Quantum Computing

Prof. Dr. Jean-Pierre Seifert
jpseifert@sec.t-labs.tu-berlin.de
http://www.sec.t-labs.tu-berlin.de/
Outline for this lecture

Peter Shor’s Factorization Algorithm
Perhaps the most significant success of quantum computing so far has been Shor’s polynomial time algorithm for factorization to be presented in this lecture. This is a highly nontrivial algorithm that uses a new technique, that of Quantum Fourier Transform, that will also be illustrated in this chapter.

The fastest classical algorithm to factor $m$ bit numbers requires time $O(e^{cm^{1/3}(\lg m)^{2/3}})$.

Shor’s factorization algorithm requires $O(m^2 \lg^2 m \lg \lg m)$ time on a quantum computer.

Of interest and importance is also another Shor’s polynomial time algorithm, for discrete logarithm, to be also presented in this chapter. We will present also another than Shor’s approach to quantum factorization.

Correctness and efficiency of Shor’s algorithm is based on several simple results from number theory to be presented first.
First reduction

Lemma 0.1 If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

\[ a^2 \equiv 1 \pmod{n}, \]

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

Proof. Let \( a \neq \pm 1 \) be such that \( a^2 \equiv 1 \pmod{n} \). Since

\[ a^2 - 1 = (a + 1)(a - 1), \]

if \( n \) is not prime, then a prime factor of \( n \) has to be a prime factor of either \( a + 1 \) or \( a - 1 \).

By using Euclid’s algorithm to compute

\[ \gcd(a + 1, n) \text{ and } \gcd(a - 1, n) \]

we can find, in \( O(\lg n) \) steps, a prime factor of \( n \).
Second reduction

The second key concept is that of period of the functions

\[ f_{n,x}(k) = x^k \mod n. \]

It is the smallest integer \( r \) such that

\[ f_{n,x}(k + r) = f_{n,x}(k) \]

for any \( k \), i.e. the smallest \( r \) such that

\[ x^r \equiv 1 \pmod{n}. \]

**AN ALGORITHM TO SOLVE EQUATION** \( x^2 \equiv 1 \pmod{n} \).

1. Choose randomly \( 1 < a < n \).
2. Compute \( \gcd(a, n) \). If \( \gcd(a, n) \neq 1 \) we have a factor.
3. Find period \( r \) of function \( a^k \mod n \).
4. If \( r \) is odd or \( a^{r/2} \equiv \pm 1 \pmod{n} \), then go to step 1; otherwise stop.

If this algorithm stops, then \( a^{r/2} \) is a non-trivial solution of the equation

\[ x^2 \equiv 1 \pmod{n}. \]
Efficiency of reduction

Lemma 0.2 If $1 < a < n$ satisfying $\gcd(n, a) = 1$ is selected in the above algorithm randomly and $n$ is not a power of prime, then

$$Pr\{r \text{ is even and } a^{r/2} \neq \pm 1\} \geq \frac{9}{16}.$$  

1. Choose randomly $1 < a < n$.
2. Compute $\gcd(a, n)$. If $\gcd(a, n) \neq 1$ we have a factor.
3. Find period $r$ of function $a^k \mod n$.
4. If $r$ is odd or $a^{r/2} \equiv \pm 1 \pmod{n}$, then go to step 1; otherwise stop.

Corollary 0.3 If there is a polynomial time randomized [quantum] algorithm to compute the period of the function

$$f_{n,a}(k) = a^k \mod n,$$

then there is a polynomial time randomized [quantum] algorithm to find non-trivial solution of the equation $a^2 \equiv 1 \pmod{n}$ (and therefore also to factorize integers).
Recap from lecture on Simon’s problem

Recap
Simon’s Problem

- Simon has discovered a simple problem with an expected quantum polynomial time algorithm, but having no polynomial time randomized algorithm.

- Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a function such that either $f$ is one-to-one or $f$ is two-to-one and there exists a single $0 \neq s \in \{0, 1\}^n$ such that

  \[
  \forall x \neq x' \ ( f(x) = f(x') \iff x' = x \oplus s ).
  \]

- The task is to determine which of the above conditions holds for $f$ and, in the second case, to determine also $s$.

- To solve the problem two registers are used, both with $n$ qubits, and the initial states $|0^n\rangle$, and (expected) $O(n)$ repetitions of the following version of the so-called Hadamard-twice scheme:
Simon’s Problem

- Apply the Hadamard transformation on the first register with the initial value $|0^n\rangle$, to produce the superposition
  
  $$|0^n, 0^n\rangle \xrightarrow{H_n} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} |x, 0^n\rangle.$$ 

- Apply $U_f$ to compute $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0, 1\}^n} |x, f(x)\rangle$.

- Apply Hadamard transformation on the first register to get
  
  $$\frac{1}{2^n} \sum_{x, y \in \{0, 1\}^n} (-1)^{x \cdot y} |y, f(x)\rangle.$$ 

- Observe the resulting state to get a pair $(y, f(x))$. 
FROM SIMON’s PROBLEM TO FACTORIZATION

One can see Simon’s problem as the one to find the unknown period of a function on \( n \)-bit integers that is “periodic” under bit-wise modulo-2 addition.

One can see the factorization problem as the one to find a period of integer functions \( f_b(x) = b^x \mod n \) under ordinary addition. That is to find such an \( r \) that \( f_b(x + r) = fb(x) \) for all \( x \), that is the smallest integer \( r \) such that \( b^r \equiv 1(\mod n) \).

A large difficulty of this task is connected with the fact that values of the function \( f \) between \( f_b(x + r) \) and \( f_b(x) \) are almost randomly distributed and therefore knowledge of some of them give almost no clue about others.
choose randomly \( a \in \{2, \ldots, n-1\} \)

compute \( z = \gcd(a, n) \)

no \( z = 1? \) yes

\( z = 1? \) no yes

quantum subroutine

find period \( r \) of function \( a^x \mod n \)

\( r \) is even? no yes

\[ z = \max\{\gcd(n, a^{r/2}-1), \gcd(n, a^{r/2}+1)\} \]

\( z = 1? \) no

output \( z \)
SHOR’S ALGORITHM

1. For given $n, q = 2^d$, a create states

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, 0\rangle$$ and $$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \mod n\rangle$$

2. By measuring the last register the state collapses into the state

$$\frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |n, a, q, jr + l, y\rangle$$ or, shortly $$\frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |jr + l\rangle,$$

where $A$ is the largest integer such that $l + Ar \leq q$, $r$ is the period of $a^x \mod n$ and $l$ is the offset.

3. In case $A = \frac{q}{r} - 1$, the resulting state has the form.

$$\sqrt{\frac{r}{q}} \frac{1}{\sqrt{r}} \sum_{j=0}^{q-1} |jr + l\rangle$$

4. By applying quantum Fourier transformation we get then the state

$$\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{2\pi i jl/r} |j, q\rangle.$$

5. By measuring the resulting state we get $c = \frac{ia}{r}$ and if $\gcd(j, r) = 1$, what happens with sufficient large probability, then from $c$ and $q$ we can determine the period $r$. 

12
PERIOD COMPUTATION for \( f_{n,a}(x) = a^x \mod n \)

Hadamard transform applied to the state \( |0^{(d)}, 0^{(d)}\rangle \) yields

\[
|\phi\rangle = \frac{1}{\sqrt{2^d}} \sum_{x=0}^{q-1} |x, 0^{(d)}\rangle
\]

and an application of the unitary transformation

\[
U_{f_{n,a}} : |x, 0^{(d)}\rangle \rightarrow |x, a^x \mod n\rangle
\]

provides the state

\[
|\phi_1\rangle = U_{f_{n,a}} |\phi\rangle = \frac{1}{\sqrt{2^d}} \sum_{x=0}^{q-1} |x, f_{n,a}(x)\rangle
\]

Note 1: All possible values of the function \( f_{n,a} \) are encoded in the second register in the state \( |\phi_1\rangle \).

