Python Implementation of Quantum Resistant Lattice Cryptography

Liangyu Wang

***Abstract*—In August 2015, the U.S. National Security Agency (NSA) released a major policy statement urging the need to move away from Elliptic Curve-based cryptographic systems to post-quantum cryptography. One such candidate for quantum-resistant cryptographic system is lattice cryptography. Lattice cryptography is an umbrella term for cryptographic systems constructed using mathematical objects called lattices. The security of lattice cryptographic system depends on the difficulty of solving several mathematical problems involving lattice, such as shortest vector problem, closest vector problem, bounded distance decoding problem, shortest integer solution problem; these problems are believed to be difficult for quantum computer to solve. In our paper, we look mainly look at Learning with Error cryptographic system, Ring Learning with Error cryptographic system, Ring Learning with Error Diffie-Hellman key exchange, and their implementation in the Python programming language.**

Keywords— lattice cryptography, quantumresistant, learning with error, ring learning with error, diffie-hellman key exchange

I. Introduction

Since its creation in the 1980s, NSA has been a major proponent of Elliptic Curve cryptographic (ECC) systems, it has recommended ECC to be used to protect the communications of all US government agencies. However in 2015, NSA released a major policy statement on its website on the need to move away from elliptic curve cryptography and develop standards for post-quantum cryptography:

"Unfortunately, the growth of elliptic curve use has bumped up against the fact of continued progress in the research on quantum computing, which has made it clear that elliptic curve cryptography is not the long term solution many once hoped it would be. Thus, we have been obligated to update our strategy."

This has prompted widespread speculations that NSA has found a way to break elliptic curve cryptographic systems.

In 2016, NIST(National Institute of Standards and Technology) began the Post-Quantum Cryptography Standardization project in order to "standardize one or more quantum-resistant public-key cryptographic algorithms" that is publicly disclosed worldwide and "capable of protecting sensitive government information well into the foreseeable future". Of the 26 submissions that advanced to the second round, most fall into three large families: lattice (12), code-based (7), multivariate polynomial (4). Lattice cryptography is based on hard to solve problems from the mathematical theory of lattices; Code-based cryptography is based on the hardness of decoding linear error-correcting codes; Multivariate polynomial cryptography is based on the hardness of solving multi-variate system of polynomial equations. Older proposals of multivariate polynomial cryptography has already been broken, and recent proposals have not yet been thoroughly studied; Code-based cryptography is generally considered not very efficient and requires very large key size. This combined with performance that is competitive with or even better than classical encryption algorithms like RSA and ECC leaves lattice cryptography as the strongest contender for post-quantum cryptography.

Lattice cryptography is considered quantum-resistant because at present there is no known quantum algorithm that can solve the hard lattice problems faster than conventional computer. The best known algorithms for solving lattice problems either run in exponential time or have very bad approximation ratios.

II. From Lattices to Lattice Cryptography

Lattices are abstract mathematical constructs from algebra and group theory. Given any basis of R^n, the subgroup of all linear combinations with integer coefficients of the basis vectors forms a lattice. We can view a lattice as an orderly arrangement of points in R^n, and each point is a vector that extends from the origin. Given a lattice L of dimension n, L divides R^n into equal copies called the fundamental domains, the n-dimensional volume of a fundamental domain is called the determinant of the lattice L.

Difficult lattice problems have been studied by mathematicians for over 200 years. The most famous and widely studied lattice problem are the Shortest Vector Problem (SVP): given a lattice L, find the shortest non-zero vector in L; and the Closest Vector Problem (CVP): given a vector v not in a lattice L, find the point in L that is closest to v. In the 18th century, Carl Friedrich Gauss discovered an algorithm to compute the shortest vector in 2-dimensional lattices. In the 19th century, Herman Minkowski and Charles Hermite proved that every lattice L of dimension n contains a short non-zero vector with length bounded by the dimension and determinant of L. In 1982, Arjen Lenstra, Hendrik Lenstra and László Lovász invented the celebrated LLL Lattice Basis Reduction algorithm that can approximate the shortest vector in a lattice, but can also be used to break existing cryptosystems like knapsack cryptosystem and RSA. Despite its long history, to date, there has not been major advances in solving these difficult lattice problems.

The first generation of lattice-based cryptographic systems came in the early 1990s, they are Ajtai-Dwork cryptosystem, GGH cryptosystem of Goldreich, Goldwasser and Halevi, and the NTRU cryptosystem proposed by Hoffstein, Pipher and Silverman.

