


DEFINITION 1 (LATTICE) *Given n linearly independent vectors $b_1, b_2, \dots, b_n \in \mathbb{R}^m$, the lattice generated by them is defined as*


$$\mathcal{L}(b_1, b_2, \dots, b_n) = \left\{ \sum x_i b_i \mid x_i \in \mathbb{Z} \right\}.$$

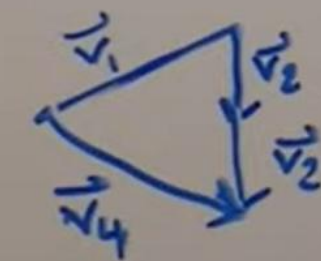
We refer to b_1, \dots, b_n as a *basis* of the lattice.

Given n linearly independent vectors $b_1, b_2, \dots, b_n \in \mathbb{R}^m$ the lattice generated by them is defined as –
$$\mathcal{L}(b_1, b_2, b_3 \dots b_n) = \sum x_i b_i \mid x_i \in \mathbb{Z}$$

$$\vec{v}_1 =$$


$$\vec{v}_2 =$$


$$\vec{v}_1 + \vec{v}_2 =$$


$$\vec{v}_1 + (-2)\vec{v}_2 =$$


basis
" $\{\vec{v}_1, \vec{v}_2\}$

$$\vec{v}_3 \in \text{lattice}(\vec{v}_1, \vec{v}_2)$$

$$\vec{v}_4 \in \text{lattice}(\vec{v}_1, \vec{v}_2)$$

$$\vec{v}_1 = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_i \end{bmatrix}$$

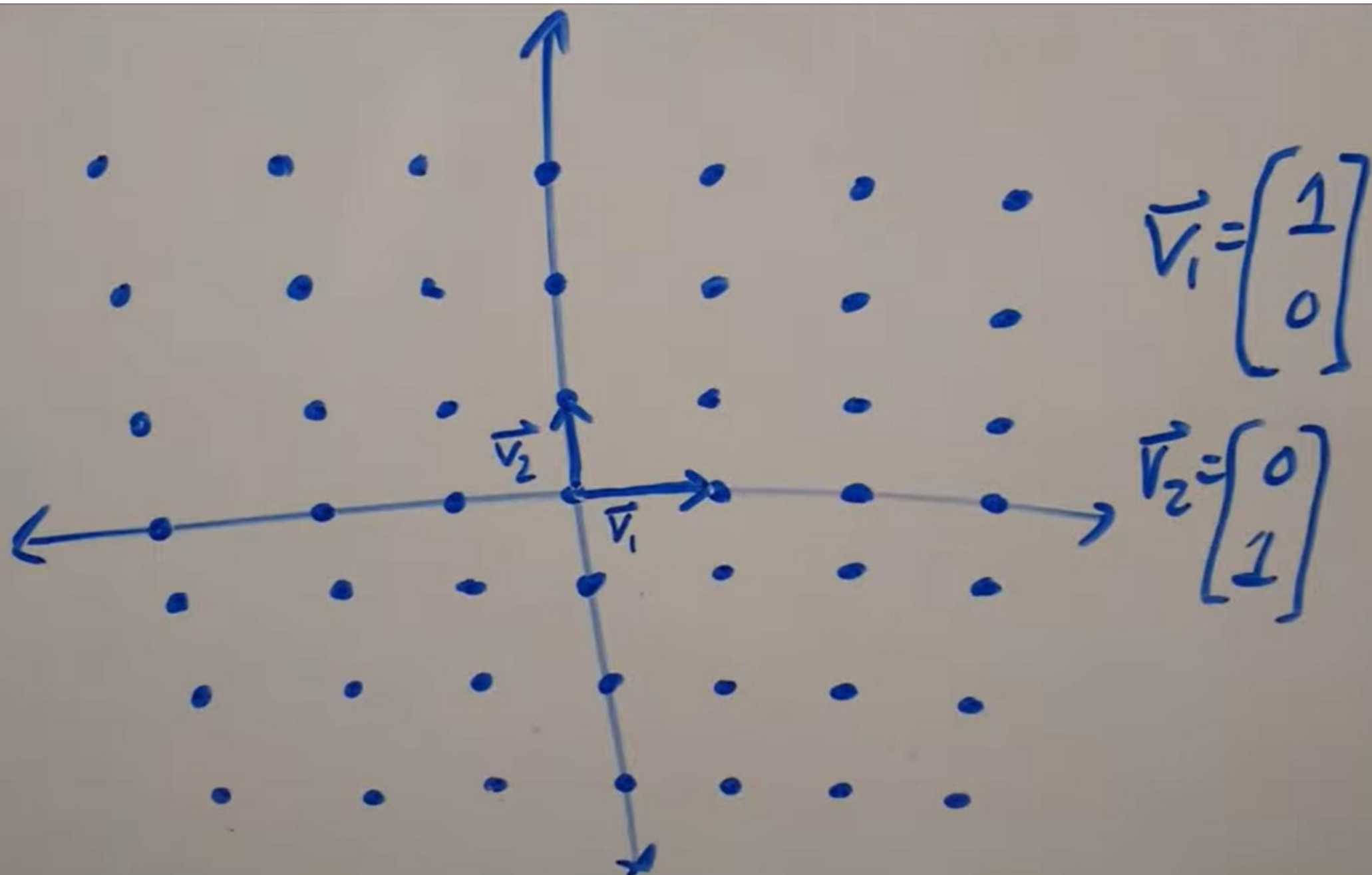
$$\vec{v}_2 = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \end{bmatrix}$$

