**Lecture 11: Comparison among Cryptosystems**

**LEARNING OUTCOME**

**By the end of this lesson a student will be able to:**

1. understand a concept of general cryptosystems
2. have an overview of a transition among popular cryptosystems
3. see a transition from interger to polynomial over finite fields.

In general, a public key infrastructure(PKI) cryptosystem is operating on at least 2 domain rings. For instance, RSA operates publically on a ring modulo N. At the same time, there are 2 more smaller private rings over prime P and Q fields. A ring modulo N is public and a ring modulo prime P is private. However, a popular ECC is not operating in this mode. It follows a one way function.

On one hand, RSA is operating on integers. On the other hand, there are 2 ECC modes. An ECC over prime P field and another operates on finite field over an irreducible polynomial. Next, NTRU will make use of polynomial prowes. NTRU operates on a smaller rings mod a small p=3 and a slightly larger relatively prime q=256. Traditionally, it is more practical to operate and communicate on an integer ring. Operating on polynomials is more efficient and programming friendly.

In terms of popularity, we will give an estimate RSA~60-70%, ECC~20-30% and NTRU~5-10%. A good analogy here, Microsoft Windows 60-70%, Apple 20-30% and Linux~10% but now Android is on the way from mobile devices to a tab (a small notebook).

In term of efficiency, RSA is slow, ECC is faster but NTRU is the fastest. In term of running time RSA take O(*n*3) due to a power mod operation, ECC takes O(*n*3) due to point projection but NTRU takes O(*n*2) due to multiply modulo operation. Ring size varies but popular size is RSA~2048-bit, ECC~256-bit but NTRU~511-bit.

Table 1. Comparison among popular cryptosystems

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CryptoSystem | Year | Key Bit Size | Speed | Running Time | Market Popularity | IP Control | Post Quantum |
| AES | 2000 | 128 | Fastest | O(*c⋅ n*) | 90-95% | Open | Immune |
| NTRU | 1995 | 6400 | Faster | O(*n*2) | 5-10% | Open | Immune |
| ECC | 1985 | 256 | Fast | O(*n*3) | 20-30% | Strong | Vulnerable |
| RSA | 1978 | 2048 | Slow | O(*n*3) | 60-70% | Standard | Breakable |

The popular NTRU public-key algorithm competes with RSA and elliptic curve ECC. NTRUEncrypt was approved by the financial services standards body, the Accredited Standards Committee X9. The X9.98 standard specifies how to use NTRU, as it's called for short, in financial transactions.

Difficult Problems among Cryptosystem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CryptoSystem | Year | Bit Size | Speed | Problem |
| NTRU | 1995 | 6400 | Fastest | Square Lattice |
| ECC | 1985 | 256 | Faster | Discrete Log on EC |
| DH | 1979 | 2048 | Fast | Discrete Log |
| RSA | 1978 | 2048 | Fast | Integer Factoring |
| GM | 1982 | 2048 | Fast | Quadratic Residue |
| Knapsack | 1970 | 128 | Fast | Subset Sum=NPC |

A difficult Computer Science problem is NPC. A difficult mathematical problem is grouped as {IF, DL, QR}. Another group is called a square lattice problem which is being developed for a post quantum computer(PQC) cryptosystem.

Given a quantum computer, {IF, DL, QR} can be solved in real time. Is there going to be a quantum computer soon?

There is a Shor Algorithm to solve IF in polynomial time. It was developed in 1994 by the American mathematician Peter Shor.

There is RSA Number Challenge since 1991. The current record stand at 829-bit on RSA-250.

Why no one has been able to factor RSA number using quantum computer?

Simple questions:

1. How ECC can perform faster than RSA? At the same security level
2. How NTRU can perform faster than ECC? At the same security level
3. How fast is an AES in an encryption and decryption process?

