

Bob



Python and Crypto: Learning With Errors (LWE) and Ring LWE

Prof Bill Buchanan OBE, The Cyber Academy

<http://asecuritysite.com>

Alice



Eve



**CYBER
ACADEMY**

Bob



Alice



Learning With Errors (LWE)

Prof Bill Buchanan OBE, The Cyber Academy

<http://asecuritysite.com>

Eve



**CYBER
ACADEMY**

Quantum Methods

- Quantum Computers will crack most public key methods, such as RSA and Elliptic Curve encryption.
- We need new methods which define hard problems which will not be cracked by quantum computers.
- **Lattice-based cryptography** [[Lattice](#)] – This classification shows great potential and is leading to new cryptography, such as for fully homomorphic encryption [here], and code obfuscation. An example is given in the following section.
- **Code-based cryptography** [[McEliece](#)] – This method was created in 1978 with the McEliece cryptosystem but has barely been using in real applications. The McEliece method uses linear codes that are used in error correcting codes, and involves matrix-vector multiplication. An example of a linear code is Hamming code [here].
- **Multivariate polynomial cryptography** [[UOV](#)] – These focus on the difficulty of solving systems of multivariate polynomials over finite fields. Unfortunately, many of the methods that have been proposed have already been broken.
- **Hash-based signatures** [[GMSS](#)] – This would involve created digital signatures using hashing methods. The drawback is that a signer needs to keep a track of all of the messages that have been signed, and that there is a limit to the number of signatures that can be produced.

LWE

With LWE we use a random matrix (**A**) secret matrix (**s**) and an error matrix (**e**).

Presentation based on Introduction to post-quantum cryptography and learning with errors, Summer School on real-world crypto and privacy, Šibenik, Croatia • June 11, 2018 [[Link](#)].

LWE

random
 $\mathbb{Z}_{13}^{7 \times 4}$

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

\times

secret
 $\mathbb{Z}_{13}^{4 \times 1}$

6
9
11
11

$+$

small noise
 $\mathbb{Z}_{13}^{7 \times 1}$

0
-1
1
1
1
0
-1

$=$

$\mathbb{Z}_{13}^{7 \times 1}$

4
7
2
11
5
12
8

LWE

A

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

```
A=np.array([[4 ,1, 11, 10],[5, 5 ,9 ,5],[3, 9 ,0 ,10],[1, 3 ,3 ,2],[12, 7 ,3 ,4],[6, 5 ,11 ,4],[3, 3, 5, 0]])
```

All operations are (mod q). Let $q = 13$

LWE

A

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

sA

6
9
11
11

X

+

eA

0
-1
1
1
1
0
-1

```
A=np.array([[4 ,1, 11, 10],[5, 5 ,9 ,5],[3, 9 ,0 ,10],[1, 3 ,3 ,2],[12, 7 ,3 ,4],[6, 5 ,11 ,4],[3, 3, 5, 0]])
```

```
sA = np.array([[6],[9],[11],[11]])
```

```
eA = np.array([[0],[-1],[1],[1],[1],[0],[-1]])
```

All operations are mod q. q=13

LWE

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

×

6
9
11
11

+

0
-1
1
1
1
0
-1

=

4
7
2
11
5
12
8

q=13

```
A=np.array([[4 ,1, 11, 10],[5, 5 ,9 ,5],[3, 9 ,0 ,10],[1, 3 ,3 ,2],[12, 7 ,3 ,4],[6, 5 ,11 ,4],[3, 3, 5, 0]])
```

```
sA = np.array([[6],[9],[11],[11]])
```

```
eA = np.array([[0],[-1],[1],[1],[1],[0],[-1]])
```

```
bA = np.matmul(A,sA)%q
```

```
bA = np.add(bA,eA)%q
```

```
print bA
```

4	1	11	10
5	5	9	5
3	9	0	10
1	3	3	2
12	7	3	4
6	5	11	4
3	3	5	0

×

6
9
11
11

+

0
-1
1
1
1
0
-1

=

4
7
2
11
5
12
8

LWE - Encrypting

Next we have a single bit message (M) we sample A and B to produce:

$$u = \sum A_{samples}$$
$$v = \sum B_{samples} - \frac{q}{2} \cdot M$$

The encrypted value is then (u, v)

LWE - Decrypting

To decrypt we take (u,v) and calculate:

$$Dec = v - su \pmod{q}$$

If Dec is less than $q/2$, the message is 0.

If Dec is greater than $q/2$, the message is 1.

