

Quantum Algorithms for Estimating Gauss Sums and Calculating Discrete Logarithms

Wim van Dam^{1*} and Gadiel Seroussi²

¹ MIT, Center for Theoretical Physics, 77 Massachusetts Avenue,
Cambridge, MA 02139-4307, USA; vandam@mit.edu

² HP Labs, Information Theory Research, 1501 Page Mill Road,
Palo Alto, CA 94304-1126, USA; seroussi@hpl.hp.com

Abstract. An efficient quantum algorithm for estimating Gauss sums over finite fields and finite rings is presented. Also a quantum algorithm for the discrete logarithm problem is described, which is different from Shor's algorithm. A reduction from the discrete logarithm problem to Gauss sum estimation gives evidence that the latter is classically a hard problem. A crucial ingredient of the algorithms is a quantum state that needs to be constructed before we can perform the computation. After one copy of this state is created, the algorithm can be executed arbitrarily many times.

1 Introduction

The theory of quantum computation investigates how the laws of quantum physics allow us to process information in a more efficient way than is possible in the classical, Turing machine model of computation. Strong support for the claim that quantum computers are indeed more powerful than classical ones was given in 1994 by Peter Shor who proved the existence of efficient quantum algorithms for factoring and the discrete logarithm problem [13]. More recently, Hallgren showed that also Pell's equation can be solved in polynomial time on a quantum computer [8]. Currently it is not well understood for which other problems quantum computers are more efficient than classical computers. Hence the design of new and useful quantum algorithms has become an important goal among researchers in theoretical computer science and quantum physics. See [11] by Nielsen and Chuang for a thorough introduction to this field.

Let $\chi : R \rightarrow \mathbb{C}$ be a multiplicative character and $e : R \rightarrow \mathbb{C}$ an additive character over a finite ring R . The *Gauss sum* G of this triplet (R, χ, e) is the inner product between χ and e , that is: $G(R, \chi, e) := \sum_{x \in R} \chi(x)e(x)$. Gauss sums are useful on many fronts for the analysis of finite fields $R = \mathbb{F}_p$ and rings $R = \mathbb{Z}/n\mathbb{Z}$. In combination with the closely related *Jacobi sums*, they have been

* WvD's work is supported in part by funds provided by the U.S. Department of Energy (DOE) and cooperative research agreement DF-FC02-94ER40818, and by a CMI postdoctoral fellowship. Part of this work was done while WvD was at HP Labs and MSRI supported by an HP-MSRI postdoctoral fellowship.

used to prove theorems about finite field equations, difference sets, primality testing, zeta-functions, et cetera. One can view Gauss sums as the finite versions of the gamma function $\Gamma(s) := \int_0^\infty x^{s-1} e^{-x} dx$, where x^{s-1} is a multiplicative, and e^{-x} is an additive character over \mathbb{R} . See Brendt et al. [2] for a book entirely devoted to these topics.

In Section 5 of this article we describe a quantum algorithm that, given the specification of the characters χ and e over a finite field \mathbb{F}_{p^r} efficiently approximates the corresponding Gauss sum G . Because determining the norm $|G|$ of a Gauss sum is straightforward, our algorithm focuses on estimating the angle $\gamma \bmod 2\pi$ in the equation $G = |G| \cdot e^{i\gamma}$. We describe a quantum transform that induces this angle as a relative phase in a mapping $|0\rangle + |1\rangle \mapsto |0\rangle + e^{i\gamma}|1\rangle$. Because this transformation can be implemented efficiently, we can sample this transformation $O(\frac{1}{\varepsilon})$ times to get an estimation $\tilde{\gamma}$ of the angle γ with expected error ε . The time complexity of this algorithm is thus $O(\frac{1}{\varepsilon} \cdot \text{poly}(\log p^r))$. Using a reduction from the discrete log problem to the approximation of Gauss sums in Section 6, we provide evidence that this is a hard task on a classical computer.

An important part of the Gauss sum estimation algorithm is the preparation of a quantum state $\sum_x \chi(x)|x\rangle$ that is defined by the values of the multiplicative character χ . With the use of this state it is also possible to efficiently solve the discrete logarithm problem over R , as explained in Section 4. The preparation of this ‘chi state’ should be viewed as a form of preprocessing because the state does not get destroyed during the computation (see Section 3). The Gauss sum estimation algorithm over finite fields can be used to estimate Jacobi sums and Gauss sums over finite rings $\mathbb{Z}/n\mathbb{Z}$. This is done in Section 7.

There are several predecessors to this article. The fact that the angle of Gauss sum appears as a general phase after one applies a quantum Fourier transform to a chi state was already mentioned in [5]. In the report [6] we described a quantum algorithm for estimating Gauss sums that uses the Shor’s discrete logarithm algorithm. The properties of the multiplicative χ state were used earlier by Watrous in [14].

In this article we use the χ state to construct an alternative quantum algorithm for the discrete logarithm problem (Algorithm 2 in Section 4). Our algorithm uses two quantum registers and two Fourier transforms whereas Shor’s algorithm requires three registers and four Fourier transforms. The chi state is also used in the Gauss sum estimation algorithm in Section 5. This algorithm is an improvement over the algorithm of [6] as it does not rely anymore on Shor’s discrete logarithm algorithm, which reduces its error rate and makes it more time and space efficient as well. In Lemma 1 we present a new implementation of the Fourier transform over \mathbb{F}_{p^r} with a circuit depth that is identical to the circuit depth of the Fourier transform over \mathbb{F}_p . Throughout the article, results that are already known are indicated as ‘facts’.

2 Background and Definitions

2.1 Characters, Discrete Logarithms and Gauss Sums over \mathbb{F}_{p^r}

Let ζ_p denote a p th root of unity $\zeta_p := e^{2\pi i/p}$. The *trace* of an element x of the finite field \mathbb{F}_{p^r} over \mathbb{F}_p is $\text{Tr}(x) := x + x^p + \cdots + x^{p^{r-1}}$. It can be shown that for every $x \in \mathbb{F}_{p^r}$, its trace is an element of the base-field: $\text{Tr}(x) \in \mathbb{F}_p$. For any $\beta \in \mathbb{F}_{p^r}$ we also have the related functions $x \mapsto \text{Tr}(\beta x)$. These trace functions are all the linear functions $\mathbb{F}_{p^r} \rightarrow \mathbb{F}_p$ (note that $\beta = 0$ gives the trivial function 0). When we write $\zeta_p^{\text{Tr}(x)}$ we interpret the value $\text{Tr}(x)$ as an element of the set $\{0, 1, \dots, p-1\} \subset \mathbb{Z}$. For $\beta \in \mathbb{F}_{p^r}$, the functions $e_\beta(x) := \zeta_p^{\text{Tr}(\beta x)}$ describe all possible additive characters $\mathbb{F}_{p^r} \rightarrow \mathbb{C}$.

