Quantum Computing 4/27/2022

Lecture 24: Integer Factorization and Discrete Log

Instructor: Dieter van Melkebeek Scribe: Ivan Hu

In this lecture, we wrap up our discussion of Shor's algorithm by applying the order-finding algorithm to efficiently solve integer factorization. The quantum algorithms for factoring integers and computing discrete logarithms imply that many widely used cryptographic systems, such as RSA encryption and Diffie-Hellman key exchange, are broken. This creates many challenges in creating new cryptosystems that are secure against quantum computers, but there are also opportunities to exploit quantum computers for cryptography. Of course, quantum computing is not omnipotent, as some cryptographic primitives like perfect bit commitment remain impossible.

1 Integer Factorization

Integer factorization is the problem of taking a positive integer μ and writing it as the product of primes. Note that the complexity of the algorithm is measured in $n = \log \mu$ – the size of the input is n because μ is written in binary. So the trivial algorithm of checking all divisors below μ has a time complexity of $2^{\widetilde{O}(n)}$. Even knowing that μ has a factor less than $\sqrt{\mu}$, given μ is composite, does not help improve the exponent by more than a constant factor.

Currently known classical algorithms such as the general number field sieve do better than that, but they are still exponential, far slower than the quantum algorithm. The heuristic time complexity for the classical algorithm is not proven, but is based on reasonable conjectures on the distribution of primes.

- Classical time complexity: $2^{\widetilde{O}(n^{1/2})}$ (rigorous), $2^{\widetilde{O}(n^{1/3})}$ (heuristic)
- Quantum time complexity: $\widetilde{O}(n^2)$

To solve integer factorization, we can reduce the problem to *splitting*, or writing μ as the product of two smaller integers. By recursively splitting the smaller factors, we can eventually get the prime factorization.

1.1 Splitting to Order Finding

Next, we reduce the problem of splitting to order finding, for which we have a quantum algorithm developed last class. Recall the definition of order finding:

- Input: $a, \mu \in \mathbb{N}$ such that $\gcd(a, \mu) = 1$
- Output: smallest positive $r \in \mathbb{N}$ such that $a^r \equiv 1 \bmod \mu$

Lemma 1. *If* $b^2 \equiv 1 \bmod \mu$ *and* $b \not\equiv \pm 1 \bmod \mu$*, then* $\gcd(b-1, \mu)$ *and* $\gcd(b+1, \mu)$ *are nontrivial factors of* μ*.*

Proof. We can rewrite the conditions in terms of divisibility: μ divides $b^2 - 1 = (b-1)(b+1)$, but μ divides neither $b-1$ nor $b+1$. From this, $b-1$ and $b+1$ both contain some, but not all, the factors of μ. □

Lemma 2. *Suppose at least k distinct primes divide μ. For at least $1 - \frac{1}{2^{k-1}}$ of the values of $a \in \mathbb{Z}_\mu^\times$, the order r of a is even, and $b \equiv a^{r/2} \bmod \mu$ satisfies $b \not\equiv \pm 1 \bmod \mu$.*

Proof. Omitted; use the Chinese Remainder Theorem. □

This is very useful for finding a factor of μ when μ is composite. If we choose a random element $a \in \mathbb{Z}_\mu^\times$, there is at least a $1/2$ chance that a satisfies the conditions in Lemma **??**. With an order finding algorithm, we can compute $b \equiv a^{r/2} \bmod \mu$, and then find the nontrivial factor $\gcd(b+1, \mu)$, allowing us to solve splitting for μ. Repeatedly trying different values of a will greatly increase the chance of success.

Of course, if we somehow choose $a \notin \mathbb{Z}_\mu^\times$, that is fine as well, because $\gcd(a, \mu)$ will be a nontrivial factor of μ (assuming $1 < a < \mu$).

1.2 Full Integer Factorization Algorithm

1. Check whether μ is a prime. If yes, return μ.
2. If $\mu = (\mu')^\ell$ for some integers μ' and $\ell > 1$; then, recursively factor μ' and repeat the factorization ℓ times
3. Pick a random $a \in \{1, \ldots, \mu - 1\}$.

   If $\gcd(a, \mu) \neq 1$ then $\mu' := \gcd(a, \mu)$. Else, compute the order r of a mod μ.

   If r is even and $b \equiv a^{r/2} \bmod \mu$ satisfies $b \not\equiv \pm 1 \bmod \mu$; then, $\mu' := \gcd(b + 1, \mu)$

   Else, try again with a new value of a.
4. Recursively factor μ' and $\frac{\mu}{\mu'}$.

