MAA PROGRAMS Virtual Programming

MATHEMATICAL ASSOCIATION OF AMERICA

RSA is dead, long live PQC!

Teaching cryptography in the quantum era

Joshua Holden (he/him/his)

Rose-Hulman Institute of Technology

ENERCELL

3 Nov
2022
(Part 2)

MAA Recording Policy

- The MAA is recording this event.
- We retain the right to show it again and to distribute it.
- By participating, you are agreeing that your contributions (e.g., text in the chat, your video if enabled, etc.) become part of the recording.
  - *Note:* Breakout rooms are NOT recorded.
- The recording is available by request to registered participants. Please email programs@MAA.org to request.

Letters of Participation

- If you would like a letter of participation, you must request that by emailing programs@MAA.org within 48 hours of attendance.
- Letters will not be sent to those who only view the recording, or those who are in attendance for less than 30 minutes.
- The MAA staff will verify attendance and then send a PDF of the letter to each requester.

Land Acknowledgement

This talk is being broadcast from land that is part of the traditional territories of the Očeti Šakówiŋ (Sioux), Kiikaapoi (Kickapoo), and Myaamia (Miami) nations. These peoples, and many others, are still fighting for the rights promised them by treaties with the United States government.

BIPOC lives and heritages matter.

What is Post-Quantum Cryptography?

Not ...

- ..."The thing that comes after Quantum Cryptography"
- ...Quantum Key Distribution
- ...Quantum Computation

Post-Quantum Cryptography (PQC) **is**

- cryptography that we can run on today's computers
- which will be resistant to cryptanalysis by quantum computers.

NIST has so far selected four submissions for standardization.

- CRYSTALS-Kyber (key-establishment for most use cases)
- CRYSTALS-Dilithium (digital signatures for most use cases)
- FALCON (digital signatures for use cases requiring smaller signatures)
- SPHINCS+ (digital signatures not relying on the security of lattices)

The first three are lattice systems; the fourth is a hash function system.

CRYSTALS-Kyber is based on a version of the Learning With Errors (LWE) problem.

Given a vector $\mathbf{t}$ of the form

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

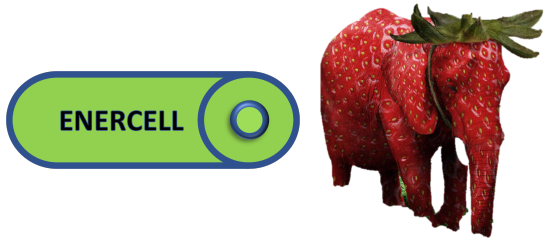
where

- A is a public matrix with at least as many rows as columns
- $\mathbf{e}$ is a "small error vector" drawn from some probability distribution

find the secret vector $\mathbf{s}$.

This seminar will introduce three toy cryptosystems based on variations of LWE:

- Last time: Smeagol (a simplified version of Frodo), based on LWE
- Phantom (a simplified version of NewHope), based on Ring LWE
- Alkaline (a simplified version of Kyber), based on Module LWE



ElectroNic ExeRcises for CiphEricaL Learning

We can replace the vectors with polynomials to get the Ring LWE (RLWE) problem.

Given a polynomial $t(x)$ of the form

$$t(x) \equiv a(x)s(x) + e(x) \pmod{x^n + 1} \pmod{q}, \tag{2}$$

where

- $a(x)$ is a public polynomial in $R = \{\text{polynomials in } x \text{ of degree} \leq n\}$
- $e(x)$ is an "error polynomial" with small coefficients

find the secret polynomial $s(x)$ in R.

Phantom key generation:

Boba Fett's private key is a polynomial $s(x)$ in R and his public key is $(a(x), t(x))$, where

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

for an error polynomial $e(x)$.

[Lucasfilm Ltd.]

Phantom example ($n = 4, q = 23$)

Boba Fett first generates the random polynomials

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

He then uses these to compute the public polynomial

$$\begin{aligned} t(x) &= (a(x)s(x) + e(x)) \operatorname{MOD} (x^n + 1) \operatorname{MOD} q \\ &= \left(18x^6 - 8x^5 - 6x^4 - 44x^3 - 37x^2 - 27x - 6\right) \operatorname{MOD} (x^4 + 1) \operatorname{MOD} 23 \\ &= ??? \end{aligned}$$

Phantom example ($n = 4, q = 23$)

