Supplementary Information
An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem

Jun Li, Xinhua Peng, Jiangfeng Du, Dieter Suter

I. GAUSS SUM EVALUATION

In the Gauss sum

$$G(a,\chi)=\sum_{m=0}^{N-1}\chi_N(m)e^{2\pi iam/N}, \tag{1}$$

the Jacobi symbol $\chi_N(m)$ is defined in terms of the Legendre symbols $\chi_{p_i}(m)$ with respect to the prime factors of N

$$\chi_N(m)=\chi_{p_1}(m)^{\alpha_1}\chi_{p_2}(m)^{\alpha_2}\cdots\chi_{p_k}(m)^{\alpha_k}. \tag{2}$$

Here, the Legendre symbol $\chi_{p_i}(m)$ for a prime number p_i is

$$\chi_{p_i}(m)=\begin{cases} 0 & if\; m\equiv 0(\mathrm{mod} p_i); \\ +1 & if\; m\not\equiv 0(\mathrm{mod} p_i)\;\mathrm{and}\; m=x^2(\mathrm{mod} p_i)\text{for some integer } x; \\ -1 & \text{otherwise.}\end{cases} \tag{3}$$

If N is square–free, the Jacobi symbol is a primitive Dirichlet character $modN$. As shown in Ref. [1], the Gauss sum then has the following important properties

$$G(a,\chi)=\varepsilon_N\chi(a)\sqrt{N}. \tag{4}$$

If, in addition, $(a,N)>1$, by definition of the Jacobi symbol we'll have

$$G(a,\chi)=0. \tag{5}$$

The general formula for the case that N is not square–free can be found in ref. [1] (p94, Exercise 12). Here, we only need the special case where $(a,N)=1$. Then, the general formula evaluates to

$$G(a,\chi)=0. \tag{6}$$

II. QUANTUM NETWORK FOR EXTENDED EUCLIDEAN ALGORITHM

In the main paper and methods, we point out that U_1 (greatest common divisor computation)

$$\frac{1}{\sqrt{N}}\sum_{m=0}^{N-1}|m\rangle\,|N\rangle \to \frac{1}{\sqrt{N}}\sum_{m=0}^{N-1}|m\rangle\,|(m,N)\rangle, \tag{7}$$

and the key part of U_2 (Jacobi symbol computation)

$$\frac{1}{\sqrt{\varphi(N)}}\sum_{(m,N)=1}|m\rangle\,|1\rangle \to \frac{1}{\sqrt{\varphi(N)}}\sum_{(m,N)=1}|m\rangle\,|\chi(m)\rangle. \tag{8}$$

can be efficiently solved based on the famous extended Euclid algorithm. There is a variant of this algorithm, called the binary extended Euclidean algorithm [2], which can be more conveniently performed on a binary computer. The classical binary algorithms for GCD and Jacobi symbol are presented in supplementary Fig. 1 and supplementary Fig. 2 respectively.

We now try to construct a quantum version of the binary algorithms. The classical algorithm contains several conditional statements that translate into conditional control operations in the quantum algorithm. For some of them auxiliary registers are needed. Take binary GCD algorithm for example, corresponding to the conditional statements in supplementary Fig. 1, we add three additional registers called “terminate or go on control register” (statement 7), “even or odd control register” (statement 8, 9) and “comparison control register” (statement 10). The three kinds of registers function as (1) “terminate or go on”: $0 \leftrightarrow v = 0$ and $1 \leftrightarrow v > 0$; (2) “even or odd”: $0 \leftrightarrow$ *odd* and $1 \leftrightarrow$ *even*; (3)”comparison”: $0 \leftrightarrow u < v$ and $1 \leftrightarrow u > v$ respectively. As illustrated in supplementary Fig. 3, the complete algorithm requires seven registers. Details of the construction are illustrated in supplementary Fig. 5. Like in the classical case, the quantum circuit complexity for the GCD problem is of the order of $O((\log N)^2)$.

Analogously, the quantum network for Jacobi symbol computation can be constructed. Note that the conditional statements 5, 9, 12 in supplementary Fig. 2 are easy to implement, because the modular properties of u and v with respect to 4 and 8 are only determined by their lowest digits. Besides, in the network, if $\chi(m) = -1$, we represent it by $r = N - 1$. The explicit network for Jacobi symbol computation is illustrated in supplementary Fig. 4 and supplementary Fig. 6.

