Alkaline: A Simplified Post-Quantum Encryption Algorithm for Classroom Use

Joshua Holden
Department of Mathematics
Rose-Hulman Institute of Technology
Terre Haute, IN 47803, USA
holden@rose-hulman.edu
+1 812-877-8320

Alkaline: A Simplified Post-Quantum Encryption Algorithm for Classroom Use

Abstract: This paper describes Alkaline, a size-reduced version of Kyber, which has recently been announced as a prototype NIST standard for post-quantum public-key cryptography. While not as simple as RSA, I believe that Alkaline can be used in an undergraduate classroom to effectively teach the techniques and principles behind Kyber and post-quantum cryptography in general. Classroom experiences with individual concepts used in Alkaline support this belief. In addition to cryptography, linear algebra and abstract algebra classes would be good candidates for the use of Alkaline. A few exercises suitable for use in these classes are included.

Keywords: post-quantum, Kyber, classroom, cryptography, public-key

1 INTRODUCTION

On July 5, 2022, after much delay, the National Institute of Standards and Technology (NIST) announced the first set of algorithms to be standardized for Post-Quantum Cryptography. Post-Quantum Cryptography, despite the name, is intended to be used on current ("classical") computers; it is designed to be resistant to breaking by the quantum computers expected to be developed in the next few decades. This is possible because while quantum computers are predicted to be able to solve many problems much faster than classical computers, this speed-up does not apply equally to all problems. For example, symmetric-key cryptographic systems such as the Advanced Encryption Standard (AES) are expected to remain reasonably secure, while all public-key systems currently in widespread use, including RSA, Diffie-Hellman, and Elliptic Curve Cryptography, are expected to be completely broken.

While the evaluation process is still underway, NIST has so far selected one encryption method and three digital signature algorithms to be standardized. The encryption method and one of the signature algorithms were developed by a project known as the CRYptographic SuiTe for Algebraic LatticeS, or CRYSTALS, and both are named after crystals from popular fictional universes. The encryption system is known as Kyber, after the power source for lightsabers in *Star Wars*, while the signature algorithm is named after the Dilithium crystals used to power spaceships in *Star Trek*.[1]

Given the classroom success of simplified versions of other modern cryptosystems, including S-DES [14], S-AES [11], Demitasse [8], and JHA [9], I thought that it would be useful to develop similar versions of Kyber and Dilithium for classroom use. I call this project DIgital Resources About CiphEricaL Learning, or DIRACELL. This paper introduces a simplified version of Kyber which I call Alkaline. In a future paper, I hope to introduce a simplified signature algorithm, tentatively named Lithium.

2 LEARNING WITH ERRORS

Like most public-key cryptography, Kyber is based on a hard mathematical problem. In this case, the problem is from a family of related problems known as Learning With Errors. Despite the name, Learning With Errors doesn't really have anything much to do with Machine Learning or Artificial Intelligence. In fact, the original Learning With Errors

[1]Note that competing Post-Quantum proposals included NewHope and SABER, which has a version called LightSABER.

problem, known as LWE, is a fairly straightforward-sounding problem in Linear Algebra, or maybe Statistics: given a vector $\mathbf{t}$ of the form

$$\mathbf{t} \equiv A\mathbf{s} + \mathbf{e} \pmod{q}, \tag{1}$$

where A is a public matrix with at least as many rows as columns and $\mathbf{e}$ is a "small error vector" drawn from some probability distribution, find the secret vector $\mathbf{s}$.

Notice that if there was no error vector, the equation would be solvable simply by row reduction of A. Likewise, if the error vectors are *too* small, or not random enough, then the problem is easy to solve. On the other hand, if the error vectors are too large, then there will be more than one solution $\mathbf{s}$, which will be a problem.

A public-key cryptosystem based on LWE was proposed by Regev in 2009 [12], but it suffers from requiring very large keys and producing large ciphertexts. We present the "dual" version from [6, Sec 7.1], which has very similar properties but is closer to the version of LWE used in Kyber. To encrypt a message of ℓ bits, the private key is a matrix S of ℓ columns and the public key is (A, T), where

$$T \equiv AS + E \pmod{q} \tag{2}$$

for an error matrix E of the appropriate size. (This matrix equation is equivalent to a system of vector equations of the form (1).) If $\mathbf{p}$ is a plaintext message represented as a vector of bits, the ciphertext is obtained by choosing a random ℓ-bit nonce vector $\mathbf{r}$ and small random errors $\mathbf{e}_1$ and $\mathbf{e_2}$, and computing $(\mathbf{u}, \mathbf{v})$, where

$$\mathbf{u} = \left(A^T\mathbf{r} + \mathbf{e}_1\right) \operatorname{MOD} q, \qquad \mathbf{v} = \left(T^T\mathbf{r} + \mathbf{e}_2 + \lfloor q/2 \rceil\, \mathbf{p}\right) \operatorname{MOD} q.$$

($\lfloor x \rceil$ denotes the nearest integer to x, halves rounded up. MOD indicates the action of reducing to the least nonnegative residue, as opposed to the congruence relation.)

Decrypting the ciphertext works by testing each coordinate of $\mathbf{v} - S^T\mathbf{u}$ to see if it is closer to 0 or to $q/2$ modulo q:

$$\mathbf{p}' = \left\lfloor \lfloor q/2 \rceil^{-1} \left(\mathbf{v} - S^T\mathbf{u}\right) \right\rceil \operatorname{MOD} 2.$$

Since

$$\begin{aligned} \mathbf{v} - S^T\mathbf{u} &\equiv T^T\mathbf{r} + \lfloor q/2 \rceil\, \mathbf{p} - S^T A^T \mathbf{r} \pmod{q} \\ &\equiv (AS)^T\mathbf{r} + E^T\mathbf{r} + \lfloor q/2 \rceil\, \mathbf{p} - S^T A^T \mathbf{r} \pmod{q} \\ &\equiv E^T\mathbf{r} + \lfloor q/2 \rceil\, \mathbf{p} \pmod{q} \end{aligned}$$

we can see that as long as E is small enough that the coordinates of $E^T\mathbf{r}$ have magnitude less than $q/4$, $\mathbf{p}'$ will be equal to $\mathbf{p}$.