Boba Fett first generates the random polynomials

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

He then uses these to compute the public polynomial

$$\begin{array}{rl}
 & 18x^2 - 8x - 6 \\
\hline
x^4+1 \;) & 18x^6 - 8x^5 - 6x^4 - 44x^3 - 37x^2 - 27x - 6 \\
 & -18x^6 \qquad\qquad\quad - 18x^2 \\
\hline
 & \quad -8x^5 - 6x^4 - 44x^3 - 55x^2 - 27x \\
 & \quad\;\; 8x^5 \qquad\qquad\qquad\quad + 8x \\
\hline
 & \qquad\quad -6x^4 - 44x^3 - 55x^2 - 19x - 6 \\
 & \qquad\quad\;\; 6x^4 \qquad\qquad\qquad\quad + 6 \\
\hline
 & \qquad\qquad\quad -44x^3 - 55x^2 - 19x
\end{array}$$

Phantom example ($n = 4, q = 23$)

Boba Fett first generates the random polynomials

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

He then uses these to compute the public polynomial

$$\begin{aligned} t(x) &= \left(-44x^3 - 55x^2 - 19x\right) \text{MOD } 23 \\ &= 2x^3 + 14x^2 - 19x \end{aligned}$$

and publishes $(a(x), t(x))$ as his public key.

Phantom encryption:

$p(x)$ is a plaintext message represented as a polynomial in R with 0's and 1's as coefficients.

Ahsoka Tano  chooses

- a random nonce polynomial $r(x)$ in R
- small random error polynomials $e_1(x)$ and $e_2(x)$.

She computes the ciphertext $(u(x), v(x))$, where

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

($\lfloor x \rceil$ means round x to the nearest integer. MOD is the "coder's mod".)

Phantom example (continued)

Ahsoka Tano will send the same message "h", or 1000 in binary, as we saw earlier. Now it corresponds to

$$p(x) = 1x^3 + 0x^2 + 0x + 0 = x^3.$$

She computes random vectors

$$r(x) = x^3 + 1, \qquad e_1(x) = -x, \qquad e_2(x) = -x^3 + x^2 - 1.$$

She then computes

$$\begin{aligned} u(x) &= (a(x)r(x) + e_1(x)) \operatorname{MOD} (x^4 + 1) \operatorname{MOD} 23 = x^3 - 8x^2 + 11x - 16 \\ v(x) &= (t(x)r(x) + e_2(x) + \lfloor q/2 \rceil\, p(x)) \operatorname{MOD} (x^4 + 1) \operatorname{MOD} 23 \\ &= \left[(2x^3 + 12x^2 + 13x + 19) + (-x^3 + x^2 - 1) + 12x^3\right] \operatorname{MOD} 23 \\ &= 13x^3 + 13x^2 + 13x + 18 \qquad \text{and sends } (u(x), v(x)) \text{ to Boba Fett.} \end{aligned}$$

Phantom decryption:

Boba Fett again tests each coefficient to see if it is closer to 0 or to $q/2$ modulo q:

$$p'(x) = \left\lfloor \frac{(v(x) - s(x)u(x)) \bmod (x^n + 1) \bmod q}{\lfloor q/2 \rceil} \right\rceil \bmod 2,$$

where the rounding is done component-wise.

Once again, $p'(x)$ will be equal to $p(x)$ as long as the coefficients of $e(x)r(x) + e_2(x) - s(x)e_1(x)$ have magnitude less than $q/4$.

Phantom example (concluded)

You are Boba Fett  . You receive the message:

$$u(x) = x^3 - 8x^2 + 11x - 16 \qquad v(x) = 13x^3 + 13x^2 + 13x + 18$$

from Ahsoka Tano  . Recall your private key is:

$$s(x) = x^3 - x^2 - x - 1$$

You compute

$$p'(x) = \left\lfloor \frac{(v(x) - s(x)u(x)) \operatorname{MOD} (x^4 + 1) \operatorname{MOD} 23}{\lfloor 23/2 \rfloor} \right\rceil \operatorname{MOD} 2 = ???$$

Do you get the message that Ahsoka sent?

RLWE is more efficient, but possibly less secure.

$$a(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0, \quad s(x) = s_{n-1}x^{n-1} + \ldots + s_1 x + s_0$$

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

$$\text{Then } A\mathbf{s} = \begin{pmatrix} a_0 s_0 - a_{n-1}s_1 - \cdots - a_2 s_{n-2} - a_1 s_{n-1} \\ a_0 s_1 + a_0 s_1 - \cdots - a_3 s_{n-2} - a_2 s_{n-1} \\ \vdots \\ a_{n-2}s_0 + a_{n-3}s_1 + \cdots + a_0 s_{n-2} - a_{n-1}s_{n-1} \\ a_{n-1}s_0 + a_{n-2}s_1 + \cdots + a_1 s_{n-2} + a_0 s_{n-1} \end{pmatrix}$$

corresponds to $a(x)s(x)$ reduced modulo $x^n + 1$.
So RLWE is equivalent to LWE with shorter keys but more structure.

