Jouko Teeriaho / Feb 2026

Post-Quantum Cryptography (PQC)

PQC = developing cryptographic algorithms designed to secure data against future, powerful quantum computers that could break current encryption methods.

Current public-key algorithms (e.g., RSA, DH, Elliptic Curves) rely on mathematical problems that quantum computers can solve efficiently using algorithms like Shor’s.

Quantum resistant and non-quantum resistant algorithms

In protocols like TLS data encryption algorithm AES is considered to be quantum resistant. But in Key Agreement Protocols most common algorithms RSA, DH and ECDHE are not quantum resistant.

LWE based encryption algorithm

LWE (Learning With Errors) based on a set of linear equations.

Assume we have an overdetermined set of linear equations, where each equation is of the form $a_{i1}s_1+a_{i1}s_2+....+a_{in}s_n = b_i$. Coefficients s_1,s_2,... are the variables, which are assumed to be integers.

This equation can be written in matrix form as follows

- A . **s** = **b** , where A is the n × m coefficient matrix of the system of equations

Knowing matrix A and vector **b** the unknown vector **s** can be solved.

1. Generation of public key and private key

We make to the matrix equation following modifications:
1) Add to the left hand side an error vector **e** = $(e_1, e_2, \ldots, e_n)$, where components e_i are small integers.
2) Add a prime modulus q to the equation

After the modifications equation can be written

$$\mathrm{A} . \mathrm{s} + \mathbf{e} = \mathbf{b} \pmod{\mathrm{q}} \tag{1}$$

After the modifications vector **s** is very hard to solve from **A** and **b** without knowing the error vector **e**.
This mathematically hard problem is a basis of a cryptosystem.

The keys of a user are following

1. **Public key** is the pair **(A, b)**
2. **Private key** is vector **s**

2. Algorithms for encryption and decryption

2.1 Message m

The algorithm is used in Key Exchange Protocols for sending 256 bit AES keys to the other party of communication. It encrypts the binary message bit by bit. That is why we assume that the message m is either 0 or 1.

2.2 Encryption

Algorithm steps:
1) Choose a random **sample R** of the equations of the linear system.
2) Calculate two parameters **u** and **v**:
Vector **u** is obtained by summing the equations of the sample R (mod q). We can write $\mathbf{u} = \sum_R A$ (mod q)

Constant **v** is the sum of constants **b** of the right hand side of the equation (1) belonging to the sample R. To the sum we add the message **m** scaled with a factor, which is half of the modulus q rounded to the nearest integer

$$\mathbf{v} = \sum_R b \pmod{q} + m \left[\frac{q}{2}\right]$$

Thus for encryption of a single bit **m** , we use recipients public keys (A,b) and send a pair **{u,v}**.

$$\text{cipher } \mathbf{c} = \{\mathbf{u}, \mathrm{v}\} = \left\{\sum_R A \pmod{q},\ \sum_R b + m\left[\frac{q}{2}\right] \pmod{q}\right\} \tag{2}$$

R is a random sample of rows in the system of equations

2.3 Decryption

For decryption of cipher c recipient uses his private key **s**

$$\text{decryption}: \quad v - \mathbf{s}^T \mathbf{u} \pmod{q} \tag{3}$$

If the result of the formula $<$ q/2, then m = 0, else m = 1.

2.4 Another formulation of algorithm

We can express the choosing of random sample R as a binary vector $\mathbf{r} = (r_1, r_2, \ldots, r_2)$ as in (1, 0, 1, 1, 0, 0),
where 1 means that the row is included in the sample R while 0 means the it is not.

$$c = \{\mathbf{u}, v\} = \left\{A^T.r \pmod{q},\ b^T.r + m\left\lceil \frac{q}{2} \right\rfloor \pmod{q}\right\} \tag{4}$$

$$\begin{aligned} \text{decryption} \quad g &= v - \mathbf{s}^T \mathbf{u} \pmod{q} \\ m &= \left\lceil \frac{2g}{q} \right\rfloor \qquad (\text{rounds message to } 0 \text{ or } 1) \end{aligned} \tag{5}$$

2.5 Solved example

A: Generating keys for Alice

```
A = ( 15  -7   9  11
       6 -10   6   2
      18  12  -7   6
       4 -14  20   8
       6   3  17  -5
      -3  -6  19  21 ); (* coeff.matrix, first part of public key *)

s = {9, 3, 7, 11};      (* private key *)
e = {2, -1, 3, 2, -2, 1}; (* error vector *)
q = 89           (* prime modulus *);
```