A *multiplicative character* of a finite field is a function $\chi : \mathbb{F}_{p^r} \rightarrow \mathbb{C}$ such that $\chi(xy) = \chi(x)\chi(y)$ for all $x, y \in \mathbb{F}_{p^r}$. Let g be a primitive element of \mathbb{F}_{p^r} , i.e. the multiplicative group $\langle g \rangle$ generated by g equals $\mathbb{F}_{p^r}^\times := \mathbb{F}_{p^r} \setminus \{0\}$. For each $0 \leq \alpha \leq p^r - 2$, the function $\chi(g^j) := \zeta_{p^r-1}^{\alpha j}$ (complemented with $\chi(0) := 0$) is a multiplicative character $\mathbb{F}_{p^r} \rightarrow \mathbb{C}$. Also, every multiplicative character can be written as such a function. The *discrete logarithm* with respect to g is defined for every $x = g^j \in \mathbb{F}_{p^r}^\times$ by $\log_g(x) := j \bmod p^r - 1$. Hence, every multiplicative character can be expressed as $\chi(x) := \zeta_{p^r-1}^{\alpha \log_g(x)}$ for $x \neq 0$ and $\chi(0) := 0$. The trivial multiplicative character is denoted by $\chi^{(0)}$ and is defined by $\chi^{(0)}(0) = 0$ and $\chi^{(0)}(x) = 1$ for all $x \neq 0$. Using the equality $\chi^{(\alpha)}(x)\chi^{(\beta)}(x) = \zeta_{p^r-1}^{\alpha \log_g(x)} \zeta_{p^r-1}^{\beta \log_g(x)} = \zeta_{p^r-1}^{(\alpha+\beta) \log_g(x)} = \chi^{(\alpha+\beta)}(x)$, it is easy to see that the set of characters $\{\chi^{(\alpha)} \mid \alpha \in \{0, \dots, p^r - 2\}\}$ with pointwise multiplication defines a group, which we will denote by $\hat{\mathbb{F}}_{p^r}^\times$. The relation $\chi^{(\alpha)} \cdot \chi^{(\beta)} = \chi^{(\alpha+\beta)}$ thus establishes the isomorphism $\hat{\mathbb{F}}_{p^r}^\times \cong \mathbb{Z}/(p^r - 1)\mathbb{Z}$. By $\chi^{(\alpha)}(x) = (\chi^{(1)}(x))^\alpha$, the index α can also be viewed as a power, which allows us to drop the parentheses around the indices of the characters and write simply χ^α . The inverse of χ obeys $\chi^{(-1)}(x) = \overline{\chi(x)}$ for all x and where \bar{z} denotes the complex conjugate of z .

Definition 1 (Gauss sums over finite fields). For a finite field \mathbb{F}_{p^r} , a multiplicative character χ , and an additive character e_β , we define the complex valued Gauss sum G by

$$G(\mathbb{F}_{p^r}, \chi, \beta) := \sum_{x \in \mathbb{F}_{p^r}} \chi(x) \zeta_p^{\text{Tr}(\beta x)}.$$

Obviously, $G(\mathbb{F}_{p^r}, \chi^{(0)}, 0) = p^r - 1$, $G(\mathbb{F}_{p^r}, \chi^{(0)}, \neq 0) = -1$, and $G(\mathbb{F}_{p^r}, \chi, 0) = 0$ for $\chi \neq \chi^{(0)}$. In general, for $\beta \neq 0$ we have the following fact.

Fact 1. For $\beta \neq 0$ it holds that $G(\mathbb{F}_{p^r}, \chi, \beta\delta) = \chi(\beta^{-1})G(\mathbb{F}_{p^r}, \chi, \delta)$.

Proof. For $G(\mathbb{F}_{p^r}, \chi, \beta\delta)$ we have $\sum_x \chi(x) \zeta_p^{\text{Tr}(\beta\delta x)} = \chi(\beta^{-1}) \sum_z \chi(z) \zeta_p^{\text{Tr}(\delta z)}$, with the summations over \mathbb{F}_{p^r} and where we used the substitution $x \leftarrow z\beta^{-1}$ and the multiplicativity of χ . \square

From now on we will assume that the Gauss sum concerns nontrivial characters $\chi \neq \chi^0$ and $\beta \neq 0$. It is known that the norm of a Gauss sum obeys $|G(\mathbb{F}_{p^r}, \chi, \beta)| = \sqrt{p^r}$.

2.2 Discrete Logarithm and Approximate Gauss Sum Problem

The discrete logarithm problem is a well-known problem in number theory. Part of its importance lies in the presumed hardness of the problem, which makes it a useful primitive in cryptography.[9]

Definition 2 (Discrete logarithm problem over finite fields). *Let \mathbb{F}_{p^r} be a finite field with a known generator g . Given a value $x \in \mathbb{F}_{p^r}^\times$, the base- g discrete logarithm problem asks for the value $\log_g(x)$, which is the integer $0 \leq j \leq p^r - 2$ such that $g^j = x$.*

The input of a discrete logarithm problem is \mathbb{F}_{p^r} , g and x , hence the input length is $O(\log p^r)$.

If we want to define the problem of estimating Gauss sums as a computational task, we have to make clear what the length of the input is. As stated above, any multiplicative character χ can be described by a triplet (p^r, g, α) , where $g \in \mathbb{F}_{p^r}^\times$ is a generator of $\mathbb{F}_{p^r}^\times$ and $\alpha \in \mathbb{Z}/(p^r - 1)\mathbb{Z}$ the index of χ . As a result, the specification of the problem “What is $G(\mathbb{F}_{p^r}, \chi, \beta)$?” as defined below, requires no more than $O(\log p^r)$ bits of information.

Definition 3 (Gauss sum problem over finite fields). *Let \mathbb{F}_{p^r} be a finite field, χ a nontrivial character over \mathbb{F}_{p^r} and $\beta \in \mathbb{F}_{p^r}^\times$. What is (approximately) the angle $\gamma \bmod 2\pi$ in the Gauss sum equation $G(\mathbb{F}_{p^r}, \chi, \beta) = \sqrt{p^r} \cdot e^{i\gamma}$?*

A quadratic character is a nontrivial χ such that $\chi(x) \in \{0, 1, -1\}$ for all x . By the isomorphism $\hat{\mathbb{F}}_{p^r}^\times \cong \mathbb{Z}/(p^r - 1)\mathbb{Z}$ one sees that such a character is only possible if p is odd and where χ is defined by $\chi(g^j) = (-1)^j$. Unlike the case of general characters, the Gauss sums of such quadratic characters are known completely: $G(\mathbb{F}_{p^r}, \chi, 1) = -(-1)^r \sqrt{p^r}$ if $p \equiv 1 \pmod{4}$, and $G(\mathbb{F}_{p^r}, \chi, 1) = -(-i)^r \sqrt{p^r}$ if $p \equiv 3 \pmod{4}$. (See Theorem 11.5.4 in [2] for a proof.)

