_1 2^1 + y_0 2^0) = 7^{x_3 2^3 + x_2 2^2 + x_1 2^1 + x_0 2^0} \bmod 15$. The gadget G_0 computes 2^{x_0}, and G_1 computes $\times 4^{x_1} \bmod 15$ since $7^2 = 4 \bmod 15$. The computations on x_2 and x_3 can be omitted since $7^4 \equiv 7^8 \equiv 1 \pmod{15}$.

FIGURE 14. Our transpiled circuits of instance II (top) and III (bottom) to solve the DLP instance $2^z \equiv 2 \pmod 3$. The last observing gates are omitted.

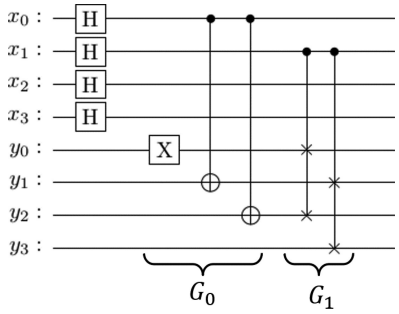


FIGURE 15. Modular-multiplication gadgets for a proof-of-concept circuit of Shor's factoring algorithm with parameters $(a, N) = (7, 15)$ and size of QFT is 4.

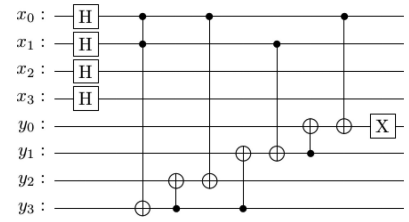


FIGURE 16. Modular-multiplication gadgets for a proof-of-concept circuit of Shor's factoring algorithm with parameters $(a, N) = (2, 15)$ and size of QFT is 4. The number of CNOT gates after transpiling is slightly reduced to the circuit of Fig. 15.

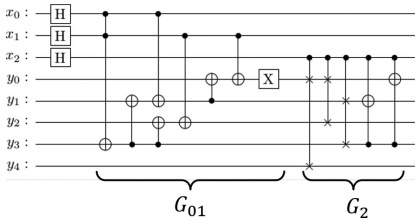


FIGURE 17. Modular-multiplication gadgets for a proof-of-concept circuit of Shor's factoring algorithm with parameters $(a, N) = (2, 21)$ and size of QFT is 3.

Construction of 15B: For the parameters $(a, N) = (2, 15)$, the size of factoring circuit can be slightly reduced by using the information that $2^x \bmod 15 \in \{1, 2, 4, 8\}$. The logical relations between $Y = (y_3 2^3 + y_2 2^2 + y_1 2^1 + y_0 2^0)$ and $X = x_1 2 + x_0$ are written as

$$y_3 = x_0 \cdot x_1, \quad y_2 = x_0 \cdot \overline{x_1}$$
$$y_1 = \overline{x_0} \cdot x_1 \quad y_0 = \overline{x_0} \cdot \overline{x_1}$$

and they are interpreted to the following formulas and the circuit in Fig. 16:

$$y_3 = x_0 \cdot x_1$$
$$y_2 = y_3 \oplus x_0$$
$$y_1 = y_3 \oplus x_1$$
$$y_0 = \overline{y_1 \oplus x_0}.$$

As the situation of $a = 7$, the computations on x_2 and x_3 can be omitted since $2^4 \equiv 2^8 \equiv 1 \pmod{15}$.

