Network Security

Professor Gaurav S. Kasbekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Week - 12
Lecture - 74
Post-Quantum Cryptography

Hello, in this lecture, we will discuss post-quantum cryptography. The idea is that quantum computers are being developed, and there have been shown to be some algorithms which can break the traditional public key schemes such as RSA and Diffie-Hellman. As quantum computers become more powerful, these existing schemes such as RSA and Diffie-Hellman will no longer be secure. New public key schemes have to be developed, and that's the domain of post-quantum cryptography, which we will discuss. Our discussion will be brief in the interest of time.

First of all, what is a quantum computer? It's a computer that exploits quantum mechanical phenomena such as superposition and entanglement. A scalable quantum computer is expected to be able to perform some calculations exponentially faster than classical computers. Examples of such calculations are as follows. One is factoring large products of prime numbers.

For example, if n is p times q, where p and q are prime numbers, then we discussed earlier that the security of RSA is based on the fact that there is no known efficient algorithm to factor n into its factors, p and q. But an algorithm called Shor's algorithm has been shown to be able to efficiently factor products of prime numbers using a quantum computer. Another example is unstructured search. For example, searching for a key in symmetric key encryption which converts a particular piece of plaintext to a ciphertext. So, that's searching through the key space, which is unstructured. So, a quantum computer can perform unstructured search faster than a classical computer.

One common misconception is the following: that a quantum computer is faster than a classical computer for all tasks. Since Moore's law is slowing down, eventually all classical computers will be replaced by quantum computers. But this is not true. There is only a narrow set of problems for which a quantum computer would be faster, and examples are those which we discussed earlier. The property of a quantum computer that makes it excel

at some tasks is that it can compute with a storage size of n, as if it were operating on 2^n values in parallel.

- Property of a quantum computer that makes it excel at some tasks is that:
  - it can compute, with only storage size n, as if it were operating on 2^n values in parallel

But it also has some serious limitations. For example, a state is generally the superposition of multiple binary states. For example, the counterpart of a bit in the classical domain is a so-called qubit. And a qubit is in the superposition of 0 and 1, but if you read or measure the state of a qubit, then you'll only see either a 0 or a 1, and the superposition vanishes. A typical quantum program tries to ensure that the quantum computation it performs raises the probability that when you finally measure a result, it will be a useful value.

So, a typical quantum program initializes some qubits, then performs some operations on them, and produces a result which, when read, will be useful with high probability. Another common misconception is the following: A quantum computer can solve NP-hard problems, such as the traveling salesperson problem and the set cover problem, in polynomial time. So, this is also almost certainly not true. Nobody has proved that it is impossible for a quantum computer to solve NP-hard problems, but on the other hand, there is no known quantum algorithm for solving such problems which is that powerful.

Although in principle a quantum computer can do any calculation that a classical computer can do, for most calculations, quantum computers would be no faster than conventional computers. So, it would be better to use classical computers for most tasks. The reason is that quantum computers are likely to be much more expensive to build and operate than classical computers. For example, most designs would need to operate at temperatures very close to absolute zero. So, the cost involved in cooling is very large.

For this reason, whenever possible, we'll perform our tasks using classical computers. So, in summary, it's not likely that quantum computers will ever serve more than a small niche market. For the rest of the computational tasks, classical computers will continue to be used. As we mentioned earlier, instead of bits, a quantum computer uses so-called qubits, where a qubit's state can be a mixture of a 0 and a 1. So, with some probability, the state is a 1, and with some probability, it is a 0.

This is known as superposition. A qubit is in the superposition of 0 and 1. Now, let's discuss some quantum algorithms that are relevant to cryptography. One is Grover's algorithm,

which we discussed on this slide, and the other is Shor's algorithm, which we discussed on the next slide. We start with Grover's algorithm.

It makes brute-force search faster. For example, if you want to find the key that converts a particular piece of plaintext to a particular piece of ciphertext, then a brute-force search can be used to find the key. And in the case of cryptographic hash function, a brute-force search can find a particular input for which the hash value is a particular output. That is, a brute-force search can find a pre-image for a cryptographic hash function. So. Grover's algorithm can do brute- force search in the square root of the time that it would take on a classical computer.

- Can do brute-force search in the square root of the time it would take on a classical computer, e.g.:
  - assume that we want to know which n-bit secret key k encrypts plaintext m to ciphertext c
  - on a classical computer, we could try all possible ($N = 2^n$) keys, and in the average case (respectively, worst case), it would take $\frac{N}{2}$ (respectively, N) guesses to find the right key
  - on a quantum computer running Grover's algorithm, the number of iterations to get n qubits into a state where it is very probable that the state will be read as the key k is proportional to $\sqrt{N} = 2^{n/2}$
- Squaring the size of the space being searched (e.g., using an encryption key twice as long) is adequate to protect against Grover's algorithm

For example, assume that we want to know which n-bit secret key k encrypts plaintext m to ciphertext c. On a classical computer, we could try all possible keys. Notice that there are $N = 2^n$ possible keys. Since the secret key is n bits in length, so there are $N = 2^n$ possible keys. On a classical computer, in the average case, it would take N/2 guesses to find the right key. And in the worst case, it would take N guesses to find the right key.

So, that is, the time required is of the order of 2^n. But on a quantum computer running Grover's algorithm, the number of iterations to get n qubits into a state where it is very probable that the state will be read as the key k is proportional to only $\sqrt{N}$, that is, $2^{n/2}$. Hence, the amount of time required is reduced. So, the time required is only square root of the time it would require on a classical computer. But it's not difficult to counter Grover's algorithm.