2.3 Quantum States and Quantum Transformations

A quantum state ψ of n quantum bits (qubits) is described by a 2^n dimensional complex valued vector, which represents a superposition over all possible n bit strings $\{0, 1\}^n$. In the ‘bra-ket’ notation this is expressed as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle,$$

with $\alpha_x \in \mathbb{C}$ and the normalization restriction $\sum_x |\alpha_x|^2 = 1$. When observing this state ψ in the computational basis $\{0, 1\}^n$, the probability of observing a specific string $x \in \{0, 1\}^n$ is $|\alpha_x|^2$ (hence the normalization restriction).

A coherent quantum mechanical transformation T can always be described by a linear transformation that preserves the normalization restriction of the state vectors. Thus the transformation T of an n qubit system can be represented as a $2^n \times 2^n$ dimensional unitary matrix: $T \in \mathbf{U}(2^n)$. Any such transformation is reversible. A local, k -qubit quantum gate G is a unitary transformation of k qubits, such that $G \in \mathbf{U}(2^k)$.

2.4 Efficient Quantum Algorithms

A quantum circuit $C \in \mathbf{U}(2^n)$ on n qubits can be defined as sequence of 2 qubit gates (note that this includes single qubit gates and the two-qubit SWAP operation for the wiring of the circuit). The outcome of a circuit on a given input string $x \in \{0, 1\}^n$ is the probability distribution of the output state $C|x\rangle$ over the computational basis states $\{0, 1\}^n$. The depth of such a circuit equals its time complexity and we consider a circuit efficient if its depth is upper bounded by $O(\text{poly}(\log n))$. Almost all quantum algorithms known to date rely on the properties and the efficiency of the quantum Fourier transform.

Definition 4 (Efficient quantum Fourier transform over \mathbb{F}_{p^r}). Let $\beta \in \mathbb{F}_{p^r}^\times$. The quantum Fourier transform F_β over the finite field \mathbb{F}_{p^r} is defined as the unitary mapping

$$F_\beta : |x\rangle \mapsto \frac{1}{\sqrt{p^r}} \sum_{y \in \mathbb{F}_{p^r}} \zeta_p^{\text{Tr}(\beta xy)} |y\rangle$$

for every $x \in \mathbb{F}_{p^r}$. When $\beta = 1$, we will simply write F .

It is well-known that this quantum Fourier transform can be implemented efficiently (in $\text{poly}(\log p^r)$ time) on a quantum computer. Similarly, the Fourier transform over the group $\mathbb{Z}/n\mathbb{Z}$ can also be executed in an efficient way.

Usually, we would express the elements of the finite field \mathbb{F}_{p^r} as r -dimensional vectors in \mathbb{F}_p with a basis $\{B_1, \dots, B_r\}$. If $\text{Tr}(B_i B_j) = \delta_{ij}$ for all i, j , then we call this basis *self-dual*. If p is even, or if both p and r are odd, then \mathbb{F}_{p^r} has a self-dual basis. If p is odd and r is even then there exists an ‘almost’ self-dual basis with $\text{Tr}(B_i B_j) = \delta_{ij}$ except for $\text{Tr}(B_1 B_1) = c$ with $c \notin \{0, 1\}$. [12]

When we do our computations in a self-dual basis the implementation of the Fourier transform over \mathbb{F}_{p^r} is much more simple because we no longer have to compute the trace function. Instead it is sufficient to compute, in parallel, the Fourier transforms over \mathbb{F}_p on the r components of the x register. We thus have the following result.

Lemma 1 (Parallel Fourier transform over \mathbb{F}_{p^r}). Using a self-dual or almost self dual basis of \mathbb{F}_{p^r} , the Fourier transform over a finite field \mathbb{F}_{p^r} can be implemented by a parallel circuit with width $O(\log(p^r))$ and depth equal to the depth of a Fourier transform over \mathbb{F}_p .

Proof. Assume that \mathbb{F}_{p^r} has a self-dual basis $\{B_1, \dots, B_r\}$. We thus express the $x = x_1 B_1 + \dots + x_r B_r \in \mathbb{F}_{p^r}$ as strings $(x_1, \dots, x_r) \in \mathbb{F}_p^r$. Because $\text{Tr}(B_i B_j) = \delta_{ij}$ and the linearity of Tr , it is straightforward to see that for the Fourier transform of x we have

$$\begin{aligned} F : |x_1, \dots, x_r\rangle &\mapsto \frac{1}{\sqrt{p^r}} \sum_{y \in \mathbb{F}_{p^r}} \zeta_p^{\text{Tr}(xy)} |y\rangle \\ &= \frac{1}{\sqrt{p^r}} \sum_{(y_1, \dots, y_r) \in \mathbb{F}_p^r} \zeta_p^{x_1 y_1 + \dots + x_r y_r} |y_1, \dots, y_r\rangle \\ &= \frac{1}{\sqrt{p^r}} \bigotimes_{j=1}^r \zeta_p^{x_j y_j} |y_j\rangle. \end{aligned}$$

In other words: the Fourier transform over \mathbb{F}_{p^r} of $x = (x_1, \dots, x_r)$ can be calculated by performing r parallel and independent Fourier transforms over \mathbb{F}_p of the entries $x_j \in \mathbb{F}_p$. If \mathbb{F}_{p^r} does not have a self-dual basis, we can use an almost self-dual basis to obtain the same complexity. \square

For general p^r or n , it is not known how to implement the quantum Fourier transform exactly. Hence, the following results and the estimation algorithms of the next sections should be understood as approximate algorithms that can be made arbitrarily accurate. Sometimes we use the hat notation in $F : |\psi\rangle \mapsto |\hat{\psi}\rangle$ to denote the Fourier transform of a state.

Fact 2 (Quantum phase estimation). *Let γ be an unknown phase mod 2π of the qubit state $|x_\gamma\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\gamma}|1\rangle)$. If we measure this qubit in the orthogonal basis $|m_\phi\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\phi}|1\rangle)$ and $|m_\phi^\perp\rangle := \frac{1}{\sqrt{2}}(|0\rangle - e^{i\phi}|1\rangle)$, then the respective outcome probabilities are $\text{Prob}(m_\phi|x_\gamma) = \frac{1}{2} + \frac{1}{2} \cos(\gamma - \phi)$ and $\text{Prob}(m_\phi^\perp|x_\gamma) = \frac{1}{2} - \frac{1}{2} \cos(\gamma - \phi)$. Similarly, from the state $\frac{1}{\sqrt{t+1}} \sum_{j=0}^t e^{i\gamma j} |j\rangle$ we can efficiently obtain an estimate $\tilde{\gamma}$ of the unknown γ within an expected error of $O(\frac{1}{t})$ if we measure in the ‘Fourier basis’ $|\hat{k}\rangle := \frac{1}{\sqrt{t+1}} \sum_{j=0}^t \zeta_{t+1}^{kj} |j\rangle$ for $k = 0, 1, \dots, t$. See [3] for more details.*

3 Chi States, Their Properties and Preparation

Before we can describe the quantum algorithms in this article, we first will define the ‘chi states’, its properties and show how these state can be prepared in an efficient way.