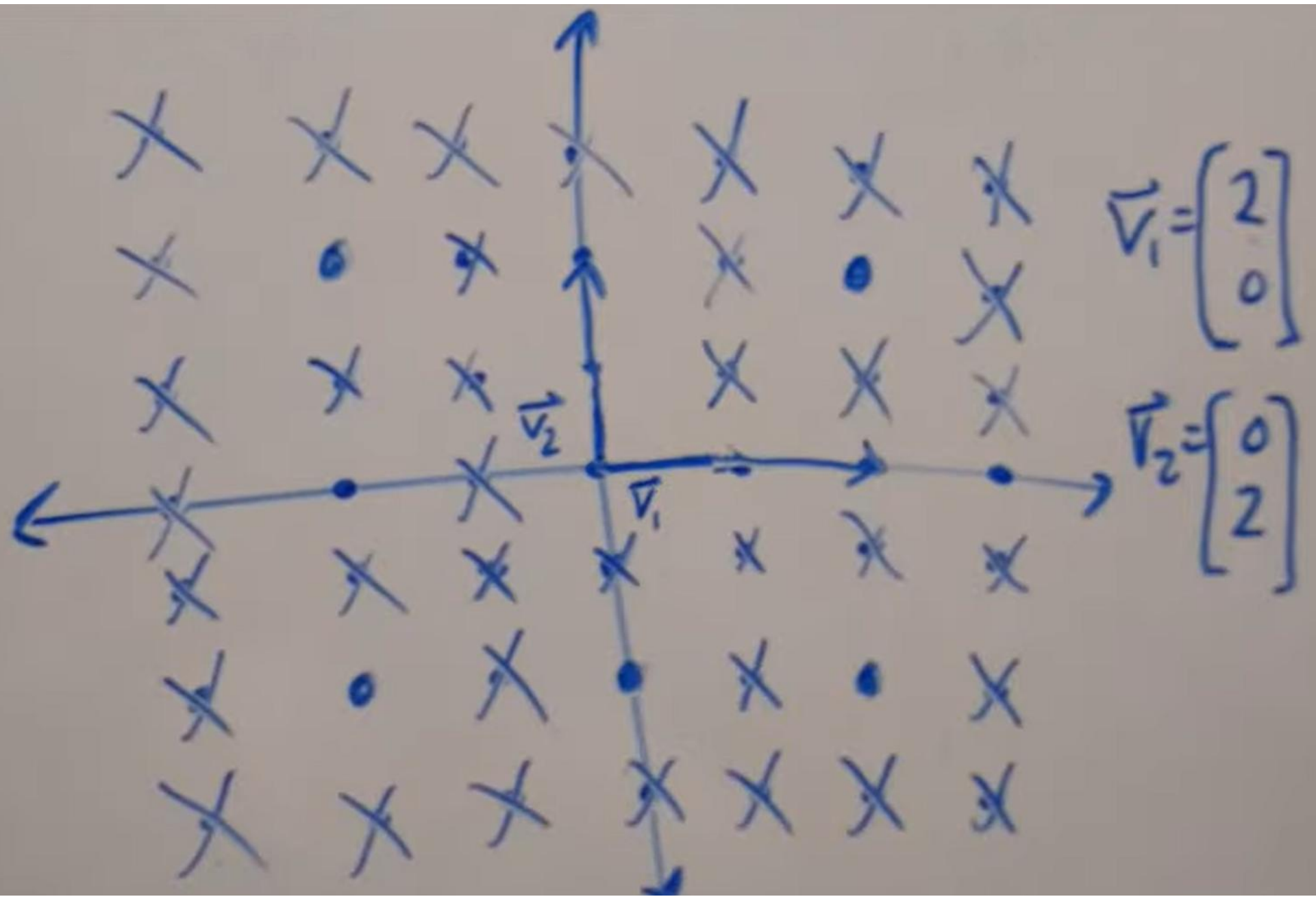
basis

$$\{\vec{v}_1, \vec{v}_2\}$$

then $a_j, b_j \in \mathbb{Z}$
for $1 \leq j \leq i$

$$\vec{v} = x \vec{v}_1 + y \vec{v}_2 \quad \text{where } \vec{v} \in \text{lattice} \\ \text{if } x, y \in \mathbb{Z}$$





$$\vec{v}_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$\vec{v}_2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

basis: $\{ \vec{v}_1, \vec{v}_2, \vec{v}_3, \dots, \vec{v}_n \}$

where $\vec{v}_1, \dots, \vec{v}_n$ all have m entries
for $m \geq n$

$$\text{matrix } A = \left[\begin{array}{cccc} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 & \dots & \vec{v}_n \end{array} \right] \begin{matrix} \overbrace{\hspace{1.5cm}}^n \\ \underbrace{\hspace{1.5cm}}_m \end{matrix}$$