Note 2: We are interested in the period of the function \( f_{n,a} \) and not in particular values of \( f_{n,a} \).

Could we get period by measuring, several times, at first the second register and then the first one?
Example

For $n = 15, a = 7$, $f_{n,a}(x) = 7^x \mod 15$, $q = 16$, the state

$$U_{f_{n,a}}|\phi\rangle = \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x, f_{n,a}(x)\rangle$$

has the form

$$\frac{1}{4}(|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + \ldots + |14\rangle|4\rangle + |15\rangle|13\rangle).$$

If we measure at this point the second register, then we get as the outcome one of the numbers 1, 4, 7 or 13, and the following table shows the corresponding post-measurement states in the second column.

<table>
<thead>
<tr>
<th>result</th>
<th>post-measurement state</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{1}{2}(</td>
<td>0\rangle +</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{1}{2}(</td>
<td>2\rangle +</td>
</tr>
<tr>
<td>7</td>
<td>$\frac{1}{2}(</td>
<td>1\rangle +</td>
</tr>
<tr>
<td>13</td>
<td>$\frac{1}{2}(</td>
<td>3\rangle +</td>
</tr>
</tbody>
</table>

The corresponding sequences of values of the first register are periodic with period 4 but they have different offsets (pre-periods) listed in column 3 of the table.
DISCRETE FOURIER TRANSFORM

Discrete Fourier Transform maps a vector $a = (a_0, a_1, \ldots, a_{n-1})^T$ into the vector $DFT(a) = A_n a$, where $A_n$ is an $n \times n$ matrix such that $A_n[i, j] = \frac{1}{\sqrt{n}} e^{i \frac{2\pi ij}{n}}$ for $0 \leq i, j < n$ and $\omega = e^{2\pi i/n}$ is the $n$th root of unity. The matrix $A_n$ has therefore the form

$$A_n = \frac{1}{\sqrt{n}} \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\
1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2}
\end{pmatrix}.$$

The Inverse Discrete Fourier Transform is the mapping

$$DFT^{-1}(a) = A_n^{-1} a,$$

where

$$A_n^{-1}[i, j] = \frac{1}{\sqrt{n}} e^{-i \frac{2\pi ij}{n}}.$$
Insides into DFT

There is a close relation between Discrete Fourier Transform and polynomial evaluation and interpolation. Let us consider a polynomial

$$p(x) = \sum_{i=0}^{n-1} a_i x^i.$$  

Such a polynomial can be uniquely represented in two ways: either by a list of its coefficients $a_0, a_1, \ldots, a_{n-1}$, or by a list of its values at $n$ distinct points $x_0, x_1, \ldots, x_{n-1}$.

The process of finding the coefficient representation of the polynomial given its values at points $x_0, x_1, \ldots, x_{n-1}$ is called interpolation.

Computing the Discrete Fourier Transform of a vector $(a_0, a_1, \ldots, a_{n-1})$ is equivalent to converting the coefficient representation of the polynomial $\sum_{i=0}^{n-1} a_i x^i$ to its value representation at the points $\omega^0, \omega^1, \ldots, \omega^{n-1}$.

Likewise, the Inverse Discrete Fourier Transform is equivalent to interpolating a polynomial given its values at the $n$-th roots of unity.
FOURIER TRANSFORM

- The Quantum Fourier Transform is a quantum variant of the Discrete Fourier Transform (DFT).
- It maps a discrete function to another discrete one with equally distant points as its domain.