3 RING LEARNING WITH ERRORS

Cryptographers with an algebraic inclination have known for a long time that a matrix multiplication like $\mathbf{t} \equiv A\mathbf{s} + \mathbf{e}$ can be compactly represented by a polynomial equation, provided that the vectors are of a particular form. (See, for example, Daemen and Rijmen [3, Sec. 2.2], where they appear to take the fact for granted.)

Let's start by fixing a positive integer n and considering the set R of polynomials in x of degree less than n, where polynomials are multiplied modulo $x^n + 1$. In other words, whenever two polynomials are multiplied, perform regular polynomial multiplication, then divide by $x^n + 1$ and take the remainder to get a third polynomial in R. Students who are familiar with AES or NTRU will probably have encountered similar polynomial modular arithmetic.

Now we can introduce our second problem: given a polynomial $t(x)$ of the form

$$t(x) \equiv a(x)s(x) + e(x) \pmod{x^n + 1} \pmod{q}, \tag{3}$$

where $a(x)$ is a public polynomial in R and $e(x)$ is an "error polynomial" with small coefficients, find the secret polynomial $s(x)$ in R. A structure like R where you can add, subtract, and multiply, but not necessarily divide without a remainder, is called a ring, and this new problem is called the Ring Learning With Errors (RLWE) problem.

Example 1. Let $n = 4$ and $q = 23$. Suppose

$$a(x) = 18x^3 + 10x^2 + 22x + 6, \qquad s(x) = x^3 - x^2 - x - 1, \qquad e(x) = x^2 + x.$$

$$a(x)s(x) = 18x^6 - 8x^5 - 6x^4 - 44x^3 - 38x^2 - 28x - 6$$

$$\begin{array}{rl}
 & 18x^2 - 8x - 6 \\
\cline{2-2}
x^4+1 \big) & 18x^6 - 8x^5 - 6x^4 - 44x^3 - 38x^2 - 28x - 6 \\
 & -18x^6 \qquad\qquad\quad - 18x^2 \\
\cline{2-2}
 & \quad - 8x^5 - 6x^4 - 44x^3 - 56x^2 - 28x \\
 & \quad\; 8x^5 \qquad\qquad\qquad\quad + 8x \\
\cline{2-2}
 & \quad\quad - 6x^4 - 44x^3 - 56x^2 - 20x - 6 \\
 & \quad\quad\; 6x^4 \qquad\qquad\qquad\quad + 6 \\
\cline{2-2}
 & \quad\quad\quad - 44x^3 - 56x^2 - 20x
\end{array}$$

So $t(x) \equiv (-44x^3 - 56x^2 - 20x) + (x^2 + x) \pmod{23} \equiv 2x^3 + 14x^2 + 4x \pmod{23}$. (Of course, we could have reduced modulo q at any step of the long division as well as at the end.)

Now suppose we have two polynomials $a(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0$ and $s(x) = s_{n-1}x^{n-1} + \ldots + s_1x + s_0$ in R. Notice that when $a(x)s(x)$ is reduced modulo $x^n + 1$, any powers x^m with $m \le n$ will be reduced to $-x^{m-n}$. Let A and $\mathbf{s}$ be the matrix and vector

$$A = \begin{pmatrix} a_0 & -a_{n-1} & \cdots & -a_2 & -a_1 \\ a_1 & a_0 & \cdots & -a_3 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \cdots & a_0 & -a_{n-1} \\ a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \end{pmatrix}, \qquad \mathbf{s} = \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-2} \\ s_{n-1} \end{pmatrix}. \tag{4}$$

Then

$$A\mathbf{s} = \begin{pmatrix} a_0s_0 - a_{n-1}s_1 - \cdots - a_2s_{n-2} - a_1s_{n-1} \\ a_0s_1 + a_0s_1 - \cdots - a_3s_{n-2} - a_2s_{n-1} \\ \vdots \\ a_{n-2}s_0 + a_{n-3}s_1 + \cdots + a_0s_{n-2} - a_{n-1}s_{n-1} \\ a_{n-1}s_0 + a_{n-2}s_1 + \cdots + a_1s_{n-2} + a_0s_{n-1} \end{pmatrix}.$$

But the entries in this vector are exactly the coefficients of $a(x)s(x)$ reduced modulo $x^n + 1$. The matrix A is called a "negacyclic matrix", and the operation on polynomials is sometimes called "negacyclic convolution". So a modular multiplication of two polynomials with n coefficients each can be thought of as representing a product of a matrix with n^2 entries (of special form) by a vector of n entries. If that matrix is part of a cryptographic key, representing it by a much smaller polynomial is clearly helpful. The tradeoff is extra structure in the matrix which might (or might not) reduce the security.

Example 2. With $a(x), s(x), n$, and q as above, we have

$$A = \begin{pmatrix} 6 & -18 & -10 & -22 \\ 22 & 6 & -18 & -10 \\ 10 & 22 & 6 & -18 \\ 18 & 10 & 22 & 6 \end{pmatrix}, \qquad S = \begin{pmatrix} -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 \end{pmatrix}, \qquad AS = \begin{pmatrix} 0 & 44 & 56 & 20 \\ -20 & 0 & 44 & 56 \\ -56 & -20 & 0 & 44 \\ -44 & -56 & -20 & 0 \end{pmatrix}$$

Once again we can define a cryptosystem based on this problem [10]. To encrypt a message of n bits, the private key is a polynomial $s(x)$ in R and the public key is $(a(x), t(x))$, where

$$t(x) \equiv a(x)s(x) + e(x) \pmod{x^n + 1} \pmod{q} \tag{5}$$

for an error polynomial $e(x)$. If $p(x)$ is a plaintext message represented as a polynomial in R with 0's and 1's as coefficients, the ciphertext is obtained by choosing a random polynomial $r(x)$ in R and small random error polynomials $e_1(x)$ and $e_2(x)$, and computing $(u(x), v(x))$, where

$$u(x) = a(x)r(x)+e_1(x) \operatorname{MOD} x^n+1 \operatorname{MOD} q, \qquad v(x) = t(x)r(x)+e_2(x)+\lfloor q/2 \rceil\, p(x) \operatorname{MOD} x^n+1 \operatorname{MOD} q.$$

To decrypting the ciphertext, let

$$p'(x) = \Big\lfloor \lfloor q/2 \rceil^{-1} (v(x) - s(x)u(x)) \Big\rceil \operatorname{MOD} 2,$$

where the rounding is done component-wise. Once again, $p'(x)$ will be equal to $p(x)$ as long as the coefficients of $e(x)r(x)$ have magnitude less than $q/4$.

