If the gcd is equal to 1, then reducing modulo $x^N - 1$ yields $\boldsymbol{a}(x) \star \boldsymbol{u}(x) = 1$ in R_q. Conversely, if $\boldsymbol{a}(x)$ is a unit in R_q, then we can find a polynomial $\boldsymbol{u}(x)$ such that $\boldsymbol{a}(x) \star \boldsymbol{u}(x) = 1$ in R_q. By definition of R_q, this means that

$$\boldsymbol{a}(x)\boldsymbol{u}(x) \equiv 1 \pmod{(x^N - 1)},$$

so by definition of congruences, there is a polynomial $\boldsymbol{v}(x)$ satisfying

$$\boldsymbol{a}(x)\boldsymbol{u}(x) - 1 = (x^N - 1)\boldsymbol{v}(x) \quad \text{in } (\mathbb{Z}/q\mathbb{Z})[x]. \qquad \square$$

Example 7.46. We let $N = 5$ and $q = 2$ and give the full details for computing $(1 + x + x^4)^{-1}$ in R_2. First we use the Euclidean algorithm to compute the greatest common divisor of $1 + x + x^4$ and $1 - x^5$ in $(\mathbb{Z}/2\mathbb{Z})[x]$. (Note that since we are working modulo 2, we have $1 - x^5 = 1 + x^5$.) Thus

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

So the gcd is equal to 1, and using the usual substitution method yields

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

Hence

$$(1 + x + x^4)^{-1} = 1 + x^2 + x^3 \quad \text{in } R_2.$$

(See Exercise 1.12 for an efficient computer algorithm and Fig. 1.3 for the "magic box method" to compute $\boldsymbol{a}(x)^{-1}$ in R_q.)

Remark 7.47. The ring R_q makes perfect sense regardless of whether q is prime, and indeed there are situations in which it can be advantageous to take q composite, for example $q = 2^k$. In general, if q is a power of a prime p, then in order to compute the inverse of $\boldsymbol{a}(x)$ in R_q, one first computes the inverse in R_p, then "lifts" this value to an inverse in R_{p^2}, and then lifts to an inverse in R_{p^4}, and so on. (See Exercise 7.27.) Similarly, if $q = q_1 q_2 \cdots q_r$, where each $q_i = p_i^{k_i}$ is a prime power, one first computes inverses in R_{q_i} and then combines the inverses using the Chinese remainder theorem.

7.10 The NTRU Public Key Cryptosystem

Cryptosystems based on the difficulty of integer factorization or the discrete logarithm problem are group-based cryptosystems, because the underlying hard problem involves only one operation. For RSA, Diffie–Hellman, and Elgamal, the group is the group of units modulo m for some modulus m that

may be prime or composite, and the group operation is multiplication modulo m. For ECC, the group is the set of points on an elliptic curve modulo p and the group operation is elliptic curve addition.

Rings are algebraic objects that have two operations, addition and multiplication, which are connected via the distributive law. In this section we describe NTRUEncrypt, the NTRU public key cryptosystem. NTRUEncrypt is most naturally described using convolution polynomial rings, but the underlying hard mathematical problem can also be interpreted as SVP or CVP in a lattice. We discuss the connection with lattices in Sect. 7.11.

7.10.1 NTRUEncrypt

In this section we describe NTRUEncrypt, the NTRU (pronounced *en-trū*) public key cryptosystem. We begin by fixing an integer $N \ge 1$ and two moduli p and q, and we let R, R_p, and R_q be the convolution polynomial rings

$$R = \frac{\mathbb{Z}[x]}{(x^N-1)}, \qquad R_p = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{(x^N-1)}, \qquad R_q = \frac{(\mathbb{Z}/q\mathbb{Z})[x]}{(x^N-1)},$$

described in Sect. 7.9. As usual, we may view a polynomial $\boldsymbol{a}(x) \in R$ as an element of R_p or R_q by reducing its coefficients modulo p or q. In the other direction, we use center-lifts to move elements from R_p or R_q to R. We make various assumptions on the parameters N, p and q, in particular we require that N be prime and that $\gcd(N,q) = \gcd(p,q) = 1$. (The reasons for these assumptions are explained in Exercises 7.32 and 7.37.)

We need one more piece of notation before describing NTRUEncrypt.