Steps 1 and 2 can be done classically using polynomial time in n. Step 1 can even be done deterministically with the AKS primality test [**?**]. For step 2, we know that if μ' exists, then $\mu' \geq 2$. Therefore, we need only check $\ell = 2, \ldots, \lfloor \log_2 \mu \rfloor$, which can be brute forced.

When we reach step 3, we know that μ must have at least two distinct prime factors, so Lemma **??** guarantees at least a $1/2$ probability of success in step 3. With a quantum order finding algorithm, each iteration of step 3 can be completed using polynomial time in n. The expected number of iterations is 2, and the probability that more than k iterations are required is $1/2^k$, an exponential decrease.

Complexity Polynomial in n (polylogarithmic in μ) with a quantum computer.

Why is this algorithm important? The hardness of factoring integers is used prominently throughout many different cryptography systems. Being able to factor integers quickly breaks many commonly-used schemes, like RSA, which is often used to secure e-commerce transactions.

2 RSA Cryptosystem

RSA is a public-key cryptosystem wherein every participating entity has a publicly-viewable encryption key and a private decryption key that only they know. This protocol relies on the hardness of factoring for security. The acronym is composed of the first letter of each of the three computer scientists who developed the protocol, Rivest, Shamir, and Adleman.

The private key is composed of two randomly chosen distinct primes p and q, and an integer $d \in \mathbb{N}$ relatively prime to $(p-1)$ and $(q-1)$. The public key is the product $\mu = p \cdot q$, and $e \in \mathbb{N}$ such that $de \equiv 1 \bmod (p-1)(q-1)$. In other words, e is the modular inverse of d.

Before the protocol begins, the parties planning to use RSA must share a public key. For example, if Bob wants to send a message to Alice, he needs to know her public key in order to encrypt his message. Alice will need her private key in order to decrypt the message sent from Bob. The advantage of public key cryptography is that the public key does not need to be sent secretly.

Encryption Let $M \in \mathbb{Z}_\mu$ be the secret message. Bob computes the encrypted message $E \equiv M^e \bmod \mu$, using Alice's public key e, and sends it to Alice.

Decryption Alice receives the encrypted message E and calculates $D \equiv E^d \equiv M^{ed} \equiv M \bmod \mu$.

Proposition 3. *The decryption step recovers the original message, i.e., $D = M$.*

Proof. We need to prove that for all $M \in \mathbb{Z}_\mu$ and an integer n, $M^{ed} \equiv M^{n(p-1)(q-1)+1} \equiv M \bmod \mu$. To prove this, recall Fermat's little theorem, which states that $a^{p-1} \equiv 1 \bmod p$ for prime p and a satisfying $\gcd(a,p) = 1$. Therefore, for any integer k, $a^{k(p-1)} \equiv 1 \bmod p$ when $\gcd(a,p) = 1$. Going further,

$$a^{k(p-1)+1} \equiv a \pmod{p},$$

and this is even true when $a \equiv 0 \bmod p$.

Using this fact, we can deduce that $M^{n(p-1)(q-1)+1} \equiv M \bmod p$ and mod q, by using $k = n(q-1)$ and $n(p-1)$ respectively. Therefore,

$$M^{n(p-1)(q-1)+1} \equiv M \bmod \mu$$

by the Chinese remainder theorem. □

The security of this protocol depends on preventing the attacker from gaining d from e given μ. If the attacker can factorize an integer efficiently, then she can find p and q from μ, which allows her to know $(p-1)(q-1)$. Because calculating a modular inverse is efficient, the attacker can produce d from e.

Efficient integer factorization would allow an attacker to execute the steps described above. Classically, it is an open question as to whether RSA is secure, as no efficient classical integer factorization algorithm is known (although we cannot rule out its existence either). In addition, it is unknown if breaking RSA and integer factorization are equivalent, so there is the unlikely scenario that RSA can be broken without requiring efficient integer factorization.

3 Discrete Logarithm

For a prime p, we can define the *multiplicative group* $\mathbb{Z}_p^\times = \{x \in \mathbb{Z}_p : \gcd(x,p) = 1\} = \{1, 2, \ldots, p-1\}$, with the group operation of multiplication mod p. In fact, for prime p, this group is a cyclic group – it is isomorphic to $\mathbb{Z}_{p-1}$. Therefore, some elements $g \in \mathbb{Z}_p^\times$ are *generators*. This means that the order of $g \bmod p$ is $p-1$, the maximum possible order, and the set $\{g, g^2, \ldots, g^{p-1}\}$ is a permutation of the elements of $\mathbb{Z}_p^\times$.

