DILITHIUM/2 = LITHIUM? POST-QUANTUM SIGNATURES FOR UNDERGRADUATE CLASSES

JOSHUA HOLDEN

Abstract. This paper continues the ElectroNic ExeRcises for CiphEricaL Learning (ENERCELL) project by introducing Lithium, a simplified version of the Dilithium digital signature scheme which is currently being standardized by the National Institute of Standards and Technology. This scheme is "post-quantum" in the sense that while it does not require a quantum computer to run, it is expected to be secure against practical quantum computers, which could be developed in the near future. No previous knowledge of quantum or postquantum cryptography is necessary for this paper or to teach Lithium in your classes. Versions of the system will be presented which are suitable for linear algebra, abstract algebra, and/or cryptography classes, along with some analysis of the results in a cryptography class.

1. INTRODUCTION

Today's Internet would not be possible without two ideas invented in the 1970's: asymmetric encryption and digital signatures. Asymmetric encryption is the idea that it is possible to send secret messages such that anyone on earth can encrypt them, but only the intended receiver can decrypt. Digital signatures are the dual notion: only one person can sign a message, but anyone on earth can verify that the signature is correct. We use these notions anytime we send a credit card number (for instance) over the Internet. Asymmetric encryption makes sure that only one person can read our card number, and digital signatures make sure that the one person is who we think it is.

Unfortunately, many experts predict that quantum computers will be able to break all of the currently used asymmetric encryption and digital signature schemes in the next few decades. In response, the National Institute of Standards and Technology (NIST) is selecting a 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, but it is designed to be resistant to breaking by quantum computers. 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.

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, now officially ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism) was originally called Kyber, after the power source

Key words and phrases. post-quantum, Dilithium, classroom, cryptography, public-key.

for lightsabers in *Star Wars.* The signature algorithm, now officially ML-DSA (Module-Lattice-Based Digital Signature Algorithm) was 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 (Schaefer, 1996), S-AES (Musa et al., 2003), Demitasse (Holden, 2013a), and JHA (Holden, 2013b), I thought that it would be useful to develop similar versions of Kyber and Dilithium for classroom use. I call this project ElectroNic ExeRcises for CiphEricaL Learning, or ENERCELL. An earlier paper (Holden, 2024) introduced a simplified version of Kyber, called Alkaline. In this paper, I describe outline a simplified signature algorithm, called Lithium.

2. LEARNING WITH ERRORS

Asymmetric encryption schemes are generally based on hard mathematical problems; for example, the well-known RSA scheme is based on the difficulty of factoring a large number into prime factors. As with Kyber, the mathematical hard problem underlying Dilithium is a matrix algebra problem known as Learning With Errors. Matrix algebra is already an important part of many cryptography courses (see, for example, Sections 1.6 and 4.5 of Holden (2018) for some examples and justification), so this problem can be introduced easily enough in those classes.

The original Learning With Errors problem, known as LWE, is a fairly straightforward-sounding problem: given a fixed prime q and a vector $\mathbf{t}$ of the form

$$\mathbf{t} \equiv A\mathbf{s_1} + \mathbf{s_2} \pmod{q}, \tag{1}$$

where A is a public matrix with at least as many rows as columns and $\mathbf{s_2}$ is a "small error vector" drawn from some probability distribution, find the secret vector $\mathbf{s_1}$. (All matrix and vector components are required to be integers. $a \equiv b \pmod{q}$ means that the difference of a and b is divisible by q.)

The difficulty of this problem stems from the integer requirement and from a careful choice of the size of the error vector compared to the other vectors. Notice that if there was no error vector, equation (1) 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 $\mathbf{s_1}$, which will be a problem. Notice also that a least-squares approach will not work here due to the integer requirement and the modulus. For more information about LWE, see Holden (2023) and Holden (2024).

3. IDENTIFICATION SCHEMES AND FIAT-SHAMIR

Digital signature schemes based on LWE fall into two broad categories (Lyubashevsky, 2009): one using the "hash-and-sign" paradigm and one using "Fiat-Shamir". Dilithium falls into the second category, which involves starting with a "commit-challenge-response" identification scheme. The general idea of such a scheme is that Aretha (Know Your Meme, 2009) publishes a "public key" and wants to prove to Bernie (Know Your Meme, 2021)

[1]Note that competing Post-Quantum proposals included NewHope and SABER, which has a version called LightSABER. One of the other two digital signature algorithm chosen for standardization is named FALCON; I have not been able to determine whether this is a Millennium Falcon or some other kind.

that she knows the "secret key" that goes with it, without revealing the secret key. (See Holden (forthcoming) for more on the Fiat-Shamir paradigm.)

In our case, Aretha wants to prove to Bernie that she knows the secret key, a pair of matrices (S_1, S_2) with "small" entries. She publishes a public key which in this case will be (A, T), where

$$T \equiv AS_1 + S_2 \pmod{q}. \tag{2}$$

(This matrix equation is equivalent to a system of vector equations of the form (1).)

Aretha picks random $\mathbf{y_1}$ and $\mathbf{y_2}$ with small coefficients (relative to q) and sends Bernie a commitment $\mathbf{w} \equiv A\mathbf{y_1} + \mathbf{y_2} \pmod{q}$. Bernie responds with a randomly picked challenge vector $\mathbf{c}$. Aretha then combines S_i, $\mathbf{c}$, and $\mathbf{y_i}$ into a response $(\mathbf{z_1}, \mathbf{z_2}) \equiv (\mathbf{y_1} + S_1\mathbf{c}, \mathbf{y_2} + S_2\mathbf{c})$ $\pmod{q}$ and sends it back to Bernie. Bernie checks that $A\mathbf{z_1} + \mathbf{z_2} \equiv \mathbf{w} + T\mathbf{c} \pmod{q}$. This procedure is illustrated in Figure 1. On the assumption that Aretha cannot solve the LWE problem, Bernie can verify that Aretha could not have constructed $(\mathbf{z_1}, \mathbf{z_2})$ without knowing (S_1, S_2).

The challenge is necessary so that if Frank the Forger overhears $\mathbf{w}$ and $\mathbf{z_i}$, he can't use them to impersonate Aretha in the future. The commitment allows Aretha to mix the proof that she knows $\mathbf{s_i}$ with a proof that she knows some random $\mathbf{y_i}$. This prevents Bernie from gaining information about $\mathbf{s_i}$. However, Aretha needs to commit to her $\mathbf{w}$ *before* the challenge. Otherwise, Frank could work backwards from $\mathbf{c}$ and S_i to find $\mathbf{w}$ and $\mathbf{z_i}$ that will satisfy Bernie. Ideally, the resulting scheme is a "zero-knowledge proof of identity", where Aretha can prove to Bernie that she knows the secret key without Bernie or Frank getting any other useful information.

In order to turn this identification scheme into a digital signature, we need to remove the steps where Aretha and Bernie send information back and forth. Fiat and Shamir (1987) showed how to do this by replacing Bernie's challenge with the output of a publicly known *hash function.* A cryptographically secure hash function H should:

a. be fast to compute,
b. distribute hash values evenly,
1. given c, be resistant to finding a preimage M such that $\mathrm{H}(M) = c$,
2. given M, be resistant to finding a second preimage M' such that $\mathrm{H}(M') = \mathrm{H}(M)$,
3. make it hard to find any two colliding M_1 and M_2 such that $\mathrm{H}(M_1) = \mathrm{H}(M_2)$.

(For more on hash functions, see for example, Holden (2013b).)

Aretha starts by picking random $\mathbf{y_1}$ and $\mathbf{y_2}$ as before and calculating $\mathbf{w} = (A\mathbf{y_1} + \mathbf{y_2}) \,\mathrm{MOD}\, q$. ($\mathrm{MOD}\, q$ indicates the action of reducing to the least nonnegative residue, as opposed to the congruence relation $a \equiv b \pmod{q}$ mentioned earlier.) Instead of sending it to Bernie, she calculates $\mathbf{c} = H(M, \mathbf{w})$, where M is a list of numbers representing the message she wants to sign. She uses this $\mathbf{c}$ when she creates her response $\mathbf{z_1}, \mathbf{z_2}$. The signature is then the triple $(\mathbf{z_1}, \mathbf{z_2}, \mathbf{c})$, which Aretha sends to Bernie along with the message. Bernie can verify that $A\mathbf{z_1} + \mathbf{z_2} \equiv \mathbf{w} + T\mathbf{c} \pmod{q}$ as before. Furthermore, by checking that $\mathbf{c} = H(M, \mathbf{w})$, and assuming the properties (1)–(3) of the hash function, Bernie knows that Frank did not alter the message M and that he did not work backwards from $\mathbf{c}$ and S_i to find $\mathbf{w}$.

Aretha

- picks $k \times k$ matrix A with uniform random entries between 0 and $q-1$
- picks $k \times r$ matrices S_1, and S_2 with uniform random entries between $-\eta$ and η
- computes $T = (AS_1 + S_2) \operatorname{MOD} q$

- posts public key (A, T)
- keeps private key (S_1, S_2) secret

(repeat the following until Bernie is satisfied)

Aretha

- picks vectors $\mathbf{y_1}$ and $\mathbf{y_2}$ with uniform random entries between $-(\gamma-1)$ and $\gamma-1$

$$\mathbf{y_1}, \mathbf{y_2}$$

$$\downarrow A$$

Aretha $\rightarrow \mathbf{w} = (A\mathbf{y_1} + \mathbf{y_2}) \operatorname{MOD} q \rightarrow$ Bernie

$\leftarrow$ uniform random vector $\mathbf{c}$ with entries between -1 and 1 $\leftarrow$

$$\rightarrow (\mathbf{z_1}, \mathbf{z_2}) = (\mathbf{y_1} + S_1\mathbf{c}, \mathbf{y_2} + S_2\mathbf{c}) \operatorname{MOD} q \rightarrow$$

Bernie

$$(\mathbf{z_1}, \mathbf{z_2}, \mathbf{w}, \mathbf{c})$$

$$\downarrow (A, T)$$

$$A\mathbf{z_1} + \mathbf{z_2} \overset{?}{\equiv} \mathbf{w} + T\mathbf{c} \pmod{q}$$

(repeat as necessary)

- if they match, Bernie accepts that Aretha knows the secret

FIGURE 1. An LWE identification scheme

4. FIAT-SHAMIR WITH ABORTS

There is a problem with this scheme, which is that it is not quite zero-knowledge: the values of $\mathbf{z_1}$ and $\mathbf{z_2}$ are going to be biased in directions related to S_1 and S_2 (National Institute of Standards and Technology, 2024, Sec. 3.3). Given enough signatures, Bernie can get enough information to guess S_1 and S_2. This was used to break several of the early attempts at signature schemes based on LWE and similar problems (Hoffstein et al., 2014, Sec. 7.12.2). One solution is to use "Fiat-Shamir with Aborts", which was first proposed by Lyubashevsky in 2009 (Lyubashevsky, 2009). Lyubashevsky used an idea from statistics called rejection sampling (Hoffstein et al., 2014, Sec. 7.12.3), which changes one random distribution into another by rejecting some of the samples.