Definition 5 (Chi states). *Given a finite field \mathbb{F}_{p^r} and a generator $g \in \mathbb{F}_{p^r}^\times$ we define the chi state by*

$$|\chi\rangle := \frac{1}{\sqrt{p^r - 1}} \sum_{j=0}^{p^r-2} \zeta_{p^r-1}^j |g^j\rangle,$$

where χ refers to the multiplicative character defined by g in $\chi(g^j) = \zeta_{p^r-1}^j$. For every $\alpha \in \mathbb{Z}/(p^r - 1)\mathbb{Z}$ we also define the α -th power of the chi state by

$$|\chi^\alpha\rangle := \frac{1}{\sqrt{p^r - 1}} \sum_{j=0}^{p^r-2} \zeta_{p^r-1}^{\alpha j} |g^j\rangle.$$

Note that $|\chi^0\rangle$ is the uniform superposition of the elements of $\mathbb{F}_{p^r}^\times$.

The following division operation can be performed efficiently under the assumption that multiplication and division in \mathbb{F}_{p^r} can be done efficiently. (Which implies that using repeated powering $x \mapsto x^2 \mapsto x^4 \dots$, we can efficiently calculate any power x^j for $-p^r < j < p^r$.)

Definition 6 (Division operator). For a finite field \mathbb{F}_{p^r} , the following two reversible ‘division operators’ are defined:

$$D^\alpha : |x, y\rangle \mapsto |x, y/x^\alpha\rangle \quad \text{and} \quad D_x : |\alpha, y\rangle \mapsto |\alpha, y/x^\alpha\rangle,$$

for all $x, y \in \mathbb{F}_{p^r}^\times$ and $\alpha \in \mathbb{Z}/(p^r - 1)\mathbb{Z}$.

Using the D^α operation, chi states can be copied to arbitrary χ^α states. It is straightforward to check that if we apply a D^α operation to a state $|g^s\rangle|\chi\rangle$ we will induce the phase change $|g^s, \chi\rangle \mapsto \zeta_{p^r-1}^{\alpha s} |g^s, \chi\rangle$. Hence, if we apply D^α to a uniform superposition of $\mathbb{F}_{p^r}^\times$ and a χ -state, we obtain a new χ^α state without losing the original $|\chi\rangle$:

$$D^\alpha : \frac{1}{\sqrt{p^r - 1}} \sum_{x \in \mathbb{F}_{p^r}^\times} |x\rangle|\chi\rangle \mapsto |\chi^\alpha\rangle|\chi\rangle.$$

In general we have in fact the mapping $D^\alpha : |\chi^\beta\rangle|\chi^\gamma\rangle \mapsto |\chi^{\beta+\alpha\gamma}\rangle|\chi^\gamma\rangle$. Under the assumption that it is easy to create the uniform superposition $|\chi^0\rangle$, we thus see that we can efficiently create arbitrary $|\chi^\alpha\rangle$ states, as soon as we have an initial state $|\chi\rangle$. To create the first chi state, we use the following zero error procedure.

Algorithm 1 (Chi state preparation algorithm). Let g be the generator of the group $\mathbb{F}_{p^r}^\times = \{g, g^2, \dots, g^{p^r-1} = g^0 = 1\}$, which defines the multiplicative character χ .

1. Initialize two $\log(p^r - 1)$ qubit registers to $|0, 0\rangle$ and apply the Fourier transform over $\mathbb{Z}/(p^r - 1)\mathbb{Z}$ to the left one. Next, calculate in the right register powers g^j where the exponent j is read from the left register. This step gives the transformation

$$|0, 0\rangle \mapsto \frac{1}{\sqrt{p^r - 1}} \sum_{j=0}^{p^r-2} |j, g^j\rangle.$$

2. Apply the Fourier transform over $\mathbb{Z}/(p^r - 1)\mathbb{Z}$ to the first register:

$$F \otimes I : \frac{1}{\sqrt{p^r - 1}} \sum_{j=0}^{p^r - 2} |j, g^j\rangle \mapsto \frac{1}{p^r - 1} \sum_{k, j=0}^{p^r - 2} \zeta_{p^r - 1}^{jk} |k, g^j\rangle.$$

Note that this state equals $\sum_k |k, \chi^k\rangle / \sqrt{p^r - 1}$.

3. Measure the k -register. If $\gcd(k, p^r - 1) \neq 1$, go back to step 1 and repeat the protocol. Otherwise, continue with the state $|k, \chi^k\rangle$.
4. Clear the k register and replace it with the uniform superposition of elements of G such that we obtain the state $|\chi^0, \chi^k\rangle$.
5. Apply $D^{(1/k)}$ (as $1/k := k^{-1}$ is well-defined in $\mathbb{Z}/(p^r - 1)\mathbb{Z}$), such that we get the transformation $|\chi^0, \chi^k\rangle \mapsto |\chi^1, \chi^k\rangle$. Remove the right register, yielding the desired $|\chi\rangle$.

All steps in the above algorithm can be done in time $\text{poly}(\log m)$. The probability that the observed k in Step 1 is co-prime with m is $\phi(m)/m$, which is lower bounded by $\Omega(1/\log(\log m))$. Hence the expected number of times that we have to repeat the algorithm until we reach Step 4 is $O(\log(\log m))$. In all, and assuming that we can perform the Fourier transform exactly, this shows that this algorithm produces the state $|\chi\rangle$ with zero error probability and has expected running time $\text{poly}(\log m)$. Using amplitude amplification [1] and knowledge about $\phi(m)$ we could make this algorithm exact, but because we need to prepare $|\chi\rangle$ only once, we do not bother. (Note again that copying the χ -state via the operation $D^1 : |\chi^0, \chi\rangle \mapsto |\chi, \chi\rangle$ is deterministic and more simple than the just described chi preparation algorithm.)

4 Alternative Discrete Logarithm Quantum Algorithm

Here we present a quantum algorithm for the discrete logarithm problem that uses a preprocessed chi state of the previous section. The algorithm uses two quantum registers and two Fourier transforms whereas Shor's original algorithm requires three registers and four Fourier transforms.

The crucial property of the chi state that we will use in the logarithm algorithm is its phase changing behavior when we apply D_x to it. Given an element $x = g^j \in G$, the D_x transform on $|\alpha\rangle$ and $|\chi\rangle$ has the following effect (which is shown with the help of the equality $\sum_k \zeta_{p^r - 1}^k |g^k/g^{j\alpha}\rangle = \sum_k \zeta_{p^r - 1}^{\alpha j + k} |g^k\rangle$):

$$D_x : |\alpha\rangle|\chi\rangle \mapsto \zeta_{p^r - 1}^{\alpha j} |\alpha\rangle|\chi\rangle, \quad (1)$$

with $j := \log_g x$. This 'multiplicative phase kick-back trick' (cf. [3] for the additive version) has been described earlier by Watrous in [14]. Here we use it to calculate $\log_g x$ in the following algorithm.