Definition. For any positive integers d_1 and d_2, we let

$$\mathcal{T}(d_1,d_2) = \left\{ \boldsymbol{a}(x) \in R : \begin{array}{l} \boldsymbol{a}(x) \text{ has } d_1 \text{ coefficients equal to } 1, \\ \boldsymbol{a}(x) \text{ has } d_2 \text{ coefficients equal to } -1, \\ \boldsymbol{a}(x) \text{ has all other coefficients equal to } 0 \end{array} \right\}.$$

Polynomials in $\mathcal{T}(d_1,d_2)$ are called *ternary* (or *trinary*) *polynomials*. They are analogous to *binary polynomials*, which have only 0's and 1's as coefficients.

We are now ready to describe NTRUEncrypt. Alice (or some trusted authority) chooses public parameters (N,p,q,d) satisfying the guidelines described earlier (or see Table 7.4). Alice's private key consists of two randomly chosen polynomials

$$\boldsymbol{f}(x) \in \mathcal{T}(d+1,d) \qquad \text{and} \qquad \boldsymbol{g}(x) \in \mathcal{T}(d,d). \tag{7.31}$$

Alice computes the inverses

$$\boldsymbol{F}_q(x) = \boldsymbol{f}(x)^{-1} \quad \text{in } R_q \qquad \text{and} \qquad \boldsymbol{F}_p(x) = \boldsymbol{f}(x)^{-1} \quad \text{in } R_p. \tag{7.32}$$

(If either inverse fails to exist, she discards this $\boldsymbol{f}(x)$ and chooses a new one. We mention that Alice chooses $\boldsymbol{f}(x)$ in $\mathcal{T}(d+1,d)$, rather than in $\mathcal{T}(d,d)$, because elements in $\mathcal{T}(d,d)$ never have inverses in R_q; see Exercise 7.24.)

Alice next computes

$$\boldsymbol{h}(x) = \boldsymbol{F}_q(x) \star \boldsymbol{g}(x) \quad \text{in } R_q. \tag{7.33}$$

The polynomial $\boldsymbol{h}(x)$ is Alice's public key. Her private key, which she'll need to decrypt messages, is the pair $\big(\boldsymbol{f}(x), \boldsymbol{F}_p(x)\big)$. Alternatively, Alice can just store $\boldsymbol{f}(x)$ and recompute $\boldsymbol{F}_p(x)$ when she needs it.

Bob's plaintext is a polynomial $\boldsymbol{m}(x) \in R$ whose coefficients satisfy $-\frac{1}{2}p < m_i \le \frac{1}{2}p$, i.e., the plaintext $\boldsymbol{m}$ is a polynomial in R that is the center-lift of a polynomial in R_p. Bob chooses a random polynomial (a random element) $\boldsymbol{r}(x) \in \mathcal{T}(d,d)$ and computes[7]

$$\boldsymbol{e}(x) \equiv p\boldsymbol{h}(x) \star \boldsymbol{r}(x) + \boldsymbol{m}(x) \pmod{q}. \tag{7.34}$$

Bob's ciphertext $\boldsymbol{e}(x)$ is in the ring R_q.

On receiving Bob's ciphertext, Alice starts the decryption process by computing

$$\boldsymbol{a}(x) \equiv \boldsymbol{f}(x) \star \boldsymbol{e}(x) \pmod{q}. \tag{7.35}$$

She then center lifts $\boldsymbol{a}(x)$ to an element of R and does a mod p computation,

$$\boldsymbol{b}(x) \equiv \boldsymbol{F}_p(x) \star \boldsymbol{a}(x) \pmod{p}. \tag{7.36}$$

Assuming that the parameters have been chosen properly, we now verify that the polynomial $\boldsymbol{b}(x)$ is equal to the plaintext $\boldsymbol{m}(x)$.

NTRUEncrypt, the NTRU public key cryptosystem, is summarized in Table 7.4.

Proposition 7.48. *If the NTRUEncrypt parameters (N,p,q,d) are chosen to satisfy*

$$q > (6d+1)\,p, \tag{7.37}$$

then the polynomial $\boldsymbol{b}(x)$ computed by Alice in (7.36) is equal to Bob's plaintext $\boldsymbol{m}(x)$.

Proof. We first determine more precisely the shape of Alice's preliminary calculation of $\boldsymbol{a}(x)$. Thus

$$\begin{aligned}
\boldsymbol{a}(x) &\equiv \boldsymbol{f}(x) \star \boldsymbol{e}(x) \pmod{q} && \text{from (7.35)},\\
&\equiv \boldsymbol{f}(x) \star \big(p\boldsymbol{h}(x) \star \boldsymbol{r}(x) + \boldsymbol{m}(x)\big) \pmod{q} && \text{from (7.34)},\\
&\equiv p\boldsymbol{f}(x) \star \boldsymbol{F}_q(x) \star \boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{q} && \text{from (7.33)}.\\
&\equiv p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{q} && \text{from (7.32)}.
\end{aligned}$$

[7] Note that when we write a congruence of polynomials modulo q, we really mean that the computation is being done in R_q.