Kyber uses a combination of LWE and RLWE called the Module Learning With Errors (MLWE) problem.

Given a vector $\mathbf{t}(x)$ of k polynomials in R, with the form

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

where

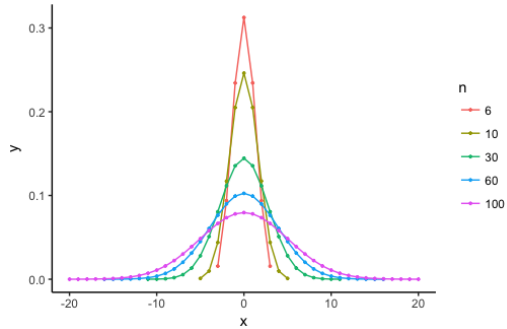
- $A(x)$ is a $k \times k$ public matrix of polynomials in R with at least as many rows as columns, and
- $\mathbf{e}(x)$ is a "small error vector" of k polynomials in R drawn from some probability distribution

find the secret vector of k polynomials $\mathbf{s}(x)$.

What about that probability distribution?

Kyber uses "centered binomial coefficients":

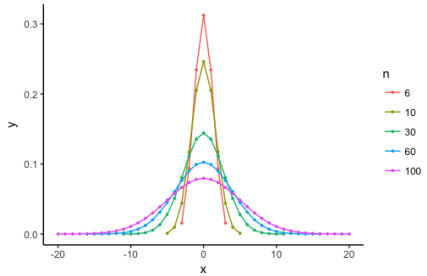
- generate 2η uniformly random bits $a_1, \ldots, a_\eta$ and $b_1, \ldots, b_\eta$;
- calculate $\sum_{i=1}^{\eta}(a_i - b_i)$.



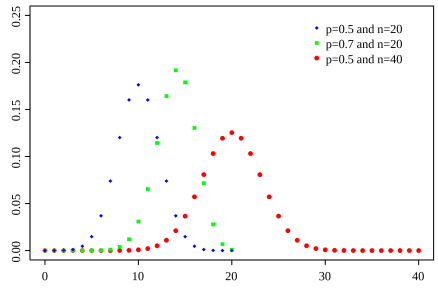
[www.stackoverflow.com/questions/42684993]

This gives an opportunity for probability students.

- Prove that this is the same distribution obtained by flipping 2η fair coins, counting the number of heads, and subtracting η.
- Or, calculate the probabilities for a small value of η and show that the distributions are equal.



[www.stackoverflow.com/questions/42684993]



[Wikipedia User Tayste]

Alkaline key generation:

Bernie

- picks $k \times k$ matrix $A(x)$ with uniform random entries in $R_q = \{\text{polynomials in } R \text{ with coefficients in } [0, q)\}$
- generates $2nk\eta$ random bits
- uses random bits to compute centered binomial coefficients for vectors $\mathbf{s}(x)$ and $\mathbf{e}(x)$ of k polynomials in R
- 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

(The notation $a(x) \star t(x)$ means $a(x)t(x) \,\mathrm{MOD}(x^n + 1) \,\mathrm{MOD}\, q$. All additions will be modulo q, and similarly for matrices and vectors.)

Alkaline example ($n = 4, k = 2, q = 23$)

Bernie first generates the random 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.

Alkaline encryption:

Aretha

- encodes plaintext as a polynomial $p(x)$ in R
- looks up Bernie's encryption key $(A(x), \mathbf{t}(x))$
- generates $2nk\eta + 2n\eta$ random bits
- uses random bits to compute centered binomial coefficients for vectors $\mathbf{r}(x)$ and $\mathbf{e}_1(x)$ of k polynomials in R each and one polynomial $e_2(x)$ in R
- Computes ciphertext $\mathbf{u}(x) = A(x)^T \star \mathbf{r}(x) + \mathbf{e}_1(x)$, $v(x) = \mathbf{t}(x)^T \star \mathbf{r}(x) + e_2(x) + \lfloor q/2 \rceil \, p(x)$