Algorithm 2 (Discrete logarithm algorithm). Given the generator g , a state $|\chi\rangle$ and the input value $x = g^j$, perform the following 3 steps.

1. Create a uniform superposition of α 's by applying the Fourier transform over $\mathbb{Z}/(p^r - 1)\mathbb{Z}$ to 0:

$$|0\rangle \mapsto \frac{1}{\sqrt{p^r - 1}} \sum_{\alpha=0}^{p^r-2} |\alpha\rangle.$$

2. With the χ state as the second register, apply the D_x transform to this superposition, giving (see Equation 1):

$$D_x : \frac{1}{\sqrt{p^r - 1}} \sum_{\alpha=0}^{p^r-2} |\alpha\rangle |\chi\rangle \mapsto \frac{1}{\sqrt{p^r - 1}} \sum_{\alpha=0}^{p^r-2} \zeta_{p^r-1}^{\alpha \log_g(x)} |\alpha\rangle |\chi\rangle.$$

3. Recover the logarithm j by applying an inverse Fourier transform over $\mathbb{Z}/(p^r - 1)\mathbb{Z}$ to the first register, yielding the final state $|\log_g(x)\rangle |\chi\rangle$.

The complexity of the algorithm consists of two Fourier transforms over $\mathbb{Z}/(p^r - 1)\mathbb{Z}$ and one implementation of D_x , which can all be done in time $\text{poly}(\log p^r)$. If these transformations are performed perfectly and the state χ is exact, then the above algorithm finds the discrete logarithm $\log_g(x)$ with probability 1. Note also that the chi state did not get destroyed in the computation, and hence can be reused.

5 Estimating Gauss Sums over Finite Fields

With the ingredients of the last two sections, we are now ready to describe the quantum algorithm that estimates the angle γ of the Gauss sum $G = |G| \cdot e^{i\gamma}$ over finite fields.

5.1 Quantum Algorithm for Gauss Sum Estimation

The crucial part of our algorithm relies on the interaction between the Fourier transform F_β and the multiplicative character χ . Using the fact that for nontrivial characters $\hat{\chi} = G(\mathbb{F}_{p^r}, \chi, \beta) / \sqrt{p^r} \cdot \bar{\chi}$, we are able to perform a γ -phase change. By sampling this unknown phase factor we can obtain an arbitrary precise estimation of γ and thus of $G(\mathbb{F}_{p^r}, \chi, \beta)$.

Algorithm 3 (Gauss sum phase change algorithm). Consider a finite field \mathbb{F}_{p^r} , a nontrivial character χ and a $\beta \in \mathbb{F}_{p^r}^\times$. By applying the quantum Fourier transform F_β over this field to the state $|\chi\rangle$, followed by a base change $|y\rangle \mapsto |y^{-1}\rangle$, we generate an overall phase change according to

$$|\chi\rangle := \frac{1}{\sqrt{p^r - 1}} \sum_{x \in \mathbb{F}_{p^r}} \chi(x) |x\rangle \mapsto \frac{G(\mathbb{F}_{p^r}, \chi, \beta)}{\sqrt{p^r}} |\chi\rangle.$$

Proof. First, we note that the output after the Fourier transform F_β looks like

$$|\hat{\chi}\rangle := \frac{1}{\sqrt{p^r(p^r-1)}} \sum_{y \in \mathbb{F}_{p^r}} \left(\sum_{x \in \mathbb{F}_{p^r}} \chi(x) \zeta_p^{\text{Tr}(\beta xy)} \right) |y\rangle.$$

The expression between the big parentheses equals $G(\mathbb{F}_{p^r}, \chi, \beta y)$, which equals $\chi(y^{-1})G(\mathbb{F}_{p^r}, \chi, \beta)$ for $y \neq 0$ and is zero if $y = 0$. In sum, we thus see that

$$|\hat{\chi}\rangle = \frac{G(\mathbb{F}_{p^r}, \chi, \beta)}{\sqrt{p^r(p^r-1)}} \sum_{y \in \mathbb{F}_{p^r}^\times} \chi(y^{-1})|y\rangle,$$

such that indeed after $|y\rangle \mapsto |y^{-1}\rangle$ we have created $(G(\mathbb{F}_{p^r}, \chi, \beta)/\sqrt{p^r})|\chi\rangle$.

By Algorithm 1, we know that the preparation of $|\chi\rangle$ can be done efficiently. Also the Fourier transform over \mathbb{F}_{p^r} (Lemma 1) and the base change $|y\rangle \mapsto |y^{-1}\rangle$ can be done in time $O(\log p^r)$, hence the overall algorithm is efficient. \square

With the above algorithm we are now able to efficiently estimate the angle γ in the equation $G(\mathbb{F}_{p^r}, \chi, \beta) = \sqrt{p^r} \cdot e^{i\gamma}$.

Theorem 1 (Quantum algorithm for Gauss sum estimation over \mathbb{F}_{p^r}). *For any $\varepsilon > 0$, there exists a quantum algorithm that estimates the phase γ in $G(\mathbb{F}_{p^r}, \chi, \beta) = \sqrt{p^r} \cdot e^{i\gamma}$, with expected error $\mathbb{E}[|\gamma - \tilde{\gamma}|] < \varepsilon$. The time complexity of this algorithm is bounded by $O(\frac{1}{\varepsilon} \cdot \text{poly}(\log p^r))$.*

Proof. By the earlier algorithm, we know that we can induce the phase change $|\chi\rangle \mapsto e^{i\gamma}|\chi\rangle$ in $\text{poly}(\log p^r)$ time. If we do this in superposition with a ‘stale’ component \emptyset , then we have produced the relative phase shift $(|\emptyset\rangle + |\chi\rangle)/\sqrt{2} \mapsto (|\emptyset\rangle + e^{i\gamma}|\chi\rangle)/\sqrt{2}$. By repeating this process $t = 1/\varepsilon$ times, we can create the state $(|0\rangle + e^{i\gamma}|1\rangle + \dots + e^{it\gamma}|t\rangle)/\sqrt{t+1}$. As mentioned in Fact 2, we can efficiently estimate from this state the phase γ with expected error $\mathbb{E}[|\gamma - \tilde{\gamma}|] < \varepsilon$. \square

6 Classical Complexity of Approximating Gauss Sums

The obvious next question now is: How difficult it is to estimate Gauss sums with classical computers? Although we are not able to prove that this is hard, we will give in this section some arguments why it is unlikely that there exists a classical polynomial time algorithm for Gauss sum estimation.

6.1 Connection with Discrete Logarithm Problem

Although we can only approximate Gauss sums, this is at least as hard—classically—as calculating discrete logarithms over finite fields.