Calculation of the right hand side vector **b**

```
b = Mod[A.s + e, q]     (* second part of public key *)

Out[•]= {33, 87, 40, 46, 36, 53}
```

B: Encryption of message m = 1 to Alice

```
m = 1; (* message (1 bit) *)
r = {1, 0, 1, 1, 0, 0} (* sample of equations *);
```

Calculation of the cipher {**u**,v}

```
In[•]:=
{u, v} = {Mod[Transpose[A].r, q],
  Mod[Transpose[b].r + m * Round[q / 2], q]}

Out[•]= {{37, 80, 22, 25}, 74}
```

C: Decryption of the cipher

In[ ]:=
```
g = Mod[v - Transpose[s].u, q]
```
Out[ ]= 51

In[ ]:=
```
m = Round[2 * g / q]
```
Out[ ]= 1

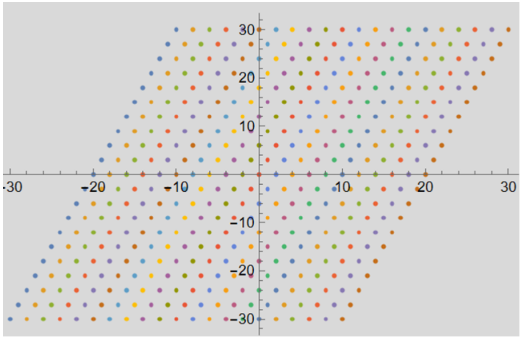
Point Lattice -based encryption

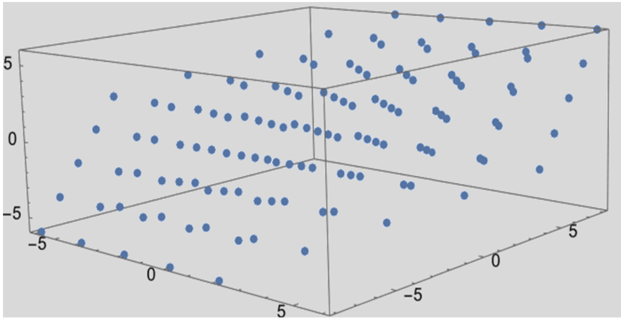
Point Lattices

Definition. Let $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}\}$ be a set of linearly independent vectors and let each $\boldsymbol{v_i}$ be of form $(x_1, x_2, \ldots, x_n)$, where all x_i are integers. The set of vectors $L = \{a_1 \mathbf{v_1} + a_2\, \mathbf{v_2} + \ldots + a_n \boldsymbol{v_n}\}$, where coefficients a_i get all possible integer values, is called a *lattice*. Vector set V is called the basis of the lattice L.

Example. Let **a** =(a_1,a_2) and **b** =(b_1,b_2) be two linearly independent vectors with integer components. Vectors **a** and **b** generate a lattice L = {m a + n b}, where m and n are integer coefficients. Vectors **a** and **b** form a basis of lattice L

Pictures of lattices in 2D and 3D spaces.





Changing base of lattices

Definition. A n x n square matrix P is called *unimodular*, if its determinant det(P) equals either 1 or -1.

Lemma. Let $V = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ be a basis of lattice L and let $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$ a set of vectors obtained with a linear transformation from vectors of V. Then U is also a base of lattice L, if the transformation matrix is unimodular.

Motivation for the above lemma is following: The components of basis of lattice L must be integers. In order that transformation matrix P would map integer vector to integer vectors in both directions, the inverse matrix P^{-1} should also have only integer elements. This is possible only, when det(P) = ±1.

Example. Let L be a lattice L with basis vectors **a** =(2,0) and **b** =(1,3).

a) Create a new basis {**u**,**v**} using unimodular transformation matrix P = $\begin{pmatrix} 7 & 1 \\ 6 & 1 \end{pmatrix}$.

We have $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} 7 & 1 \\ 6 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}$ => The new basis vectors are

u = 7 **a** + **b** = 7 (2,0) + (1,3) = (15,3) and

v = 6 **a** + **b** = 6 (2,0) +(1,3) = (13,3)

b) Present the lattice point 19 **a** - 7 **b** in the new basis **{u,v}** .

Calculation of coordinates gives 19 **a** - 7 **b** = 19 (2,0) - 7(1,3) = (31,-21)

Coefficients r and s in the new base are obtained from (31,-21) = r (15,3) + s (13,3),

which gives pair of equations $\begin{cases} 31 = 15\,r + 13\,s \\ -12 = 3\,r + 3\,s \end{cases}$. Its solution is: r = 61 and s = -68

Creating unimodular matrices

Lemma. Assume that L is a lower triangular matrix and U is an upper triangular matrix and assume that both L and U have only numbers -1 or 1 on their diagonals. Then the product matrix L.U is unimodular.

