Lecture Topic: Applications in Security

Overview

The question of many people's minds is whether and when "quantum computers would kill RSA"? We will formalize this and review some recent work in these directions:

- generating random strings
- quantum key distribution
- Shor factoring
- Grover-based factoring
- variational factoring.

While the first two happily live in "vendor-land", the latter three are more involved.

Generating random strings

US authorities now recommend using random strings only from quantum effects, rather than pseudorandom generators. In some cases , quantum random number generators (RNG) come with strong guarantees, but often, it seems an overkill to utilize a quantum computer to generate random numbers. There are now purpose-built devices that can generate random strings at 17 Gbps, exceeding what can be done with near-term quantum computers. The purpose-built devices can be bought, e.g., from ID Quantique.

Quantum key distribution

An important quantum technology in security is quantum key distribution, which makes it possible to certify that the communication has not been intercepted. There are two approaches:

- *Prepare-and-measure*: measuring an unknown quantum state changes it.
- *Entanglement-based*: measuring one of two entangled quantum systems affects the other.

Either way, one can calculate the amount of information that has been intercepted by measurement. ID Quantiq showcased quantum key distribution at 307 km, and sells related devices. Toshiba demonstrated QKD at 100 km of fiber in 2004 and the first with a continuous key rate exceeding 10 Mbit/second in 2017. CTU has purchased such devices from both ID Quantiq and Toshiba.

Factoring integers

Much of modern cryptoprimitives are built on factoring of large integers. A textbook version of public-key cryptography, here cited from in verbatim, is as follows:

1. Select two large prime numbers, p and q.
2. Compute the product $n = pq$.
3. Select at random a small odd integer, e, that is relatively prime to $\phi(n) = (p-1)(q-1)$.
4. Compute d, the multiplicative inverse of e, modulo $\phi(n)$.
5. The RSA public key is the pair $P = (e, n)$. The RSA secret key is the pair $S = (d, n)$.

The encryption of message M on $\log n$ bits involves $M^e \mod n$ to obtain $E(M)$, while decryption requires $E(M)^d \mod n$.

Factoring integers

What is the complexity of factoring n to p and q?

- poly(log(n)) is the runtime of factoring algorithms on a BSS machine. Testing whether an integer is a prime is in P, but does not provide the factors, when the number is not prime.
- $O(n^{1/4})$ is the runtime of the best deterministic factoring algorithms for factoring an integer n with $\log n$ bits in length.
- $O(\exp(c(\log n)^{1/3}(\log\log n)^{2/3}))$ is the runtime of the best randomized algorithms, for some constant c and integer n. The runtime is thus subexponential, but not polynomial time: $O(\exp(\sqrt{(\log n)(\log\log n)}) = O(n)$. It is thus unlikely that factoring is NP-Complete. The elliptic curve method (ECM) is the fastest known algorithm for small numbers, e.g. within 100 digits. The the number field sieve (NFS) is the best classical algorithm for large numbers, and has been used to factor a 240-digit (795-bit) number in 900 core-years.
- $O((\log n)^2(\log\log n)(\log\log\log n))$ is the runtime of a quantum algorithm introduced by Peter Shor

Shor factoring

Peter Shor introduced an algorithm for factoring integers, which based on two facts of number theory, makes it possible to reduce factoring to order finding, i.e., determining r in $f(x + r) = f(x)$ for $f(x) = a^x$.

When one receives a composite number n, it uses $O(\log^3 n)$ order-finding operations to produce a non-trivial factor of n with a constant probability.

Shor factoring

Peter Shor introduced an algorithm for factoring integers, which based on two facts of number theory, makes it possible to reduce factoring to order finding, i.e., determining r in $f(x + r) = f(x)$ for $f(x) = a^x$.

When one receives a composite number n, it uses $O(\log^3 n)$ order-finding operations to produce a non-trivial factor of n with a constant probability.

Shor factoring

Suppose that n is an L-bit composite number, and x is a non-trivial solution to the equation $x^2 = 1 \mod n$ in the range $1 \leq x \leq n$, i.e., neither $x = 1 \mod N$ nor $x = n. - 1 = -1 \mod n$. Then at least $\gcd(x-1, n)$ and $\gcd(x+1, n)$ is a non-trivial factor of n can be computed using $O(L^3)$ operations.