In the discrete log problem, we are given a prime p and a generator $g \in \mathbb{Z}_p^\times$ as parameters, and the input is some $a \in \mathbb{Z}_p^\times$. The output is the exponent l such that $g^l \equiv a \bmod p$.

Similar to integer factorization, the discrete log problem has no known efficient classical algorithm, but there is an efficient quantum algorithm based on the hidden subgroup problem for abelian groups covered in lecture 20. The respective time complexities are shown below, where $n = \log p$.

- Classical time complexity: $2^{\widetilde{O}(n^{1/2})}$ (rigorous), $2^{\widetilde{O}(n^{1/3})}$ (heuristic)
- Quantum time complexity: $\widetilde{O}(n^2)$

4 Diffie-Hellman Key Exchange

The Diffie-Hellman protocol leverages the difficulty of the discrete log problem to develop a key exchange cryptosystem. DH allows two parties to communicate over an insecure communication channel by creating a shared secret key. The shared key that is created is used in subsequent messages sent between the two communicating parties. This method is known as a symmetric cryptography system. This system was developed by Whitfield Diffie and Martin Hellman in the late 1970s.

In DH, there are two public parameters: a prime p and a generator $g \in \mathbb{Z}_p^\times$. When Alice and Bob want to create a shared secret key, they use the protocol below.

1. Alice picks $a \in \mathbb{Z}_p^\times$ uniformly randomly, and sends $A \equiv g^a \bmod p$ to Bob. Bob picks $b \in \mathbb{Z}_p^\times$ uniformly randomly, and sends $B \equiv g^b \bmod p$ to Alice.
2. Alice computes $K_A \equiv B^a \bmod p$ and Bob computes $K_B \equiv A^b \bmod p$. Since $K_A \equiv K_B \equiv g^{ab} \bmod p$, Alice and Bob can use K_A and K_B as the shared key.

Note that if x is uniformly distributed in $\mathbb{Z}_p^\times$, then g^x is also uniformly distributed in $\mathbb{Z}_p^\times$, because the set $\{g, g^2, \ldots, g^{p-1}\}$ is a permutation of $\{1, 2, \ldots, p-1\}$. This prevents any easy brute force attack.

DH is vulnerable to any attacker that can efficiently calculate discrete logs. First, a and b can be extracted using discrete log from the intercepted messages A and B. From this, the attacker can compute ab and then the shared key $g^{ab} \bmod p$. In fact, only one of a or b is necessary, as the attacker can choose either A^b or $B^a \bmod p$ to compute the secret key.

The hardness of DH relies on a problem that is possibly easier than discrete log: given g^a and $g^b \bmod p$, compute $g^{ab} \bmod p$. It is unknown if breaking DH is equivalent to solving the full discrete log problem efficiently.

5 Quantum and Crypto

5.1 Challenges

The quantum algorithms for integer factorization and HSP on abelian groups threaten all cryptosystems that rely on the hardness of integer factorization and discrete log. Even algorithms relying on elliptic curves are compromised, as the underlying group formed by points on an elliptic curve is an abelian group.

An important challenge is constructing a new classical cryptosystem that is resilient against quantum computers. One option is lattice based cryptosystems. These systems base their hardness on the shortest vector problem: given a basis of integer vectors in $\mathbb{R}^n$, find an integer linear combination of that basis that results in the shortest vector. This problem can be reduced to solving HSP on dihedral groups, which are nonabelian. Currently, no efficient quantum algorithms for this instance of HSP are known, so these systems are conjectured to be safe against quantum computers.

One could go further and construct a cryptosystem based on the hardness of an NP-hard problem. Although the versions of the shortest vector problem used in cryptosystems are not known to be NP-hard, there are other problems such as quadratic programs which are NP-hard and lend themselves to cryptography. It is more plausible that quantum computers cannot solve NP-hard problems efficiently, but these systems have some drawbacks, such as large key sizes.

The largest integer factored with Shor's algorithm to date is 21, which means we still have time to devise a post-quantum cryptosystem.

5.2 Opportunities

There are still classical cryptosystems that are known to be secure against quantum computers, such as the zero knowledge proofs discussed in the next lecture. These systems are based on information-theoretic security.