For example, suppose you want to pick random points evenly distributed throughout a circle of radius 1. You could pick a random distance from the center uniformly between 0 and 1 and a random angle, but that will tend to make the points bunch up in the center. You could try to adjust the distribution of distances so that small distances are picked less often, in just the right way to even everything out. Or, you could pick points evenly across a square circumscribed around the circle, and then throw away (or "do over") any point which is outside the circle. A classroom demonstration of this can be found online (Holden, 2025).

In our case, we are going to pick a fixed interval around the origin, not depending on the secret key, such that all of the values of $\mathbf{z_i}$ inside that interval are distributed evenly. To sign a message, Aretha follows the same procedure as above, except that if either $\mathbf{z_1}$ or $\mathbf{z_2}$ is outside our interval, she aborts and starts over with a different $\mathbf{y_1}$ and $\mathbf{y_2}$.

We can now outline a digital signature based on LWE which is more or less the same as Lyubashevsky's. To sign a message of r bits, the private key is a pair (S_1, S_2) of matrices of r columns each with small coefficients. The public key is (A, T), as before. Aretha chooses random r-entry nonce vectors $\mathbf{y_1}$ and $\mathbf{y_2}$, also with small coefficients, and calculates

$$\mathbf{c} = \mathrm{H}(M, (A\mathbf{y_1} + \mathbf{y_2}) \,\mathrm{MOD}\, q)$$

for some vector-valued hash function H which outputs "small" vectors. (MOD indicates the action of reducing to the least nonnegative residue, as opposed to $\pmod q$, which indicates a relation between two numbers.) She then computes $(\mathbf{z_1}, \mathbf{z_2})$, where

$$\mathbf{z_1} = \mathbf{y_1} + S_1\mathbf{c}\ , \qquad \mathbf{z_2} = \mathbf{y_2} + S_2\mathbf{c}\ .$$

Note that since $\mathbf{y_1}$, $\mathbf{y_2}$, S_1, S_2, and $\mathbf{c}$ have small coefficients relative to q, it should not be necessary to reduce these calculations modulo q. If any coefficient of $\mathbf{z_1}$ or $\mathbf{z_2}$ is outside our chosen interval, then Aretha starts over with a new $\mathbf{y_1}$ and $\mathbf{y_2}$. Otherwise, she sends Bernie the message and the signature $(\mathbf{z_1}, \mathbf{z_2}, \mathbf{c})$.

Bernie verifies the signature by first checking to make sure $\mathbf{z_1}$ or $\mathbf{z_2}$ are inside our interval — if they are not, it may indicate that someone is trying to forge Aretha's signature. If they are in the interval, Bernie calculates

$$\mathbf{c}' = \mathrm{H}(M, (A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c}) \,\mathrm{MOD}\, q).$$

Since

$$\begin{aligned} A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c} &\equiv A\,(\mathbf{y_1} + S_1\mathbf{c}) + (\mathbf{y_2} + S_2\mathbf{c}) - (AS_1 + S2)\,\mathbf{c} \pmod q \\ &\equiv (A\mathbf{y_1} + AS_1\mathbf{c}) + (\mathbf{y_2} + S_2\mathbf{c}) - (AS_1\mathbf{c} + S_2\mathbf{c}) \pmod q \\ &\equiv A\mathbf{y_1} + \mathbf{y_2} \pmod q, \end{aligned}$$

$\mathbf{c}'$ should be equal to the $\mathbf{c}$ in the signature. If it is, then Bernie accepts the signature as valid. Figure 2 summarizes this process.

Example 1 (The LWE signature scheme)**.** Aretha and Bernie agree to use the system described above with $r = 4$, $q = 41$, and 4×4 matrices for the public and private keys. We also have the "smallness" parameters $\eta = 1$ (for S_i), $\gamma = 16$ (for $\mathbf{y_i}$) and $\gamma - \beta = 15$ (for $\mathbf{z_i}$). (These parameters will be explained in more detail in Section 8.)

Aretha first generates the random matrices

$$A = \begin{pmatrix} 4 & 14 & 4 & 14 \\ 27 & 28 & 8 & 40 \\ 9 & 5 & 8 & 19 \\ 8 & 9 & 14 & 3 \end{pmatrix}, \qquad S_1 = \begin{pmatrix} -1 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & -1 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

$$S_2 = \begin{pmatrix} 0 & -1 & -1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

She then uses these to compute the public matrix

$$T = (AS_1 + S_2) \,\mathrm{MOD}\, 41 = \begin{pmatrix} 27 & 9 & 36 & 22 \\ 24 & 19 & 14 & 21 \\ 21 & 37 & 33 & 23 \\ 3 & 37 & 33 & 30 \end{pmatrix}$$

and publishes (A, T) as her public verification key.

Aretha wants to send the message "hola" to Bernie. She computes random

$$\mathbf{y_1} = (-3\ 0\ \ -1\ 14)^T \qquad \text{and} \qquad \mathbf{y_2} = (0\ \ -12\ 9\ \ -2)^T$$

with coefficients between $-\gamma$ and γ and calculates $\mathbf{w} = (A\mathbf{y_1} + \mathbf{y_2}) \,\mathrm{MOD}\, 41 = (16\ 8\ \ -6\ 2)^T$. The hash function that we will introduce in Section 5 gives

$$H(M, (16\ 8\ \ -6\ 2)^T) = (0\ 0\ 0\ 1)^T$$

Aretha continues calculating

$$\mathbf{z_1} = \mathbf{y_1} + S_1\mathbf{c} = (-2\ 1\ 0\ 14)^T$$
$$\mathbf{z_2} = \mathbf{y_2} + S_2\mathbf{c} = (0\ \ -13\ 10\ \ -3)^T$$

The coefficients are smaller (in absolute value) than $\gamma - \beta$, so the check passes. Aretha sends the message and the signature $(\mathbf{z_1}, \mathbf{z_2}, \mathbf{c})$ to Bernie.

Finally, Bernie receives the message $M =$ `"hola"` and the signature

$$\mathbf{z_1} = (-2\ 1\ 0\ 14)^T$$
$$\mathbf{z_2} = (0\ \ -13\ 10\ 3)^T$$
$$\mathbf{c} = (0\ 0\ 0\ 1)^T$$

He checks that the coefficients of $\mathbf{z_i}$ are smaller than $\gamma - \beta$ and calculates

$$\mathrm{H}(M, (A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c}) \,\mathrm{MOD}\, 41)$$

Aretha

- picks $k \times k$ matrix A with uniform random entries between 0 and $q-1$
- picks $k \times r$ matrices S_1, and S_2 with uniform random entries between $-\eta$ and η
- computes $T = (AS_1 + S_2) \operatorname{MOD} q$

- posts public key (A, T)
- keeps private key (S_1, S_2) secret

Aretha

- converts her message M into a list of numbers
- picks vectors $\mathbf{y_1}$ and $\mathbf{y_2}$ with uniform random entries between $-(\gamma-1)$ and $\gamma-1$

$$\mathbf{y_1}, \mathbf{y_2}$$

$$\downarrow A$$

$$\mathbf{w} = (A\mathbf{y_1} + \mathbf{y_2}) \operatorname{MOD} q$$

$$\downarrow M$$

$$\mathbf{c} = H(M, \mathbf{w})$$

$$\downarrow \mathbf{c}$$

Aretha $\rightarrow (\mathbf{z_1}, \mathbf{z_2}) = (\mathbf{y_1} + S_1\mathbf{c}, \mathbf{y_2} + S_2\mathbf{c}) \rightarrow$ Bernie

$$\rightarrow \mathbf{c} \rightarrow$$

$$\rightarrow M \rightarrow$$

Bernie

$$(M, \mathbf{z_1}, \mathbf{z_2}, \mathbf{c})$$

$$\downarrow (A, T)$$

$$\mathbf{c} \overset{?}{=} \mathrm{H}(M, (A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c}) \operatorname{MOD} q)$$

- if they match (and $\mathbf{z_i}$ are small), Bernie accepts the signature

FIGURE 2. An LWE signature scheme

Since $(A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c}) \operatorname{MOD} 41 = (16\ 8\ \ -6\ 2)^T$, this gives the correct result:

$$\mathrm{H}(M, (16\ 8\ \ -6\ 2)^T) = (0\ 0\ 0\ 1)^T.$$

Bernie accepts the signature.

5. A "HASH-FREE" HASH FUNCTION

In order to turn this into a complete digital signature system, we need to specify a few elements more completely, namely the size of the matrices, vectors, and modulus, and the hash function H. The choice of hash function proved to require the most thought of any part of the design of Lithium. In many cryptography classes (including mine in the past), hash functions are not covered. While in practice, hash functions are used to add to the speed and security of most digital signatures, including RSA (Schneier, 1996, Sec. 2.6; Paar et al., 2024, Sec. 11.1), many signature schemes can be and are taught without them. While there are hash functions designed to be used in a classroom (for instance, see Holden (2013b)), I wanted my new system to be usable without necessarily requiring class time to explain an additional new topic.