Public parameter creation	
A trusted party chooses public parameters (N, p, q, d) with N and p prime, $\gcd(p, q) = \gcd(N, q) = 1$, and $q > (6d + 1)p$.	
Alice	**Bob**
Key creation	
Choose private $\boldsymbol{f} \in \mathcal{T}(d + 1, d)$ that is invertible in R_q and R_p. Choose private $\boldsymbol{g} \in \mathcal{T}(d, d)$. Compute $\boldsymbol{F}_q$, the inverse of $\boldsymbol{f}$ in R_q. Compute $\boldsymbol{F}_p$, the inverse of $\boldsymbol{f}$ in R_p. Publish the public key $\boldsymbol{h} = \boldsymbol{F}_q \star \boldsymbol{g}$.	
Encryption	
	Choose plaintext $\boldsymbol{m} \in R_p$. Choose a random $\boldsymbol{r} \in \mathcal{T}(d, d)$. Use Alice's public key $\boldsymbol{h}$ to compute $\boldsymbol{e} \equiv p\boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod q$. Send ciphertext $\boldsymbol{e}$ to Alice.
Decryption	
Compute $\boldsymbol{f} \star \boldsymbol{e} \equiv p\boldsymbol{g} \star \boldsymbol{r} + \boldsymbol{f} \star \boldsymbol{m} \pmod q$. Center-lift to $\boldsymbol{a} \in R$ and compute $\boldsymbol{m} \equiv \boldsymbol{F}_p \star \boldsymbol{a} \pmod p$.	

Table 7.4: NTRUEncryt: the NTRU public key cryptosystem

Consider the polynomial

$$p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x), \tag{7.38}$$

computed exactly in R, rather than modulo q. We need to bound its largest possible coefficient. The polynomials $\boldsymbol{g}(x)$ and $\boldsymbol{r}(x)$ are in $\mathcal{T}(d, d)$, so if, in the convolution product $\boldsymbol{g}(x) \star \boldsymbol{r}(x)$, all of their 1's match up and all of their -1's match up, the largest possible coefficient of $\boldsymbol{g}(x) \star \boldsymbol{r}(x)$ is $2d$. Similarly, $\boldsymbol{f}(x) \in \mathcal{T}(d+1, d)$ and the coefficients of $\boldsymbol{m}(x)$ are between $-\frac{1}{2}p$ and $\frac{1}{2}p$, so the largest possible coefficient of $\boldsymbol{f}(x) \star \boldsymbol{m}(x)$ is $(2d+1) \cdot \frac{1}{2}p$. So even if the largest coefficient of $\boldsymbol{g}(x) \star \boldsymbol{r}(x)$ happens to coincide with the largest coefficient of $\boldsymbol{r}(x) \star \boldsymbol{m}(x)$, the largest coefficient of (7.38) has magnitude at most

$$p \cdot 2d + (2d + 1) \cdot \frac{1}{2}p = \left(3d + \frac{1}{2}\right) p.$$

Thus our assumption (7.37) ensures that every coefficient of (7.38) has magnitude strictly smaller than $\frac{1}{2}q$. Hence when Alice computes $\boldsymbol{a}(x)$ modulo q

(i.e., in R_q) and then lifts it to R, she recovers the exact value (7.38). In other words,

$$\boldsymbol{a}(x) = p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x) \tag{7.39}$$

exactly in R, and not merely modulo q.

The rest is easy. Alice multiplies $\boldsymbol{a}(x)$ by $\boldsymbol{F}_p(x)$, the inverse of $\boldsymbol{f}(x)$ modulo p, and reduces the result modulo p to obtain

$$\begin{aligned}
\boldsymbol{b}(x) &\equiv \boldsymbol{F}_p(x) \star \boldsymbol{a}(x) \pmod{p} && \text{from (7.36),}\\
&\equiv \boldsymbol{F}_p(x) \star \big(p\boldsymbol{g}(x) \star \boldsymbol{r}(x) + \boldsymbol{f}(x) \star \boldsymbol{m}(x)\big) \pmod{p} && \text{from (7.39),}\\
&\equiv \boldsymbol{F}_p(x) \star \boldsymbol{f}(x) \star \boldsymbol{m}(x) \pmod{p} && \text{reducing mod } p,\\
&\equiv \boldsymbol{m}(x) \pmod{p}. && \text{from (7.32).}
\end{aligned}$$