4 KYBER

As I suggested earlier, there is some evidence that the extra structure in the RLWE problem might make it susceptible to an attack, although no serious attack has been developed so far. For this reason, the developers of Kyber decided to combine the LWE and RLWE problems by using a vector of a small number k of polynomials instead of a single polynomial. Technically, a vector over a ring no longer belongs to a vector space, but rather a module [5, Chapter 10], so we now have the Module Learning With Errors (MLWE) problem: given a vector $\mathbf{t}(x)$ of k polynomials in R, with the form

$$\mathbf{t}(x) \equiv A(x)\mathbf{s}(x) + \mathbf{e}(x) \pmod{x^n + 1} \pmod{q}, \tag{6}$$

where $A(x)$ is a $k \times k$ public matrix of polynomials in R with at least as many rows as columns, and $\mathbf{e}(x)$ is a "small error vector" of k polynomials in R drawn from some probability distribution, find the secret vector of k polynomials $\mathbf{s}(x)$.

We could now define a cryptosystem exactly parallel to those above. However, the developers discovered some tricks that lead to even greater efficiency. These fall into three main categories:

1. using hash function and extendable output functions to generate keys from smaller random numbers,
2. using the "number-theoretic transform" (NTT) to speed up polynomial modular multiplication,
3. using compression functions to discard some bits in the ciphertext which are unlikely to affect the decryption.

This paper will not say too much about point (1), which is not generally taught much in relation to public-key cryptography. (But see Stinson and Paterson [15, Secs. 5.4.1 and 12.3].) The number-theoretic transform in point (2) has a lot of mathematical interest, but is rather advanced for many cryptography students. Nevertheless, several versions of Alkaline have been chosen so that the NTT could be used. Point (3) is not crucial, but it is also not too difficult to understand. I hope to introduce a version of Alkaline incorporating compression (and perhaps the NTT) in a future paper.

One more thing that I haven't discussed is the form of the error distribution. It turns that the exact form of the distribution is not critical, so the Kyber designers chose to use the "centered binomial distribution", which can be easily and efficiently generated as follows [1, Sec. 1.5]: for a small parameter η, generate 2η uniformly random bits $a_1, \ldots, a_\eta$ and $b_1, \ldots, b_\eta$ and calculate $\sum_{i=1}^{\eta}(a_i - b_i)$. Student of probability might recognize that this is the same distribution obtained by flipping 2η fair coins, counting the number of heads, and subtracting η. The proof that these are the same will not be hard for students

	n	k	q	η	δ
Alkaline N	4	2	17	1	$2^{-2.9} \approx 0.13$
Alkaline AA	4	2	23	1	$2^{-4.7} \approx 0.04$
Alkaline C	4	2	29	$2/1^*$	$2^{-2.8} \approx 0.14$
Alkaline D	4	2	41	2	$2^{-4.3} \approx 0.05$
Kyber512	256	2	3329	$3/2^{**}$	2^{-139}
Kyber768	256	3	3329	2	2^{-164}
Kyber1024	256	4	3329	2	2^{-174}

*Alkaline C uses $\eta = 2$ for $\mathbf{s}$, $\mathbf{e}$, and $\mathbf{r}$, and $\eta = 1$ for $\mathbf{e}_1$ and e_2
**Kyber512 uses $\eta = 3$ for $\mathbf{s}$, $\mathbf{e}$, and $\mathbf{r}$, and $\eta = 2$ for $\mathbf{e}_1$ and e_2

Table 1. Parameter sets for Alkaline and Kyber

with a strong probability background; other students could be encouraged to calculate the probabilities for a small value of η and show that the distributions are equal. All vectors and polynomials described as "errors" will have their coordinates and/or coefficients chosen from this distribution, as will the secret key and the random nonce.

5 ALKALINE

Putting all of these pieces together gives us Figure 1, showing the entire process of Aretha sending a message to Bernie. In addition to R, we will also use R_q to denote the set of polynomials of degree less than n with integer coefficients in the range $[0, q)$. For brevity, I will use the notation $a(x) \star t(x)$ for $a(x)t(x) \operatorname{MOD} x^n + 1 \operatorname{MOD} q$. All additions will be modulo q, and matrix and vector operations will also use these modular reductions.

Some options for parameter choices are shown in Table 1, with comparison to the options for some of the versions of Kyber. We also list the failure probability δ as given for Kyber in [1] and as reported for Alkaline by the Python script available on GitHub [4].[2]

Parameters for Alkaline were chosen to balance ease of student use, including working without a computer, with illustration of the major features of Kyber. Security was not considered particularly important, and the target for decryption failures was in the 5–10% range — large enough so that the possibility can be easily demonstrated, but not so large that students get frustrated. Once difference from Kyber is that in Alkaline q is allowed to change, while k is held constant. This makes the system slightly harder to program, but easier to use by hand.

Inevitably, no single parameter set will serve all purposes.

- Alkaline D has a good failure rate and a realistic value of η, but q is really too large to work with by hand.
- Alkaline C has a slightly more manageable q than Alkaline D, and a realistic η, but a slightly higher than ideal failure rate. Also, the two different values for η is a complication that may not be worth the trade-off. (Using a single $\eta = 2$ produced an estimated failure rate of larger than 1.)
- Alkaline AA is probably workable by hand and has a failure rate near the target zone. The value of η is less realistic, and since $q - 1$ is not divisible by n, it cannot be used to demonstrate the number-theoretic transform.
- Alkaline N is the most workable by hand, but the failure rate is somewhat high and η is less realistic. (Smaller values of q have unacceptably high failure rates.)

Example 3. In order to create a key for Alkaline AA, Bernie first generates the random

[2]This script was clearly not designed for such small parameters and the result should be taken with a grain of salt, although limited testing suggests the failure is not significantly greater than reported. A version of the script modified to run on small parameters without crashing is available on GitHub [?redacted].