Construction of 21A: We construct a factoring circuit with parameters $(a, N) = (2, 21)$ and size of QFT 3. We follow the construction of Amico *et al.* [13]. Our circuit to compute $Y = (y_4 2^4 + y_3 2^3 + y_2 2^2 + y_1 2^1 + y_0 2^0) = 2^{x_2 2^2 + x_1 2^1 + x_0 2^0} \bmod 21$ is displayed in Fig. 17. In the gadget circuit G_{01}, it computes $Y = 2^{x_1 2^1 + x_0 2^0}$ by using the precomputed information that 2^X for $X = 0, 1, 2, 3$ is 1, 2, 4, and 8 respectively. This gadget is the same as circuit 15B. Then, the next gadget computes $\times 2^{4x_2} \bmod 21$. The trick here uses the fact for $Y = \{1, 4, 8\}$, the results of $16Y \bmod 21$ and $16Y \bmod 63$ are the same. This allows us to use the

IEEE Transactions on Quantum Engineering

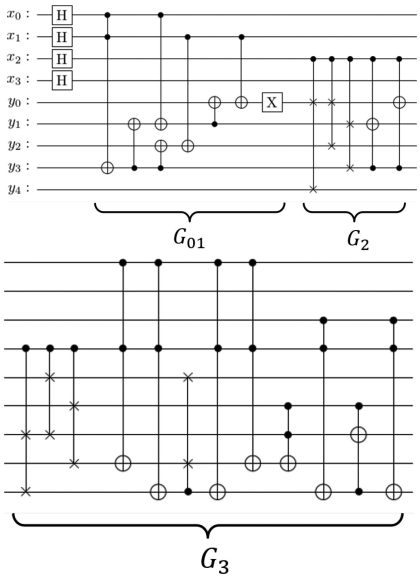


FIGURE 18. Modular-multiplication gadgets for a proof-of-concept circuit of Shor's factoring algorithm with parameters $(a, N) = (2, 21)$ and size of QFT is 4.

TABLE 6. Values That Are Necessary to Be Modified After the First Three CSWAPs in G_3

Input Y after G_2	Y after cswaps in G_3	correct value $4Y$ mod 21
1	4	4
2	8	8
4	16	16
8	2	11
11	14	2
16	1	1

bit-rotation technique described in Section IV-A, which is implemented by the first three CSWAP gates in G_2. After the CSWAP gates applied, $Y = 01000 = 8$ corresponding to $X = 101 = 5$ is only the value to modify. The last two CCX gates modify it to $2^5 \bmod 21 = 11$.

Construction of 21B: A circuit with parameters $(a, N) = (2, 21)$ and size of QFT 4 is constructed by a similar method to that of 21A. In Fig. 18, the gadgets G_{01} and G_2 are the same as Fig. 17. To construct the $(\times 2^{8x_3} \bmod 21) = (\times 4^{x_3} \bmod 21)$ circuits, we use the equivalence between $4Y \bmod 21$ and $4Y \bmod 63$ for $Y = 1, 2, 4, 16$. The first three CSWAPs change the values of Y, as described in Table 6. The remaining part transforms $Y = 2, 14$ to 11 and 2, respectively without changing other values.

ACKNOWLEDGMENT

The results presented in this article were obtained in part using an IBM Qunatum quantum computing system as a project of the Quantum Computing Center, Keio University. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Quantum team.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997, doi: 10.1137/S0097539795293172.

[2] J. Gambetta and S. Sheldon, "Cramming more power into a quantum device," 2019. [Online]. Available: https://www.ibm.com/blogs/research/2019/03/power-quantum-device/

[3] J. Gambetta, "IBM's roadmap for scaling quantum technology," 2020. [Online]. Available: https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/

[4] P. Chapman, "Scaling IonQ's quantum computers: The roadmap," 2020. [Online]. Available: https://ionq.com/posts/december-09-2020-scaling-quantum-computer-roadmap

[5] "Our quantum computing journey," Accessed: Jul. 13, 2022. [Online]. Available: https://quantumai.google/learn/map

[6] "Get to know Honeywell's latest quantum computer system model H1," 2020. Accessed: Jul. 13, 2022. [Online]. Available: https://www.honeywell.com/us/en/news/2020/10/get-to-know-honeywell-s-latest-quantum-computer-system-model-h1

[7] "IBM quantum experience," Accessed: 13 Jul. 2022. [Online]. Available: https://research.ibm.com/blog/heavy-hex-lattice

[8] C. P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Conf. Theory Appl. Cryptol.*, 1990, pp. 239–252, doi: 10.1007/0-387-34805-0_22.