Lemma 2 (Reduction from discrete logarithm to Gauss sums). *Let \mathbb{F}_{p^r} be a finite field with primitive element g , $\chi(g^j) := \zeta_{p^r-1}^j$ a multiplicative character and x an element of $\mathbb{F}_{p^r}^\times$. With an oracle that ε -approximates the angle γ of the Gauss sum $G(\mathbb{F}_{p^r}, \chi, \beta)$ for arbitrary β , we can determine, classically, the discrete logarithm $\log_g x$ in time $O(\text{poly}(1/\varepsilon, \log p^r))$.*

Proof. With $x = g^\ell$, we try to determine this $0 \leq \ell \leq p^r - 2$. For $k = 1, 2, 3, \dots$ we observe, using Lemma 1, that: $G(\mathbb{F}_{p^r}, \chi, x^k)/G(\mathbb{F}_{p^r}, \chi, 1) = \chi(g^{-k\ell}) = \zeta_{p^r-1}^{-k\ell}$; call this angle $\gamma_k := -2\pi k\ell/(p^r - 1)$. Using the ‘powering algorithm’ ($x \mapsto x^2 \mapsto x^4 \dots$ et cetera) we can calculate x^k for any $0 \leq k \leq p^r - 2$ in $\text{poly}(\log p^r)$ time, hence we can use our oracle to ε -approximate γ_k for any such k . Via the equality $-\frac{\gamma_k}{2\pi}(p^r - 1) = k\ell \pmod{p^r - 1}$ this will give us information on the value of $\ell \pmod{p^r - 1}$ depending on k . By estimating γ_k for $k = 1, 2, 4, 8, \dots, \approx p^r$, we can thus get a reliable estimation of all $\log(p^r)$ bits of ℓ , thereby calculating the desired value $\log_g(x) = \ell$. \square

6.2 Homomorphisms of $\mathbb{Q}(\zeta_{p^r-1}, \zeta_{p^r})$

The previous lemma shows that it is not trivial to estimate the Gauss sum $G(\mathbb{F}_{p^r}, \chi, \beta)$ even if we already know the value $G(\mathbb{F}_{p^r}, \chi, 1)$. A similar result seems to hold for the estimation of $G(\mathbb{F}_{p^r}, \chi^\alpha, \beta)$ while having information on $G(\mathbb{F}_{p^r}, \chi, \beta)$.

As noted earlier, $G(\mathbb{F}_{p^r}, \chi, \beta)$ is an element of $\mathbb{Q}(\zeta_{p^r-1}, \zeta_p)$. Compare now the two expressions for $G(\mathbb{F}_{p^r}, \chi, \beta)$ and $G(\mathbb{F}_{p^r}, \chi^\alpha, \beta)$, respectively,

$$\sum_{j=0}^{p^r-2} \zeta_{p^r-1}^j \zeta_p^{\text{Tr}(\beta g^j)} \quad \text{and} \quad \sum_{j=0}^{p^r-2} \zeta_{p^r-1}^{\alpha j} \zeta_p^{\text{Tr}(\beta g^j)}.$$

This shows that under the homomorphism $\sigma_\alpha : \mathbb{Q}(\zeta_{p^r-1}, \zeta_p) \rightarrow \mathbb{Q}(\zeta_{p^r-1}, \zeta_p)$, with $\sigma_\alpha : \zeta_{p^r-1} \mapsto \zeta_{p^r-1}^\alpha$, we have $\sigma_\alpha : G(\mathbb{F}_{p^r}, \chi, \beta) \mapsto G(\mathbb{F}_{p^r}, \chi^\alpha, \beta)$. Note that if $\gcd(\alpha, p^r - 1) = 1$, then this mapping is a Galois automorphism $\sigma_\alpha \in \text{Gal}(\mathbb{Q}(\zeta_{p^r-1}, \zeta_p)/\mathbb{Q}(\zeta_p))$. If $\gcd(\alpha, p^r - 1) \neq 1$ then the mapping σ_α is not necessarily bijective, and hence not an automorphism.

This result suggests that knowledge about the Gauss sum $G(\mathbb{F}_{p^r}, \chi, \beta)$ is sufficient to determine $G(\mathbb{F}_{p^r}, \chi^\alpha, \beta)$ for all other α . However, observe that the degree $[\mathbb{Q}(\zeta_{p^r-1}, \zeta_p) : \mathbb{Q}(\zeta_p)]$ equals $\phi(p^r - 1)$, which is exponential in the input size $\log(p^r)$. As a result, the σ_α mapping concerns an exponential number of coefficients, and is hence not efficient.

6.3 Gauss Sums as Pseudorandom Sums in \mathbb{C}

Let the finite field be a base field \mathbb{F}_p . For every $x \neq 0$, the terms $\chi(x)e_\beta(x)$ in $\sum_x \chi(x)e_\beta(x)$ are unit norm vectors in \mathbb{C} that together describe a sum in \mathbb{C} (of $p - 1$ terms) from 0 to the final outcome $G(\mathbb{F}_p, \chi, \beta)$. An obvious classical attempt to approximate g consists of trying to estimate the ‘average direction’ of the terms $\chi(x)e_\beta(x)$ by sampling a small number of x values. As the sum g has norm \sqrt{p} , a significant average direction can only be obtained with a sample size that is linear in p , hence this sampling method is not efficient.

In fact, the sequence $\chi(0)e_\beta(0), \chi(1)e_\beta(1), \dots, \chi(p-1)e_\beta(p-1)$ shares many of the properties that a truly random sequence in \mathbb{C} would have. Not only does

the sum coincide with the expected norm of a random sum, but also the sequence of terms exhibits the nonregularity of a random process. It is easy to verify that the autocorrelation of the sequence $\chi(0)e(0), \chi(1)e(1), \dots$ is near-zero: $\mathbb{E}[\chi(j)e(j)\bar{\chi}(j+s)\bar{e}(j+s)] = -e(-s)/(p-1)$ for $s \neq 0$. These pseudorandom characteristics do not change when we indexing of the summation (and hence of the sequence) to $\chi(1)e(1), \chi(g)e(g), \chi(g^2)e(g^2), \dots$ with g a generator of \mathbb{F}_p^\times , as this sequence obeys $\mathbb{E}[\chi(g^j)e(g^j)\bar{\chi}(g^{j+s})\bar{e}(g^{j+s})] = -\chi(-s)/(p-1)$.

7 Two Corollaries to the Gauss Sum Quantum Algorithm

7.1 Jacobi Sums over Finite Fields

Definition 7 (Jacobi sums over finite fields [2]). For a finite field \mathbb{F}_{p^r} and k multiplicative characters χ_1, \dots, χ_k , the Jacobi sum $J \in \mathbb{C}$ is defined by

$$J(\chi_1, \dots, \chi_k) := \sum_{x_1 + \dots + x_k = 1} \chi_1(x_1) \cdots \chi_k(x_k).$$

From Chapter 8 in [7] we repeat the following facts.

Fact 3 (Some properties of Jacobi sums). With χ^0 the trivial character, $J(\chi^0, \dots, \chi^0) = p^{r(k-1)}$. If some but not all of the characters χ_1, \dots, χ_k are trivial, then $J(\chi_1, \dots, \chi_k) = 0$.