Bernie

- picks $k \times k = 2 \times 2$ matrix $A(x)$ with uniform random entries in R_q
- generates $2nk\eta = 16\eta$ random bits
- uses random bits to compute centered binomial coefficients for vectors $\mathbf{s}(x)$ and $\mathbf{e}(x)$ of $k = 2$ polynomials in R each
- computes $\mathbf{t}(x) = A(x) \star \mathbf{s}(x) + \mathbf{e}(x)$

- posts public encryption key $(A(x), \mathbf{t}(x)))$
- keeps private key $\mathbf{s}(x)$ secret

Aretha

- encodes plaintext as a polynomial $p(x)$ in R

- looks up Bernie's encryption key $(A(x), \mathbf{t}(x))$

- generates $2nk\eta + 2n\eta = 20\eta$ random bits
- uses random bits to compute centered binomial coefficients for vectors $\mathbf{r}(x)$ and $\mathbf{e}_1(x)$ of $k = 2$ polynomials in R each and one polynomial $e_2(x)$ in R

$$p(x)$$

$$\downarrow (A(x), \mathbf{t}(x))$$

$$\mathbf{u}(x) = A(x)^T \star \mathbf{r}(x) + \mathbf{e}_1(x)$$

$$v(x) = \mathbf{t}(x)^T \star \mathbf{r}(x) + e_2(x) + \lfloor q/2 \rceil \, p(x)$$

Aretha $\rightarrow (\mathbf{u}(x), v(x)) \rightarrow$ Bernie 

Bernie

$$(\mathbf{u}(x), v(x))$$

$$\downarrow \mathbf{s}(x)$$

$$p'(x) = \left\lfloor \lfloor q/2 \rceil^{-1} \left(v(x) - \mathbf{s}(x)^T \star \mathbf{u}(x) \right) \right\rceil \text{ MOD } 2$$

Figure 1. The Alkaline system

public matrix

$$A(x) = \begin{pmatrix} 4x^3 + 4x^2 + 10x & 11x^3 + 15x^2 + 10x + 3 \\ 12x^3 + 22x^2 + 4x + 12 & 6x^3 + x + 11 \end{pmatrix}$$

and 16 random bits, which he uses to compute secret vectors

$$\mathbf{s}(x) = \begin{pmatrix} -x^3 + x^2 + 1 \\ -x^2 \end{pmatrix}, \qquad \mathbf{e}(x) = \begin{pmatrix} x^2 - x - 1 \\ -x + 1 \end{pmatrix}.$$

He then uses these to compute the public vector

$$\begin{aligned} \mathbf{t}(x) &= A(x) \star \mathbf{s}(x) + \mathbf{e}(x) \\ &= \begin{pmatrix} 4x^3 + 6x^2 + 20x + 20 \\ 3x^3 + 12x^2 + 19x - 5 \end{pmatrix} \end{aligned}$$

and publishes $(A(x), \mathbf{t}(x))$ as his public key.

Example 4. Aretha wants to send the message "hi" to Bernie, using Alkaline AA, so she looks up Bernie's public key, which is

$$\begin{aligned} A(x) &= \begin{pmatrix} 4x^3 + 4x^2 + 10x & 11x^3 + 15x^2 + 10x + 3 \\ 12x^3 + 22x^2 + 4x + 12 & 6x^3 + x + 11 \end{pmatrix}, \\ \mathbf{t}(x) &= \begin{pmatrix} 4x^3 + 6x^2 + 20x + 20 \\ 3x^3 + 12x^2 + 19x - 5 \end{pmatrix} \end{aligned}$$

For the first letter, Aretha encodes "h" as the number 8, or 1000 in binary, corresponding to

$$\mathbf{p}(x) = 1x^3 + 0x^2 + 0x + 0 = x^3.$$

She generates 20 random bits and uses them to compute

$$\mathbf{r}(x) = \begin{pmatrix} x^3 + x^2 + x + 1 \\ -x^3 \end{pmatrix}, \qquad \mathbf{e_1}(x) = \begin{pmatrix} x^3 + x - 1 \\ -x^3 + x^2 - 1 \end{pmatrix}, \qquad e_2(x) = -x^2 + x + 1$$

She then computes

$$\begin{aligned} \mathbf{u}(x) &= A(x)^T \star \mathbf{r}(x) + \mathbf{e}_1(x) = \begin{pmatrix} 7x^3 + 22x^2 + 2x - 15 \\ 4x^3 + x^2 - 13x + 13 \end{pmatrix} \\ v(x) &= \mathbf{t}(x)^T \star \mathbf{r}(x) + e_2(x) + \lfloor q/2 \rceil \, p(x) \\ &= \left[(9x^3 + 22x^2 + 19x + 9) + (-x^2 + x + 1) + 12x^3 \right] \text{MOD } 23 \\ &= 21x^3 + 21x^2 + 20x + 10 \end{aligned}$$

and sends $(\mathbf{u}(x), v(x))$ to Bernie.

Example 5. Bernie receives the message

$$\begin{aligned} \mathbf{u}(x) &= \begin{pmatrix} 7x^3 + 22x^2 + 2x - 15 \\ 4x^3 + x^2 - 13x + 13 \end{pmatrix} \\ v(x) &= 21x^3 + 21x^2 + 20x + 10 \end{aligned}$$

from Aretha. He computes

$$\begin{aligned} &\left\lfloor \lfloor q/2 \rceil^{-1} \left(v(x) - \mathbf{s}(x)^T \star \mathbf{u}(x) \right) \right\rceil \text{MOD } 2 \\ &= \left\lfloor \frac{1}{12}(-16x^3 + 20x^2 - x + 21) \right\rceil \text{MOD } 2 \\ &= \left\lfloor -\frac{4}{3}x^3 + \frac{5}{3}x^2 - \frac{1}{12}x + \frac{7}{4} \right\rceil \text{MOD } 2 \\ &= (-x^3 + 2x^2 + 0x + 2) \text{ MOD } 2 = x^3 + 0x^2 + 0x + 0, \end{aligned}$$

yielding the correct plaintext.

6 CRYPTANALYSIS OF ALKALINE

LWE is considered a lattice problem, meaning that the best way of solving it seems to be looking at lattices such as

$$\begin{aligned}\Lambda &= \{\mathbf{y} \mid \mathbf{y} \equiv A\mathbf{s} \pmod q \text{ for some vector } \mathbf{s} \text{ with integer coordinates}\} \\ &= \{\mathbf{y} \mid \mathbf{y} = A\mathbf{s} + q\mathbf{s}' \text{ for some vectors } \mathbf{s} \text{ and } \mathbf{s}' \text{ with integer coordinates}\}.\end{aligned}$$

If we are given A and $\mathbf{t}$ from (1) and want to find the secret vector $\mathbf{s}$, it is sufficient to find $A\mathbf{s}$ modulo q and then row reduce. We can do this if we can solve the Closest Vector Problem in Λ, because if $\mathbf{e}$ is small then $A\mathbf{s}+q\mathbf{s}'$ for some $\mathbf{s}'$ is likely to be the closest vector in Λ to our given $\mathbf{t}$.

