10 Lattice-Based Cryptography

10.1 Rings of Polynomials

A ring is in essence a set in which we can do addition and multiplication, but not necessarily division Formally:

Definition 10.1. A *ring* is a set R together with two operations $+$ and $\cdot$, such that

1. R is an abelian group under the operation $+$, with identity 0;
2. Multiplication is commutative, and has identity element 1;
3. and we have the distributive law $k(x+y) = kx + ky$.

Example 10.2. 1. Every field is a ring.

2. The integers $\mathbb{Z}$ form a ring.
3. $\mathbb{Z}/m\mathbb{Z}$, the integers modulo m, form a ring for any m. (They form a field if and only if m is prime).
4. The set of all $m \times n$ matrices does *not* form a ring under this definition, because I required multiplication to be commutative.
5. The set of functions $f : \mathbb{R} \to \mathbb{R}$ is a ring under pointwise addition and pointwise multiplication.
6. The set of polynomials in one variable with rational coefficients $\mathbb{Q}[x]$ form a ring.
7. In fact, if R is any ring, then the set of polynomials with coefficients in R forms a ring $R[x]$.

This last example is the one we're most interested in. In particular we want to look at one particular type of polynomial ring:

Definition 10.3. The ring of polynomials with integer coefficients is $\mathbb{Z}[x] = \{a_0 + a_1 x + \cdots + a_n x^n : a_i \in \mathbb{Z}, n \in \mathbb{N}\}$.

The ring of polynomials with mod m coefficients is $\mathbb{Z}/m\mathbb{Z}[x] = \{a_0 + a_1 x + \cdots + a_n x^n : a_i \in \mathbb{Z}/m\mathbb{Z}, n \in \mathbb{N}\}$.

Example 10.4. Consider the ring $\mathbb{Z}/5\mathbb{Z}[x]$. Let $f(x) = x^2 + 3x + 1, g(x) = x^3 + 2x^2 + 4 \in \mathbb{Z}/5\mathbb{Z}[x]$. Then we compute $f(x) + g(x) = x^3 + 3x^2 + 3x$ and

$$\begin{aligned} f \cdot g(x) &= x^5 + 3x^4 + x^3 + 2x^4 + x^3 + 2x^2 + 4x^2 + 2x + 4 \\ &= x^5 + 2x^3 + x^2 + 2x + 4. \end{aligned}$$

Even though our coefficients are computed mod m, this ring still has infinitely many elements. We'd like to cut it down to finitely many elements so we can do cryptography with it. When we cut the integers down from an infinite set to a finite set, we modded out by an integer. To cut these polynomials down to a finite set, we need to mod out by a polynomial.

Definition 10.5. Let R be a ring and $r_1, \ldots, r_n \in R$. The *ideal* generated by r_i, written $\langle r_1, \ldots, r_n \rangle$, is the set of all linear combinations of the r_i. That is,

$$\langle r_1, \ldots, r_n \rangle = \{r_1 s_1 + r_2 s_2 + \cdots + r_n s_n : s_i \in R\}.$$

In particular, if $f \in \mathbb{Z}/m\mathbb{Z}[x]$ then $\langle f \rangle = \{f(x)g(x) : g(x) \in \mathbb{Z}/m\mathbb{Z}[x]\}$.

If R is a ring and I is an ideal in R, and $r, s \in R$, we say $r = s + I$ or $r = s \mod I$ if $r - s \in I$. We write R/I for the set of equivalence classes of R modulo I. R/I is a ring under the operations inherited from R.

Example 10.6. If $R = \mathbb{Z}$ and $I = \langle m \rangle$ then $R/I = \mathbb{Z}/m\mathbb{Z}$. This is the usual modular arithmetic you're already familiar with.

If $R = \mathbb{Z}[x]$ and $I = \langle m \rangle$ for $m \in \mathbb{Z}$ then $R/I = \mathbb{Z}/m\mathbb{Z}[x]$.

Example 10.7. The example we're most interested in is the case where $R = \mathbb{Z}/m\mathbb{Z}[x]$ and $I = \langle x^n + 1 \rangle$ for some $n = 2^k$. (This polynomial is the *$2n$th cyclotomic polynomial* $\Phi_{2n}(x)$, but we won't need to worry about that broader context).

In R/I we have $x^n = -1 + I$, so any time we have an x^n we can replace it with a -1. (And similarly we can replace x^{n+1} with $-x$ and x^{n+2} with $-x^2$, and so on). Thus a complete set of representatives for $R/I = \mathbb{Z}/m\mathbb{Z}[x]/\langle x^n + 1 \rangle$ is $\{a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} : a_i \in \mathbb{Z}/m\mathbb{Z}\}$.