Practically, AES performs faster than any PKI cryptosystems. In the year 2000, a reference code in c++ can encrypt at 1 megabit per second. In the year 2010, an optimised c plus assembly language can encrypt at 1 gigabit per second. In the year 2020, a crypto chip of AES can perform an encryption at 10 Gigabit per second.

Table 2. Comparison among popular cryptosystems within 20 years

|  |  |  |
| --- | --- | --- |
| CryptoSystem | Bit Size  Year 2000 | Bit Size  Year 2020 |
| AES | 128 | \*256 |
| NTRU | 251͂ | 511 |
| ECC | 160 | 256 |
| RSA | 1024 | 2048 |

RSA(1978-1980), ECC(1985), NTRU(1995-2000)

It was invented in the mid-1990s. Unlike RSA, NTRU is not widely used, and in fact the NTRU cryptosystem needed changes early on to improve its security by addressing weaknesses and performance. But today NTRU is recognised as faster than the widely used RSA algorithm.

NTRU at a high security level, is much faster than RSA (around five orders of magnitude) and ECC (around three orders of magnitude). This type of lattice design makes it more resistant than an algorithm like RSA to so-called quantum computing attacks.

Classical Computer Era

Quantum Computer Era

ECDH

1970

1980

1990

2000

2010

2020

DH

RSA

ECCP

Prime Field

ECCK

Polynomial Field

AES

NTRU

Figure 1. A transition from a prime field to a polynomial field within the last 5 decades.

AES and NTRU is expected to survive a full fledge quantum computer prowess. Currently, several research groups claim to have a large quantum computer in terms of several thousands qubits. However, no one has ever show that he can break medium size RSA via these quantum computers even though there is an explicit Shor’s algorithm to do so.

There are RSA challenge numbers put forward by RSA Laboratories back on 18 March 1991. Even though the challenge was ended in 2007, a succes of factoring on remaining RSA numbers is still considered a new world record. The RSA challenge numbers ranges from 330 bits to 2048 bits.

RSA has been designed based on an intractable integer factoring problem. Shor's algorithm shows that factoring integers can be efficiently done on an ideal quantum computer. And (in fact) there is an explicit Shor’s algorithm to factor an integer via a quantum computer. However, no one has managed to factor a viable RSA modulo N=PQ. Why no one has been able to show a simple factoring manuver on a practical RSA modulo?

Presumably, there will be a viable quantum computer in the near future. Any mathematically related to RSA cryptosystems will be vulnerable to quantum computer prowess. There is a call for a post quantum algorithm to secure future communication.

**Quantum Computer**

Omni presence: It is present everywhere(at least 2 places) but one place at a time. An element in physic which is not bounded by time and space or physical distance. Philosopically, a good example here is a soul.

Virtual presence: Something that is not present but accounted for. A virtual reality over an internet is a good example.

Spiritual Organisation: These people teach and live by aura or energy. A major group is Sai Baba. A middle layer member can see and describe other people aura. Now, there is a camera that can capture an aura.

An atom is a matter. A matter is a soul of an atom. It is one and many. Split an atom but what it is more important, we split a matter and send into 2 different distant places. If we touch one at a venue A, the other at another venue B will feel the touch. If we change the colour of the matter at A, another entangled twin matter at B will also change. This atomic split is a starting point on teleportation.

0

1

Let us introduce a qubit: *x* = 0 + 1*i*. If an *x* is on the upper hemisphere, then it is 0. If an *x* is on the lower hemisphere, then it is 1.

In the last 20 years, I have observed in Quantum computing, every other people will say we will or might have quantum computer soon in the immediate future. This is the same story of going to Mars.

There is an algorithm : Shor Algorithm to factor N=PQ and there is supposedly a quantum computer; why so far someone has factor 3\*5 = 15 only via a quantum computer?

There are several package which are efficient to go into at this juncture for you.

1. Graphical Processing Unit(GPU) instead of parallel processing
2. Quantum Computing over the cloud through Qiskit.