Shor factoring

Suppose that $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_m^{\alpha_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq n-1$ and x is co-prime to n. Let r be the order of x mod n. Then the probability r is even and $x^{r/2} \neq -1 \mod n$ is greater or equal to $1 - \frac{1}{2^m}$.

Shor Factoring

1. If n is even, return 2.
2. If $n = a^b$ for $a \geq 1$ and $b \geq 2$, return a.
3. Choose x in [1, n-1]. If $\gcd(x, n) \geq 1$, return $\gcd(x, n)$.
4. Use order-finding to find the order r of x modulo n. If r is even and $x^{r/2} \neq -1 \mod n$ and either of $\gcd(x^{r/2} - 1, n)$ and $\gcd(x^{r/2} + 1, n)$ is non-trivial, return the non-trivial factor.
5. Repeat from 3 otherwise.

Shor Factoring

Shor's order-finding works as follows:

1. creates an initial, Q-qubit state $|0\rangle^{\otimes Q}$
2. apply Hadamard transform on it: $\frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} |k\rangle$
3. apply the function $f(x) = a^x \bmod N$ using $U_f|x, 0^n\rangle = |x, f(x)\rangle$ to obtain

$$U_f \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, 0^n\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, f(x)\rangle$$

   such that the value we are looking for is in the phase
4. apply the quantum Fourier transform: $\frac{1}{Q} \sum_{x=0}^{Q-1} \sum_{y=0}^{Q-1} \omega^{xy} |y, f(x)\rangle$
5. obtain y by measuring the first register. The probability of measuring $|y, z\rangle$ is

$$\frac{1}{Q^2} \frac{\sin^2(\frac{\pi m r y}{Q})}{\sin^2(\frac{\pi r y}{Q})}$$

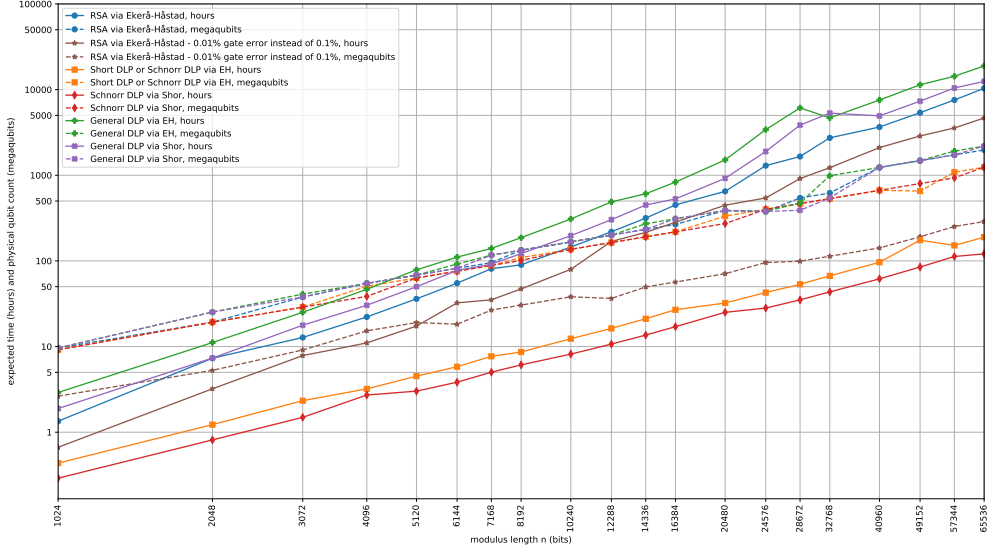
Shor Factoring of 15

1. Let us consider $n = 15$ and a random number x coprime (having no non-trivial common factors) with n, e.g., $x = 7$.
2. Compute the order r of x modulo n, as follows: apply Hadamard transform to the first register of $|0\rangle|0\rangle$. Compute $f(k) = x^k \mod n$ in the second register

$$\frac{1}{\sqrt{2^t}}[|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle + |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + \cdot].$$

   When inverse Fourier transform is applied to the first register (seen as $2^t = 2048$ frequencies) and the second register is measured, one obtains one of 1, 7, 4, or 13. Eventually, we obtain $r = 4$ as the order of $x = 7$.
3. When r is even, and $x^{r/2} \mod n = 7^2 \mod 15 = 4 \neq -1 \mod 15$, we run $\gcd(x^2 - 1, 15) = 3$ and $\gcd(x^2 + 1, 15) = 5$ to obtain two factors.