Note this definition is different from earlier; this n is fixed by our choice of $I = \langle x^n + 1 \rangle$.

Remark 10.8. We could also view this ring as $\mathbb{Z}[x]/\langle m, x^n + 1 \rangle$; it doesn't matter what order we do the modding in.

Example 10.9. Let $R = \mathbb{Z}/5\mathbb{Z}[x]/\langle x^4+1\rangle$. Let $f(x) = x^2+3x+1, g(x) = x^3+2x^2+4 \in R$. Then we compute $f(x)+g(x) = x^3+3x^2+3x$ and

$$\begin{aligned} f \cdot g(x) &= x^5 + 3x^4 + x^3 + 2x^4 + x^3 + 2x^2 + 4x^2 + 2x + 4 \\ &= (-1)x + 2x^3 + x^2 + 2x + 4 \\ &= 2x^3 + x^2 + x + 4. \end{aligned}$$

10.2 Ring-LWE

Let $f(x) = x^n + 1 = \Phi_{2n}$ where $n = 2^k$. (This guarantees among other things that f has no rational roots). Let q be a large prime with $q \equiv 1 \mod 2n$, and set $R_q = \mathbb{Z}/q\mathbb{Z}[x]/\langle f(x)\rangle = \mathbb{Z}[x]/\langle q, f(x)\rangle$.

We want to be able to talk about small errors, which means we need some idea of the "size" of a polynomial. There are a number of choices we could make here, but the simplest is to look at the largest coefficient of the polynomial.

Definition 10.10. Let $f(x) \in R$ and write $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ where each $a_i \in \left\{\frac{-(q-1)}{2}, \dots, -1, 0, 1, \dots, \frac{q-1}{2}\right\}$. That is, we choose each coefficient to be as close to zero as possible.

We define the *infinity norm of f* to be $\|f\|_\infty = \max\{|a_i|\}$. Thus the norm is the magnitude of the largest-magnitude coefficient.

To set up a Ring Learning with Errors problem, we first need to specify a probability distribution on polynomials that guarantees that the polynomials are "probably" "small". The simplest example of this is to pick some bound b, and then choose each coefficient uniformly at random from the set $\{0, 1, \dots, b\}$. Then:

1. Let $a_i(x)$ be a set of random known polynomials in R.
2. Let e_i be a set of random unknown polynomials that are small with respect to the bound b.
3. Let $s(x)$ be an unknown polynomial with $\|s\|_\infty \leq b$.
4. Set $b_i(x) = (a_i(x) \cdot s(x)) + e_i(x)$.

The *Ring-LWE Decision Problem* is, given a list of pairs $(a_i(x), b_i(x))$, to determine whether the $b_i(x)$ were generated from the $a_i(x)$ by this process, or whether they were independently randomly generated.

The *Ring-LWE Search Problem* is, given a list of pairs $(a_i(x), b_i(x))$ generated by this process, to determine s.

Theorem 10.11 (Lyubashevsky, Peikert, Regev)**.** *The Ring-LWE Search Problem is at least as hard as the worst-case approximate shortest vector problem, even on a quantum computer.*

This proof is done by turning the Ring-LWE question into a question about lattices, much as we did to knapsack encrpytion in week 9. In fact, these polynomial rings naturally form mod-q lattices of dimension n, generated by the elements $1, x, x^2, \ldots, x^{n-1}$.

The Ring-LWE problem fundamentally involves the introduction of error terms, and thus has fundamental randomness. Thus it's never possible to solve a problem with 100% certainty; however, we can force the chances of error to be as low as we wish by careful choice of parameters.

10.3 Rounding and Masking

There are a number of different variations and implementations of Ring-LWE that have been proposed; Google has been using an algorithm called New Hope for the past year. Here we will present an algorithm from [Singh(2015)].

We need a way to convert ring elements into binary strings (and vice versa), that is robust to a certain amount of error. We do this with a rounding procedure.

The basic idea is this: we will divide $\mathbb{Z}/q\mathbb{Z}$ up into four quadrants. We label:

$$\begin{aligned}
I_0 &= \mathbb{Z}/q\mathbb{Z} \cap [0, q/4) \\
I_1' &= \mathbb{Z}/q\mathbb{Z} \cap [q/4, q/2) \\
I_0' &= \mathbb{Z}/q\mathbb{Z} \cap [q/2, 3q/4) \\
I_1 &= \mathbb{Z}/q\mathbb{Z} \cap [3q/4, q)
\end{aligned}$$

Every integer in $\mathbb{Z}/q\mathbb{Z}$ will be in exactly one of these intervals.