The first lattice-based cryptosystem was Ajtai-Dwork cryptosystem created by Miklós Ajtai and Cynthia Dwork in 1996. Ajtai introduced a new difficult lattice problem called the Shortest Integer Solution (SIS) problem, which can be viewed as a variant of subset-sum problem over a particular additive group, and used it to create cryptographic primitives such as one way collision-resistant hash functions, identification schemes, digital signatures, but not public key encryption. However, Nguyen and Stern subsequently showed that any practical and efficient implementation of the Ajtai-Dwork system is insecure. [1]

In 1997, Oded Goldreich, Shafi Goldwasser and Shai Halevi proposed the GGH cryptosystem. In GGH, the public key is some "bad" basis W of a lattice L and some random uni-modular integer matrix U, and the private key is some "good" basis V of L such that $W = UV$. "bad" basis consists of long, non-orthogonal vectors, "good" basis consists of short, orthogonal vectors. Plaintext is a binary vector m. Encryption is done by perturbing mW by a small random error vector e, so that ciphertext = $mW + e$. Decryption is done by using the private key V to find the vector v in L closest to e. However, GGH cryptosystem was broken in 1999 by Phong Q. Nguyen. [1]

In 1996, mathematicians Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman developed the NTRU cryptosystem. In NTRU, operations are performed in convolution polynomial quotient rings R^p and R^q. The private key consists of a random polynomial f with coefficients of either 1, 0 or -1 that is invertible in R^p and R^q and a random polynomial g with coefficients of either 1, 0 or -1. f_p is the inverse of f in R_p, and f_q is the inverse of f in R_q. The public key is $h = f_q \cdot g$. The plaintext is a binary polynomial $m \in R_p$. Encryption is done by choosing a small random error polynomial e and calculate ciphertext $c = pe \cdot h + m$. Decryption is done by computing $f \cdot c = pg \cdot e + f \cdot m$ (mod q), center-lift to a polynomial $a \in R$ and then compute plaintext = $f_p \cdot a$ (mod p). [1] The best attacks against NTRU uses the LLL Lattice Basis Reduction algorithm, but becomes impractical for NTRU using very large dimension lattices. NTRU is the oldest secure lattice cryptosystem, but its decryption algorithm can sometimes fail to correctly decrypt ciphertexts. [2]

III. Learning With Error

In 2005, Oded Regev introduced the first second generation lattice cryptographic system based on the learning with errors (LWE) problem. The LWE problem can be viewed as an "public key encryption-enabling" dual of the SIS problem introduced by Ajtai-Dwork, such that Regev's LWE-based cryptosystem operates very similarly to the Ajtai-Dwork cryptosystem.[3] There are two versions of the learning with error problem: search and decision version.

Search Learning With Error Problem: given m independent samples (a_i, b_i), find uniformly random secret s such that $a_i s + e = b_i$, where e is a small error randomly sampled from a Gaussian distribution.

Decision Learning with Error Problem: Let s be a uniformly random secret and e a small error randomly sampled from a Gaussian distribution. Given m independent samples (a_i, b_i), distinguish which is (a_i, b_i) = (a_i, $a_i s + e$) and which is a uniform sample.

Regev proved that having a procedure for solving the search-LWE problem would automatically yield a solution to the decision-LWE problem and vice versa.

In addition, Regev proved that the search-LWE problem is at least as hard as the hardest lattice problems, via a quantum reduction.

Regev's proposed public key cryptographic system operates as follows:

Alice chooses a modulus q, an $m \times n$ matrix A, a secret key s and a "small" error vector e sampled from a Gaussian distribution.

Alice calculate $b = sA + e$, then publishes (A, b, q) as her public key. Here, the dimension of the matrix A is a security parameter that determines the strength of the cryptographic system.

Bobby wants to send a message to Alice. He convert the message into a binary vector x , then calculate $u = Ax$ and $v = bx + bit \cdot q/2$, where bit is an element of x . Bobby then send the ciphertext (u , v) to Alice.

When Alice receives (u , v) she decrypt the ciphertext by computing $v - su$. The key insight here is that the term ex is very small compared to $q/2$, therefore $v - su \approx bit \cdot q/2$. Thus if Alice's calculation evaluates to a value close to zero, she will know Bobby sent her a bit of 0; if her calculation evaluates to a value close to $q/2$, she will know Bobby sent her a bit of 1.

Regev's system is semantically secure against passive eavesdroppers, assuming that decision-LWE problem is hard, then LWE encrypted ciphertext will look indistinguishable from random numbers sampled from a Gaussian distribution.