Hence $\boldsymbol{b}(x)$ and $\boldsymbol{m}(x)$ are the same modulo p. □

Remark 7.49. The condition $q > (6d+1)p$ in Proposition 7.48 ensures that decryption never fails. However, an examination of the proof shows that decryption is likely to succeed even for considerably smaller values of q, since it is highly unlikely that the positive and negative coefficients of $\boldsymbol{g}(x)$ and $\boldsymbol{r}(x)$ will exactly line up, and similarly for $\boldsymbol{f}(x)$ and $\boldsymbol{m}(x)$. So for additional efficiency and to reduce the size of the public key, it may be advantageous to choose a smaller value of q. It then becomes a delicate problem to estimate the probability of decryption failure. It is important that the probability of decryption failure be very small (e.g., smaller than 2^{-80}), since decryption failures have the potential to reveal private key information to an attacker.

Remark 7.50. Notice that NTRUEncrypt is an example of a probabilistic cryptosystem (Sect. 3.10), since a single plaintext $\boldsymbol{m}(x)$ has many different encryptions $p\boldsymbol{h}(x)\star\boldsymbol{r}(x)+\boldsymbol{m}(x)$ corresponding to different choices of the random element $\boldsymbol{r}(x)$. As is common for such systems, cf. Remark 7.37 for GGH, it is a bad idea for Bob to send the same message twice using different random elements, just as it is inadvisable for Bob to use the same random element to send two different plaintexts; see Exercise 7.34. Various ways of ameliorating this danger for GGH, which also apply *mutatis mutandis* to NTRUEncrypt, are described in Exercises 7.20 and 7.21.

Remark 7.51. The polynomial $\boldsymbol{f}(x) \in \mathcal{T}(d+1,d)$ has small coefficients, but the coefficients of its inverse $\boldsymbol{F}_q(x) \in R_q$ tend to be randomly and uniformly distributed modulo q. (This is not a theorem, but it is an experimentally observed fact.) For example, let $N = 11$ and $q = 73$ and take a random polynomial

$$\boldsymbol{f}(x) = x^{10} + x^8 - x^3 + x^2 - 1 \in \mathcal{T}(3,2).$$

Then $\boldsymbol{f}(x)$ is invertible in R_q, and its inverse

$$\boldsymbol{F}_q(x) = 22x^{10}+33x^9+15x^8+33x^7-10x^6+36x^5-33x^4-30x^3+12x^2-32x+28$$

has random-looking coefficients. Similarly, in practice the coefficients of the public key and the ciphertext,

$$\boldsymbol{h}(x) \equiv \boldsymbol{F}_q(x) \star \boldsymbol{g}(x) \pmod{q} \quad \text{and} \quad \boldsymbol{e}(x) \equiv p\boldsymbol{r}(x) \star \boldsymbol{h}(x) + \boldsymbol{m}(x) \pmod{q},$$

also appear to be randomly distributed modulo q.

Remark 7.52. As noted in Sect. 7.7, a motivation for using lattice-based cryptosystems is their high speed compared to discrete logarithm and factorization-based cryptosystems. How fast is NTRUEncrypt? The most time consuming part of encryption and decryption is the convolution product. In general, a convolution product $\boldsymbol{a}\star\boldsymbol{b}$ requires N^2 multiplications, since each coefficient is essentially the dot product of two vectors. However, the convolution products required by NTRUEncrypt have the form $\boldsymbol{r}\star\boldsymbol{h}$, $\boldsymbol{f}\star\boldsymbol{e}$, and $\boldsymbol{F}_p\star\boldsymbol{a}$, where $\boldsymbol{r}$, $\boldsymbol{f}$, and $\boldsymbol{F}_p$ are ternary polynomials. Thus these convolution products can be computed without any multiplications; they each require approximately $\frac{2}{3}N^2$ additions and subtractions. (If d is smaller than $N/3$, the first two require only $\frac{2}{3}dN$ additions and subtractions.) Thus NTRUEncrypt encryption and decryption take $\mathcal{O}(N^2)$ steps, where each step is extremely fast.