III. COMPLEXITY ESTIMATION OF THE ALGORITHM

In the complexity estimation of the algorithm (see methods), we made two arguments

1. for an odd integer N, if Ω proceeds to M_2 with measurement result o and (o, N) is a square number, then (o, N) must be the square part s^2 of N;

2. for an odd integer N, application of Ω will directly yield the square part of N with probability larger than $(\varphi(N)/N)^2$.

Now we explain why the above arguments are valid.

A. Proof of argument *1*

We now prove that at measurement M_2, if (o, N) is a square number z^2, then it must be equal to s^2. To prove this, set $o = tz^2$ and $N = xz^2$.

We need to evaluate $G(tz^2, \chi)$. First it is rewritten as

$$G(tz^2,\chi) = \sum_{m=0}^{N-1} \chi(m) e^{2\pi i t z^2 m/N} = \sum_{(m,N)=1} \chi(m) e^{2\pi i t m/x}. \tag{9}$$

Let $m = kx + j$, where $0 \le k \le z^2 - 1$ and $0 \le j \le x - 1$ such that $(kx + j, N) = 1$. As

$$\chi(kx+j) = \left(\frac{kx+j}{xz^2}\right) = \left(\frac{kx+j}{x}\right) = \chi_x(j), \tag{10}$$

there is

$$G(tz^2,\chi) = \sum_{(j,N)=1} \left(\chi_x(j) e^{2\pi i t j/x} \cdot \sum_{(kx+j,N)=1} 1 \right). \tag{11}$$

Since for a certain j

$$\sum_{(kx+j,N)=1} 1 = \frac{\sum\limits_{(j,x)=1} 1 \cdot \sum\limits_{(kx+j,N)=1} 1}{\sum\limits_{(j,x)=1} 1} = \frac{\varphi(N)}{\varphi(x)}, \tag{12}$$

hence

$$G(tz^2,\chi) = \frac{\varphi(N)}{\varphi(x)} \sum_{j=0}^{x-1} \chi_x(j) e^{2\pi i t j/x} = \frac{\varphi(N)}{\varphi(x)} G(t,\chi_x). \tag{13}$$

According to the Gauss sum formula, $G(tz^2, \chi) \neq 0$ if and only if x is square-free. Then as the square-free decomposition is unique, x must be r and z^2 must be s^2.

B. Proof of argument *2*

For $o = ts^2$ and $(t, N) = 1$, we then have

$$G(ts^2, \chi) = \frac{\varphi(N)}{\varphi(r)} \sum_{j=0}^{r-1} \chi_r(j) e^{2\pi i t j/r} = \chi(t)\varepsilon_r \sqrt{r} \frac{\varphi(N)}{\varphi(r)}. \tag{14}$$

Now it is obvious that the probability of obtaining o such that $o = ts^2$ and $(t, N) = 1$ is

$$\frac{\sum_{(t,N)=1} \left|G(ts^2, \chi)\right|^2}{\varphi(N)N} = \frac{\varphi(r) r \varphi^2(N)}{\varphi(N)N\varphi^2(r)} = \frac{\varphi(N)r}{N\varphi(r)}. \tag{15}$$

As the probability that the algorithm not terminate at measurement M_1 is $(\varphi(N)/N)$, the probability that the algorithm proceeds to measurement M_2 and gives the square part s^2 would be

$$\frac{r}{\varphi(r)} \left(\frac{\varphi(N)}{N}\right)^2 \geq \left(\frac{\varphi(N)}{N}\right)^2 \tag{16}$$

IV. REFERENCES

[1] Cohen, H. *Number Theory, Volume I: Tools and Diophantine Equations* (Springer-Verlag, New York, 2007). 1

[2] Bach, E. & Shallit, J. *Algorithmic Number Theory, Vol. 1: Efficient Algorithms* (MIT Press, Cambridge, 1996).

2

Algorithm – *binary GCD algorithm.*

Input : *N* and *m.*
Output: *r*, such that *r*=(*m, N*).

```
1.  Set $u=N$, $v=m$, $r=1$;
2.  While ($u$ is even) and ($v$ is even) do {
3.      $u=u/2$;
4.      $v=v/2$;
5.      $r=2r$;
6.  }
7.  While $v \neq 0$ do {
8.      while ($u$ is even) do $u=u/2$;
9.      while ($v$ is even) do $v=v/2$;
10.     if $u>v$ then $u=(u-v)/2$;
11.     else $v=(v-u)/2$;
12. }
13. $r=u \cdot r$;
14. Return $r$.
```