Remark 10.12. These sets do not all have the same number of elements: three of them will contain $(q-1)/4$ elements each, and the remaining set will contain $(q+3)/4$ elements. In order to maintain real-world security, this bias needs to be eliminated, so some of the edge-case numbers are moved over with 50% probability to avoid bias. We're going to ignore this subtlety for now.

Now given an $v \in \mathbb{Z}/q\mathbb{Z}$, we define the modular rounding function to be

$$\lfloor v \rceil_2 = \begin{cases} 0 & v \in I_0 \cup I_1 \quad 0 \leq v < q/4 \text{ or } 3q/4 \leq v < 0 \\ 1 & v \in I_0' \cup I_1' \qquad\qquad q/4 \leq v < 3q/4 \end{cases}$$

Modular rounding essentially tells us whether v is closer to 0 or to $q/2$, mod q. We will use this function to convert elements of $\mathbb{Z}/q\mathbb{Z}$ into individual bits, and thus to convert elements of $R = \mathbb{Z}/m\mathbb{Z}[x]/\langle x^n + 1\rangle$ into strings of n bits.

However, we expect to get our elements transmitted with some error; we need some correction (or "reconciliation") method to make sure that we are getting the same bit string that our partner is.

We define the cross rounding function to be

$$\langle v \rangle_2 = \begin{cases} 0 & v \in I_0 \cup I_0' \quad 0 \leq v < q/4 \text{ or } q/2 \leq v < 3q/4 \\ 1 & v \in I_1 \cup I_1' \quad q/4 \leq v < q/2 \text{ or } 3q/4 \leq v < q \end{cases}$$

This cross rounding function allows us to correctly compute $\lfloor a \rceil_2$ from a transmission of $w = v + e$ for small error e. In particular, we will assume the error is of size $\leq q/8$. We set $E = [-q/8, q/8) \cap \mathbb{Z}$, and we define

$$\operatorname{rec}(w, b) = \begin{cases} 0 & w \in I_b + E \mod q \\ 1 & \text{otherwise} \end{cases}$$

Proposition 10.13. *If $w = v + e \mod q$ for $a \in \mathbb{Z}/q\mathbb{Z}, e \in E$, then* $\operatorname{rec}(w, \langle v \rangle_2) = \lfloor v \rceil_2$.

Proof. Suppose $w = \langle v \rangle_2 = 0$. Then $v \in I_0 \cup I_0'$ by definition, and we compute $I_0 + E = [-q/8, 3q/8)$.

If $v \in I_0$, then we have $0 \leq v < q/4$ and $-q/8 \leq e < q/8$, so $-q/8 \leq v + e = w < 3q/8$ and $w \in I_b + E$, so $\operatorname{rec}(w, b) = 0 = \lfloor v \rceil_2$. Conversely, if $v \in I_0'$ then we have $q/2 \leq v < 3q/4$ so $3q/8 \leq v + e = w < 7q/8$ and $\operatorname{rec}(w, b) = 1 = \lfloor v \rceil_2$.

If $v \in I_1$ we can confirm $\operatorname{rec}(w, b) = \lfloor v \rceil_2$ by the same argument. □

Thus if you have access to $v + e$ and to $\langle v \rangle_2$, then you can compute $\lfloor v \rceil_2$, even if you don't know v or e.

10.4 Ring-LWE Diffie-Hellman

Algorithm 10.1. To generate a key:

A trusted party chooses parameters $n = 2^k$, q an odd prime with $q \equiv 1 \mod 2n$, $a \in R = \mathbb{Z}/q\mathbb{Z}[x]/\langle x^n + 1\rangle$, and a probability distribution over R that produces small elements.

1. Alice generates two random elements $s_0, s_1 \in R$. This is her private key.
2. Alice computes $b = s_1 \cdot a + s_0$. This is her public key.

When Bob receives Alices public key, he generates a shared secret via *encapsulation*.