Shortest Vector Problem: Find the shortest non-zero vector in a lattice given the basis of that lattice

Closest Vector Problem: Given a basis of a lattice and a vector, v , *not* in the lattice, find the closest vector to v that is in the lattice

Bounded Distance Decoding: Similar to the Closest Vector Problem, find the lattice point, s , closest given that v is known to be close to s

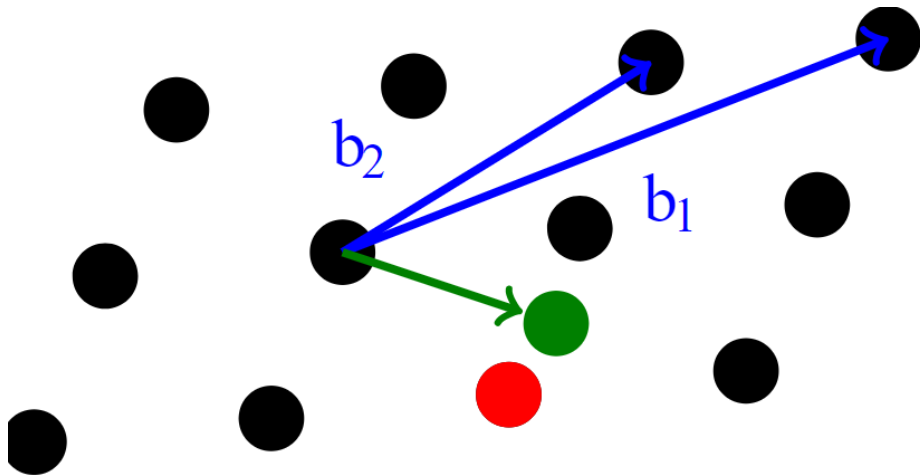
Covering Radius Problem: Given a basis for a lattice find the smallest sphere that when placed at every lattice point it includes 2 lattice points

Covering radius problem (CRP): Given a basis for the lattice, the algorithm must find the largest distance (or in some versions, its approximation) from any vector to the lattice.