- For example it maps a $q$-dimensional complex

$$(f(0), f(1), \ldots, f(q-1))$$

into

$$(\overline{f}(0), \overline{f}(1), \ldots, \overline{f}(q-1))$$

where

$$\overline{f}(c) = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} e^{2\pi i ac/q} f(a)$$

and $c \in \{0, \ldots, q-1\}$. 
QUANTUM FOURIER TRANSFORM

The quantum version of DFT (QFT) is the unitary transformation

\[
\text{QFT}_q : |x\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} e^{2\pi i ax/q} |a\rangle,
\]

\(x \in \{0, \ldots, q-1\}\) for the unitary matrix

\[
F_q = \frac{1}{\sqrt{q}} \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \omega & \omega^2 & \cdots & \omega^{(q-1)} \\
1 & \omega^2 & \omega^4 & \cdots & \omega^{2(q-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega^{(q-1)} & \omega^{2(q-1)} & \cdots & \omega^{(q-1)^2}
\end{pmatrix},
\]

and \(\omega = e^{2\pi i / q}\) is the \(q^{th}\) root of unity.
Property of QFT

- If applied to a quantum superposition, \( \text{QFT}_q \) performs as

\[
\text{QFT}_q : \sum_{a=0}^{q-1} f(a) \left| a \right\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \bar{f}(c) \left| c \right\rangle,
\]

\[
= \frac{1}{\sqrt{q}} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sum_{a=0}^{q-1} e^{2\pi i ac/q} f(a) \left| c \right\rangle
\]

where \( \bar{f}(c) \) is as defined as before.

- Note that

\[
\text{QFT}_q : \left| 0^n \right\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{i=0}^{q-1} \left| i \right\rangle.
\]
SHOR phase I

Design of states whose amplitudes have the same period as \( f_{n,a} \)

Given an \( m \) bit integer \( n \) we choose a \( n^2 \leq q = 2^d \leq 2n^2 \) and start with five registers in states \( |n, a, q, 0, 0\rangle \), where the last two registers have \( m = \lceil \lg n \rceil \) qubits.

An application of the Hadamard transformation to the fourth register yields the state

\[
\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, 0\rangle.
\]

and using quantum parallelism we compute \( a^x \mod n \) for all \( x \) in one step, to get

\[
\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \mod n\rangle.
\]

As the next step we perform a measurement on the last register. Let \( y \) be the value obtained, i.e. \( y = a^l \mod n \) for the smallest \( l_y \) with this property. If \( r \) is the period of \( f_{n,a} \), then \( a^{ly} \equiv a^{jr+l_y} \mod n \) for all \( j \). Therefore, the measurement actually selects the sequence of \( x \)'s values (in the fourth register), \( l_y, l_y + r, l_y + 2r, \ldots, l_y + Ar \), where \( A \) is the largest integer such that \( l_y + Ar \leq q - 1 \). Clearly, \( A \approx \frac{q}{r} \). The post-measurement state is then

\[
|\phi_y\rangle = \frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |n, a, q, jr + l_y, y\rangle = \frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |jr + l_y\rangle.
\]

after omitting some registers being fixed from now on.
Amplitude amplification by QFT

From now on we consider in detail only a special case. Namely that \( r \) divides \( q \). In such a case \( A = \frac{q}{r} - 1 \). In such a case the last state can be written in the form

\[
|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q-1}{r}} |jr + l_y\rangle
\]

and after \( \text{QFT}_{q} \) is applied on \( |\phi_l\rangle \) we get:

\[
\text{QFT}_{q}|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q-1}{r}} e^{2\pi i c (jr + l_y)/q} |c\rangle = \sqrt{\frac{r}{q}} \sum_{c=0}^{q-1} e^{2\pi i l_y c/q} \left( \sum_{j=0}^{\frac{q-1}{r}} e^{2\pi i j cr/q} \right) |c\rangle = \sum_{c=0}^{q-1} \alpha_c |c\rangle
\]