The Closest Vector Problem (CVP) is thought to be hard; there are a number of algorithms for solving it but all of them require time exponential in the dimension of the lattice [7, Secs. 7.5 and 7.13]. Some of the more obvious ideas are also ruled out in the MLWE case because we have the same number of rows and columns rather than more rows [1, Sec. 5.1.1]. On the other hand, in our situation we can take advantage of the fact that not only $\mathbf{e}$ but also $\mathbf{s}$ are small. This leads to two competitive attacks, known as the primal attack [1, Sec. 5.1.2] and the dual attack [1, Sec. 5.1.3]. We will focus on the primal attack, since it is somewhat more direct and also currently seems to be slightly faster.

The primal attack on (1) starts with the observation that since

$$A\mathbf{s} + \mathbf{e} - \mathbf{t} \equiv \mathbf{0} \pmod q,$$

we can consider matrices

$$\mathbf{s}'' = \left(\, \mathbf{s}^T \middle| \mathbf{e}^T \middle| 1 \middle| \mathbf{s}'^T \,\right), \qquad M = \left(\begin{array}{c} A^T \\ \hline I \\ \hline -\mathbf{t}^T \\ \hline qI \end{array}\right)$$

such that $\mathbf{s}''M = \mathbf{0}$. In other words, $\mathbf{s}''$ is in the left kernel of M, which is constructed from public information. (We transpose our matrices in this section because most descriptions of lattice algorithms, as well as most software packages, treat matrices as made up of row vectors rather than column vectors.)

Since we know that $\mathbf{s}$ and $\mathbf{e}$ are small, a good strategy is to look for small (nonzero) vectors in the lattice of integer vectors in the left kernel of M. The first step is to find a basis for this lattice, which can be done by constructing the matrix $M' = (\, M \,|\, I \,)$ and reducing it to integer row echelon form. This is fairly intuitive for students who know Gaussian elimination, although note that even for Alkaline this matrix will be too large to deal with conveniently by hand. One source is Cohen [2, Algorithm 2.4.10], which goes farther and reduces to Hermite Normal Form, which is unnecessary but not problematic. The rows of the right side of the reduced matrix (coming from I) which correspond to the zero rows of the left side (coming from M) form a basis for the integer left kernel.

Example 6. Consider the LWE cryptosystem from Section 2 above, with $\ell = 1$ and a public key

$$A = \begin{pmatrix} 18 & 10 \\ 16 & 4 \end{pmatrix}, \qquad T = \begin{pmatrix} 15 \\ 0 \end{pmatrix}.$$

If Eve wants to recover S, she can first set up the matrix

$$M' = \left(\begin{array}{cc|cccccccc} 18 & 16 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline -15 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 23 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

Since the first two columns of M‘ contain a copy of the identity matrix, the reduction can start in the same way Gaussian reduction would:

$$M' \to \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -18 & -16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -10 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 15 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -23 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -23 & 0 & 0 & 1 \end{pmatrix}$$

Now the greatest common denominator of 15 and -23 in the fifth column is 1, and the extended Euclidean algorithm gives us $-3(15) - 2(-23) = 1$, so we can replace the fifth row with -3(fifth row) $-$ 2(sixth row) and the sixth row with -23(fifth row) $-$ 15(sixth row) to get

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -18 & -16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -10 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -3 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -23 & -15 & 0 \\ 0 & 0 & 0 & 0 & 0 & -23 & 0 & 0 & 1 \end{pmatrix}$$

Swapping the last two rows puts the matrix into echelon form and shows that a basis for the integer left kernel is {(1 0 $-$18 $-$16 0 0 0), (0 1 $-$10 $-$4 0 0 0), (0 0 1 0 -3 -2 0), (0 0 0 -23 0 0 1), (0 0 0 0 -23 -15 0)}.

Given a basis, the gold standard for finding short vectors in a high-dimensional lattice is the Block Korkine-Zolotarev (BKZ) algorithm [7, Sec. 7.13.4], but this is overkill for breaking toy cryptosystems. The Lenstra-Lenstra-Lovász (LLL) algorithm works well enough for our purposes,[3] is built into most Computer Algebra Systems, and is simple enough to explain in one or two class periods if desired. (I have taught both classes where I explained the algorithm and justified its results, and classes where I treated it as a “black box”.) I believe most students with a programming class could even program it if desired. I recommend Hoffstein et al. [7, Sec. 7.13.2] for a readable explanation of the complete algorithm. Cohen [2, Algorithm 2.7.2] notes that the previous matrix reduction step can in fact be combined with LLL to get a faster but more complex algorithm.

Example 7. Performing LLL on the matrix composed of the integer left kernel from the previous example yields

$$\begin{pmatrix} 1 & 0 & -18 & -16 & 0 & 0 & 0 \\ 0 & 1 & -10 & -4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -3 & -2 & 0 \\ 0 & 0 & 0 & -23 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -23 & -15 & 0 \end{pmatrix} \overset{LLL}{\to} \begin{pmatrix} 0 & 0 & 1 & 0 & -3 & -2 & 0 \\ -2 & 2 & 1 & 1 & -1 & 0 & 1 \\ 1 & 2 & 0 & -1 & 1 & -1 & -1 \\ 1 & 1 & 2 & 3 & 2 & 0 & -1 \\ -1 & -1 & 5 & -3 & 0 & 1 & 1 \end{pmatrix}.$$

Looking for a short vector with a 1 in the correct position for $\mathbf{s}''$ produces (1 2 0 $-$1 1 $-$1 $-$1), corresponding to $\mathbf{s} = (1\ 2)^T$ and $\mathbf{e} = (0\ -1)^T$, which is in fact correct.

For the MLWE equation (6), the only difference is that the matrices and vectors over R need to be expanded using the negacyclic matrices and expanded vectors from (4). Despite the extra structure of the matrices, the best known attacks are still the same as for LWE [1, Sec. 5.1].