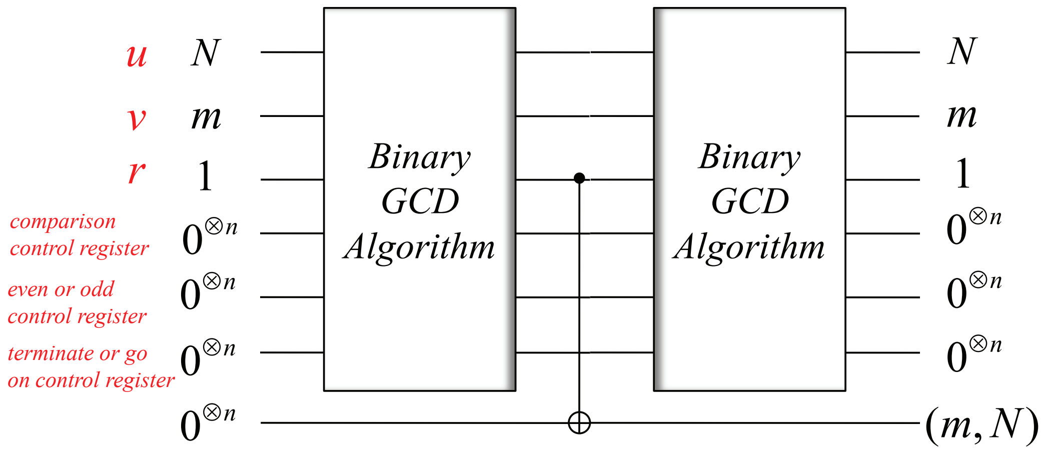
Supplementary Figure S1: Classical binary GCD algorithm for computing greatest common divisor.

Algorithm – *binary Jaocbi symbol algorithm.*

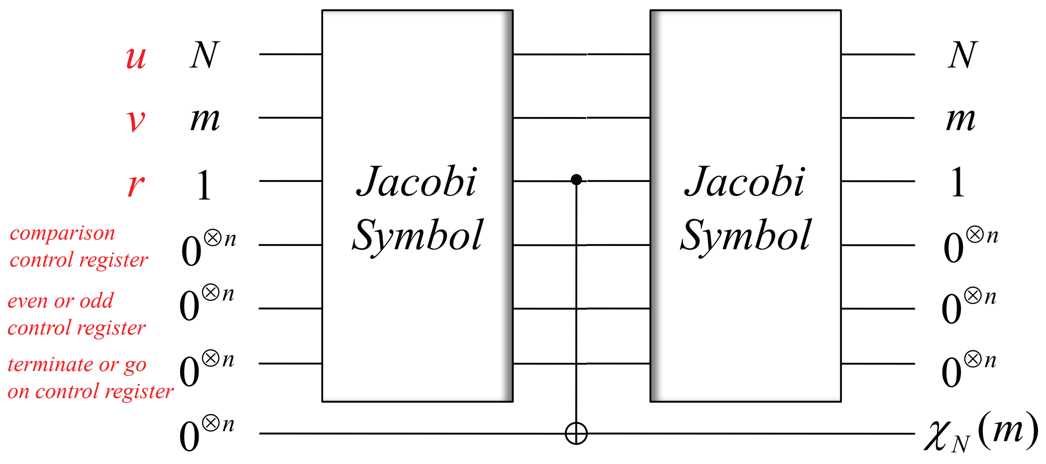
Input : N and m, N is odd, $(m,N)=1$.
Output: r, such that $r=\chi_N(m)$.

```
1.   Set $u=N$, $v=m$, $r=1$;
2.   While $u \neq 1$ do {
3.       While $v$ is even do {
4.           $v=v/2$;
5.           if ($u$ mod 8=3) or ($u$ mod 8=5) then $r=-r$;
6.       }
7.       if $v<u$ {
8.           swap $(u,v)$;
9.           if ($v$ mod 4=3) and ($u$ mod 4=3) then $r=-r$;
10.      }
11.      $v=(v-u)/2$;
12.      if ($u$ mod 8=3) or ($u$ mod 8=5) then $r=-r$;
13.  }
14.  Return $r$.
```