Unlike with RSA, using some sort of hash function is a necessity with Fiat-Shamir signatures. There needs to be a way to mix the message with the commitment and use them equally to produce the challenge. The standardized Dilithium uses a three-stage process to hash M and $\mathbf{w}$ into a challenge vector $\mathbf{c}$. First, the SHAKE256 variant of the SHA-3 hash function (Paar et al., 2024, Sec. 11.5) is used to hash the message together with a representation of the public key. This gives a convenient starting point for future stages, and also serves to prevent certain attacks which involve changing the intended key for signing a message. In the second stage, the result of the first stage is concatenated with the challenge and hashed using SHAKE256 again.

In the third stage, the actual challenge is generated. In the standardized Dilithium, the output of the third stage should be a polynomial with 256 integer coefficients between -1 and 1, where exactly τ coefficients are nonzero for a parameter $\tau \leq 64$. (See Section 6 for the details.) Since most of the coefficients need to be zero, the most obvious ways to generate a random polynomial are either very inefficient or produce a non-uniform distribution of polynomials. The procedure used to generate the polynomials uniformly is based on the "inside-out Fisher-Yates shuffle" (Knuth, Sec. 3.4.2), shown in Figure 3. One feature of this implementation is that the number of random bits needed is not fixed, but depends on the number of times Step 2a needs to be repeated. Therefore, the standardized version of Dilithium uses SHAKE256 as an *extendable-output function* (XOF) in the third stage, rather than a hash function strictly speaking, in order to produce the random bits necessary. The difference is that the output of the XOF can be arbitrarily long, rather than fixed as in a hash function.

For simplicity, I made a number of changes to this three-stage process. The first stage I eliminated entirely. For the second stage, I decided to use a hash function that was simple, but not cryptographically secure. This is a compromise, but has some justification. It is generally thought (and proved in some cases) (Neven et al., 2009; Schnorr, 1991) that the hash function used in Fiat-Shamir does not have to have full collision-resistance (property (3)), since Frank the Forger does not have complete control over the commitment $\mathbf{w}$. (More on this in Section 9.)

```
(1) $\mathbf{c} = (000 \cdots 0)^T$, $m = 0$, $\mathbf{s}$ the list of things to shuffle, $\mathbf{h}$ a list of random bits
(2) for $i$ from 0 to length($\mathbf{c}$) $- 1$ do
    (a) while $\mathbf{h}_{[m,\ldots,m+\log_2 \text{length}(\mathbf{c})-1]} > i$ do $m = m + \log_2 \text{length}(\mathbf{c})$
    (b) $j = \mathbf{h}_{[m,\ldots,m+\log_2 \text{length}(\mathbf{c})-1]}$        [$j$ is a new pseudorandom index $\leq i$]
    (c) $\mathbf{c}_i = \mathbf{c}_j$
    (d) $\mathbf{c}_j = \mathbf{s}_i$
    (e) $m = m + \log_2 \text{length}(\mathbf{c})$
(3) return $\mathbf{c}$
```

FIGURE 3. The "inside-out Fisher-Yates shuffle"

In fact, recent work (Chen et al., 2021) suggests that the hash function only has to satisfy properties (1) and (2) in a weak sense, namely that an arbitrary preimage found can most likely be recognized as not a legitimate message. For example, to take advantage of this one could require that messages be encoded in ASCII, and only messages that consist of all capital letters be considered legitimate.

In particular, instead of a traditional hash function I decided to use a simple *diffusion box*, or D-box. This is designed only to get a thorough mixing of the message with the commitment. I also chose not to try to make an extendable-output function, but rather use a fixed-length output and abort if it was not long enough. (The chance of this abort needs to be added to the chance discussed above of aborting based on large coefficients of $\mathbf{z_1}$ and $\mathbf{z_2}$.) Since the message and commitment are both in the form of vectors, I could now use the relatively simple D-box

$$D(M, \mathbf{w}) = \left\lfloor \frac{(2M + \mathbf{1}) \cdot (2\mathbf{w} + \mathbf{1})}{2} \right\rfloor \operatorname{MOD} 2^d,$$

where $\mathbf{1}$ is the vector of all 1's and d is a parameter indicating the number of bits of output. This is not cryptographically secure, but it does have the desirable property that changing one bit of the input affects every bit of the output, in a way that is at least somewhat non-trivial to predict, and the output is distributed more evenly than a simple dot product. (See Appendix C.) There are a number of ways this D-box could be improved; Exercise 6 explores one of them. You might challenge your students to think of others.

The first τ bits of D-box output are used to determine the signs of the non-zero coefficients, and the rest of the bits are fed into the Fisher-Yates shuffle. The resulting algorithm is shown in Figure 4.

Example 2 (Hashing, continuation of Example 3)**.** Recall that Aretha wanted to send Bernie the message "hola", hashed with the commitment $\mathbf{w} = (A\mathbf{y_1} + \mathbf{y_2}) \operatorname{MOD} 41 = (16\ 8\ {-6}\ 2)^T$. We will use a 6-bit output ($d = 6$), and $\tau = 1$. Aretha encodes the message as $M = (8\ 15\ 12\ 1)$ and calculates

$$D(M, \mathbf{w}) = \left\lfloor \frac{(2M + \mathbf{1}) \cdot (2\mathbf{w} + \mathbf{1})}{2} \right\rfloor \operatorname{MOD} 2^d = 30 = (011110)_2$$

Since $\tau = 1$, and we want a hash vector $\mathbf{c}$ with 4 entries, we will start with the second bit and take $\log_2 4 = 2$ bits at a time. We start the counter at $i = 3$. The second and third bit of the D-box give $(11)_2 = 3$, so we swap position 3 with position 3 (zero-based) of $\mathbf{c} = (0\ 0\ 0\ 0)^T$.

```
(1) $\mathbf{c} = (000\cdots 0)^T$, $m = \tau$, $\mathbf{h} = D(M, \mathbf{w})$
(2) for $i$ from length($\mathbf{c}$) $- \tau$ to length($\mathbf{c}$) $- 1$ do
    (a) while $\mathbf{h}_{[m,\ldots,m+\log_2 \text{length}(\mathbf{c})-1]} > i$ do $m = m + \log_2 \text{length}(\mathbf{c})$
    (b) $j = \mathbf{h}_{[m,\ldots,m+\log_2 \text{length}(\mathbf{c})-1]}$        [$j$ is a new pseudorandom index $\leq i$]
    (c) $\mathbf{c}_i = \mathbf{c}_j$
    (d) $\mathbf{c}_j = (-1)^{\mathbf{h}_{i+\tau-\text{length}(\mathbf{c})}}$        [$\mathbf{h}_{i+\tau-\text{length}(\mathbf{c})}$ is a new pseudorandom bit]
    (e) $m = m + \log_2 \text{length}(\mathbf{c})$
(3) return $\mathbf{c}$
```

FIGURE 4. The Lithium hashing algorithm

Then we use the first bit (0) to set position 3 of $\mathbf{c}$ equal to $(-1)^0$. This ends the loop, giving $\mathbf{c} = (0\ 0\ 0\ 1)^T$, as we saw earlier.

6. MODULE LEARNING WITH ERRORS

Like Kyber, the standardized version of Dilithium is based on the Module Learning With Errors (MLWE) problem rather than the basic LWE. Instead of matrices and vectors of integers, the entries of the matrices and vectors are taken from the set R of integer polynomials in x of degree less than some n, taken modulo the polynomial $x^n + 1$. (In other words, elements of the ring $R = \mathbb{Z}[x]/(x^n + 1)$. See the Alkaline paper (Holden, 2024) for a more detailed description.) Students who are familiar with AES or NTRU will probably have encountered similar polynomial modular arithmetic.

The Module Learning With Errors problem is then: given a vector $\mathbf{t}(x)$ of k polynomials in R, with the form

$$\mathbf{t}(x) \equiv A(x)\mathbf{s_1}(x) + \mathbf{s_2}(x) \pmod{x^n + 1} \pmod{q}, \tag{3}$$

where $A(x)$ is a $k \times \ell$ public matrix of polynomials in R and $\mathbf{s_2}(x)$ is a "small error vector" of k polynomials in R drawn from some probability distribution, find the secret vector of k polynomials $\mathbf{s_1}(x)$. As before, we will use also this in the matrix form

$$T(x) \equiv A(x)S_1(x) + S_2(x) \pmod{x^n + 1} \pmod{q}, \tag{4}$$

where $S_1(x)$ and $S_2(x)$ are matrices of polynomials in R with r columns in each polynomial. The whole Lithium signature scheme is shown in Figure 5. In addition to R, we 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 $a(x)t(x) \,\mathrm{MOD}(x^n + 1)\, \mathrm{MOD}\, q$ and $\boxplus$ for addition modulo q. Matrix and vector operations will also use these modular reductions.