[9] Digital Signature Standard (DSS), National Institute of Standards and Technology, Gaithersburg, MD, USA. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[10] V. S. Miller, "Use of elliptic curves in cryptography," in Proc. Conf. Theory Appl. Cryptographic Techn., 1985, pp. 417–426, doi: 10.1007/3-540-39799-X_31.

[11] E. Lucero *et al.*, "Computing prime factors with a Josephson phase qubit quantum processor," *Nature Phys.*, vol. 8, pp. 719–723, 2012, doi: 10.1038/nphys2385.

[12] T. Monz *et al.*>, "Realization of a scalable Shor algorithm," *Science*, vol. 351, no. 6277, pp. 1068–1070, 2016, doi: 10.1126/science.aad9480.

[13] M. Amico, Z. H. Saleem, and M. Kumph, "Experimental study of Shor's factoring algorithm using the IBM Q experience," *Phys. Rev. A*, vol. 100, no. 1, Jul. 2019, Art. no. 012305, doi: 10.1103/PhysRevA.100.012305.

[14] U. Skosana and M. Tame, "Demonstration of Shor's factoring algorithm for $n = 21$ on IBM quantum processors," *Sci. Rep.*, vol. 11, no. 1, Aug. 2021, Art. no. 16599, doi: 10.1038/2Fs41598-021-95973-w.

[15] K. Oonishi, T. Tanaka, S. Uno, N. Yamamoto, and N. Kunihiro, "Proposal of efficient quantum modular addition circuit and its implementation," *Inst. Electron., Inf. Commun. Eng.*, Tokyo, Japan, Tech. Rep., 2019.

[16] J. A. Smolin, G. Smith, and A. Vargo, "Oversimplifying quantum factoring," *Nature*, vol. 499, no. 7457, pp. 163–165, Jul. 2013, doi 10.1038/nature12290.

[17] T. Satoh, Y. Ohkura, and R. Van Meter, "Subdivided phase oracle for NISQ search algorithms," *IEEE Trans. Quantum Eng.*, vol. 1, 2020, Art. no. 3100815, doi: 10.1109/TQE.2020.3012068.

[18] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Phys. Rev. A*, vol. 100, Sep. 2019, Art. no. 032328, doi: 10.1103/PhysRevA.100.032328.

[19] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: 10.1038/s41586-019-1666-5.

[20] B. Barak, C. Chou, and X. Gao, "Spoofing linear cross-entropy benchmarking in shallow quantum circuits," in *Proc. 12th Innov. Theor. Comput. Sci. Conf.*, 2021, pp. 30:1–30:20, doi: 10.4230/LIPIcs.ITCS.2021.30.

[21] J. H. Cheon, "Discrete logarithm problems with auxiliary inputs," *J. Cryptol.*, vol. 23, pp. 457–476, 2010, doi: 10.1007/s00145-009-9047-0.

[22] M. Ekerå and J. Håstad, "Quantum algorithms for computing short discrete logarithms and factoring RSA integers," in *Proc. Int. Workshop Post-Quantum Cryptography*, 2017, pp. 347–363, doi: 10.1007/978-3-319-59879-6_20.

[23] M. R. Bremner, *Lattice Basis Reduction: An Introduction to the LLL Algorithm and Its Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, 2011.

[24] M. Ekerå, "Quantum algorithms for computing general discrete logarithms and orders with tradeoffs," *J. Math. Cryptol.*, vol. 15, no. 1, pp. 359–407, 2020, doi: 10.1515/jmc-2020-0006.