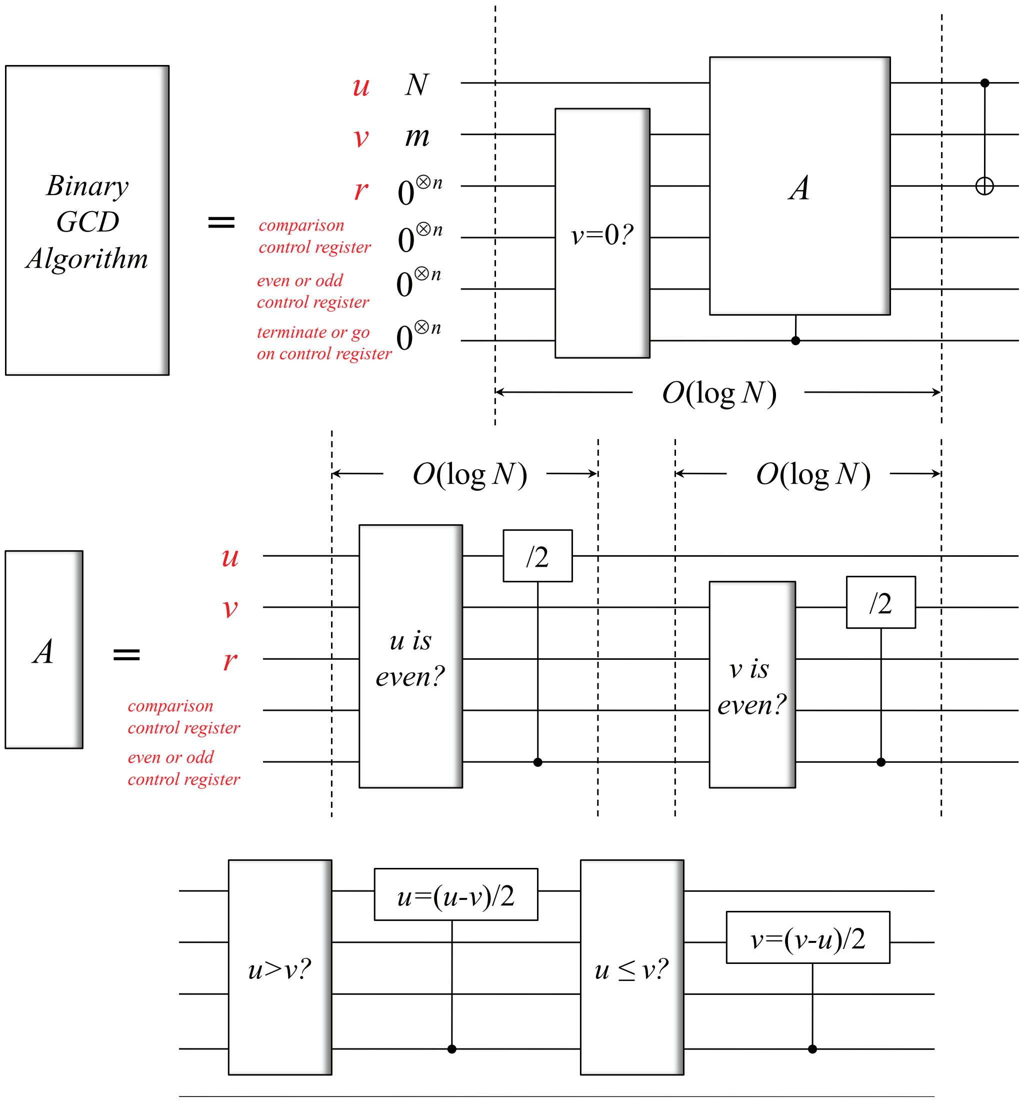
Supplementary Figure S2: Classical binary algorithm for computing Jacobi symbol.



Supplementary Figure S3: Outline of the circuit for binary GCD algorithm. This represents the quantum generalization of the classical binary GCD algorithm. Each horizontal line represents a quantum register with n qubits. The first three lines labeled u, v, r represent the corresponding variables from the classical algorithm, as represented in supplementary Fig. 1.