*Proof. From the general rule det(AB)= det(A) det(B) it follows that det(L.U) = det(L)*det(U). Now det(L.U) equals -1 or 1, because determinants of triangular matrices equal to the products of their diagonal elements.*

Example: Create a unimodular matrix using triangular matrices L = $\begin{pmatrix} -1 & 0 \\ 6 & 1 \end{pmatrix}$ and U = $\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$.

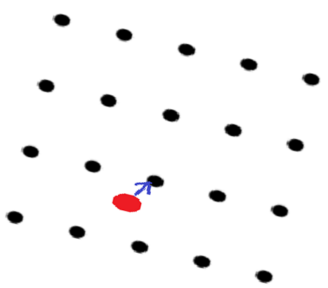
Solution: Multiplication L.U gives $\begin{pmatrix} -1 & -3 \\ 6 & 19 \end{pmatrix}$, which is unimodular, because determinant -1*19-6*(-3) = -1.

Closest Vector Problem (CVP)

Let $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}\}$ be a basis of lattice L and let **a** $=\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n\}$ be **a** point, which is not point of lattice L.

The problem of finding the point of lattice L, which is closest to **a** is called the Closest Vector Problem (CVP).

Picture of CVP in 2 D lattice



When the dimension n is sufficiently large, CVP problem belongs to complexity class NP *) and cannot be solved in reasonable time even with quantum computers. CVP is a hard problem which can be used in Post Quantum Cryptography.

*) Term ”NP ” comes from ”Non-deterministics Polynomial Time”

Babai's algorithm for solving CVP

Babai's Rounding Method for finding closest lattice point to vector a
Let $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}\}$ be a basis of lattice L. Write the target vector **a** as a linear combination of base vectors.
Solve coefficients t_i from $\mathbf{a} = t_1\boldsymbol{v_1}+t_2\boldsymbol{v_2}+\ldots+t_n\boldsymbol{v_n}$ and round t_i's to nearest integers $[t_i]$.
Then the closest lattice point is $[t_1]\boldsymbol{v_1}+[t_2]\boldsymbol{v_2}+\ldots+[t_n]\boldsymbol{v_n}$

Weakness: The quality of the result depends entirely on how “good” (near the orthogonal) the basis is. If the basis is poor, the found point may be far from the actual nearest point.

Example. Lattice L has basis of vectors $\boldsymbol{w}_1$ = (3,0) and $\boldsymbol{w}_2$=(2,5). Find the closest lattice point for point **a** = (18,16).

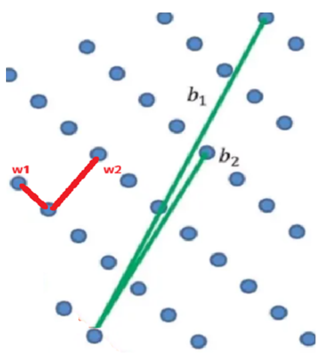
Solution. Solving coefficients a_1 and a_2 from equation $\mathbf{a} = a_1\,\boldsymbol{w}_1 + a_2\,\boldsymbol{w}_2$ gives pair of equations $\begin{cases} 18 = 3\,a_1 + 2\,a_2 \\ 16 = 5\,a_2 \end{cases}$, which has solution a_1 = 3.87 and a_2 = 3.2 , which are rounded to $[a_1]$ = 4 and $[a_2]$ =3
Substitution gives the closest lattice point 4 (3,0) + 3 (2,5) = (12+6, 15) = (18,15).

Good and bad bases

From the point of view of solving CVP some bases can be “good” and some other bases “bad”

Good basis vectors are orthogonal or nearly orthogonal. Babai's algorithm works well.

Bad basis vectors are almost parallel and rather long. Babai's algorithm works badly.



Picture : Good and Bad bases

Public key encryption algorithm based on Point Lattices.

Alice’s keys

Every used has two keys, which are different bases of the same lattice L.

- Private key is a “good basis” W = ($\boldsymbol{w}_1$,$\boldsymbol{w}_2$, …., $\boldsymbol{w}_n$)
- Public key is a “bad basis” B = ($\boldsymbol{b}_1$,$\boldsymbol{b}_2$, …., $\boldsymbol{b}_n$)

Encryption

Another user Bob sends an encrypted message **m**, which is a vector with integer components (m_1,m_2,…,m_3) to Alice using Alice’s public key, which is the ”the bad basis” B in the following way.