For simplicity, we stick to the $k = 2$ case from Alkaline and Kyber512. Then if

$$A(x) = \begin{pmatrix} f(x) & g(x) \\ h(x) & j(x) \end{pmatrix}, \qquad \mathbf{s}(x) = \begin{pmatrix} y(x) \\ z(x) \end{pmatrix},$$

[3]To be exact, there is a trade-off between speed and failure rate in these algorithms. LLL runs in time polynomial in the dimension, but has a failure rate exponential in the dimension. BKZ can be tuned to have a polynomial failure rate, but then has an exponential running time. See Hoffstein et al. [7, Sec. 7.13.4] for more details.

We expand these into

$$A = \left(\begin{array}{ccccc|ccccc} f_0 & -f_{n-1} & \cdots & -f_2 & -f_1 & g_0 & -g_{n-1} & \cdots & -g_2 & -g_1 \\ f_1 & f_0 & \cdots & -f_3 & -f_2 & g_1 & g_0 & \cdots & -g_3 & -g_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ f_{n-2} & f_{n-3} & \cdots & f_0 & -f_{n-1} & g_{n-2} & g_{n-3} & \cdots & g_0 & -g_{n-1} \\ f_{n-1} & f_{n-2} & \cdots & f_1 & f_0 & g_{n-1} & g_{n-2} & \cdots & g_1 & g_0 \\ \hline h_0 & -h_{n-1} & \cdots & -h_2 & -h_1 & j_0 & -j_{n-1} & \cdots & -j_2 & -j_1 \\ h_1 & h_0 & \cdots & -h_3 & -h_2 & j_1 & j_0 & \cdots & -j_3 & -j_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{n-2} & h_{n-3} & \cdots & h_0 & -h_{n-1} & j_{n-2} & j_{n-3} & \cdots & j_0 & -j_{n-1} \\ h_{n-1} & h_{n-2} & \cdots & h_1 & h_0 & j_{n-1} & j_{n-2} & \cdots & j_1 & j_0 \end{array}\right), \qquad \mathbf{s} = \left(\begin{array}{c} y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \\ \hline z_0 \\ z_1 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{array}\right)$$

and then proceed as above. The matrix M will have (in this case) dimensions $(6n+1) \times 2n$ and M' will have dimensions $(6n+1) \times (8n+1)$. The recovered vector thus has $6n+1$ entries; the first n are the coefficients of $y(x)$, and the second n are the coefficients of $z(x)$.

Example 8. Suppose Eve wants to read the messages that have been sent using Bernie's key. She knows Bernie's public key

$$A(x) = \begin{pmatrix} 4x^3 + 4x^2 + 10x & 11x^3 + 15x^2 + 10x + 3 \\ 12x^3 + 22x^2 + 4x + 12 & 6x^3 + x + 11 \end{pmatrix},$$
$$\mathbf{t}(x) = \begin{pmatrix} 4x^3 + 6x^2 + 20x + 20 \\ 3x^3 + 12x^2 + 19x - 5 \end{pmatrix}$$

and expands it into the matrix and vector

$$A = \left(\begin{array}{cccc|cccc} 0 & -4 & -4 & -10 & 3 & -11 & -15 & -10 \\ 10 & 0 & -4 & -4 & 10 & 3 & -11 & -15 \\ 4 & 10 & 0 & -4 & 15 & 10 & 3 & -11 \\ 4 & 4 & 10 & 0 & 11 & 15 & 10 & 3 \\ \hline 12 & -12 & -22 & -4 & 11 & -6 & 0 & -1 \\ 4 & 12 & -12 & -22 & 1 & 11 & -6 & 0 \\ 22 & 4 & 12 & -12 & 0 & 1 & 11 & -6 \\ 12 & 22 & 4 & 12 & 6 & 0 & 1 & 11 \end{array}\right), \qquad \mathbf{t} = \left(\begin{array}{c} 20 \\ 20 \\ 6 \\ 4 \\ \hline -5 \\ 19 \\ 12 \\ 3 \end{array}\right)$$

She then constructs the matrix

$$M' = \left(\begin{array}{cccccccc|cccccccccccccccccccccccc}
0 & 10 & 4 & 4 & 12 & 4 & 22 & 12 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-4 & 0 & 10 & 4 & -12 & 12 & 4 & 22 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-4 & -4 & 0 & 10 & -22 & -12 & 12 & 4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-10 & -4 & -4 & 0 & -4 & -22 & -12 & 12 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 10 & 15 & 11 & 11 & 1 & 0 & 6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-11 & 3 & 10 & 15 & -6 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-15 & -11 & 3 & 10 & 0 & -6 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-10 & -15 & -11 & 3 & -1 & 0 & -6 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
-20 & -20 & -6 & -4 & 5 & -19 & -12 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

And reduces it to

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 2 & 9 & 6 & 14 & 4 & 1 & 8 & 7 & 6 & 2 & 1 & -2 & 6 & 4 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 19 & 19 & 11 & 19 & 1 & 11 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 21 & 9 & 13 & 21 & 12 & 5 & 9 & 8 & 7 & 1 & 1 & -2 & 6 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 3 & 0 & 22 & 4 & 0 & 9 & 8 & 8 & 2 & 1 & -1 & 7 & 4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 17 & 1 & 21 & 18 & 1 & 6 & 21 & 11 & 10 & 9 & 3 & 1 & -3 & 10 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 16 & 2 & 8 & 17 & 3 & 11 & 14 & 22 & 19 & 18 & 5 & 3 & -6 & 18 & 11 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 9 & 12 & 15 & 3 & 5 & 20 & 11 & 19 & 17 & 16 & 4 & 2 & -4 & 15 & 9 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 19 & 4 & 10 & 21 & 9 & 3 & 14 & 5 & 5 & 4 & 1 & 0 & -2 & 4 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 5 & 8 & 18 & 15 & 2 & 0 & 22 & 11 & 10 & 10 & 3 & 1 & -3 & 9 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 21 & 14 & 17 & 9 & 19 & 22 & 15 & 13 & 13 & 3 & 2 & -4 & 12 & 7 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 20 & 20 & 6 & 4 & -5 & 19 & 12 & 3
\end{pmatrix}$$

(The reduction in this example was done with Sage [13], which reduces all the way to Hermite Normal Form.) A basis for the left integer kernel is then