1. Bob generates three random elements $e_0, e_1, e_2 \in R$.
2. Bob computes $u = e_0 \cdot a + e_1, v = e_0 \cdot b + e_2 \in R$.
3. Bob computes $\mu = \lfloor v \rceil_2 \in \{0,1\}^n$. Recall each element of $\mathbb{Z}/m\mathbb{Z}$ gives us one bit, so an element of R gives us n bits. This is the shared secret key.
4. Bob computes $\langle v \rangle_2 \in \{0,1\}^n$.
5. Bob transmits the ciphertext $c = (u, \langle v \rangle_2) \in R \times \{0,1\}^n$.

When Alice receives a ciphertext (u, v'), she recovers the shared secret via *decapsulation*.

1. Alice computes $w = u \cdot s_1$ using her private s_1.
2. Then Alice computes $\mu = \operatorname{rec}(w, v')$.

Proposition 10.14. *Alice and Bob recover the same secret bitstring in Algorithm 10.1. That is,* $\mu = \operatorname{rec}(u \cdot s_1, \langle v \rangle_2)$.

Proof. Alice computes $w = u \cdot s_1 = e_0 \cdot a \cdot s_1 + e_1 \cdot s_1$, and Bob has computed

$$v = e_0 \cdot b + e_2 = e_0(s_1 \cdot a + s_0) + e_2 = e_0 \cdot a \cdot s_1 + e_0 \cdot s_0 + e_2.$$

Thus we have $w = v + (e_0 \cdot s_0 - e_1 \cdot s_1 + e_2)$. Since all five of these numbers are small, proposition 10.13 tells us that $\operatorname{rec}(w, \langle v \rangle_2) = \lfloor v \rceil_2$.

□

What would Eve have to do in order to intercept the key? Eve gets to see $bs_1 \cdot a + s_0$ and $u = e_0 \cdot a + e_1$, and needs to deduce something close to $e_0 \cdot a \cdot s_1$. (Eve has also seen $\langle v \rangle_2$ but after normalizing the rounding properly this conveys exactly zero information about the shared secret).

Thus Eve would need to be able to find s_1 and e_0, which would roughly involve solving two distinct Ring-LWE problems.

Example 10.15. We choose parameters $n = 4, q = 17, a = 3x^3 + 7x^2 + 12x$.

Alice generates two random small elements $s_0 = x^3 + x + 1$ and $s_1 = x^2 + 3x + 2$. She computes

$$\begin{aligned} b = s_1 \cdot a + s_0 &= (x^2 + 3x + 2)(3x^3 + 7x^2 + 12x) + x^3 + x + 1 \\ &= 3x^5 + 7x^4 + 12x^3 + 9x^4 + 21x^3 + 36x^2 + 6x^3 + 14x^2 + 24x + x^3 + x + 1 \\ &= -3x - 7 + 12x^3 - 9 + 4x^3 + 2x^2 + 6x^3 + 14x^2 + 7x + x^3 + x + 1 \\ &= 6x^3 + 16x^2 + 5x + 2. \end{aligned}$$

She transmits this $b = 6x^3 + 16x^2 + 5x + 2$ to Bob.

When Bob receives this b, he generates $e_0 = x^3 - x - 1, e_1 = x^2 + 2x - 2, e_2 = -x^3 + 2x^2 + 1$. He computes

$$\begin{aligned} u = e_0 a + e_1 &= (x^3 - x - 1)(3x^3 + 7x^2 + 12x) + x^2 + 2x - 2 \\ &= 3x^6 + 7x^5 + 12x^4 - 3x^4 - 7x^3 - 12x^2 - 3x^3 - 7x^2 - 12x + x^2 + 2x - 2 \\ &= -3x^2 - 7x - 12 + 3 - 7x^3 - 12x^2 - 3x^3 - 7x^2 - 12x + x^2 + 2x - 2 \\ &= 7x^3 - 4x^2 + 0x + 6 \\ v = e_0 \cdot b + e_2 &= (x^3 - x - 1)(6x^3 + 16x^2 + 5x + 2) - x^3 + 2x^2 + 1 \\ &= 6x^6 + 16x^5 + 5x^4 + 2x^3 - 6x^4 - 16x^3 - 5x^2 - 2x - 6x^3 - 16x^2 - 5x - 2 - x^3 + 2x^2 + 1 \\ &= -6x^2 + x - 5 + 2x^3 + 6 + x^3 - 5x^2 - 2x - 6x^3 + x^2 - 5x - 2 - x^3 + 2x^2 + 1 \\ &= -4x^3 - 8x^2 - 6x. \end{aligned}$$

To compute $\mu = \lfloor v \rceil_2$ we observe that $I_0 = [0, 17/4) \cap \mathbb{Z}/q\mathbb{Z} = \{0, 1, 2, 3, 4\}$. Similarly, we have $I_1' = \{5, 6, 7, 8\}, I_0' = \{9, 10, 11, 12\}, I_1 = \{13, 14, 15, 16\}$. So $\mu = \lfloor -4x^3 - 8x^2 - 6x \rceil_2 = \lfloor 13x^3 + 9x^2 + 11x + 0 \rceil_2 = (0, 1, 1, 0)$.