Another exciting possibility is using quantum computers to devise new cryptosystems that were previously impossible with classical computers. For example, in the classical setting, informationally-secure key exchange is impossible, but surprisingly, this is possible with a quantum communications channel. Using an informationally-secure key exchange allows two parties to perform the one time pad, which allows informationally-secure two way communication. However, some possibilities remain out of reach, such as perfectly-secure bit commitment.

6 Bit Commitment

In bit commitment, Alice wants to commit to a bit b to Bob, while not revealing it until later. Think of a rock-paper-scissors game, where Alice does not want Bob to know her choice until Bob has decided, and Bob does not want Alice to change her choice afterwards to cheat.

Any bit commitment scheme $V(c, w, b)$ has two stages:

- Commit phase: Alice sends a random piece of information c that commits her to a bit b.
- Reveal phase: Alice then sends a witness w to Bob that lets him figure out the value of b.

For each bit $b = 0, 1$, we let $c^{(b)}$ denote a commitment for b. We will be loose with notation by using $c^{(b)}$ to also refer to the distribution of possible commitments for b.

We can say that a bit commitment scheme must be *hiding*, in that Bob cannot figure out b before the reveal phase, and *binding*, in that Alice cannot reveal a different bit from the b decided in the commit phase. More formally:

- Hiding: Any $c^{(0)}, c^{(1)}$ must be indistinguishable to Bob.
- Binding: Alice cannot find a bad witness w that makes $V(c^{(b)}, w, 1-b)$ accept.

These hiding and binding requirements can have varying levels of strictness.

- Hiding
  - Perfectly hiding: the distributions of $c^{(0)}$ and $c^{(1)}$ are identical.
  - Statistically hiding: the statistical distance between the distributions of $c^{(0)}, c^{(1)}$ is very small.
  - Computationally hiding: no efficient algorithm can distinguish between $c^{(0)}$ and $c^{(1)}$.
- Binding
  - Perfectly binding: no bad witnesses can exist.
  - Statistically binding: with a given $c^{(b)}$, there is a very small chance that a bad witness exists.
  - Computationally binding: no efficient algorithm can produce a bad witness.

The two properties of hiding and binding clash with one another. As it turns out it is impossible to satisfy both classically, at least in the case of perfect security.

Proposition 4. *Any classical bit commitment scheme that is perfectly hiding is not binding at all, i.e., for any commitment, Alice can always produce a witness for the wrong bit, assuming she is computationally unlimited.*

Proof. Any commitment $c^{(0)}$ must also have a witness w that causes $V(c^{(0)}, w, 1)$ to accept. If not, then the set of valid commitments for $b = 0$ and $b = 1$ are different, violating perfect hiding. However, this fact immediately violates perfect binding, because Alice can use this witness to reveal $b = 1$ after committing to $b = 0$. □

Perfectly hiding and binding schemes assume that Alice has unlimited computational power. In practice, it may be difficult to find a bad witness even though it exists. Most bit commitment schemes use the weaker requirement of computational hiding or binding.

6.1 Perfect Quantum Bit Commitment is Impossible

Originally, it was believed that perfectly secure bit commitment was still possible with quantum computing, just like with key exchange. Several flawed proposals for such a scheme were made until this problem was proven impossible for quantum computers.

Before proving that statement, we must first adapt our terminology to involve quantum states. Now, the scheme includes a quantum state represented by a density operator ρ. To reference the fact that Alice and Bob can only access their own qubits, we can use the reduced density operators ρ_A, ρ_B to reference the information each party can access. In this new framework:

- Hiding: $\rho_B^{(0)}$ is indistinguishable from $\rho_B^{(1)}$.
- Binding: Alice cannot pass verification with a value other than b.

Theorem 5. *Any quantum bit commitment scheme that is perfectly hiding is not binding at all, assuming Alice is computationally unlimited.*

The main idea behind the proof is decomposing the state using the Schmidt decomposition.

Lemma 6 (Schmidt Decomposition). *Given a state $|\psi_{AB}\rangle$, there exist orthonormal bases, $\{|\psi_{A,i}\rangle\}_i$ for Alice's part of the state, and $\{|\psi_{B,i}\rangle\}_i$ for Bob's part of the state, and nonnegative reals, λ_i such that $|\psi_{AB}\rangle = \sum_i \lambda_i |\psi_{A,i}\rangle |\psi_{B,i}\rangle$.*

Proof. This follows from singular value decomposition (SVD), by viewing $|\psi_{AB}\rangle$ as a matrix in the tensor product of Alice and Bob's subspaces.