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 19 & 19 & 11 & 19 & 1 & 11 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 19 & 21 & 9 & 13 & 21 & 12 & 5 & 9 & 8 & 7 & 1 & 1 & -2 & 6 & 4 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 3 & 0 & 22 & 4 & 0 & 9 & 8 & 8 & 2 & 1 & -1 & 7 & 4 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 17 & 1 & 21 & 18 & 1 & 6 & 21 & 11 & 10 & 9 & 3 & 1 & -3 & 10 & 6 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 16 & 2 & 8 & 17 & 3 & 11 & 14 & 22 & 19 & 18 & 5 & 3 & -6 & 18 & 11 & 2 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 9 & 12 & 15 & 3 & 5 & 20 & 11 & 19 & 17 & 16 & 4 & 2 & -4 & 15 & 9 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 19 & 4 & 10 & 21 & 9 & 3 & 14 & 5 & 5 & 4 & 1 & 0 & -2 & 4 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 5 & 8 & 18 & 15 & 2 & 0 & 22 & 11 & 10 & 10 & 3 & 1 & -3 & 9 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 21 & 14 & 17 & 9 & 19 & 22 & 15 & 13 & 13 & 3 & 2 & -4 & 12 & 7 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 23 & 20 & 20 & 6 & 4 & -5 & 19 & 12 & 3
\end{pmatrix},$$

and applying LLL gives Eve

$$\begin{pmatrix}
1 & 0 & 1 & -1 & 0 & 0 & -1 & 0 & -1 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 2 & -1 & 0 & -1 & -1 & 1 & -1 & -2 & 2 & -3 & 2 & 1 & 1 & 0 & 2 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & -1 \\
1 & 1 & -1 & 0 & -1 & -1 & -1 & -2 & 0 & 1 & -2 & 2 & -1 & -3 & -2 & 2 & 1 & -1 & -1 & 0 & 2 & -1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 1 & 1 & -1 & 1 & 1 & -2 & -4 & -1 & 0 & -1 & -2 & -2 & 0 & -1 & 0 & 0 \\
-1 & 1 & 1 & -2 & 0 & -2 & 0 & 1 & -4 & 1 & 0 & -2 & -1 & 1 & 2 & 2 & 1 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & -3 & 0 & 0 & 3 & -1 & 3 & 0 & 3 & -2 & -1 & -1 & 0 & 0 & 0 & -2 & -1 & 0 \\
0 & 1 & 0 & -2 & 2 & -3 & -2 & 0 & 1 & -1 & 0 & 1 & 0 & 1 & -2 & -2 & 2 & -2 & 0 & 0 & 2 & -2 & 0 & 1 & 0 \\
3 & 0 & 1 & 0 & 1 & -2 & -2 & 1 & -1 & 3 & -1 & -1 & 0 & 1 & -1 & -3 & 2 & 0 & 0 & 1 & 1 & -2 & 2 & -1 & -2 \\
2 & -3 & 0 & 2 & -1 & 3 & -3 & 0 & -1 & 2 & 1 & 0 & 0 & -1 & -1 & 4 & 0 & 0 & -2 & 1 & 0 & -1 & 1 & 1 & 1 \\
2 & 1 & -1 & 1 & -1 & 0 & 2 & -2 & -3 & 2 & 2 & -1 & -3 & -1 & 0 & -2 & 1 & 2 & 0 & -1 & 0 & -1 & 1 & -2 & -1 \\
1 & -1 & 0 & 1 & 0 & 0 & 0 & 3 & -1 & 2 & 2 & 2 & -2 & -4 & -1 & 2 & -3 & -1 & -1 & 1 & -1 & 0 & -1 & -1 & -2 \\
1 & 1 & 0 & 1 & 0 & -2 & -1 & 2 & 0 & -1 & 6 & -1 & -1 & 3 & -1 & -1 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & -3 \\
1 & -3 & -1 & 1 & -1 & 3 & 1 & 4 & -1 & -1 & -2 & 0 & 0 & 1 & 0 & 1 & -2 & 2 & 1 & 2 & -2 & -1 & -1 & 0 & 0 \\
-1 & 1 & -1 & 1 & -2 & -1 & 0 & -1 & -2 & -2 & -2 & 0 & 3 & -4 & 2 & 2 & -1 & -1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 \\
0 & -1 & -1 & 0 & 1 & -3 & 3 & 0 & -2 & 2 & -1 & -1 & 1 & 0 & -2 & -3 & 1 & 1 & 2 & 1 & 1 & -3 & 3 & 0 & 1 \\
0 & -3 & 2 & 0 & -2 & 1 & -1 & 2 & -2 & 1 & 0 & -1 & -3 & 3 & -2 & 0 & -1 & 0 & 1 & 3 & 0 & 2 & 1 & 0 & 2 \\
1 & 2 & 0 & 1 & 0 & -2 & 2 & 0 & 0 & -4 & 0 & 2 & -1 & 1 & -3 & 2 & 1 & 2 & 2 & 0 & 0 & 0 & 2 & -1 & -3
\end{pmatrix}.$$

The top row has a 1 in the correct position for $\mathbf{s}''$, and reveals the secret key $\mathbf{s} = \left(\begin{array}{cccc|cccc} 1 & 0 & 1 & -1 & 0 & 0 & -1 & 0 \end{array}\right)^T$, or $\mathbf{s}(x) = \left(\begin{array}{c|c} -x^3 + x^2 + 1 & -x^2 \end{array}\right)^T$ (remembering that the constant term is listed first in $\mathbf{s}$).

7 CLASSROOM USE

The exercises in Appendix A are very similar to a set of NTRU exercises that I have given to multiple classes as homework. Like the ones I have used in the past, these include key generation, encryption, decryption, and cryptanalysis using LLL. I have not yet had a chance to teach a class using Alkaline, so I am not certain how long it would take to do these as in-class exercises, but I am fairly confident that they would be reasonable homework problems, with the aid of computer software as appropriate.

Note that the random numbers in the encryption and key generation exercises have been "derandomized" using information common to the class. This allows for easier grading while not giving away the answers to students who search for them online. The code used to generate the Exercises is available on GitHub [?redacted], and this can also be used to check answers.