If \( c \) is a multiple of \( \frac{q}{r} \), then \( e^{2\pi i j cr/q} = 1 \) and if \( c \) is not a multiple of \( \frac{q}{r} \), then

\[
\sum_{j=0}^{\frac{q-1}{r}} e^{2\pi i j cr/q} = 0,
\]
SHOR phase II

because the above sum is over a set of $\frac{q}{r}$ roots of unity equally spaced around the unit circle. Thus

$$\alpha_c = \begin{cases} \frac{1}{\sqrt{r}} e^{2\pi i l_c r / q}, & \text{if } c \text{ is a multiple of } \frac{q}{r}; \\ 0, & \text{otherwise}; \end{cases}$$

and therefore

$$|\phi_{out}\rangle = \text{QFT}_q |\phi_1\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{2\pi i l_y j / r} |j/r\rangle.$$  

The key point is that the trouble-making offset $l_y$ appears now in the phase factor $e^{2\pi i l_y j / r}$ and has no influence either on the probabilities or on the values in the register.
SHOR phase III

Period extraction

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the multiples $c = \lambda \frac{q}{r}$, $
\lambda \in \{0, 1, \ldots r - 1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

Observe also that in this case the QFT transforms a function with the period $r$ (and an offset $l$) to a
function with the period $\frac{q}{r}$ and offset 0. After each measurement we therefore know $c$ and $q$ and

$$\frac{c}{q} = \frac{\lambda}{r},$$

where $\lambda$ is randomly chosen.

If $gcd(\lambda, r) = 1$, then from $q$ we can determine $r$ by dividing $q$ with $gcd(c, q)$. Since $\lambda$ is chosen
randomly, the probability that $gcd(\lambda, r) = 1$ is greater than $\Omega\left(\frac{1}{\lg \lg r}\right)$. If the above computation is
repeated $O(\lg \lg r)$ times, then the success probability can be as close to 1 as desired and therefore $r$
can be determined efficiently.$^1$

In the general case, i.e., if $A \neq \frac{q}{r} - 1$, there is only a more sophisticated computation of the resulting
probabilities and a more sophisticated way to determine $r$ (using a continuous fraction method to
extract the period from its approximation).

$^1$As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.
Comments on SHOR

- Efficient implementations of $\text{QFT}_q$, concerning the number of gates, are known for the case $q = 2^d$ or $q$ is smooth (that is if factors of $q$ are smaller than $O(\lg q)$).

- Efficient implementation of modular operations (including exponentiation) are known.

- First estimation said that $300 \lg n$ gates are needed to factor $n$.

- An estimation said that to factor 130 digit integers would require two weeks on an ideal quantum computer with switching frequency 1 MHz. However to factor 260-digit number only 16 times larger time would be needed.

- It has been shown that there is polynomial time factorization even in the case only one pure qubit is available and the rest of quantumness available is in mixed states.
Comments on SHOR

- Of real practical interest is only quantum factorization of such $n = pq$, where $n$ is at least 500-digit number. In such a case if $d$ is to be the smallest integer such that $2^d > n$, then $d$ has to be around a 1700-bit number.

- Periods need to be determined precisely - in spite of the fact that they could be numbers of several hundred bits long!! However in such cases Quantum Fourier Transform circuits should work with phase factors with numbers proportional to $\frac{1}{2^j}$ for so enormously large $j$ that such small phases are practically impossible to realise. It therefore seems that there is no way practically to determine period for such large $n$.

- It can be shown that this is not the case. The reason is that phases do not have impact on discrete outcomes of measurements, only on their probabilities.

- It has been shown that relatively small "cuts" of QFT circuits, for example to delete all conditional gates that deal with wires more than 22 wires apart, are sufficient to do necessary calculations precisely enough.
General case

- Of the key importance for the efficiency of Shor’s factorization algorithm is also the fact that exponentiation in $b^x$ can be done efficiently.

- If computation would be done on classical computers than each $b^x$ could be done efficiently by computing first values $b^{2^j}$ for all $j$. However, would there be a need to do that for so many $x$ that would be enormously inefficient. However, once this is done on quantum computer using quantum parallelism this can be done "only once" and this is also behind the overall efficiency of Shor’s quantum algorithm.
General case

Let us now sketch Shor’s algorithm to compute the period of a function \( f(x) = a^x \mod n \) for the general case.