Example 3 (Key generation for Lithium signature)**.** In this example, Aretha and Bernie will use $r = 1$, $q = 41$, $n = 4$, and 2×2 matrices for the public and private keys. We again have the "smallness" parameters $\gamma = 16$, $\beta = 1$. Aretha generates the random matrices

$$A(x) = \begin{pmatrix} 21x^3 + 25x^2 + 27x + 1 & 30x^3 + 4x^2 + 10x + 19 \\ 7x^3 + 12x^2 + 12x + 30 & 40x^3 + 7x^2 + 35x + 36 \end{pmatrix},$$

$$S_1(x) = \begin{pmatrix} x^3 - x^2 - x - 1 \\ x^2 + x + 1 \end{pmatrix}, \qquad S_2(x) = \begin{pmatrix} -x^2 - x + 1 \\ -x^2 \end{pmatrix}.$$

Aretha

- picks $k \times k$ matrix $A(x)$ with uniform random entries in R_q
- picks $k \times r$ matrices $S_1(x)$, and $S_2(x)$ with uniform random entries in R such that the coefficients are between $-\eta$ and η
- computes $T(x) = (A(x) \star S_1(x)) \boxplus S_2(x)$

- posts public key $(A(x), T(x))$
- keeps private key $(S_1(x), S_2(x))$ secret

Aretha

- converts her message M into a list of numbers
- picks vectors $\mathbf{y_1}(x)$ and $\mathbf{y_2}(x)$ with uniform random coefficients between $-(\gamma - 1)$ and $\gamma - 1$

$$\mathbf{y_1}(x), \mathbf{y_2}(x)$$

$$\downarrow A(x)$$

$$\mathbf{w} = (A(x) \star \mathbf{y_1}(x)) \boxplus \mathbf{y_2}(x)$$

$$\downarrow M$$

$$\mathbf{c}(x) = H(M, \mathbf{w})$$

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

Aretha $\rightarrow \mathbf{z_1}(x) = \mathbf{y_1}(x) + (S_1(x)\mathbf{c}(x))\,\mathrm{MOD}(x^n + 1) \rightarrow$ Bernie

$$\rightarrow \mathbf{z_2}(x) = \mathbf{y_2}(x) + (S_2(x)\mathbf{c}(x))\,\mathrm{MOD}\,(x^n + 1) \rightarrow$$

$$\rightarrow \mathbf{c}(x) \rightarrow$$

$$\rightarrow M \rightarrow$$

Bernie $(M, \mathbf{z_1}(x), \mathbf{z_2}(x), \mathbf{c}(x))$

$$\downarrow (A(x), T(x))$$

$$\mathbf{c} \overset{?}{=} \mathrm{H}(M, (A(x) \star \mathbf{z_1}) \boxplus \mathbf{z_2} \boxplus (-T(x) \star \mathbf{c}(x)))$$

- if they match (and $\mathbf{z_i}$ are small), Bernie accepts the signature

FIGURE 5. The Lithium signature scheme

She then uses these to compute the public matrix

$$T(x) = (A(x)S_1(x) + S_2(x)) \,\mathrm{MOD}\, (x^4+1) \,\mathrm{MOD}\, 41 = \begin{pmatrix} -28x^3+40x^2-34x+4 \\ 40x^3+16x^2-16x-34 \end{pmatrix}$$

and publishes $(A(x), T(x))$ as her public verification key.

Example 4 (Signing; continuation of Example 3)**.** Aretha wants to send the message "hi Bernie" to Bernie. She computes random

$$\mathbf{y_1}(x) = \begin{pmatrix} 12x^3-6x^2+6x-8 \\ -11x^3-12x^2-8x-9 \end{pmatrix} \qquad \text{and} \qquad \mathbf{y_2}(x) = \begin{pmatrix} -11x^3-6x^2-8x+2 \\ -9x^3-4x^2+6x+10 \end{pmatrix}$$

and calculates

$$\mathbf{w}(x) = (A(x)\mathbf{y_1}(x) + \mathbf{y_2}(x)) \,\mathrm{MOD}\, (x^4+1) \,\mathrm{MOD}\, 41 = \begin{pmatrix} 10x^3+12x^2+16x+3 \\ 6x^3-12x^2-17x-2 \end{pmatrix}.$$

Now Aretha needs to calculate $\mathrm{H}(M, \mathbf{w}(x))$. She takes

$$\begin{aligned} D(M,\mathbf{w}) &= D((8\ 9\ 2\ 5\ 18\ 14\ 9\ 5), (3\ 16\ 12\ 10\ -2\ -17\ -12\ 6)^T) \\ &= 62 = (111110)_2 \end{aligned}$$

(Note that we process the coefficients of polynomials starting with the constant term.) Since $\tau = 1$, and we want a hash vector $\mathbf{c}(x)$ with $n = 4$ coefficients, the hashing algorithm will give us $(0\ 0\ 0\ -1)^T$ which corresponds to $\mathbf{c}(x) = (-x^3)$.

Aretha continues calculating

$$\mathbf{z_1}(x) = \mathbf{y_1}(x) + S_1(x)\mathbf{c}(x) \,\mathrm{MOD}\, x^4+1 = \begin{pmatrix} 13x^3-5x^2+5x-9 \\ -12x^3-12x^2-7x-8 \end{pmatrix}$$

$$\mathbf{z_2}(x) = \mathbf{y_2}(x) + S_2(x)\mathbf{c}(x) \,\mathrm{MOD}\, x^4+1 = \begin{pmatrix} -12x^3-6x^2-9x+1 \\ -9x^3-4x^2+5x+10 \end{pmatrix}$$

The coefficients are all smaller than $\gamma - \beta = 15$, so the check passes. Aretha sends the message and the signature $(\mathbf{z_1}(x), \mathbf{z_2}(x), \mathbf{c}(x))$ to Bernie.

Example 5 (Verification; continuation of Example 4)**.** Bernie receives the message $M =$ `"hi Bernie"` and the signature

$$\mathbf{z_1}(x) = \begin{pmatrix} 13x^3-5x^2+5x-9 \\ -12x^3-12x^2-7x-8 \end{pmatrix}$$

$$\mathbf{z_2}(x) = \begin{pmatrix} -12x^3-6x^2-9x+1 \\ -9x^3-4x^2+5x+10 \end{pmatrix}$$

$$\mathbf{c}(x) = (-x^3)$$

He checks that the coefficients of $\mathbf{z_i}$ are smaller than $\gamma - \beta$ and calculates

$$\mathrm{H}(M, (A(x)\mathbf{z_1}(x) + \mathbf{z_2}(x) - T(x)\mathbf{c}(x)) \,\mathrm{MOD}\, (x^4+1) \,\mathrm{MOD}\, 41)$$

Since

$$(A(x)\mathbf{z_1}(x) + \mathbf{z_2}(x) - T(x)\mathbf{c}(x)) \,\mathrm{MOD}\, (x^4+1) \,\mathrm{MOD}\, 41 = \begin{pmatrix} 10x^3+12x^2+16x+3 \\ 6x^3-12x^2-17x-2 \end{pmatrix},$$

this gives the correct result:

$$\mathrm{H}\left(M, \begin{pmatrix} 10x^3 + 12x^2 + 16x + 3 \\ 6x^3 - 12x^2 - 17x - 2 \end{pmatrix}\right) = (-x^3).$$

Bernie accepts the signature.

7. DILITHIUM

So far, we have closely followed the outline from Sec. 3.1 of Günyesu, et. al. (2012, Sec. 3.1), with the addition of a generalization from single polynomials to vectors and matrices of polynomials. One problem with implementing this system at cryptographic scale is the large size of the resulting signatures, a problem common to many post-quantum algorithms. Dilithium uses a number of techniques to reduce the signature size and also speed up the computation. These fall into four categories, the first three of which it shares with Kyber:

(1) using hash functions 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 signature which are unlikely to affect the verification,
(4) checking $A\mathbf{y_1}$ and $A\mathbf{z_1} - T\mathbf{c}$ for closeness rather than equality, and thereby dispensing entirely with the vectors $\mathbf{y_2}$ and $\mathbf{z_2}$.

The comments from Holden (2024, Sec. 5) on the first three of these also apply here. In addition, note that the compression in point (3) introduces an additional abort possibility in the signing procedure if too many bits are discarded. For more details on point (4), which is based on the work of Bai and Galbraith (2014), see Appendix B. This technique also introduces an abort possibility if $A\mathbf{y_1}$ and $A\mathbf{z_1} - T\mathbf{c}$ do not come out sufficiently close.

8. PARAMETERS AND VARIANTS

The parameter sets for Lithium, like those of Alkaline, were chosen to balance ease of student use with illustration of the major features of the standardized Dilithium system. (Security was not a primary consideration, except to illustrate some of the ways in which Dilithium blocks non-obvious attacks.) In particular, the "smallness" parameters γ, η, τ, and β need some further explanation.

The primary purpose of γ is to make sure that one of the underlying LWE problems is actually hard. If γ is too large, then Frank the Forger can forge a signature by picking random $\hat{\mathbf{z}}_{\boldsymbol{i}}$ and $\hat{\mathbf{c}}$, calculating

$$\mathbf{c} = \mathrm{H}(M, ((A\hat{\mathbf{z}}_\mathbf{1} + \hat{\mathbf{z}}_\mathbf{2} - T\hat{\mathbf{c}}) \operatorname{MOD} q)),$$

and then trying to find short $\mathbf{z_i}$ such that $A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c} \equiv A\hat{\mathbf{z}}_\mathbf{1} + \hat{\mathbf{z}}_\mathbf{2} - T\hat{\mathbf{c}} \pmod{q}$. Moving the known value of $T\mathbf{c}$ to the other side, this requires solving the LWE problem. A trivial way to attempt to solve this is by picking a short $\mathbf{z_1}$ at random, solving for $\mathbf{z_2}$, and hoping it is short enough. The chance that this $\mathbf{z_2}$ will have coefficients less than $\gamma - \beta$ is

$$\left(\frac{2(\gamma - \beta) - 1}{q}\right)^{n \cdot r}.$$

This should be small enough that students don't feel like it's easy to forge a signature. Lithium aims for around 0.01. (Of course, there are better ways to solve the LWE in practice.)

On the other hand, if γ is too small, then it will take many tries to generate a successful signature. The probability that a signature generated by our procedure passes checking both $\mathbf{z_i}$ is approximately

$$\left(\left(\frac{2(\gamma-\beta)-1}{2\gamma-1}\right)^{n\cdot r}\right)^2.$$

Lithium, like the CRYSTALS team, aims for this to be around 0.25. Thus it requires about 4 tries on average to generate a signature.