One disadvantage of public key system based on LWE is that it requires very large key size which is a large $m \times n$ matrix. Suppose we use a 640×256 matrix, and set q = 4093, then the public key size of the LWE cryptographic system is calculated to be 640 × 257 × $\log_2$ (4093) = 1973586 bits. Much larger than 3072-bit RSA public key or 256-bit elliptic curve public key.

IV. Ring Learning with Error

In 2012, Vadim Lyubashevsky, Chris Peikert and Oded Regev proposed a more efficient and practical version of the LWE cryptographic system called Ring Learning with Error (RLWE).

Similar to NTRU, RLWE exploits extra algebraic structures by performing encryption and decryption operations in polynomial rings over finite fields. Polynomials can be multiplied in a natural way that is not true of vectors in a Euclidean vector space. The product of two polynomials is another polynomial, while the dot product of two vectors in the euclidean vector space is a scalar. Polynomials can also be multiplied very efficiently using Fast Fourier Transform.

With RLWE we can quadratically reduce the key size, since instead of using a large $m \times n$ matrix as public key, we can just use an m th degree polynomial as the public key. For example, to achieve 128-bits of security, LWE will require a key size of 58,982,400 bits, while RLWE only need a key size of 7680 bits.

In RLWE, we work with a finite number of polynomials. Just like we can do arithmetic with only integers modulus some prime number p , we can also work only with polynomials modulus some irreducible polynomial. Just like a prime number is a positive integer not divisible by any number other than 1 and itself, an irreducible polynomial is only divisible by a constant or itself.

In RLWE, we generally work with polynomials modulus irreducible polynomial of the form $x^n + 1$, where n is a power of 2; polynomials of this form are called cyclotomic polynomials and their rich structure makes ring calculations more efficient. Here n is a security parameter that determines the strength of the cryptographic system.

The search and decision Learning with Error problems can now be reformulated for Ring Learning with Error:

Let R be a polynomial ring.

Let $a_i(x) \in R$ be a set of small random and known polynomials.

Let $e_i(x) \in R$ be a set of small random and unknown polynomials.

Let $S(x) \in R$ be a small secret and unknown polynomial.

Let $b_i(x) = (a_i(x) \times s(x)) + e_i(x)$

Search Ring Learning With Error Problem: given m random pairs ($a_i(x)$; $b_i(x)$), find $s(x)$.

Decision Ring Learning with Error Problem: given m random polynomial pairs ($a_i(x)$; $t_i(x)$), determine whether $t_i(x) = (a_i(x) \times s(x))$+ $e_i(x)$ or $t_i(x)$ is randomly generated from R .

The Ring learning with Error public key cryptographic system is defined as follows:

Alice Selects a polynomial ring R_p , and a polynomial $a \in R_p$ such that the coefficients of a are uniformly random positive integers less than p , as well as two random "small" polynomials $s, e \in R$ where the coefficients of s and e are sampled from a discrete Gaussian error distribution X . Here s is Alice's the secret private key. Alice calculates $b = a \cdot s + e$ and publishes (a , b) as her public key.

In order to send an n-bit binary message m , Bobby encodes the message m into the 0-1 coefficients of a polynomial z . Bobby then chooses 3 random "small" polynomials $r, e_1, e_2 \in R$ from X .

To encrypt the message, Bobby calculate $u = a \cdot r + e_1$ (mod p) and $v = b \cdot r + e_2 + z \cdot p/2$ (mod p), then send (u , v) as the ciphertext.

Upon receiving the ciphertext (u , v) , Alice decrypt by calculate $v - u \cdot s$, then round each coefficient of the resulting polynomial to either 0 or $p/2$, whichever is closest modulo p, to recover the message m.

V. RLWE Diffie-Hellman Key Exchange

In 2011, Jintai Ding of the University of Cincinnati proposed the first Diffie-Hellman style key exchange protocol for RLWE. It is based on the idea that the public key polynomial a or matrix A is analogous to the generator of a (multiplicative) cyclic group, and taking noisy products is analogous to exponentiation.

The basic key exchange protocol is Alice and Bobby agree on a public key $a \in R_q$. Alice chooses small secret s_1 and s_0, and calculate $b = a \cdot s_1 + s_0$ and send b to Bobby.

Bobby chooses small secret e_0, e_1, and calculate $u = e_0 \cdot a + e_1$ and send u to Alice.

Alice can now calculate the shared secret $S = u \cdot s_1$ and Bobby picks a small secret e_2 and calculate the shared secret $S' = e_0 b + e_2$. However, S and S' are not exactly equal because of the added small noise.