If we just square the size of the space being searched, for example, using an encryption key twice as long. So, instead of an encryption key that is n bits in length, we use a key that is 2n bits in length. So that will square the size of the space being searched because the number of possible keys would be 2^{2n}, which is 2^{n^2}. So, squaring the size of the space

being searched is adequate to protect against Grover's algorithm. So, it just requires square root of the time on a classical computer, but if we square the search space, then the amount of time required is the same as before.

The complexity of encryption and decryption will increase, but that's the cost we have to pay for better security. Then another algorithm is Shor's algorithm. It can efficiently factor numbers and calculate discrete logs. As we said, a number n can be factored into its prime factors, p and q, efficiently. And also it can calculate discrete logs, which we discussed in the context of Diffie-Hellman algorithm.

If run on a sufficiently large quantum computer, Shor's algorithm would break all our widely used public key algorithms, such as RSA and Diffie-Hellman. Currently no quantum computer large enough to break the currently deployed public keys has ever been publicly demonstrated. So, there are a number of difficult engineering challenges that remain and it may never be economically viable to overcome them. But the threat exists that such a computer might be possible. Hence, it's important to convert to quantum safe public key algorithms before a quantum computer of sufficient size may exist. People might store encrypted text and then in the future once a powerful quantum computer has been developed they may try to decrypt that stored ciphertext.

So, because of this threat, it's important to convert to quantum-safe public key algorithms well in advance before a quantum computer of sufficient size might exist. The cryptographic community is actively developing and standardizing such algorithms. We'll briefly discuss the efforts of the cryptographic community to develop and standardize such algorithms. So, replacement algorithms for existing public key algorithms such as RSA and Diffie-Hellman are being developed. And these replacement algorithms are such that not even a combination of classical and quantum computers will be able to break them in a reasonable time.

These algorithms are known by different names. For example, quantum-resistant algorithms, quantum-safe algorithms, or post-quantum cryptography. It's important to start migrating away from the current public key algorithms well before a sufficiently large quantum computer might exist. Because, as we said, people can store encrypted text and then later on, when there's a powerful quantum computer available, they can try to decrypt that ciphertext. Hence, we need to migrate away from the current public key algorithms well in advance.

NIST, which stands for the National Institute of Standards and Technology, has played an important role in the standardization of cryptography. For example, it helped in standardizing AES and the SHA family of cryptographic hash functions. NIST is also playing an important role in the standardization of post-quantum algorithms. It set up a competition for designing replacement algorithms for algorithms such as RSA and Diffie-Hellman.

So, this competition proceeded as follows. In late 2017 was the deadline for submissions of new ideas for designing public algorithms. So, at this deadline NIST received about 80 proposed schemes and several rounds were conducted and in each round some algorithms were discarded because possibly some security flaws were found or they were not efficient and other algorithms were studied more closely. And very recently, that is, on August 13th, 2024, the NIST released final versions of its first three post-quantum cryptographic standards. This is the stage currently, that is, three post-quantum cryptographic standards have been released.

You can look at the website of NIST for more details about these cryptographic standards. In general, four of the best-known families of post-quantum cryptographic schemes are these: hash-based cryptography, lattice-based cryptography, code-based cryptography, and multivariate cryptography. We'll discuss them in some more detail on the next slide. Hash-based signatures: as the name suggests, they are digital signatures constructed using cryptographic hash functions. Then lattice-based cryptography involve the construction of primitives that involve lattices.

Lattices are, as the name suggests, these are a set of regularly spaced points in some space. And lattice-based cryptography is based on lattices. Code-based cryptography schemes are based on error-correcting codes. And multivariate cryptography schemes are based on the difficulty of solving systems of multivariate polynomials over finite fields. In general, post quantum cryptography schemes are based on these four ideas.

There are other ideas as well. This is not an exhaustive list. But we'll omit the details of these four types of cryptography schemes. It would take an entire course to discuss these in detail. Now, we discussed that replacement algorithms being designed for public key algorithms.

What about symmetric key algorithms and hash functions? Recall that quantum computing poses a threat to current public key algorithms. In contrast, most current symmetric cryptographic algorithms and hash functions are considered to be relatively secure against

attacks by quantum computers. We discussed Grover's algorithm which can improve attacks on such algorithms. Grover's algorithm speeds up attacks against symmetric ciphers as well as cryptographic hash functions but just by doubling the key size we can effectively block these attacks.

So, we don't need to design fundamentally new techniques for symmetric key ciphers or cryptographic hash algorithms. Hence, post quantum symmetric key cryptography as well as cryptographic hash functions don't need to be significantly different from current symmetric key cryptography. So, in summary, post quantum public key algorithms are significantly different from classical public key algorithms and we discussed the four families of post quantum public key cryptographic algorithms but symmetric key algorithms and cryptographic hash functions are very similar in the post quantum era. They are similar to classical symmetric key ciphers and cryptographic hash functions. So, in summary, we discussed briefly what a quantum computer is and we discussed the threat that quantum computers might be able to break our existing public key algorithms. We discussed the standardization efforts of NIST and the four families of the most popular post quantum public key algorithms.

We discussed that symmetric algorithms and cryptographic hash functions are very similar in the post-quantum era compared to their classical counterparts. This concludes our discussion of post-quantum cryptography. Thank you.