LWE - Example

Message to send: 0

Public Key (A): [80, 86, 19, 62, 2, 83, 25, 47, 20, 58, 45, 15, 30, 68, 4, 13, 8, 6, 42, 92]

Public Key (B): [15, 45, 2, 20, 13, 30, 32, 45, 4, 3, 34, 78, 55, 51, 23, 67, 44, 34, 17, 75]

Errors (e): [3, 3, 4, 1, 3, 3, 4, 4, 1, 4, 3, 3, 2, 2, 3, 2, 4, 4, 1, 3]

Secret key: 5

Prime number: 97

Sampling [18, 5, 8, 13, 11]

U is 34 V is 83.0 Message is a 0 [[Link](#)]

Bob



Alice



Ring Learning With Errors (Ring-LWE)

Prof Bill Buchanan OBE, The Cyber Academy

<http://asecuritysite.com>

Eve



**CYBER
ACADEMY**

Ring LWE

$$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$$

$$4 + 1x + 11x^2 + 10x^3$$

random

×

$$6 + 9x + 11x^2 + 11x^3$$

secret

+

$$0 - 1x + 1x^2 + 1x^3$$

small noise

=

$$10 + 5x + 10x^2 + 7x^3$$

Ring LWE

 $\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$

$$\mathbf{B} = \mathbf{A} \times \mathbf{sA} + \mathbf{eA}$$

$$\mathbf{A} = [4, 1, 11, 10]$$

$$\mathbf{sA} = [6, 9, 11, 11]$$

$$\mathbf{eA} = [0, -1, 1, 1]$$

$$n=4$$

$$xN_1 = [1] + [0] * (n-1) + [1]$$

$$4 + 1x + 11x^2 + 10x^3$$

random

×

$$6 + 9x + 11x^2 + 11x^3$$

secret

+

$$0 - 1x + 1x^2 + 1x^3$$

small noise

=

$$10 + 5x + 10x^2 + 7x^3$$

Ring LWE

$$\mathbb{Z}_{13}[x]/\langle x^4 + 1 \rangle$$

$$\mathbf{B} = \mathbf{A} \times \mathbf{sA} + \mathbf{eA}$$

$$\mathbf{A} = [4, 1, 11, 10]$$

$$\mathbf{sA} = [6, 9, 11, 11]$$

$$\mathbf{eA} = [0, -1, 1, 1]$$

$$\mathbf{xN_1} = [1] + [0] * (n-1) + [1]$$

$$\mathbf{A} = \text{np.floor}(\text{p.polydiv}(\mathbf{A}, \mathbf{xN_1})[1])$$

$$\mathbf{bA} = \text{p.polymul}(\mathbf{A}, \mathbf{sA}) \% q$$

$$\mathbf{bA} = \text{np.floor}(\text{p.polydiv}(\mathbf{bA}, \mathbf{xN_1})[1])$$

$$\mathbf{bA} = \text{p.polyadd}(\mathbf{bA}, \mathbf{eA}) \% q$$

$$\mathbf{bA} = \text{np.floor}(\text{p.polydiv}(\mathbf{bA}, \mathbf{xN_1})[1])$$

$$\text{print } \mathbf{bA}$$

$$4 + 1x + 11x^2 + 10x^3$$

random

$$6 + 9x + 11x^2 + 11x^3$$

secret

$$0 - 1x + 1x^2 + 1x^3$$

small noise

$$10 + 5x + 10x^2 + 7x^3$$

RLW-KEX

- $xN_1 = [1] + [0] * (n-1) + [1]$

Alice:

- $A = \text{np.floor}(p.\text{polydiv}(A, xN_1)[1])$
- $bA = p.\text{polymul}(A, sA) \% q$
- $bA = \text{np.floor}(p.\text{polydiv}(sA, xN_1)[1])$
- $bA = p.\text{polyadd}(bA, eA) \% q$

- $\text{sharedAlice} = \text{np.floor}(p.\text{polymul}(sA, bB) \% q)$
- $\text{sharedAlice} = \text{np.floor}(p.\text{polydiv}(\text{sharedAlice}, xN_1)[1]) \% q$
- $\text{sharedBob} = \text{np.floor}(p.\text{polymul}(sB, bA) \% q)$
- $\text{sharedBob} = \text{np.floor}(p.\text{polydiv}(\text{sharedBob}, xN_1)[1]) \% q$

- $xN_1 = [1] + [0] * (n-1) + [1]$

Bob:

- $sB = \text{gen_poly}(n, q)$
- $eB = \text{gen_poly}(n, q)$
- $bB = p.\text{polymul}(A, sB) \% q$
- $bB = \text{np.floor}(p.\text{polydiv}(sB, xN_1)[1])$
- $bB = p.\text{polyadd}(bB, eB) \% q$

Bob



Python and Crypto: Learning With Errors (LWE) and Ring LWE

Prof Bill Buchanan OBE, The Cyber Academy

<http://asecuritysite.com>

Alice



Eve



**CYBER
ACADEMY**