8 CONCLUSION

While not as simple to explain as RSA, I think that LWE-based cryptosystems are suitable for use in several undergraduate classes, including linear algebra, abstract algebra, and, of course, cryptography. The most sophisticated mathematics is probably the polynomial modular arithmetic, but at heart it's still just polynomial long division with remainder. If one wanted to focus on the matrices, say for a linear algebra class, one could take $n = 1$, effectively reducing to a matrix-based LWE system. On the other hand, for students unfamiliar with matrices, one might want to take $k = 1$, yielding an RLWE system. Alternatively, one could relatively easily rewrite the versions of Alkaline in terms of systems of two equations in two variables without much trouble. The cryptanalysis of Alkaline and Kyber should be reserved for a course where students are fluent with matrix reductions, but it should be no surprise that cryptanalysis takes more sophisticated mathematics than encryption and decryption. The same is certainly true for RSA if one wants to understand modern factoring methods. LLL is no harder to explain than the quadratic sieve, and probably easier than the number field sieve.

Although Alkaline could be simplified further, I think it is worth trying to keep the important features of the MLWE system. I have introduced polynomial modular arithmetic in the context of AES and NTRU to quite a few undergraduate students with no abstract algebra background. They find it challenging but certainly learnable. Likewise, matrices as a shorthand for systems of equations are not so difficult conceptually, and in fact are familiar to many students who have not had a formula linear algebra class. Finally, one could simplify the error distributions, but centered binomial coefficients are easy to generate and fairly intuitive to explain, based on my experiences teaching undergraduate probability. The combination of all of these features gives students a good idea of the wide range of mathematical techniques that go into modern cryptography!

REFERENCES

1. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M, Schwabe, P., Seiler, G., and Stehlé, D. August 4, 2021. *Algorithm Specifications And Supporting Documentation*, Technical Report (version 3.02), NIST.
2. Cohen, H. 1993. *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics, vol. 138, Berlin, Heidelberg: Springer.
3. Daemen, J. and Rijmen, V. September 3, 1999. *AES Proposal: Rijndael*, Document version 2, AES Proposals, NIST.
4. Ducas, L. and Schanck, J. January 21, 2021. *security-estimates/Kyber.py at master · pq-crystals/security-estimates*,.
5. Dummit, D. S. and Foote, R. M. 2004. *Abstract Algebra*, Hoboken, NJ: Wiley.
6. Gentry, C., Peikert, C., and Vaikuntanathan, V. 2008. *Trapdoors for hard lattices and new cryptographic constructions*. Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, pp. 197–206, DOI 10.1145/1374376.1374407.
7. Hoffstein, J., Pipher, J., and Silverman, J. H. 2014. *An Introduction to Mathematical Cryptography*, 2nd ed., Undergraduate Texts in Mathematics, New York, NY: Springer.
8. Holden, J. 2013. *Demitasse: A "Small" Version of the Tiny Encryption Algorithm and its Use in a Classroom Setting*. Cryptologia. 37(1): 74–83, DOI 10.1080/01611194.2012.660237.
9. _____. 2013. *A Good Hash Function is Hard to Find, and Vice Versa*. Cryptologia. 37: 107–119.
10. Lyubashevsky, V., Peikert, C., and Regev, O. 2010. *On Ideal Lattices and Learning with Errors over Rings*. Advances in Cryptology — EUROCRYPT 2010, pp. 1–23, DOI 10.1007/978-3-642-13190-5_1.
11. Musa, M. A., Schaefer, E. F., and Wedig, S. 2003. *A Simplified AES Algorithm and its Linear and Differential Cryptanalyses*. Cryptologia. 27(2): 148–177, DOI 10.1080/0161-110391891838.
12. Regev, O. 2005. *On lattices, learning with errors, random linear codes, and cryptography*. Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 84–93, DOI 10.1145/1060590.1060603.
13. The Sage Developers. 2021. *Sagemath, the Sage Mathematics Software System (Version 9.3)*. `https://www.sagemath.org`.
14. Schaefer, E. F. 1996. *A Simplified Data Encryption Standard Algorithm*. Cryptologia. 20(1): 77–84, DOI 10.1080/0161-119691884799.
15. Stinson, D. R. and Paterson, M. 2018. *Cryptography: Theory and Practice*, 4th Edition, Boca Raton, FL: Chapman and Hall/CRC.

A CLASS EXERCISES

For the following exercises, use the conversion "a" = 0001, "b" = 0010, ..., "o" = 1111, "p" = 0000.

Exercise 1. Finish Aretha's message to Bernie by encrypting the letter "i", or 1001 in binary. For your random bits, use the first five letters of your instructor's family name, skipping any letters "q" through "z".

Exercise 2. You are Bernie, and Aretha sends you the ciphertext

$$\mathbf{u}(x) = \begin{pmatrix} 12x^3 + 16x^2 - 9x + 22 \\ -5x^3 - 7x^2 + 21x - 2 \end{pmatrix}$$
$$v(x) = -16x^3 - 4x^2 - 20x - 11$$

Decrypt the message using the information above. (Hint: You should get the initial letter of Aretha's family name.)

Exercise 3. You are Bing Wen, and you would like to use Alkaline AA. You pick the public matrix

$$A(x) = \begin{pmatrix} 20x^3 + 15x^2 + 14x + 18 & 18x^3 + 2x^2 + 9x + 17 \\ 11x^3 + 20x^2 + 10x + 16 & 17x^3 + 14x^2 + 16x + 11 \end{pmatrix}$$

and the private key

$$\mathbf{s}(x) = \begin{pmatrix} x^2 + x \\ x^2 - 1 \end{pmatrix}.$$

Find the second part $\mathbf{t}(x)$ of the public key. For your random bits, use the first four letters of your school's name, skipping any letters "q" through "z". (Also skip the word "the", unless you go to The Ohio State University.)

Exercise 4. You are Ilham, and you overhear Aisha and Bassam using Alkaline AA with Bassam's public key

$$A(x) = \begin{pmatrix} 20x^3 + 18x^2 + 2x + 5 & 17x^3 + 2x^2 + 11x + 14 \\ 8x^2 + 19x + 20 & 7x^3 + 9x^2 + 10x \end{pmatrix}$$
$$\mathbf{t}(x) = \begin{pmatrix} 9x^3 + 13x^2 + 4x + 2 \\ 18x^3 + 10x^2 - 9x + 5 \end{pmatrix}.$$

Aisha sends Bassam the message

$$\mathbf{u}(x) = \begin{pmatrix} 10x^3 + 11x^2 + 16x + 2 \\ 2x^2 - 4x \end{pmatrix}$$
$$v(x) = 19x^3 + 12x^2 + 19x - 7.$$

Use LLL to find Bassam's private key, and then decode the message. (Hint: You should get the initial letter of the most populous country in Africa.)