Scalability of Shor's Factoring



Scalability of Shor's Factoring

Shor's factoring has been demonstrated for the number of 15 more than two decades ago, and the scalability beyond is still very much a subject of lively discussion. A Google team estimates that one could perform factoring of 2048-bit RSA integers in 8 hours using 20 million noisy qubits. The assumptions of a planar grid of qubits with nearest-neighbor connectivity, physical gate error rate of 10^{-3}, a surface code cycle time of 1 microsecond, and the use of surface codes are all quite realistic. Surface codes are textbook material , although not covered by this course.

A French team suggested that one could perform factoring of 2048-bit RSA integers in 177 days with 13436 qubits, without being very explicit about the requirement of 430 million memory qubits. Likewise, the use of 3D gauge color codes is out of reach in current qubit technologies. Otherwise, the assumptions of physical gate error rate of 10^{-3}, a processor cycle time of 1 microsecond are quite realistic.

Scalability of Shor's Factoring

Shor's factoring has been demonstrated for the number of 15 more than two decades ago, and the scalability beyond is still very much a subject of lively discussion. A Google team estimates that one could perform factoring of 2048-bit RSA integers in 8 hours using 20 million noisy qubits. The assumptions of a planar grid of qubits with nearest-neighbor connectivity, physical gate error rate of 10^{-3}, a surface code cycle time of 1 microsecond, and the use of surface codes are all quite realistic. Surface codes are textbook material , although not covered by this course.

A French team suggested that one could perform factoring of 2048-bit RSA integers in 177 days with 13436 qubits, without being very explicit about the requirement of 430 million memory qubits. Likewise, the use of 3D gauge color codes is out of reach in current qubit technologies. Otherwise, the assumptions of physical gate error rate of 10^{-3}, a processor cycle time of 1 microsecond are quite realistic.

Grover-based factoring

Bernstein et al. introduced another quantum algorithm for factoring, which they call GEECM (Grover plus Edwards Elliptic Curve Method). To gain some intuition, consider the trial division, where we would generate a small primes and perform Grover search for those that divide the n.

It reduces the number of operations of Edwards Elliptic Curve Method from $L^{\sqrt{2}+o(1)}$ to $L^{1+o(1)}$ for $L = \exp(\sqrt{\log \sqrt{n} \log\log \sqrt{n}})$.

Grover-based factoring

Bernstein et al. introduced another quantum algorithm for factoring, which they call GEECM (Grover plus Edwards Elliptic Curve Method). To gain some intuition, consider the trial division, where we would generate a small primes and perform Grover search for those that divide the n.

It reduces the number of operations of Edwards Elliptic Curve Method from $L^{\sqrt{2}+o(1)}$ to $L^{1+o(1)}$ for $L = \exp(\sqrt{\log \sqrt{n} \log\log \sqrt{n}})$.

Variational factoring

In principle, you can use drastically fewer qubits in some cases, but with lesser hopes of speed-up. Notably, the explicit, 'schoolbook" binary multiplication of p and q yields equations that have to be satisfied by bits p_i and q_i and carry bits $z_{i,j}$. One can formulate a "least-squares version" of the problem, which would minimise the sum of residuals squared, across the equations (bits). Clearly, this would be a QUBO, as in the previous lecture, and approached with, e.g., QAOA without any guarantees of finding the solution. On the flipside, one can get lucky. For instance, Karamlou et al. report factoring 1099551473989, 3127, and 6557 with 3, 4, and 5 qubits, respectively, using a QAOA.

Schnorr factoring

In a very similar spirit, a Chinese team got to the frontpages of many newspapers announcing that 2048-bit semi-prime number can be factored on a NISQ level computer with 372 physical qubits and a gate depth in the thousands. Unfortunately, they did not analyze how many runs of the circuit this would require, which analyses show would scale much worse than the runtime of the Shor factoring.

A summary of factoring

In the US, Congress passed Quantum Computing Cybersecurity Preparedness Act in December 2022, which bars federal authorities from using cryptoprimitives based on factoring. It is unlikely that this is based on the discovery of a new factoring algorithm, but rather based on the risk of there being one. In many information security standards, you need to be sure that if you encrypt today, no one will be able to decrypt without knowing the key for the next 20+ years. In "Store Now, Decrypt Later" attacks, nation states already gain access to large troves of encrypted information, in the hope that they would be able to decrypt it in the near future. Notice that for digital signatures (e.g., certificates on the web), the risk is much less: you can wait until a new factoring algorithm appears.