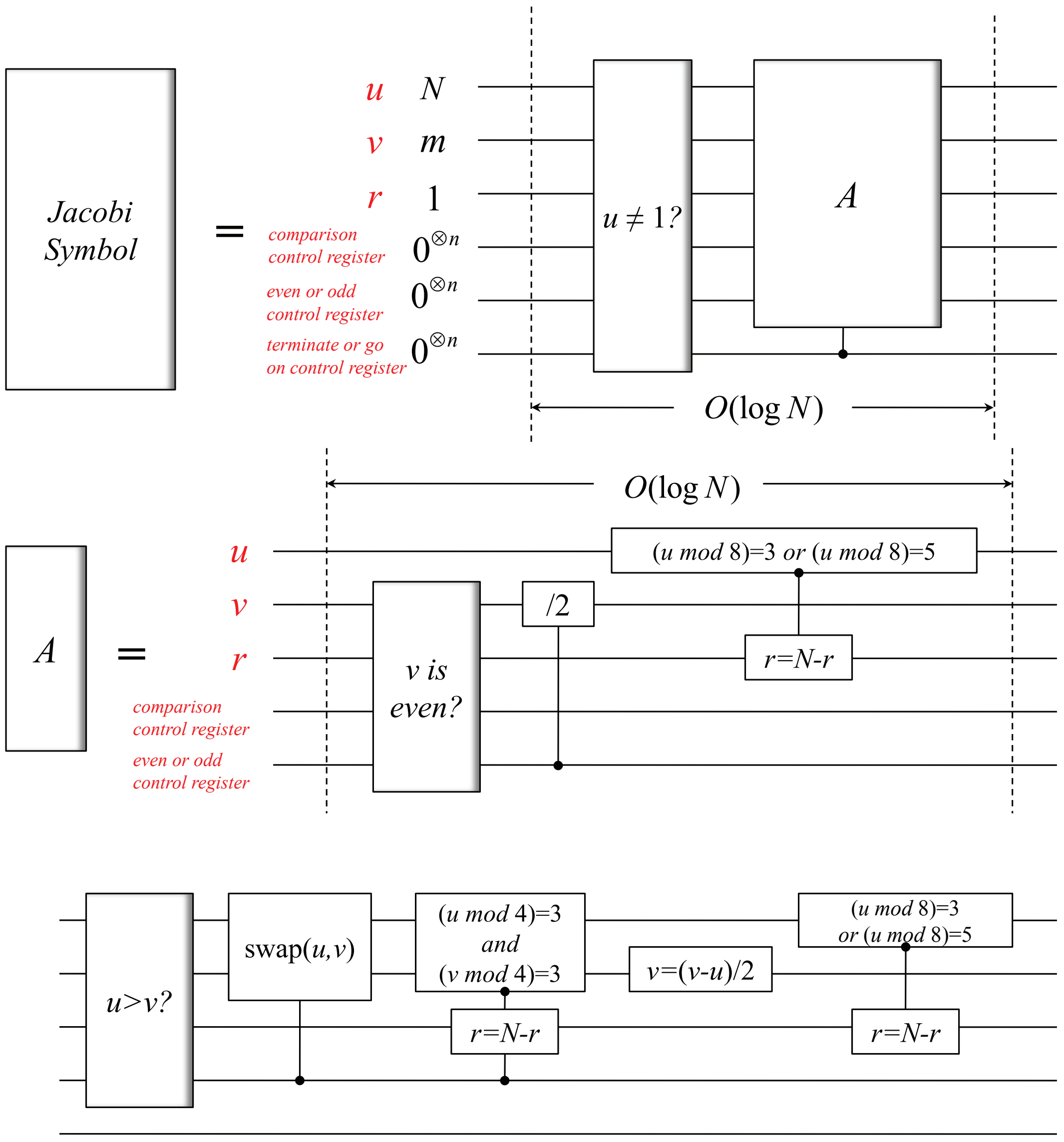


Supplementary Figure S4: Outline of the circuit for binary Jacobi symbol algorithm. This represents the quantum generalization of the classical binary Jacobi symbol algorithm. Each horizontal line represents a quantum register with n qubits. The first three lines labeled u, v, r represent the corresponding variables from the classical algorithm, as represented in supplementary Fig. 2.



Binary GCD Algorithm
=
u N
v m
r $0^{\otimes n}$
comparison control register $0^{\otimes n}$
even or odd control register $0^{\otimes n}$
terminate or go on control register $0^{\otimes n}$
v=0?
A
O(log N)
A
=
u
v
r
comparison control register
even or odd control register
O(log N)
O(log N)
u is even?
/2
v is even?
/2
u>v?
u=(u-v)/2
u ≤ v?
v=(v-u)/2

Supplementary Figure S5: **Quantum circuit for binary GCD algorithm.** The circuit is obtained through complete translation from the classical binary GCD algorithm.


Jacobi Symbol
=
u N
v m
r 1
comparison control register $0^{\otimes n}$
even or odd control register $0^{\otimes n}$
terminate or go on control register $0^{\otimes n}$
$u \neq 1?$
A
$O(\log N)$
A
=
u
v
r
comparison control register
even or odd control register
$O(\log N)$
v is even?
/2
(u mod 8)=3 or (u mod 8)=5
r=N-r
u>v?
swap(u,v)
(u mod 4)=3 and (v mod 4)=3
r=N-r
v=(v-u)/2
(u mod 8)=3 or (u mod 8)=5
r=N-r

Supplementary Figure S6: Quantum circuit for binary Jacobi symbol algorithm. The circuit is obtained through complete translation from the classical binary Jacobi symbol algorithm.