Alkaline example (continued)

Aretha wants to send the message "hi" to Bernie. For the first letter, Aretha encodes "h" as the number 8, or 1000 in binary, corresponding to

$$p(x) = 1x^3 + 0x^2 + 0x + 0 = x^3.$$

She generates 20 random bits and uses them to compute

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

$$e_2(x) = -x^2 + x + 1.$$ She then computes

$$\mathbf{u}(x) = A(x)^T \star \mathbf{r}(x) + \mathbf{e}_1(x) = \begin{pmatrix} 7x^3 + 22x^2 + 2x - 15 \\ 4x^3 + x^2 - 13x + 13 \end{pmatrix}$$

$$\begin{aligned} v(x) &= \mathbf{t}(x)^T \star \mathbf{r}(x) + e_2(x) + \lfloor q/2 \rceil\, p(x) \\ &= \left[(9x^3 + 22x^2 + 19x + 9) + (-x^2 + x + 1) + 12x^3\right] \text{MOD } 23 \\ &= 21x^3 + 21x^2 + 20x + 10 \end{aligned}$$ and sends $(\mathbf{u}(x), v(x))$ to Bernie.

Alkaline decryption:

Aretha

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

Bernie

Bernie

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

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

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

Alkaline example (concluded)

Bernie receives the message

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

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.

The developers of Kyber discovered some tricks that lead to even greater efficiency.

These fall into three main categories:

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

And, of course, they used cryptographically-sized parameters.

	n	k	q	η	$\delta =$ failure probability
Kyber512	256	2	3329	3/2*	2^{-139}
Kyber768	256	3	3329	2	2^{-164}
Kyber1024	256	4	3329	2	2^{-174}

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

Table: Parameter sets for Kyber

The best known attack on Kyber is currently the "primal attack" on generic LWE.

The primal attack on LWE starts with the observation that since

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

we can consider matrices

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

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 want to find a short vector in that left kernel.

The first step is to find a basis for the left kernel.

Consider the LWE cryptosystem 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

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

Example of finding a basis, continued

Putting this in integer echelon form gives

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

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)}.

We can then use a lattice reduction algorithm to find a short vector of the correct form.

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

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

Looking for a short vector with a 1 in the correct position for $\mathbf{s}''$ yields

$$(1\ 2\ 0\ -1\ 1\ -1\ -1),$$

corresponding to $\mathbf{s} = (1\ 2)^T$ and $\mathbf{e} = (0\ -1)^T$, which is in fact correct.

What kinds of PQC can you teach to undergraduates?

- Lattice-based: NTRU, Smeagol, Phantom, Alkaline (`https://github.com/joshuarbholden/alkaline`)
- Code-based: McEliece
- Hash-based: Merkle signature scheme
- Multivariable: Oil and Vinegar
- Elliptic curve isogenies: Charles-Goren-Lauter hash function

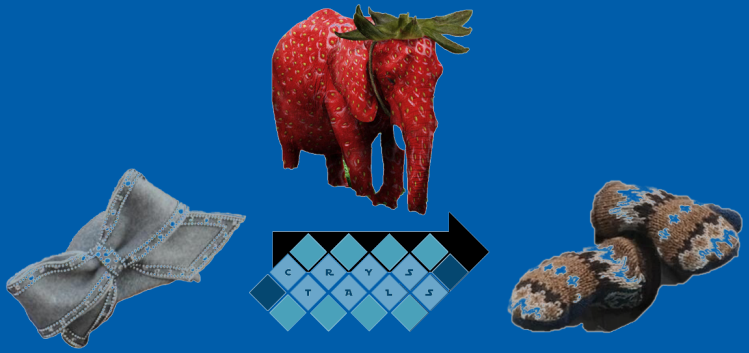
My *Resource Guide for Teaching Post-Quantum Cryptography* is at `https://arxiv.org/abs/2207.00558` (to appear in *Cryptologia*).

Also watch for the second edition of *The Mathematics of Secrets*, from Princeton University Press.

What undergraduate classes would this be appropriate for?

- Linear Algebra: Smeagol, Oil and Vinegar, McEliece
- Abstract Algebra: Phantom, Alkaline, McEliece
- Second course in Algebra or in Number Theory: CGL
- Data Structures: Merkle signature scheme
- Cryptography and/or student research project: any of these!

Thanks for joining us for this MAA virtual event!



MAA
MATHEMATICAL ASSOCIATION OF AMERICA

PROGRAMS
Virtual Programming