If χ is a nontrivial character, then $J(\chi, \chi^{-1}) = -\chi(-1)$. In general, for nontrivial χ_1, \dots, χ_k with the product $\chi_1 \cdots \chi_k$ trivial, we have $J(\chi_1, \dots, \chi_k) = -\chi_k(-1)J(\chi_1, \dots, \chi_{k-1})$.

If χ_1, \dots, χ_k and the product $\chi_1 \cdots \chi_k$ are all nontrivial, then the Jacobi can be expressed in terms of Gauss sums:

$$J(\chi_1, \dots, \chi_k) = \frac{G(\mathbb{F}_p, \chi_1, 1) \cdots G(\mathbb{F}_p, \chi_k, 1)}{G(\mathbb{F}_p, \chi_1 \cdots \chi_k, 1)}.$$

With these facts, it is obvious how to estimate Jacobi sums.

Corollary 1 (Quantum algorithm for Jacobi sum estimation). Given a finite field \mathbb{F}_{p^r} and the k characters χ_1, \dots, χ_k , it is straightforward to determine the norm $|J(\chi_1, \dots, \chi_k)|$, which depends on whether the product $\chi_1 \cdots \chi_k$ is trivial or not.

Using the Gauss sum estimation algorithm of Theorem 1, there also exists an efficient quantum algorithm that estimates the angle $\lambda \bmod 2\pi$ in $J(\chi_1, \dots, \chi_k) = e^{i\lambda}|J(\chi_1, \dots, \chi_k)|$ with expected error ε in time $O(\frac{1}{\varepsilon} \cdot \text{poly}(k, \log p^r))$.

7.2 Gauss Sums over Finite Rings

Although the quantum algorithm for estimating Gauss sums over rings $\mathbb{Z}/n\mathbb{Z}$ is not much more complicated than the version for finite fields, the theory surrounding it is somewhat more elaborate. First we will have to explain some of

the the theory of multiplicative characters over $\mathbb{Z}/n\mathbb{Z}$. These details are necessary to get a valid definition for the input size of the Gauss sum problem over $\mathbb{Z}/n\mathbb{Z}$ (see Definition 9).

Again, the n th root of unity is denoted by $\zeta_n^x := e^{2\pi i x/n}$. From [4], Section 1.4 we copy the following facts. Consider the multiplicative subgroup $(\mathbb{Z}/n\mathbb{Z})^\times$ with the prime decomposition $n = p_1^{r_1} \cdots p_k^{r_k}$. Following the Chinese remainder theorem we have $(\mathbb{Z}/n\mathbb{Z})^\times \cong (\mathbb{Z}/p_1^{r_1}\mathbb{Z})^\times \times \cdots \times (\mathbb{Z}/p_k^{r_k}\mathbb{Z})^\times$, such that $|(\mathbb{Z}/n\mathbb{Z})^\times| = \phi(n)$, with ϕ Euler's totient function. Furthermore we have

$$\begin{cases} (\mathbb{Z}/p^r\mathbb{Z})^\times \cong \mathbb{Z}/(p-1)p^{r-1}\mathbb{Z} \text{ if } p \geq 3, \\ (\mathbb{Z}/2\mathbb{Z})^\times \cong \mathbb{Z}/\mathbb{Z} \cong \{0\}, \\ (\mathbb{Z}/4\mathbb{Z})^\times \cong \mathbb{Z}/2\mathbb{Z}, \\ (\mathbb{Z}/2^r\mathbb{Z})^\times \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2^{r-2}\mathbb{Z} \text{ if } r \geq 3, \end{cases}$$

hence $(\mathbb{Z}/n\mathbb{Z})^\times$ is cyclic if and only if $n = 2, 4, p^r$ or $2p^r$, with p an odd prime.

Definition 8 (Multiplicative characters over $\mathbb{Z}/n\mathbb{Z}$). A function $\chi : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{C}$ is a multiplicative character if for all $x, y \in \mathbb{Z}/n\mathbb{Z}$ we have $\chi(x)\chi(y) = \chi(xy)$ and $\chi(x) = 0$ if and only if $\gcd(n, x) \neq 1$.

Using the multiplicative decomposition $(\mathbb{Z}/n\mathbb{Z})^\times = (\mathbb{Z}/p_1^{r_1}\mathbb{Z})^\times \times \cdots \times (\mathbb{Z}/p_k^{r_k}\mathbb{Z})^\times$, we see that all characters $\chi : (\mathbb{Z}/n\mathbb{Z}) \rightarrow \mathbb{C}$ can be decomposed as $\chi = \chi_1 \cdots \chi_k$ with $\chi_i : (\mathbb{Z}/p_i^{r_i}\mathbb{Z}) \rightarrow \mathbb{C}$ for every $1 \leq i \leq k$, and thus $\chi(x) := \chi_1(x \bmod p_1^{r_1}) \cdots \chi_k(x \bmod p_k^{r_k})$.

For p an odd prime the character $\chi : \mathbb{Z}/p^r\mathbb{Z} \rightarrow \mathbb{C}$ can be described by the expression $\chi(x) := \zeta_{\phi(p^r)}^{\alpha \log_g x}$, where g is a generator of the cyclic $(\mathbb{Z}/p^r\mathbb{Z})^\times$, $\phi(p^r) = p^{r-1}(p-1)$ and $\alpha \in \mathbb{Z}/\phi(p^r)\mathbb{Z}$. For $(\mathbb{Z}/2\mathbb{Z})^\times$ we only have the trivial character χ^0 , while for $(\mathbb{Z}/4\mathbb{Z})$ we have two possibilities: χ^0 and χ^1 with $\chi^\alpha(3) = (-1)^\alpha$ and $\chi^\alpha(1) = 1$. If χ is a character over $(\mathbb{Z}/2^r\mathbb{Z})^\times$ with $r \geq 3$, then we have to decompose the character in two terms. The group $(\mathbb{Z}/2^r\mathbb{Z})^\times$ is generated by 3 and 5 (see [4]), hence the character can be described by the pair $(\alpha, \alpha') \in (\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2^{r-2}\mathbb{Z})$ such that for all i and i' we have $\chi(3^i 5^{i'} \bmod 2^r) := (-1)^{\alpha i} \zeta_{2^{r-2}}^{\alpha' i'}$ (while $\chi(x) = 0$ if x is even).