QFT\(_q\) is applied to the first register of the state \( \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle |f(x)\rangle \) and afterwards the measurement is performed on the first register to provide an \( y_0 \in [0, \ldots, q-1] \).

To get the period \( r \) the following algorithm is realized where \( \xi = \frac{y_0}{q} \), \( a_0 = \lfloor \xi \rfloor \), \( \xi_0 = \xi - a_0 \), \( p_0 = a_0, q_0 = 1, p_1 = a_1a_0 + 1, q_1 = a_1 \)

for \( j = 1 \) until \( \xi_j = 0 \) do

• compute \( p_j \) and \( q_j \) using the recursion (for the case \( \xi_j \neq 0 \));

\[
a_j = \left\lfloor \frac{1}{\xi_{j-1}} \right\rfloor, \quad \xi_j = \frac{1}{\xi_{j-1}} - a_j, \quad p_j = a_jp_{j-1} + p_{j-2}, \quad q_j = a_jq_{j-1} + q_{j-2}
\]

• Test whether \( q_j = r \) by computing first \( m^{q_j} = \prod_i (m^{2^i})^{q_{j,i}} \mod n \), where \( q_j = \sum_i q_{j,i}2^i \) is the binary expansion of \( q_n \).

If \( a^{q_j} = 1 \mod n \), then exit with \( r = q_j \); if not continue the loop.

The non-easy task is to show, what has been done, that the above algorithm provides the period \( r \) with sufficient probability \( (\geq \frac{0.232}{\log \log n} (1 - \frac{1}{n})^2) \).
Hidden Subgroup Problem

All quantum algorithms we have been dealing with are to solve special cases of the following Hidden Subgroup Problem. 

**Given:** An (efficiently computable) function \( f : G \rightarrow R \), where \( G \) group and \( R \) a finite set.

**Promise:** There exists a subgroup \( G_0 \leq G \) such that \( f \) is constant and distinct on the cosets of \( G_0 \).

**Task:** Find a generating set for \( G_0 \) (in polynomial time (in \( \lg |G| \)) number of calls to the oracle for \( f \) and in the overall polynomial time).\(^2\)

**Deutsch’s problem,** \( G = \mathbb{Z}_2, f : \{0, 1\} \rightarrow \{0, 1\}, x - y \in G_0 \Leftrightarrow f(x) = f(y) \). Decide whether \( G_0 = \{0\} \) (and \( f \) is balanced) or \( G_0 = \{0, 1\} \) (and \( f \) is constant).

**Simon’s problem,** \( G = \mathbb{Z}_2^n, f : G \rightarrow R, x - y \in G_0 \Leftrightarrow f(x) = f(y), G_0 = \{0(n), s\}, s \in \mathbb{Z}_2^n \). Decide whether \( G_0 = \{0(n)\} \) or \( G_0 = \{0(n), s\} \), with an \( s \neq 0(n) \) (and in the second case find \( s \)).

**Order-finding problem,** \( G = \mathbb{Z}, a \in \mathbb{N}, f(x) = a^x, x - y \in G_0 \Leftrightarrow f(x) = f(y), G_0 = \{rk \mid k \in \mathbb{Z} \} \) for the smallest \( r \) such that \( a^r = 1 \). Find \( r \).

**Discrete logarithm problem,** \( G = \mathbb{Z}_r \times \mathbb{Z}_r, a^r = 1, b = a^m, a, b \in \mathbb{N}, f(x, y) = a^x b^y, f(x_1, y_1) = f(x_2, y_2) \Leftrightarrow (x_1, y_1) - (x_2, y_2) \in G_0, G_0 = \{(-km, m) \mid k \in \mathbb{Z}_r \} \). Find \( G_0 \) (or \( m \)).

\(^2\)A way to solve the problem is to show that in polynomial number of oracle calls (or time) the states corresponding to different candidate subgroups have exponentially small inner product and are therefore distinguishable.
Outline for next lecture

No more lectures this semester
Questions?