The value of η likewise affects the hardness of an LWE problem, in this case the problem of recovering the secret key (S_1, S_2) from the public key (A, T). However, it also affects whether the signature scheme is zero-knowledge. Since $\mathbf{y_i} = \mathbf{z_i} - S_i\mathbf{c}$, in order for $\mathbf{z_i}$ not to reveal information about $\mathbf{y_i}$, the range of coefficients of $\mathbf{z_i} - S_i\mathbf{c}$ should match the range of coefficients of $\mathbf{y_i}$ (Bai et al., 2021, App. B). The coefficients of S_i have absolute value less than or equal to η, and $\mathbf{c}$ is chosen to have exactly τ coefficients with absolute value 1 (and the rest zero), so the coefficients of $S_i\mathbf{c}$ have absolute value less than or equal to $\beta = \eta \cdot \tau$. Thus the coefficients of $\mathbf{z_i}$ should have absolute value less than or equal to $\gamma - \beta$.

The value of τ affects β as above, but also determines the number of possible outputs of the hash function. If this is too small, it will be easy to forge messages by guessing M and/or $\mathbf{w}$ that will match a given $\mathbf{c}$. The number of possible outputs of the hash function is $2^\tau\binom{n\cdot r}{\tau}$. The base-2 logarithm of this number is known as the "entropy" of the output, and indicates the average number of bits of output from the D-box or XOF which are actually used. This is the weakest security link in the Lithium parameter sets given. As can be seen from Table 1, the number of possible outputs is quite small. It is not clear how to improve this without making the computation very difficult to do by hand.

Finally note that the D-box parameter d is subject to three constraints. It should be larger than the range of possible numbers in the message list M and also larger than q, in order to avoid trivial collisions in the D-box involving only M or only $\mathbf{w}$. In addition, it should be large enough that the Fisher-Yates shuffle does not need to abort too often. This chance can be calculated from τ and d in a relatively straightforward way, especially when τ is small. (Note that if $\tau = 1$, then the shuffle will always use the minimum number of bits.) To make computations easier, we have tried to minimize d subject to these constraints. (However, note the observation in Section 5 that increasing d — i.e., increasing the range of preimages of the hash — relative to the range of possible numbers in M increases the security of the D-box.) With these constraints in mind, some options for parameter choices are shown in Table 1 with comparison to the options for the standardized versions of Kyber. We also list the expected number of repetitions (including the initial attempt) caused by the potential aborts, and the chance that Frank can create a "lucky" forgery by naively solving for $\mathbf{z_2}$ and hoping it is short.

Each of the listed Lithium parameter sets has strengths and weaknesses:

- Lithium N has a small prime modulus and this is easy to work with by hand, and has a low chance of a "lucky" forgery. The number of expected repetitions, however, is frustratingly high for a by-hand cipher.

r is the number of entries in the challenge vector											
n is the number of coefficients in each polynomial											
k is the number of rows in the matrix A											
ℓ is the number of columns in the matrix A											
q is the modulus											
η is the maximum absolute value of the nonce vectors $\mathbf{y_i}$											
d is the number of bits of output of the D-box											
entropy of $\mathbf{c}$ is the log base 2 of the number of possible values for $\mathbf{c}$											
"lucky" forgeries come from naively solving for $\mathbf{z_2}$ and hoping it is short											
									entropy	expected	"lucky"
	r	n	(k,ℓ)	q	τ	η	γ	d	of $\mathbf{c}$	repetitions	forgeries
Lithium N	1	4	(2,2)	41	1	1	2^3	6	3.00	9.87	0.01%
Lithium AAA	1	4	(2,2)	41	1	1	2^4	6	3.00	2.91	6.26%
Lithium AA	1	4	(2,2)	97	1	1	2^4	7	3.00	2.91	0.01%
Lithium C	1	4	(2,2)	97	2	1	2^4	8	4.58	9.73	0.004%
Lithium D	1	4	(2,2)	193	2	1	2^5	8	4.58	3.05	0.009%
Lithium LA	4	1	(4,4)	41	1	1	2^4	6	3.00	3.14	1.01%
Lithium ALG	1	4	(1,1)	41	1	1	2^3	6	3.00	3.14	1.01%
ML-DSA-44	1	256	(4,4)	8380417	39	2	2^{17}	∞	192	4.25*	$< 10^{-1500}$
ML-DSA-65	1	256	(6,5)	8380417	49	4	2^{19}	∞	225	5.5*	$< 10^{-1000}$
ML-DSA-87	1	256	(8,7)	8380417	60	2	2^{19}	∞	257	3.85*	$< 10^{-1500}$
*The expected number of repetitions for Dilithium depends on the aborts from Section 7 as well as from the Fiat-Shamir with Aborts technique.											

Table 1. Parameter sets for Lithium and Dilithium

- Lithium AAA is also easy to use by hand and has a reasonable number of expected repetitions, but the chance of a "lucky" forgery is really higher than it should be. Nevertheless, this appears to be the best option for many purposes.
- Lithium AA has a good number of expected repetitions and a small chance of "lucky" forgeries, at the expense of a larger prime which may be more difficult to work with. (Note that primes that are near round decimal numbers were chosen in all cases in order to try to ease by-hand calculations.)
- Lithium C introduces a value of τ larger than 1, which is important because the Fisher-Yates shuffle has to go through the loop more than once and there is now a possibility that the shuffle will need more than the minimum number of bits from the D-box. However, the number of expected repetitions is very large.
- Lithium D achieves all of the goals mentioned above, which the exception of requiring a rather large prime for a by-hand calculation.

Table 1 also includes two special-purpose versions of Lithium:

- Lithium LA is intended to be suitable for use in linear algebra classes where the instructor does not want to take the time to explain polynomial modular arithmetic. Taking $n = 1$ reduces the polynomials to constants and thus effectively results in a matrix-based LWE system, like the one sketched in Section 4.

- Lithium ALG is intended to be suitable for use in abstract algebra or cryptography classes where students may not be familiar with matrices. Taking $k = 1$ reduces the matrices to scalars from the polynomial ring. (This is essentially the signature scheme from Sec. 3.1 of Günyesu, et. al. (2012, Sec. 3.1))

  As an alternative to Lithium ALG, one could rewrite the $(k, \ell) = (2, 2)$ versions of Lithium in terms of systems of two equations in two variables without much trouble. (I have used this method to teach the 2×2 Hill cipher without matrices, for example.)

One caveat: I tested Alkaline AA, which uses similar parameters to Lithium AAA in my Spring 2025 cryptography class. I discovered that students made many careless mistakes simply in the multiplication of degree 3 polynomials, even putting aside the polynomial reduction. In the hopes of taking away some of the burden of computation while maintaining the basic ideas, one might try changing the parameters from $r = 1$, $n = 4$ to $r = 2$, $n = 2$. Some analysis of this is provided in Appendix A. Note that this has not yet been tested in class.

There are also two earlier versions of Lithium that I have presented in talks and classes. The one described in this paper I call Lithium Hash-Free. Another, which I call Lithium Standard, uses a standard hash function or XOF instead of a D-box. Using SHAKE256, like in standardized Dilithium, would be a reasonable choice. I have also used the obsolete SHA-1 (Schneier, 1996, Sec. 18.7) hash function in class when it was the most advanced option built into the computer algebra system I was using. (Theoretically, SHA-1 could run out of bits in the Fisher-Yates shuffle, but the chance is unbelievably small — calculating it might be a good exercise in probability!) As mentioned in Section 5, using a true hash function is more reasonable when the cipher is completely automated.

The third version I call Lithium Advanced. It uses the Bai-Galbraith technique mentioned in Section 7 and is described in Appendix B.

9. ATTACKS ON LITHIUM

There are a number of ways that Frank could attempt to forge a message from Aretha. The most obvious is to try to recover her secret key (S_1, S_2) from the public information A and $T = (AS_1 + S_2) \operatorname{MOD} q$. This is the (M)LWE problem, and the currently best-known attack is the "primal" attack described in Holden (2024, Sec. 7). This could proceed column by column, regarding each column of S_1 as the $\mathbf{s}$ vector and each column of S_2 as the $\mathbf{e}$ vector.

Keep in mind that Frank does not actually need S_1 and S_2, since these are never revealed to Bernie. Likewise, he does not actually need to create $\mathbf{y_1}$ and $\mathbf{y_2}$. He merely needs to find $\mathbf{z_1}$, $\mathbf{z_2}$, and $\mathbf{c}$ such that

$$\mathbf{c} = \mathrm{H}(M, (A\mathbf{z_1} + \boldsymbol{z_2} - T\mathbf{c}) \operatorname{MOD} q).$$

One option is to choose $\mathbf{c}$ at random, try to find $\mathbf{w}$ such that $\mathbf{c} = \mathrm{H}(M, \mathbf{w})$, and then try to find $\mathbf{z_i}$ such that $(A\mathbf{z_1} + \boldsymbol{z_2} - T\mathbf{c}) \equiv \mathbf{w} \pmod{q}$. The first part of this requires finding a preimage of the hash function, which is supposed to be hard by property (1). (Although it is not actually that hard for the given Lithium parameter sets, as explained in Section 8.)

Alternatively, Frank could pick random $\hat{\mathbf{z}}_{\boldsymbol{i}}$ and $\hat{\mathbf{c}}$ (or steal them from a valid signature). Then he could calculate

$$\mathbf{c} = \mathrm{H}(M, (A\hat{\mathbf{z}}_{\mathbf{1}} + \hat{\mathbf{z}}_{\mathbf{2}} - T\hat{\mathbf{c}}) \operatorname{MOD} q),$$

and try to find short $\mathbf{z_i}$ such that

$$\mathbf{c} = \mathrm{H}(M, (A\mathbf{z_1} + \boldsymbol{z_2} - T\mathbf{c}) \,\mathrm{MOD}\, q).$$

This requires finding a second preimage of the hash function, which is supposed to be hard by property (2).