Let $\{|e_i\rangle\}_i$ and $\{|f_i\rangle\}_i$ be the standard basis for Alice and Bob's subspaces, respectively. Then we can write

$$|\psi_{AB}\rangle = \sum_{i,j} \beta_{ij} |e_i\rangle |f_j\rangle .$$

We can interpret the entries of $|\psi_{AB}\rangle$ as a matrix $M_{AB} = (\beta_{ij})_{i,j} = \sum_{i,j} \beta_{ij} |e_i\rangle \langle f_j|$. Then by SVD, there exist unitary matrices U, V and a diagonal matrix Σ such that

$$M_{AB} = U \Sigma V^*.$$

Let $\{\lambda_i\}_i$ be the diagonal values of Σ, and let $\{|\psi_{A,i}\rangle\}_i, \{|\psi_{B,i}\rangle\}_i$ be the column vectors of U, V, respectively. Then the above equation becomes

$$M_{AB} = \sum_i \lambda_i |\psi_{A,i}\rangle \langle \psi_{B,i}|$$

which means

$$|\psi_{AB}\rangle = \sum_i \lambda_i |\psi_{A,i}\rangle |\psi_{B,i}\rangle .$$

Intuitively, SVD tells us that the bilinear form $\langle \psi_A | M_{AB} | \psi_B \rangle$ is diagonal when $|\psi_A\rangle$ is expressed in Alice's orthonormal basis and $|\psi_B\rangle$ in Bob's. This implies that $|\psi_{AB}\rangle$ becomes diagonal if Alice's part is expressed in her basis, and Bob's part in his basis. □

Corollary 7. *In terms of the reduced density operators,*

$$\rho_A = \sum_i \lambda_i^2 |\psi_{A,i}\rangle \langle \psi_{A,i}|$$

$$\rho_B = \sum_i \lambda_i^2 |\psi_{B,i}\rangle \langle \psi_{B,i}| .$$

Proof. We can write

$$\rho = |\psi\rangle \langle \psi| = \sum_i \lambda_i^2 |\psi_{A,i}\rangle \langle \psi_{A,i}| \otimes |\psi_{B,i}\rangle \langle \psi_{B,i}|$$

Taking the partial trace on the bases $\{|\psi_{A,i}\rangle\}_i, \{|\psi_{B,i}\rangle\}_i$, we get the desired result. □

We now prove the main theorem.

Proof. Suppose the scheme was perfectly hiding. Then consider the two states $\rho^{(0)}, \rho^{(1)}$ for each bit – perfect hiding implies that $\rho_B^{(0)} = \rho_B^{(1)}$. We only need to consider the purification of the overall state $\rho^{(b)}$ as $|\psi_{AB}^{(b)}\rangle$. Using the Schmidt decomposition, we have that

$$|\psi_{AB}^{(b)}\rangle = \sum_i \lambda_i^{(b)} |\psi_{A,i}^{(b)}\rangle |\psi_{B,i}^{(b)}\rangle .$$

Recall that $\rho_B^{(0)} = \rho_B^{(1)}$. By corollary **??**, we can write

$$\rho_B^{(b)} = \sum_i (\lambda_i^{(b)})^2 |\psi_{B,i}^{(b)}\rangle \langle\psi_{B,i}^{(b)}| .$$

Because identical density operators can be written to have an identical basis of eigenvectors, we can arrange $|\psi_{B,i}^{(b)}\rangle$ such that $\lambda_i^{(0)} = \lambda_i^{(1)}$ and $|\psi_{B_i}^{(0)}\rangle = |\psi_{B_i}^{(1)}\rangle$ for all i. Going back to the full state, we can see that the singular values and $|\psi_{B,i}\rangle$ do not depend on the value of b.

We can therefore write

$$|\psi_{AB}^{(b)}\rangle = \sum_i \lambda_i |\psi_{A,i}^{(b)}\rangle |\psi_{B,i}\rangle .$$

There exists a unitary U that Alice can perform that sends the orthonormal basis $|\psi_{A,i}^{(0)}\rangle$ to the orthonormal basis $|\psi_{A,i}^{(1)}\rangle$. This means that $(U \otimes I) |\psi_{AB}^{(0)}\rangle = |\psi_{AB}^{(1)}\rangle$. This violates perfect binding, because Alice can commit to $b = 0$ and, using only operations on her qubits, can change the state to accept $b = 1$. □

Of course, just like in classical systems, it may be difficult in practice to actually find the unitary U that allows Alice to change her commitment.

References

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, Sep 2004.