Example 7.53. We present a small numerical example of NTRUEncrypt with public parameters

$$(N, p, q, d) = (7, 3, 41, 2).$$

We have

$$41 = q > (6d + 1)p = 39,$$

so Proposition 7.48 ensures that decryption will work. Alice chooses

$$\boldsymbol{f}(x) = x^6 - x^4 + x^3 + x^2 - 1 \in \mathcal{T}(3, 2) \quad \text{and} \quad \boldsymbol{g}(x) = x^6 + x^4 - x^2 - x \in \mathcal{T}(2, 2).$$

She computes the inverses

$$\boldsymbol{F}_q(x) = \boldsymbol{f}(x)^{-1} \bmod q = 8x^6 + 26x^5 + 31x^4 + 21x^3 + 40x^2 + 2x + 37 \in R_q,$$
$$\boldsymbol{F}_p(x) = \boldsymbol{f}(x)^{-1} \bmod p = x^6 + 2x^5 + x^3 + x^2 + x + 1 \in R_p.$$

She stores $\big(\boldsymbol{f}(x), \boldsymbol{F}_p(x)\big)$ as her private key and computes and publishes her public key

$$\boldsymbol{h}(x) = \boldsymbol{F}_q(x) \star \boldsymbol{g}(x) = 20x^6 + 40x^5 + 2x^4 + 38x^3 + 8x^2 + 26x + 30 \in R_q.$$

Bob decides to send Alice the message

$$\boldsymbol{m}(x) = -x^5 + x^3 + x^2 - x + 1$$

using the random element

$$\boldsymbol{r}(x) = x^6 - x^5 + x - 1.$$

Bob computes and sends to Alice the ciphertext

$$\boldsymbol{e}(x) \equiv p\boldsymbol{r}(x)\star\boldsymbol{h}(x)+\boldsymbol{m}(x) \equiv 31x^6+19x^5+4x^4+2x^3+40x^2+3x+25 \pmod q.$$

Alice's decryption of Bob's message proceeds smoothly. First she computes

$$\boldsymbol{f}(x)\star\boldsymbol{e}(x) \equiv x^6+10x^5+33x^4+40x^3+40x^2+x+40 \pmod q. \quad (7.40)$$

She then center-lifts (7.40) modulo q to obtain

$$\boldsymbol{a}(x) = x^6+10x^5-8x^4-x^3-x^2+x-1 \in R.$$

Finally, she reduces $\boldsymbol{a}(x)$ modulo p and computes

$$\boldsymbol{F}_p(x)\star\boldsymbol{a}(x) \equiv 2x^5+x^3+x^2+2x+1 \pmod p. \quad (7.41)$$

Center-lifting (7.41) modulo p retrieves Bob's plaintext $\boldsymbol{m}(x) = -x^5+x^3+x^2-x+1$.

7.10.2 Mathematical Problems for NTRUEncrypt

As noted in Remark 7.51, the coefficients of the public key $\boldsymbol{h}(x)$ appear to be random integers modulo q, but there is a hidden relationship

$$\boldsymbol{f}(x)\star\boldsymbol{h}(x) \equiv \boldsymbol{g}(x) \pmod q, \quad (7.42)$$

where $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ have very small coefficients. Thus breaking NTRUEncrypt by finding the private key comes down to solving the following problem:

The NTRU Key Recovery Problem
Given $\boldsymbol{h}(x)$, find ternary polynomials $\boldsymbol{f}(x)$ and $\boldsymbol{g}(x)$ satisfying $\boldsymbol{f}(x)\star\boldsymbol{h}(x) \equiv \boldsymbol{g}(x) \pmod q$.

Remark 7.54. The solution to the NTRU key recovery problem is not unique, because if $\big(\boldsymbol{f}(x),\boldsymbol{g}(x)\big)$ is one solution, then $\big(x^k\star\boldsymbol{f}(x),x^k\star\boldsymbol{g}(x)\big)$ is also a solution for every $0 \le k < N$. The polynomial $x^k\star\boldsymbol{f}(x)$ is called a *rotation of* $\boldsymbol{f}(x)$ because the coefficients have been cyclically rotated k positions. Rotations act as private decryption keys in the sense that decryption with $x^k\star\boldsymbol{f}(x)$ yields the rotated plaintext $x^k\star\boldsymbol{m}(x)$.

More generally, any pair of polynomials $\big(\boldsymbol{f}(x),\boldsymbol{g}(x)\big)$ with sufficiently small coefficients and satisfying (7.42) serves as an NTRU decryption key. For example, if $\boldsymbol{f}(x)$ is the original decryption key and if $\boldsymbol{\theta}(x)$ has tiny coefficients, then $\boldsymbol{\theta}(x)\star\boldsymbol{f}(x)$ may also work as a decryption key.

Remark 7.55. Why would one expect the NTRU key recovery problem to be a hard mathematical problem? A first necessary requirement is that the problem not be practically solvable by a brute-force or collision search. We discuss such searches later in this section. More importantly, in Sect. 7.11.2