Yet another possibility is for Frank to pick random (or stolen) $\hat{\mathbf{z}}_{\boldsymbol{i}}$ and $\hat{\mathbf{c}}$, calculate

$$\mathbf{c} = \mathrm{H}(M, (A\hat{\mathbf{z}}_{\mathbf{1}} + \hat{\mathbf{z}}_{\mathbf{2}} - T\hat{\mathbf{c}}) \,\mathrm{MOD}\, q),$$

and try to find short $\mathbf{z_i}$ such that $A\mathbf{z_1} + \mathbf{z_2} - T\mathbf{c} \equiv A\hat{\mathbf{z}}_{\mathbf{1}} + \hat{\mathbf{z}}_{\mathbf{2}} - T\hat{\mathbf{c}} \pmod{q}$. This again requires solving the LWE problem, as discussed in Section 8.

10. DISCUSSION AND CONCLUSION

The cryptography course I taught in the Spring of 2025 was my first chance to use the ENERCELL project, and in fact the first one I have taught since NIST announced its first standardizations. Assessment in my cryptography courses is primarily done through weekly written homework and programming problems. For the last weekly assignment of the course, I included 7 exercses dealing with Alkaline, and Exercises 1, 4, and 5 from Appendix D. (The other Exericses were unfortunately not ready in time for the course, but they are similar to the Alkaline exercises (Holden, 2024, App. A).) The code used to generate the Exercises in Appendix D is available on GitHub (`?redacted`), and this can also be used to check answers.

The structure of my course allows for a great amount of freedom of which homework problems students choose to do, so not all students chose to do all problems. As it happened, no students chose to do Exercise 1 or 4. Many students did Exercise 5. Most succeeded in correctly hashing when $\tau = 1$, but only 1 of the 8 students who tried the problem successfully produced hashes with $\tau = 2$. (Ten students in total finished the class.) Although I gave them the general algorithm, the only examples I gave were for $\tau = 1$. Perhaps more students would have been successful with examples, or at least test data, for $\tau = 2$. I intend to add at least the test data for next time.

Many students chose to do some or all of the Alkaline exercises, and most were fairly successful. The two main issues were computational mistakes, especially with polynomial algebra, and forgetting to do moduli computations in the correct places. As I mentioned in Section 8, it might be good to change parameters to reduce the degree of the necessary polynomials.

As part of the Student Evaluations of Teaching at the end of the quarter, I asked the students to note whether they found useful the simplified cryptosystems we had looked at, such as S-AES, Alkaline, and Lithium. Of the 9 students who filled out the evaluations, 3 students said they thought that the systems were helpful and 1 said that they were not. The other 5 did not respond to that prompt. I hope to run the class again in the future and try to get a better response rate.

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. 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 formal linear algebra class. The combination of all of the things that go into Dilithium gives students a good idea of the wide range of mathematical techniques that go into modern cryptography!

References

1. Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. 2021. CRYSTALS-Dilithium. (Version 3.1): 38. ↑8, B
2. Bai, S. and Galbraith, S. D. 2014. An Improved Compression Technique for Signatures Based on Learning with Errors. In *Topics in Cryptology — CT-RSA 2014*, ed. Benaloh, J., 28–47. Cham: Springer. DOI 10.1007/978-3-319-04852-9_2. ↑7, B
3. Chen, Y., Lombardi, A., Ma, F., and Quach, W. 2021. Does Fiat-Shamir Require a Cryptographic Hash Function? In *Advances in Cryptology — CRYPTO 2021*, ed. Malkin, T. and Peikert, C., 334–363. Cham: Springer. DOI 10.1007/978-3-030-84259-8_12. ↑5
4. Fiat, A. and Shamir, A. 1987. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, ed. Odlyzko, A. M., 186–194. Berlin, Heidelberg: Springer Berlin Heidelberg. DOI 10.1007/3-540-47721-7_12. ↑3
5. Güneysu, T., Lyubashevsky, V., and Pöppelmann, T. 2012. Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems. In *Cryptographic Hardware and Embedded Systems — CHES 2012*, ed. Prouff, E. and Schaumont, P., 530–547. Berlin, Heidelberg: Springer. DOI 10.1007/978-3-642-33027-8_31. ↑7, 8
6. Hoffstein, J., Pipher, J., and Silverman, J. H. 2014. *An Introduction to Mathematical Cryptography*, 2nd ed., Undergraduate Texts in Mathematics, New York, NY: Springer. ↑4
7. 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. ↑1
8. Holden, J. 2013. A Good Hash Function is Hard to Find, and Vice Versa. Cryptologia. 37: 107–119. ↑1, 3, 5
9. Holden, J. 2018. *The Mathematics of Secrets: Cryptography from Caesar Ciphers to Digital Encryption*, paperback edition, Princeton, NJ: Princeton University Press. ↑2
10. Holden, J. 2023. Resource guide for teaching post-quantum cryptography. Cryptologia. 47(5): 459–465. DOI 10.1080/01611194.2022.2078077. ↑2
11. Holden, J. 2024. Alkaline: A Simplified Post-Quantum Encryption Algorithm for Classroom Use. PRIMUS. 34(1): 98–122. DOI 10.1080/10511970.2023.2235696. ↑1, 2, 6, 7, 9, 10, A
12. Holden, J. 2025. *Rejection Sampling Demonstration.* `https://enercell.github.io/rejection-sampling-demo.html`. ↑4
13. Holden, J. forthcoming. The Fiat-Shamir: Zero(-Knowledge) to Signature in Sixty Minutes (of Class Time). Cryptologia. DOI 10.1080/01611194.2025.2547602. ↑3
14. Know Your Meme. 2009. *Aretha's Hat*, `https://knowyourmeme.com/memes/arethas-hat`. ↑3
15. Know Your Meme. 2021. *Bernie Sanders Wearing Mittens Sitting in a Chair*, `https://knowyourmeme.com/memes/bernie-sanders-wearing-mittens-sitting-in-a-chair`. ↑3
16. Knuth, D. E. *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed., Reading, Mass: Addison-Wesley. ↑5
17. Lyubashevsky, V. 2009. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *Advances in Cryptology — ASIACRYPT 2009*, ed. Matsui, M., 598–616. Berlin, Heidelberg: Springer. DOI 10.1007/978-3-642-10366-7_35. ↑3, 4
18. 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. ↑1
19. National Institute of Standards and Technology. 2024. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standard (FIPS) 204. DOI 10.6028/NIST.FIPS.204. ↑4
20. Neven, G., Smart, N. P., and Warinschi, B. 2009. Hash function requirements for Schnorr signatures. Journal of Mathematical Cryptology. DOI 10.1515/JMC.2009.004. ↑5
21. Paar, C., Pelzl, J., and Güneysu, T. 2024. *Understanding Cryptography: From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms*, Berlin, Heidelberg: Springer. ↑5
22. Schaefer, E. F. 1996. A Simplified Data Encryption Standard Algorithm. Cryptologia. 20(1): 77–84. DOI 10.1080/0161-119691884799. ↑1
23. Schneier, B. 1996. *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd ed., John Wiley & Sons, Inc. ↑5, 8

24. Schnorr, C. P. 1991. Efficient signature generation by smart cards. Journal of Cryptology. 4(3): 161–174. DOI 10.1007/BF00196725. ↑5

Appendix A. ALTERNATE PARAMETERS

As mentioned in Section 8, one might want to reduce the computational burden of the Lithium system by changing the parameters from $r = 1$, $n = 4$ to $r = 2$, $n = 2$. One issue is that while the security of the LWE problems goes up as k and n go up, it goes slightly **down** as r goes up. Thus the $(r, n) = (2, 2)$ versions are less secure than the $(r, n) = (1, 4)$ versions. The security of the hash function, however, goes up as both r and n go up. This is why I suggest $(r, n) = (2, 2)$ rather than $(1, 2)$. The CRYSTALS team, on the other hand, chose $r = 1$ for all versions, since the MLWE problem was a weaker security link for them than the hash function.

Some possible parameter sets for $(r, n) = (2, 2)$ are presented in Table 2. (I call this series Lithium-22.) In all but one case, no changes to the other parameters need to be made; the number of expected repetitions goes down while the chance of a "lucky" forgery stays acceptable. In the case of Lithium AAA, however, the change of a "lucky" forgery would rise to 25.0%, which does not seem acceptable. Changing q to 61 reduces this chance to a level comparable to Lithium AAA14 (from the original series). I would nevertheless recommend Lithium N22 over AAA22 for most purposes, as the number of expected repetitions has decreased dramatically from Lithium N14.

One could do make similar changes for Alkaline N, AA, C, and D from Holden (2024). Note that neither the Alkaline-22 nor the Lithium-22 series has yet been tested in class.

	r	n	(k, ℓ)	q	τ	η	γ	d	entropy of of $\mathbf{c}$	expected repetitions	"lucky" forgeries
Lithium N22	2	2	(2,2)	41	1	1	2^3	6	3.00	3.14	1.01%
Lithium AAA22	2	2	(2,2)	61	1	1	2^4	6	3.00	1.70	5.11%
Lithium AA22	2	2	(2,2)	97	1	1	2^4	7	3.00	2.91	0.80%
Lithium C22	2	2	(2,2)	97	2	1	2^4	8	4.58	4.03	0.60%
Lithium D22	2	2	(2,2)	193	2	1	2^5	8	4.58	2.25	0.87%

Table 2. Alternate parameter sets for Lithium

Appendix B. LITHIUM ADVANCED

As mentioned in Sections 7 and 8, I have also developed a version of Lithium using the Bai-Galbraith technique (Bai and Galbraith, 2014). After presenting it in a few talks, I decided that while it was closer to standardized Dilithium, the extra complication was generally not worth it. However, I present it here for advanced students. This version follows the outline in (Bai et al., 2021, Fig. 1).