Definition 9 (Specification of characters over $\mathbb{Z}/n\mathbb{Z}$). Let

$$\begin{aligned} (\mathbb{Z}/n\mathbb{Z})^\times &\cong (\mathbb{Z}/2^{r_0}\mathbb{Z})^\times \times (\mathbb{Z}/p_1^{r_1}\mathbb{Z})^\times \times \cdots \times (\mathbb{Z}/p_k^{r_k}\mathbb{Z})^\times \\ &\cong (\mathbb{Z}/2^{r_0}\mathbb{Z})^\times \times (\mathbb{Z}/\phi_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/\phi_k\mathbb{Z}), \end{aligned}$$

with $\phi_j := (p_j - 1)p_j^{r_j - 1}$ (see Equation 2). The specification of a multiplicative character $\chi : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{C}$ is done by three sequences (p, g, α) , with the prime decomposition $p = (p_1, \dots, p_k)$ of n , the generators $g = (g_1, \dots, g_k) \in (\mathbb{Z}/p_1^{r_1}\mathbb{Z})^\times \times \cdots \times (\mathbb{Z}/p_k^{r_k}\mathbb{Z})^\times$ of the multiplicative groups, and $\alpha = ((\alpha_0, \alpha'_0), \alpha_1, \dots, \alpha_k) \in (\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2^{r-2}) \times (\mathbb{Z}/\phi_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/\phi_k\mathbb{Z})$ the specification of the characters χ_j in the definition

$$\chi(x) := \chi_0(x \bmod 2^{r_0}) \chi_1(x \bmod p_1^{r_1}) \cdots \chi_k(x \bmod p_k^{r_k})$$

with $\chi_0(3^i 5^{i'} \bmod 2^{r_0}) := (-1)^{\alpha_0 i} \zeta_{2^{r_0-2}}^{\alpha'_0 i'}$ and

$$\chi_j(x_j) := \begin{cases} \zeta_{\phi_j}^{\alpha_j \log_{g_j}(x_j)} & \text{if } x_j \in (\mathbb{Z}/p_j^{r_j}\mathbb{Z})^\times, \\ 0 & \text{if } \gcd(x, p_j) \neq 1. \end{cases}$$

With this definition, we see that the specification of a character $\chi : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{C}$ requires only $O(\log n)$ bits of information. Hence, in the context of such characters, an algorithm that requires $\text{poly}(\log n)$ steps is efficient, while a running time polynomial in n is inefficient.

The definition of Gauss sums over $\mathbb{Z}/n\mathbb{Z}$ is a natural generalization of the definition for finite fields.

Definition 10 (Gauss sums over $\mathbb{Z}/n\mathbb{Z}$). For the ring $\mathbb{Z}/n\mathbb{Z}$, a multiplicative character χ , and an additive character $e_\beta(x) := \zeta_n^{x\beta}$, the Gauss sum G is defined by

$$G(\mathbb{Z}/n\mathbb{Z}, \chi, \beta) := \sum_{x \in \mathbb{Z}/n\mathbb{Z}} \chi(x) \zeta_n^{\beta x}.$$

The Gauss sum over a ring $\mathbb{Z}/n\mathbb{Z}$ can be expressed in terms of Gauss sums over the finite fields $\mathbb{Z}/p_i\mathbb{Z}$. See Section 1.6 in [2] for how this is done (which is not entirely trivial). Using this reduction, we can construct an efficient quantum algorithm for the estimation of the Gauss sum $G(\mathbb{Z}/n\mathbb{Z}, \chi, \beta)$.

Corollary 2 (Quantum algorithm for Gauss sum estimation over $\mathbb{Z}/n\mathbb{Z}$).

For any $\varepsilon > 0$, there exists a quantum algorithm that estimates the phase γ in $G(\mathbb{Z}/n\mathbb{Z}, \chi, \beta) = |G(\mathbb{Z}/n\mathbb{Z}, \chi, \beta)| \cdot e^{i\gamma}$, with expected error $\mathbb{E}[|\gamma - \tilde{\gamma}|] < \varepsilon$. The time complexity of this algorithm is bounded by $O(\frac{1}{\varepsilon} \cdot \text{poly}(\log n))$. Also the norm $|G(\mathbb{Z}/n\mathbb{Z}, \chi, \beta)|$ can be determined in $\text{poly}(\log n)$ time.

The details of this quantum algorithm are explained elsewhere.[6]

8 Conclusion and Discussion

The algorithms that we presented in this article rely on the specific interaction between multiplicative characters and Fourier transformations. Typical for these results is the fact they are defined for finite fields or finite rings but not for groups, which indicates a departure from the Hidden Subgroup framework for quantum algorithms [10]. The results presented here describe quantum algorithms for a natural problem, which does not assume the presence of a black box function (unlike the related algorithm in [5]). The only other natural problems for which an efficient quantum algorithm has been constructed are those described by Shor [13] and Hallgren [8], both of which deal with number theory as well.

For the results of this article it remains therefore an important open question if Gauss sum estimation is hard classically, even if we have oracles for factoring and discrete logarithms. If this is indeed the case, another related question remains: Which problems reduce to Gauss sum estimation that do not reduce to

factoring or the discrete logarithm problems? A candidate for this is the estimation of the number of solutions of equations over finite fields, which can be expressed terms of Jacobi sums [2, 7].

Acknowledgments We would like to thank Harry Buhrman, Andrew Childs, Vinay Deolalikar, Hendrik Lenstra, Mike Mosca, Ashwin Nayak, Ronny Roth and Ronald de Wolf for helpful discussions about the topics in this article.

References

1. Gilles Brassard and Peter Høyer, “An exact quantum polynomial-time algorithm for Simon’s problem”, *Proceedings of Fifth Israeli Symposium on Theory of Computing and Systems (ISTCS’97)*, pages 12–23 (1997)
2. Bruce C. Berndt, Ronald J. Evans, and Kenneth S. Williams, *Gauss and Jacobi Sums*, Canadian Mathematical Society Series of Monographs and Advanced Texts, Volume 21, John Wiley & Sons (1998)
3. Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca, “Quantum algorithms revisited”, *Proceedings of the Royal Society of London A*, Volume 454, pages 339–354 (1998)
4. Henri Cohen, *A Course in Computational Algebraic Number Theory*, Springer, Graduate Texts in Mathematics 138 (1996)
5. Wim van Dam, Sean Hallgren and Lawrence Ip, “Quantum Algorithms for some Hidden Shift Problems”, *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 489–498 (2003)
6. Wim van Dam and Gadiel Seroussi, “Efficient Quantum Algorithms for Estimating Gauss Sums”, arXiv:quant-ph/0207131 (2002)
7. K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*, Second Edition, Graduate Texts in Mathematics 84, Springer (1990)
8. Sean Hallgren, “Polynomial-time quantum algorithms for Pell’s equation and the principal ideal problem”, *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 653–658, ACM Press (2002)
9. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton (1997)
10. Michele Mosca and Artur Ekert, “The hidden subgroup problem and eigenvalue estimation on a quantum computer”, *Proceedings of the 1st NASA International Conference on Quantum Computing and Quantum Communications*, Lecture Notes in Computer Science, Volume 1509, pages 174–188 (1999)
11. Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press (2000)
12. Gadiel Seroussi and Abraham Lempel, “Factorization of symmetric matrices and trace-orthogonal bases in finite fields”, *SIAM Journal on Computing*, Volume 9, No. 4, pages 758–767 (1980)
13. Peter W. Shor, “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”, *SIAM Journal on Computing*, Volume 26, No. 5, pages 1484–1509 (1997)
14. John Watrous, “Quantum algorithms for solvable groups”, *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pp. 60–67 (2001)