Ding proposed two functions to remove the noise and achieve an exact key agreement. A characteristic function SIG defined by $SIG(v) = 0$ if $v \in (-q/4, q/4)$ and $SIG(v) = 1$ otherwise. And a reconciliation function

$$Mod(v, w) = v + w*(q-1)/2 \pmod{q} \pmod{2}.$$

By applying Sig function to S' to obtain $w = SIG(S')$, Bobby can calculate the shared secret $SS = Mod(S', w)$ and send w to Alice.

Alice can then calculate the shared secret $SS = Mod(S, w)$. [4]

In 2014, Chris Peikert proposed a more accurate Reconciliation scheme.

Peikert proposed four functions: Randomized Rounding Function, Modular Rounding Function, Cross Rounding Function and Reconciliation Function.

Randomized rounding function partitions the coefficients of a polynomial into four quadrants then probabilistically distribute the coefficients on the boundaries of the quadrants such that the numbers of coefficients in quadrants representing 0 and 1 balances out.

Modular Rounding Function generates a secret key stream using the coefficients of a randomly rounded polynomial; if a coefficient $c \in (q/4, 3q/4)$, then it is represented by a bit of 1, else it is represented by a bit of 0.

Cross Rounding Function generates an array of masking bits using the coefficients of a randomly rounded polynomial; if a coefficient $c \in [q/4, q/2) \cup [3q/4, q)$, then it is represented by a masking bit of 1, else it is represented by a masking bit of 0. The generated masking bits is a hint that Bobby sends to Alice to help her remove the noise in her shared secret. The masking bits do not contain enough information that would allow an attacker to calculate the shared secret.

Reconciliation function uses the masking bits to remove the noise from shared secret S. For every coefficient $c \in S$, if the masking bit is 0, c will be represented by a key of 1 if $c \in [3q/8, 7q/7)$ or a key of 0 otherwise; and if the masking bit is 1, c will be represented by a key of 1 if $c \in [q/8, 5q/8)$ or a key of 0 otherwise. [5]

In 2015, Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe proposed a new reconciliation scheme for RLWE called New Hope, it claims to achieve greater reliability and security using high-dimensional geometry to achieve exact key agreement.

VI. PYTHON IMPLEMENTATION

We implemented Learning with Error, Ring Learning with Error and Ring Learning with Error Diffie-Hellman Key Exchange in the Python programming language, using the Numpy mathematical library.

In our Python implementation, integer matrix is randomly generated using the random.randint() function. matrix multiplications are performed using the matmul() function. polynomials are represented by integer arrays. Polynomial operations are performed using polymul(), polyadd(), polydiv() function.

Before a string is encrypted using RLWE, it is first converted into a binary string, and then the binary string is converted into a polynomial. For example, if we want to convert the character “s” into a polynomial, we first convert it into the binary string “01110011”, then we create a polynomial using the binary string as coefficients. The coefficients for the terms x^7, x^3, x^2 are 0, so these terms can be omitted, therefore the character “s” is converted into the polynomial $x^6 + x^5 + x^4 + x + 1$.

In our implementation of RLWE, we set the value of q to 255, so that ciphertext can be easily converted into base64 format. The dimension of the lattice is set to a multiple of 8 since every plaintext character is converted into an 8-bit binary string. After ciphertext is calculated, it is converted into base64 format, which allows more efficient transmission.

[1] Jeffery Hoffstein, Jill Pipher and Joseph H. Silverman, An Introduction To Mathematical Cryptography, Springer, 2014.

[2] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. "The Impact of Decryption Failures on the Security of NTRU Encryption", Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA.

[3] Chris Peikert, “A Decade of Lattice Cryptography”, February 17, 2016.

[4] Jintai Ding, Xiang Xie, Xiaodong Lin, “A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem”, University of Cincinnati.

[5] Chris Peikert, “Lattice Cryptography for the Internet”, July 16, 2014.

[6] Vikram Singh, “A Practical Key Exchange for the Internet using Lattice Cryptography”, 2008.

[7] Vikram Singh, Arjun Chopra, “Even More Practical Key Exchanges for the Internet using Lattice Cryptography”

[8] Vadim Lyubashevsky, Chris Peikert, Oded Regev “On Ideal Lattices and Learning with Errors Over Rings”, 2013

[9] Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim and Taewan Kim, “Lattice Based Cryptography for Beginners”, 2008

[10] Douglas Stebila, “Introduction to post-quantum cryptography and learning with errors”, 2018