We also compute $\langle v \rangle_2 = \langle 13x^3 + 9x^2 + 11x + 0 \rangle_2 = (1, 0, 0, 0)$.

Finally, Bob sends Alice the message $(u, \langle v \rangle_2 = (7x^3 - 4x^2 + 6, (1, 0, 0, 0))$. First Alice computes

$$\begin{aligned} w = u \cdot s_1 &= (7x^3 - 4x^2 + 6)(x^2 + 3x + 2) \\ &= 7x^5 + 21x^4 + 14x^3 - 4x^4 - 12x^3 - 8x^2 + 6x^2 + 18x + 12 \\ &= -7x - 4 - 3x^3 + 4 + 5x^3 + 9x^2 + 6x^2 + x - 5 \\ &= 2x^3 + 15x^2 + 11x + 12. \end{aligned}$$

Now she wants to compute $\operatorname{rec}(w, \langle v \rangle_2) = \operatorname{rec}(2x^3 + 15x^2 + 11x + 12, (1, 0, 0, 0))$. We see that $I_0 + E = [-q/8, 3q/8) = [-2, 6] = [15, 16] \cup [0, 6]$ and $I_1 + E = [5q/8, 9q/8) = [11, 19] =$

$[11, 16] \cup [0, 2]$. Then we have

$$\begin{aligned}
\operatorname{rec}(2, 1) &= 0 \\
\operatorname{rec}(15, 0) &= 0 \\
\operatorname{rec}(11, 0) &= 1 \\
\operatorname{rec}(12, 0) &= 1.
\end{aligned}$$

We find that this doesn't work! Alice doesn't wind up with the same secret Bob had had. Why is this?

10.5 Algorithm Analysis

Recall that this encryption algorithm is probabilistic—it works on the assumption that the error terms don't get too big relative to q, and in fact stay smaller than $q/8$. But since we took $q = 17$ we get $q/8 \approx 2$, and it's very easy for the error term to blow up larger than this.

In fact we can compute the error term $e_0s_0+e_2-e_1s_1 = 5-6x^2-6x^3$ has three coefficients that are larger than 2. In contrast, the coefficient of x is zero, so there's no error in that term. We can in fact see that the x coordinate of w is the same as the x coefficient of v, as you'd expect. And the bit corresponding to the x coefficient is in fact the only one that we got right.

You might think this probabilistic failure is a problem. And it can be; but under real-world parameter choices it really isn't. Suggested parameters are $n = 512$ and $q = 25601$, which leads to a 7680-bit public key; or $n = 1024, q = 40961$, and a 16384-bit public key. In the first case, the chances of a failure like this are $2^{-75.72}$, and in the latter the probability of this failure are $2^{-96.11}$.

So what are the advantages of this setup? There are a few.

First, even a quantum computer will find decrypting this as hard as solving the shortest-vector problem, which is NP-complete. Thus this is as secure as we could reasonably expect any sort of public-key cryptography to be.

Second, this algorithm is quite fast. The public keys and the transmitted messages are quite large, but there are very few calculations involved in the key exchange: encapsulation requires generating three random elements, doing two ring additions, two ring multiplications, and then two rounding steps. And decapsulation involves one ring multiplication and one rounding step. This is not very many operations.

On top of this, ring multiplication is very efficient. (I realize it might not feel efficient while you're doing it!) There is in fact a version of the Learning with Errors cryptosystem that doesn't involve ring operations, but it was discardes as impractical for being too slow.

In contrast, we can scale ring multiplication very efficiently using a technique called the Fast Fourier Transform. This is so efficient that for realistic key sizes, a quarter of the key exchange algorithm runtime is actually taken up just by generating the random numbers—the arithmetic is quite fast. (For $n = 1024$, [Singh(2015)] measured key generation at 112 microseconds, encapsulation at 183 microseconds, and decapsulation at 45 microseconds on a 2GHz i5.)

References

[Singh(2015)] Vikram Singh. A practical key exchange for the internet using lattice cryptography. Cryptology ePrint Archive, Report 2015/138, 2015. http://eprint.iacr.org/2015/138.