[25] H. Cohen, *A Course in Computational Algebraic Number Theory*. ser. Graduate Texts in Mathematics Series, vol. 138. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-662-02945-9.
[26] C. A. Rogers, "The number of lattice points in a set," *Proc. London Math. Soc.*, vol. s3-6, pp. 305–320, 1956, doi: 10.1112/plms/s3-6.2.305.
[27] T. Häner, M. Roetteler, and K. M. Svore, "Factoring using 2n + 2 qubits with Toffoli based modular multiplication," *Quantum Inf. Comput.*, vol. 17, nos. 7/8, pp. 673–684, Jun. 2017, doi: 10.5555/3179553.3179560.
[28] M. Ekerå, "Revisiting Shor's quantum algorithm for computing general discrete logarithms," 2019, doi: 10.48550/arXiv.1905.09084.
[29] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, no. 18, 2017, Art. no. 180509, doi: 10.1103/PhysRevLett.119.180509.
[30] S. Endo, S. C. Benjamin, and Y. Li, "Practical quantum error mitigation for near-future applications," *Phys. Rev. X*, vol. 8, no. 3, 2018, Art. no. 031027, doi: 10.1103/PhysRevX.8.031027.
[31] M. Ekerå, "On post-processing in the quantum algorithm for computing short discrete logarithms," *Des. Codes Cryptography*, vol. 88, no. 11, pp. 2313–2335, Nov. 2020, doi: 10.1007/s10623-020-00783-2.
[32] M. S. Anis *et al.*, "Qiskit: An open-source framework for quantum computing," 2021, doi: 10.5281/zenodo.2562111.
[33] "IBM quantum experience." Accessed: Jul. 13, 2022. [Online]. Available: https://quantum-computing.ibm.com
[34] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, no. 6866, pp. 883–887, Dec. 2001, doi: 10.1038/414883a.
[35] C. Miquel, J. P. Paz, and R. Perazzo, "Factoring in a dissipative quantum computer," *Phys. Rev. A*, vol. 54, no. 4, 1996, Art. no. 2605, doi: 10.1103/PhysRevA.54.2605.
[36] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, Apr. 2021, Art. no. 433, doi: 10.22331/q-2021-04-15-433.
[37] J. Sevilla and C. J. Riedel, "Forecasting timelines of quantum computing," 2020, doi: 10.48550/arXiv.2009.05045.
[38] M. Mosca and M. Piani, "Quantum threat timeline." Accessed: Jul. 13, 2022. [Online]. Available: https://globalriskinstitute.org/publications/quantum-threat-timeline/
[39] M. Mosca and M. Piani, "Quantum threat timeline report," 2020. [Online]. Available: https://globalriskinstitute.org/publications/quantum-threat-timeline-report-2020/
[40] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring," 1994. [Online]. Available: https://books.google.co.jp/books?id=MZi8HgAACAAJ
[41] Y. Nam, Y. Su, and D. Maslov, "Approximate quantum Fourier transform with $O(n\log(n))T$ gates," *NPJ Quantum Inf.*, vol. 6, no. 1, pp. 1–6, Mar. 2020, doi: 10.1038/s41534-020-0257-5.
[42] L. Egan *et al.*, "Fault-tolerant control of an error-corrected qubit," *Nature*, vol. 598, pp. 281–286, 2021, doi: 10.1038/s41586-021-03928-y.
[43] N. Heninger and H. Shacham, "Reconstructing RSA private keys from random key bits," in *Proc. Annu. Int. Cryptol. Conf.*, 2009, pp. 1–17, doi: 10.1007/978-3-642-03356-8_1.
[44] M. Ekerå, "Modifying Shor's algorithm to compute short discrete logarithms," *IACR Cryptol. ePrint Arch.*, vol. 2016, 2016, Art. no. 1128.
[45] Z.-C. Duan *et al.*, "Proof-of-principle demonstration of compiled Shor's algorithm using a quantum dot single-photon source," *Opt. Exp.*, vol. 28, no. 13, pp. 18917–18930, Jun. 2020, doi: 10.1364/OE.390209.