To sign a message of r bits, the private key is a pair of matrices (S_1, S_2) and the public key is (A, T), as before. Aretha chooses a single random nonce vector $\mathbf{y}$ with small coefficients, and calculates

$$\mathbf{c} = \mathrm{H}(M, \mathrm{HighBits}(A\mathbf{y} \,\mathrm{MOD}\, q, \gamma_2))$$

where $\mathrm{HighBits}(r, \alpha)$ and $\mathrm{LowBits}(r, \alpha)$ are defined by

$$r = \mathrm{HighBits}(r, \alpha) \cdot \alpha + \mathrm{LowBits}(r, \alpha), \qquad 0 \le \mathrm{LowBits}(r, \alpha) < \alpha$$

for nonnegative integers r, and similarly for vectors of nonnegative integers. (When α is a power of 2, this actually corresponds to taking high order and low order bits. Otherwise, it can be thought of as high order and low order digits in some non-binary base, or as division with remainder.)

Aretha then computes

$$\mathbf{z} = \mathbf{y} + S_1\mathbf{c}$$

If any coefficient of $\mathbf{z}$ is outside an interval $\{-(\gamma_1 - \beta - 1), \dots, \gamma_1 - \beta - 1\}$ or any coefficient of

$$\mathrm{LowBits}((A\mathbf{y} - S_2\mathbf{c}) \,\mathrm{MOD}\, q, 2\gamma_2)$$

is outside $\{-(\gamma_2 - \beta - 1), \dots, \gamma_2 - \beta - 1\}$, then Aretha starts over with a new $\mathbf{y}$. Otherwise, she sends Bernie the message and the signature $(\mathbf{z}, \mathbf{c})$.

Bernie verifies the signature by first checking to make sure $\mathbf{z}$ is inside the correct interval. If so, he calculates

$$\mathbf{c}' = \mathrm{H}(M, \mathrm{HighBits}((A\mathbf{z_1} - T\mathbf{c}) \,\mathrm{MOD}\, q, 2\gamma_2)).$$

Now

$$\begin{aligned} A\mathbf{z} - T\mathbf{c} &\equiv A\left(\mathbf{y} + S_1\mathbf{c}\right) - \left(AS_1 + S_2\right)\mathbf{c} \pmod q \\ &\equiv \left(A\mathbf{y} + AS_1\mathbf{c}\right) - \left(AS_1\mathbf{c} + S_2\mathbf{c}\right) \pmod q \\ &\equiv A\mathbf{y} - S_2\mathbf{c} \pmod q, \end{aligned}$$

which differs from $A\mathbf{y}$ by $S_2\mathbf{c}$. However, the coefficients of $S_2\mathbf{c}$ are smaller than or equal to β, so if $\mathrm{LowBits}(A\mathbf{y} - S_2\mathbf{c}, 2\gamma_2)$ is in the correct interval, then the difference won't overflow into the "HighBits" parts of $A\mathbf{y}$ and $A\mathbf{z} - T\mathbf{c}$. Thus, $\mathbf{c}'$ should be equal to the $\mathbf{c}$ in the signature. If it is, then Bernie accepts the signature as valid. Figure 6 summarizes this process.

Example 6 (The Lithium Advanced signature scheme)**.** Aretha and Bernie agree to use Lithium Advanced with $r = 4$, $n = 1$, $q = 41$, and 4×4 matrices for the public and private keys. We also have the "smallness" parameters $\eta = 1$ (for S_i), $\gamma_1 = 16$ (for $\mathbf{y}$), $\gamma_2 = 10$ (for HighBits), and $\tau = \eta = \beta = 1$. We will use SHAKE256 for an XOF.

Aretha first generates the random matrices

$$A = \begin{pmatrix} 4 & 14 & 4 & 14 \\ 27 & 28 & 8 & 40 \\ 9 & 5 & 8 & 19 \\ 8 & 9 & 14 & 3 \end{pmatrix}, \qquad S_1 = \begin{pmatrix} -1 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & -1 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

$$S_2 = \begin{pmatrix} 0 & -1 & -1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

She then uses these to compute the public matrix

$$T = (AS_1 + S_2) \,\mathrm{MOD}\, 41 = \begin{pmatrix} 27 & 9 & 36 & 22 \\ 24 & 19 & 14 & 21 \\ 21 & 37 & 33 & 23 \\ 3 & 37 & 33 & 30 \end{pmatrix}$$

and publishes (A, T) as her public verification key.

Aretha

- picks $k \times k$ matrix A with uniform random entries between 0 and $q-1$
- picks $k \times r$ matrices S_1, and S_2 with uniform random entries between $-\eta$ and η
- computes $T = (AS_1 + S_2) \,\mathrm{MOD}\, q$

- posts public key (A, T)
- keeps private key (S_1, S_2) secret

Aretha

- converts her message M into a list of numbers
- picks vector $\mathbf{y}$ with uniform random entries between $-(\gamma_1 - 1)$ and $\gamma_1 - 1$

$$\mathbf{y}$$

$$\downarrow A$$

$$\mathbf{w_1} = \mathrm{HighBits}(A\mathbf{y} \,\mathrm{MOD}\, q, 2\gamma_2)$$

$$\downarrow M$$

$$\mathbf{c} = H(M, \mathbf{w_1})$$

$$\downarrow \mathbf{c}$$

Aretha $\rightarrow \mathbf{z} = \mathbf{y} + S_1\mathbf{c} \rightarrow$ Bernie

$$\rightarrow \mathbf{c} \rightarrow$$

$$\rightarrow M \rightarrow$$

Bernie $(M, \mathbf{z}, \mathbf{c})$

$$\downarrow (A, T)$$

$$\mathbf{c} \overset{?}{=} \mathrm{H}(M, \mathrm{HighBits}((A\mathbf{z} - T\mathbf{c}) \,\mathrm{MOD}\, q, 2\gamma_2))$$

- if they match (and $\mathbf{z}$ is small), Bernie accepts the signature

FIGURE 6. The Lithium Advanced signature scheme

Aretha wants to send the message "hi!" to Bernie. She generates a random

$$\mathbf{y} = (-3\ 0\ \ -1\ 14)^T$$

with coefficients between $-\gamma_1$ and γ_1 and calculates $\mathbf{w} = (A\mathbf{y_1})\,\mathrm{MOD}\,41 = (16\ 20\ 26\ 4)^T$.

$$\mathbf{w} = (16\ 20\ 26\ 4)^T = (1\ 1\ 1\ 0)^T \cdot 2\gamma_2 + (-4\ 0\ 6\ 4)^T,$$

so $\mathbf{w_1} = (1\ 1\ 1\ 0)^T$.

We represent $(M, \mathbf{w_1})$ as an ASCII string `"hi!<1,1,1,0>"`. (The exact representation does not matter as long as Aretha and Bernie agree.) Passing this to SHAKE256 gives a binary string starting with $(01111101)_2$, and then the Fisher-Yates shuffle gives $\mathbf{c} = (0\ 0\ 0\ 1)^T$.

Aretha continues calculating

$$\mathbf{z} = \mathbf{y} + S_1\mathbf{c} = (-4\ 0\ 0\ 13)^T$$

The coefficients are smaller (in absolute value) than $\gamma_1 - \beta = 15$, so the first check passes.

Aretha continues calculating

$$(A\mathbf{y} - S_2\mathbf{c})\,\mathrm{MOD}\,41 = (16\ 21\ 25\ 5)^T = (1\ 1\ 1\ 0)^T \cdot 2\gamma_2 + (-4\ 1\ 5\ 5)^T.$$

The low-order bits of $A\mathbf{y} - S_2\mathbf{c}$ are smaller than $\gamma_2 - \beta = 9$, so the second check passes. Aretha sends the message and the signature $(\mathbf{z}, \mathbf{c})$ to Bernie.

Finally, Bernie receives the message $M =$ `"hi!"` and the signature

$$\mathbf{z} = (-4\ 0\ 0\ 13)^T, \qquad \mathbf{c} = (0\ 0\ 0\ 1)^T.$$

He checks that the coefficients of $\mathbf{z}$ are smaller than $\gamma_1 - \beta$ and calculates

$$\mathrm{H}(M, (A\mathbf{z_1} - T\mathbf{c})\,\mathrm{MOD}\,41)$$

Since $(A\mathbf{z} - T\mathbf{c})\,\mathrm{MOD}\,41 = (16\ 21\ 25\ 5)^T$, $\mathrm{HighBits}((A\mathbf{z} - T\mathbf{c})\,\mathrm{MOD}\,41) = (1\ 1\ 1\ 0)^T$, and Bernie gets the correct result:

$$\mathrm{H}(M, (1\ 1\ 1\ 0)^T) = (0\ 0\ 0\ 1)^T.$$

Bernie accepts the signature.

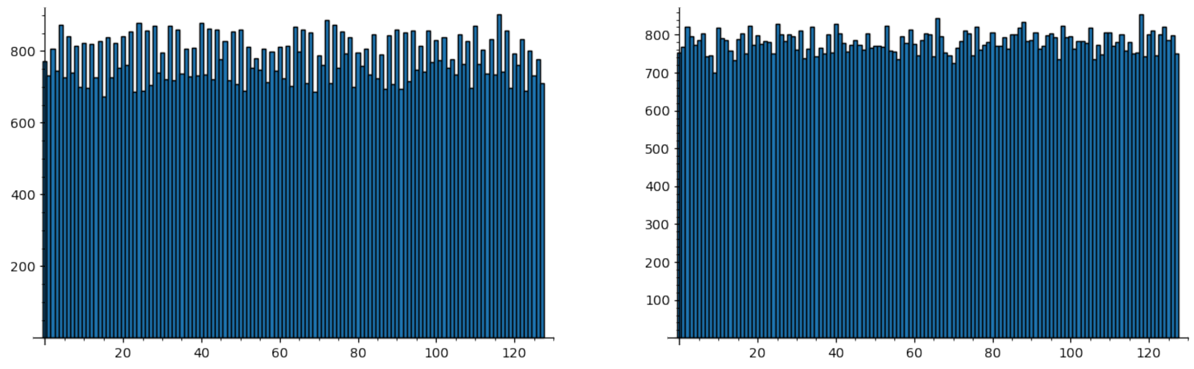
Appendix C. STATISTICAL ANALYSIS OF D-BOXES

The D-box presented in Section 5 may not seem very intuitive. My goal was a function which acted on integers from 1 to 26 and 0 to $q-1$, satisfied hash function properties (a) and (b), satisfied an avalanche criterion such that every bit of input generally affected every bit of output, used more than one arithmetic operation (for example addition and multiplication), and was relatively easy to describe to students. My first thought was to simply use the dot product modulo 2^d. I constructed a graph of the outputs from 100000 random choices of $(M, \mathbf{w})$ where M was chosen from $\{1, \ldots, 26\}^4$ and $\mathbf{w}$ was chosen from $\{0, \ldots, q-1\}^7$ for $q = 97$. As you can see in Figure 7a, the outputs are not in fact distributed evenly.