1) Bob generates a random error vector **e** = (e_1, e_2,…,e_3) in which components e_i are small.
2) Bob calculates the cipher **c** = m_1 $\boldsymbol{b}_1$+m_2 $\boldsymbol{b}_2$+ ….+ $m_2\boldsymbol{b}_n$ +**e**

Decryption

Alice applies Babai algorithm using her private key, which is the ”the good basis” W
1) Alice solves coefficients a_i from vector equation c = a_1 $\boldsymbol{w}_1$+a_2 $\boldsymbol{w}_2$+ ….+ $m_2\boldsymbol{w}_n$
2) Alice rounds the coefficients to nearest integers [a_i] and calculate the lattice point
p = [a_1] $\boldsymbol{w}_1$+ [a_2] $\boldsymbol{w}_2$+ ….+ [a_n] $\boldsymbol{w}_n$ which is the lattice point closest to **c.**
4) Alice solves coefficients m_i from equation **p** = m_1 $\boldsymbol{b}_1$+m_2 $\boldsymbol{b}_2$+ ….+ $m_2\boldsymbol{b}_n$
and retrieves the message **m** = (m_1,m_2,…,m_3)

Security

A hacker captures cipher **c** and tries to break it using Alice’s public key and Babai algorithm for solving coefficients t_i from c = t_1 $\boldsymbol{b}_1$+t_2 $\boldsymbol{b}_2$+ ….+ $t_2\boldsymbol{b}_n$ and rounding t_i ’s to nearest integers. Because B is a bad basis where Babai algorithm does not work, he is not able to find the closest Lattice point.

A solved example

Alice’s keys

- Alice’s private key : a good basis W consisting of vectors $\boldsymbol{w}_1$=(11,0,0), $\boldsymbol{w}_2$=(0,20,0), $\boldsymbol{w}_3$=(0,0,-25)
- Alice’s public key : a “bad basis” B consisting of vectors $\boldsymbol{b}_1$=(-11,40,225), $\boldsymbol{b}_2$=(-40,140,-775), $\boldsymbol{b}_3$=(-55,240,-1350)

Encryption

Bob sends a message **m** = (12, 5, 8) to Alice.

- Bob chooses a random error vector **e** = (2,4,3)
- Bob computes the cipher **c** as follows

$$\mathbf{c} = 12\, \boldsymbol{b}_1 + 5\, \boldsymbol{b}_2 + 8\, \boldsymbol{b}_3 + \mathbf{e}$$
$$= 12\ (-11,40,225) + 5\ (-40,140,-775) + 8\ (-55,240,-1350) + (2,4,3)$$
$$= (-792, 3100, -17375)$$

Decryption

Alice applies Babai algorithm using her private key , ”the good basis” B.

1. Alice solves coefficients a_i from vector equation (-792, 3100, -17375) = a_1 (11,0,0) + a_2 (0,20,0) +a_3 (0,0,-25)
and rounds a_i's to nearest integers: => [a_1]=[71.82] = 72, , [a_2]=[155.2] = 155 and [a_2]=[694.88]=695.

2. Then Alice calculates the closest lattice point **p** = [a_1] $\boldsymbol{w}_1$+ [a_2] $\boldsymbol{w}_2$+ ....+ [a_n] $\boldsymbol{w}_n$
= 72 (11,0,0) + 155(0,20,0) +695 (0,0,-25) = (-792,3100,-17375).
Because Alice uses Good Basis, this is the lattice point closest to **c**.

4) Finally Alice solves the message **m** = (m_1,m_2,m_3) from equation **p** = m_1 $\boldsymbol{b}_1$ + m_2 $\boldsymbol{b}_2$ + m_3 $\boldsymbol{b}_3$
Solution of (-792, 3100, -17375) = m_1 (-11, 40, 225) + m_2 (-40, 140, -775) + m_3(-55, 240, -1350)
is m_1 = 12, m_2 = 5 , m_3 = 8, which gives the original message m = (12, 5, 8).

Hacker’s attempt to break the cipher

A hacker captures cipher **c** and tries to break it using Alice’s public key and Babai's algorithm to solve coefficients t_i from **c** = t_1 $\boldsymbol{b}_1$+t_2 $\boldsymbol{b}_2$+$t_3$$\boldsymbol{b}_n$.

In out example we get (-792, 3100, -17375) = $t_1(-11, 40, 225)+t_2(-40, 140, -775)+t_3(-55, 240, -1350)$

Solution is t_1= -2.851, t_2= 7.04 and t_3 = 9.3 , which after rounding gives [t1] = -3, [t2] =7 and [t3] =9 .

This leads to decrypted message (-3, 7, 9) , which is incorrect. (Right message was **m** = (12, 5, 8).