Yoshinori Aono received the B.S. degree in engineering from the Musashi Institute of Technology, Tokyo, Japan, in 2005, and the M.S. and Ph.D. degrees in mathematical and computing sciences from the Tokyo Institute of Technology, Tokyo, in 2007 and 2010, respectively.

He is currently a Researcher with the National Institute of Information and Communications Technology (NICT), Tokyo. He is also an IAS Visiting Associate Professor with Yokohama National University, Yokohama, Japan. He is also a Visiting Researcher with the Keio Quantum Computing Center, Yokohama. In 2011, he joined NICT, where he conducts research on security analysis of cryptography.

Sitong Liu received the B.A. degree in environment and information studies from Keio University, Fujisawa, Japan, in 2021, where she is currently working toward the master's degree under the supervision of Prof. Van Meter.

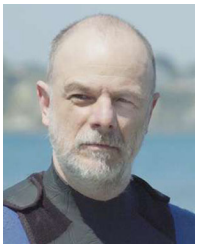
In 2019, she joined the Advancing Quantum Architecture Research Group, Keio University, Shonan Fujisawa Campus, Fujisawa. During her undergraduate study, she worked on quantum machine learning and quantum circuit optimization. Her research interests include quantum computing.

Tomoki Tanaka received the B.S. degree in science and the M.S. degree in mathematical sciences from Nagoya University, Nagoya, Japan, in 2009 and 2011, respectively, and the Ph.D. degree in quantum computing from Keio University, Tokyo, Japan, in 2022.

In 2011, he joined Mitsubishi UFJ Financial Group, Inc., Tokyo, where he is currently the Vice-President. His major is topology, especially knot theory. He joined the project IBM Quantum Network Hub @ Keio University as a Project Researcher of the Keio Quantum Computing Center, Yokohama, Japan. His research interests include quantum computing for using financial applications, such as derivative simulation, risk management, optimization, and machine learning.

Shumpei Uno received the M.Sc. and Ph.D. degrees in particle physics from Nagoya University, Nagoya, Japan, 2008 and 2011, respectively.

He is currently a Chief Consultant of the Mizuho Research and Technologies, Ltd. (MHRT), Tokyo, Japan. During the Ph.D. degree, he formulated quantum electrodynamics on finite volume lattice in order to accurately predict the light quark masses. In 2011, he joined MHRT. He became a Project Researcher of the Keio Quantum Computing Center, Yokohama, Japan, in 2018. His research interests include quantum computing for using financial applications, such as derivatives simulation, risk management, optimization, and machine learning.



Rodney Van Meter (Senior Member, IEEE) received the Ph.D. degree in computer science from Keio University, Tokyo, Japan, in 2006.

He is currently a Professor of Environment and Information Studies with Keio University, Shonan Fujisawa Campus, Fujisawa, Japan. He is also a Vice-Center Chair of the Keio Quantum Computing Center, Yokohama, Japan, a Board Member of the WIDE Project, and a Member of the Quantum Internet Task Force. His research interests include quantum networking, quantum computing, storage systems, networking, and post-Moore's law computer architecture.

Dr. Van Meter is a Member of the Association for Computing Machinery, the American Physical Society, and the American Association for the Advancement of Science.

IEEE Transactions on Quantum Engineering

Naoyuki Shinohara received the B.S., M.S., and Ph.D. degrees in mathematics from Kyushu University, Fukuoka, Japan, in 2002, 2004, and 2008, respectively.

He is currently a Research Manager of the National Institute of Information and Communications Technology, Tokyo, Japan.

Ryo Nojima received the Ph.D. degree in information processing from the Nara Institute of Science and Technology, Ikoma, Japan, in 2005.

He was a Postdoctoral Fellow with the University of Tokyo, Tokyo, Japan, in 2006. He is currently the Director of the National Institute of Information and Communications Technology, Tokyo.

Dr. Nojima was a Program Co-Chair of 2021 International Workshop on Security.