A bit of thought reveals that 3 out of every 4 randomly chosen inputs to the function

$$(a, b) \mapsto (a \cdot b)\,\mathrm{MOD}\,2^d$$

produces an even answer, which similar but smaller biases for other powers of 2 up to 2^d. Essentially, multiplying by an even number modulo 2^d is not a permutation. On the other



(a) $(M, \mathbf{w}) \mapsto (M \cdot \mathbf{w}) \,\mathrm{MOD}\, 2^7$ (b) $(M, \mathbf{w}) \mapsto D(M, \mathbf{w}) \,\mathrm{MOD}\, 2^7$

FIGURE 7. Distribution of 100000 random outputs of the given functions

hand, multiplying by an odd number modulo 2^d is a permutation, even if the domain is also restricted to odd numbers. Therefore the function

$$(a, b) \mapsto ((2a+1) \cdot (2b+1)) \,\mathrm{MOD}\, 2^d$$

distributes its outputs evenly across odd integers, the map

$$(M, \mathbf{w}) \mapsto ((2M + \mathbf{1}) \cdot (2\mathbf{w} + \mathbf{1})) \,\mathrm{MOD}\, 2^d$$

distributes its outputs evenly across integers modulo 2^d with parity equal to that of $\mathbf{1} \cdot \mathbf{1} = \mathrm{length}(M)$, and

$$D(M, \mathbf{w}) = \left\lfloor \frac{(2M + \mathbf{1}) \cdot (2\mathbf{w} + \mathbf{1})}{2} \right\rfloor \mathrm{MOD}\, 2^d$$

should distribute its outputs evenly across all integers modulo 2^d. Figure 7b shows a graph of the outputs of $D(M, \mathbf{w})$ from 100000 random choices of $(M, \mathbf{w})$ taken in the same way as above. A chi-squared test fails to detect any significant deviation from uniformity for this data set, with a significance value of $p \approx 0.522$.

It is possible to rewrite the D-box formula as

$$D(M, \mathbf{w}) = \left(2(M \cdot \mathbf{w}) + \sum_i M_i + \sum_i w_i + \lfloor d/2 \rfloor \right) \mathrm{MOD}\, 2^d.$$

This would probably be slightly faster computationally, but it seems considerably more difficult to motivate.

Exercise 1. Suppose you are signing a message using Lithium LA. The XOF of $(M, \mathbf{w})$ is "57adb935248f60745b1d954bebbe30872e0e603b" in hexadecimal. Find the vector $\mathbf{c}$. (Remember that $\tau = 1$ and the length of $\mathbf{c}$ is 4.)

Exercise 2. You are An, and you would like to sign the message "Love" using Lithium LA. You pick the public matrix

$$A = \begin{pmatrix} 29 & 22 & 18 & 5 \\ 19 & 12 & 40 & 19 \\ 12 & 27 & 2 & 20 \\ 24 & 24 & 18 & 13 \end{pmatrix}$$

and the private key

$$S_1 = \begin{pmatrix} -1 & -1 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ 0 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \end{pmatrix}, \qquad S_2 = \begin{pmatrix} -1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 1 \\ 1 & 0 & 1 & 0 \\ -1 & 0 & 1 & -1 \end{pmatrix}$$

(a) What is the other public matrix T?
(b) Suppose you pick the random nonce vectors

$$\mathbf{y_1} = (6\ 12\ 10\ -5)^T, \qquad \mathbf{y_2} = (-12\ -11\ -11\ -13)^T$$

Do $\mathbf{z_1}$ and $\mathbf{z_2}$ pass the size checks?
(c) What is the signature using the above $\mathbf{y_1}$ and $\mathbf{y_2}$?

Exercise 3. You are Botan, and you get the message "Life", supposedly from Ayame. Her public key is

$$A = \begin{pmatrix} 12 & 16 & 3 & 10 \\ 27 & 24 & 25 & 20 \\ 15 & 16 & 20 & 4 \\ 36 & 10 & 9 & 29 \end{pmatrix}, \qquad T = \begin{pmatrix} 13 & 11 & 6 & 0 \\ 18 & 33 & 22 & 23 \\ 4 & 25 & 20 & 33 \\ 3 & 34 & 24 & 40 \end{pmatrix}$$

(a) If the signature on the message is

$$\mathbf{z_1} = \begin{pmatrix} 5 & -3 & 13 & 10 \end{pmatrix}^T, \qquad \mathbf{z_2} = \begin{pmatrix} -11 & -2 & 3 & 1 \end{pmatrix}^T, \qquad \mathbf{c} = \begin{pmatrix} -1 & 0 & 0 & 0 \end{pmatrix}^T$$

does the signature verify?
(b) If the signature on the message is

$$\mathbf{z_1} = \begin{pmatrix} 13 & -11 & 8 & -12 \end{pmatrix}^T, \qquad \mathbf{z_2} = \begin{pmatrix} -1 & -1 & 12 & 6 \end{pmatrix}^T, \qquad \mathbf{c} = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$$

does the signature verify?
(c) Is it possible for more than one signature to correctly verify on the same message? Why or why not?

Exercise 4. You are Frank, and you are trying to forge a message from Aretha, using the SIS attack. Aretha's public key is in Example 1. You steal the signature from Aretha's message in that example for $\hat{\mathbf{z}}_1$, $\hat{\mathbf{z}}_1$, and $\hat{\mathbf{c}}$, and hash your message to get $\mathbf{c} = \langle 0, 1, 0, 0 \rangle$. Show that it is easy to forge a message if Bob does not check the size of the signature by solving for a $\mathbf{z_1}$ and $\mathbf{z_2}$ which don't necessarily have small coefficients.

Exercise 5. Create software that implements the hashing algorithm from Section 5. Your program should input a string representing $(M, \mathbf{w})$ (you do not have to check the format of the string), a length of $\mathbf{c}$, and a value of τ, and output a value of $\mathbf{c}$.

Exercise 6. One way to improve the D-box in Lithium Hash-Free would be to take entries for $\mathbf{w}$ in the range $\{1, \dots, q\}$ instead of $\{0, \dots, q-1\}$.

(a) Why would this improve the security of the D-box?
(b) What fraction of messages would you expect to be affected by this issue?

Exercise 7. You are Ayame, and you would like to sign the message "OpenDoor" using Lithium AAA. You pick the public matrix

$$A(x) = \begin{pmatrix} 20x^3 + 17x^2 + 6x + 8 & 10x^3 + 39x^2 + 7x + 20 \\ 39x^3 + 33x^2 + 37x + 38 & 15x^3 + 39x^2 + 38x + 30 \end{pmatrix}$$

and the private key

$$S_1(x) = \begin{pmatrix} x^3 - x^2 - 1 \\ -x^3 + x^2 + x \end{pmatrix}, \qquad S_2(x) = \begin{pmatrix} -x - 1 \\ x^2 - x + 1 \end{pmatrix}$$

(a) What is the other public matrix $T(x)$?
(b) Suppose you pick the random nonce vectors

$$\mathbf{y_1}(x) = (6\ 12\ 10\ -5)^T, \qquad \mathbf{y_2}(x) = (-12\ -11\ -11\ -13)^T$$

Do $\mathbf{z_1}(x)$ and $\mathbf{z_2}(x)$ pass the size checks?
(c) What is the signature using the above $\mathbf{y_1}(x)$ and $\mathbf{y_2}(x)$?

Exercise 8. You are Boaz, and you get the message "Entrance", supposedly from Avital. Her public key is

$$A(x) = \begin{pmatrix} 7x^3 + 17x^2 + 40x + 21 & 9x^3 + 7x + 15 \\ 35x^3 + 19x^2 + 39x + 33 & 33x^3 + 2x^2 + 23x + 24 \end{pmatrix},$$
$$T(x) = \begin{pmatrix} -21x^3 + 29x^2 + 23x + 26 \\ 34x^3 + 5x^2 - x \end{pmatrix}$$

(a) If the signature on the message is

$$\mathbf{z_1}(x) = \begin{pmatrix} 2x^2 + 8x + 3 \\ 4x^3 - 4x^2 - 8x + 12 \end{pmatrix}, \qquad \mathbf{z_2}(x) = \begin{pmatrix} -5x^3 - 6x^2 + 11x + 7 \\ 11x^3 + 8x^2 - 14x - 13 \end{pmatrix}, \qquad \mathbf{c}(x) = (x)$$

does the signature verify?
(b) If the signature on the message is

$$\mathbf{z_1}(x) = \begin{pmatrix} 8x^3 - 8x^2 - 8x - 8 \\ -13x^3 + 9x^2 + 6x + 12 \end{pmatrix}, \qquad \mathbf{z_2}(x) = \begin{pmatrix} -6x^3 + 6x - 1 \\ 2x^3 - 7x^2 - 13 \end{pmatrix}^T, \qquad \mathbf{c}(x) = (-x^2)$$

does the signature verify?
(c) Is it possible for more than one signature to correctly verify on the same message? Why or why not?

Department of Mathematics, Rose-Hulman Institute of Technology, Terre Haute, IN 47803, USA
Email address: `holden@rose-hulman.edu`