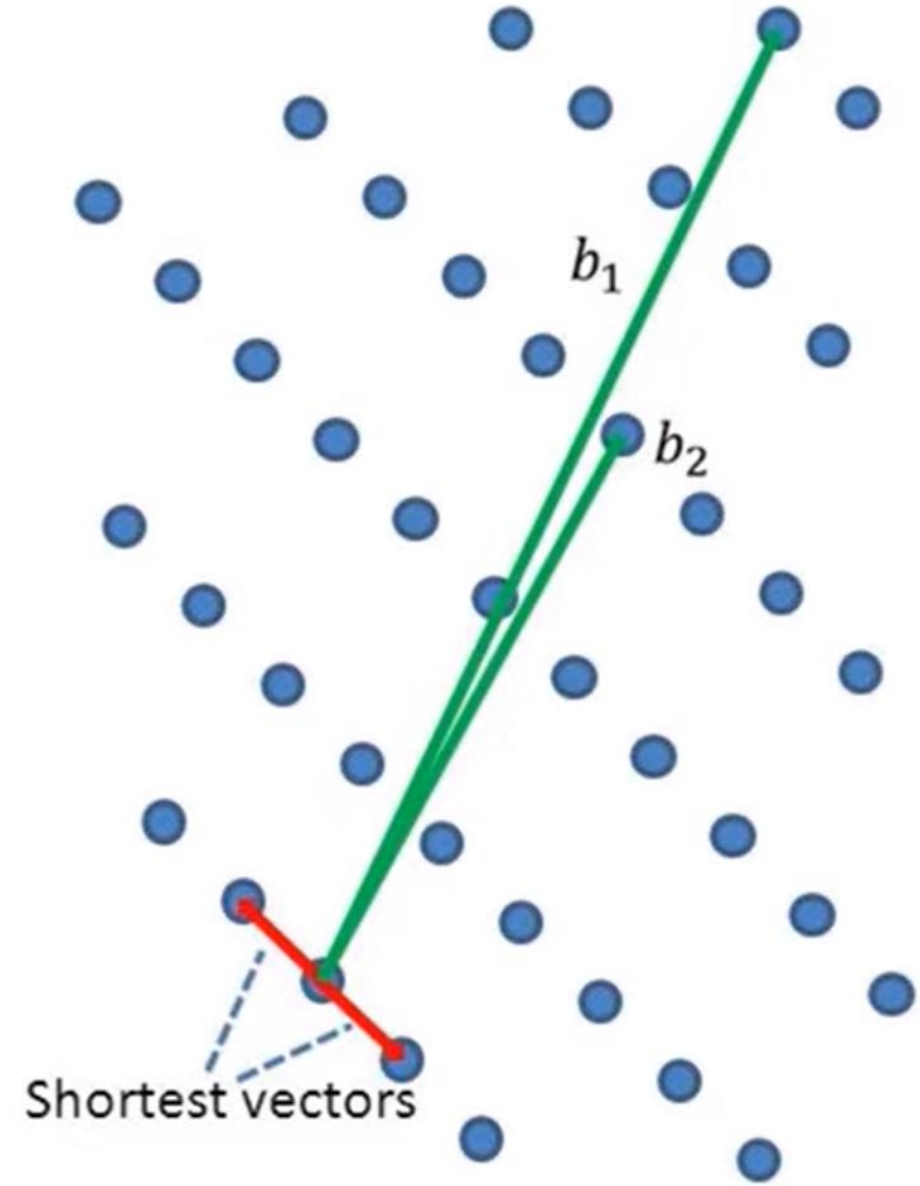
hardness of SVP implies the same hardness for CVP [Goldreich et al]

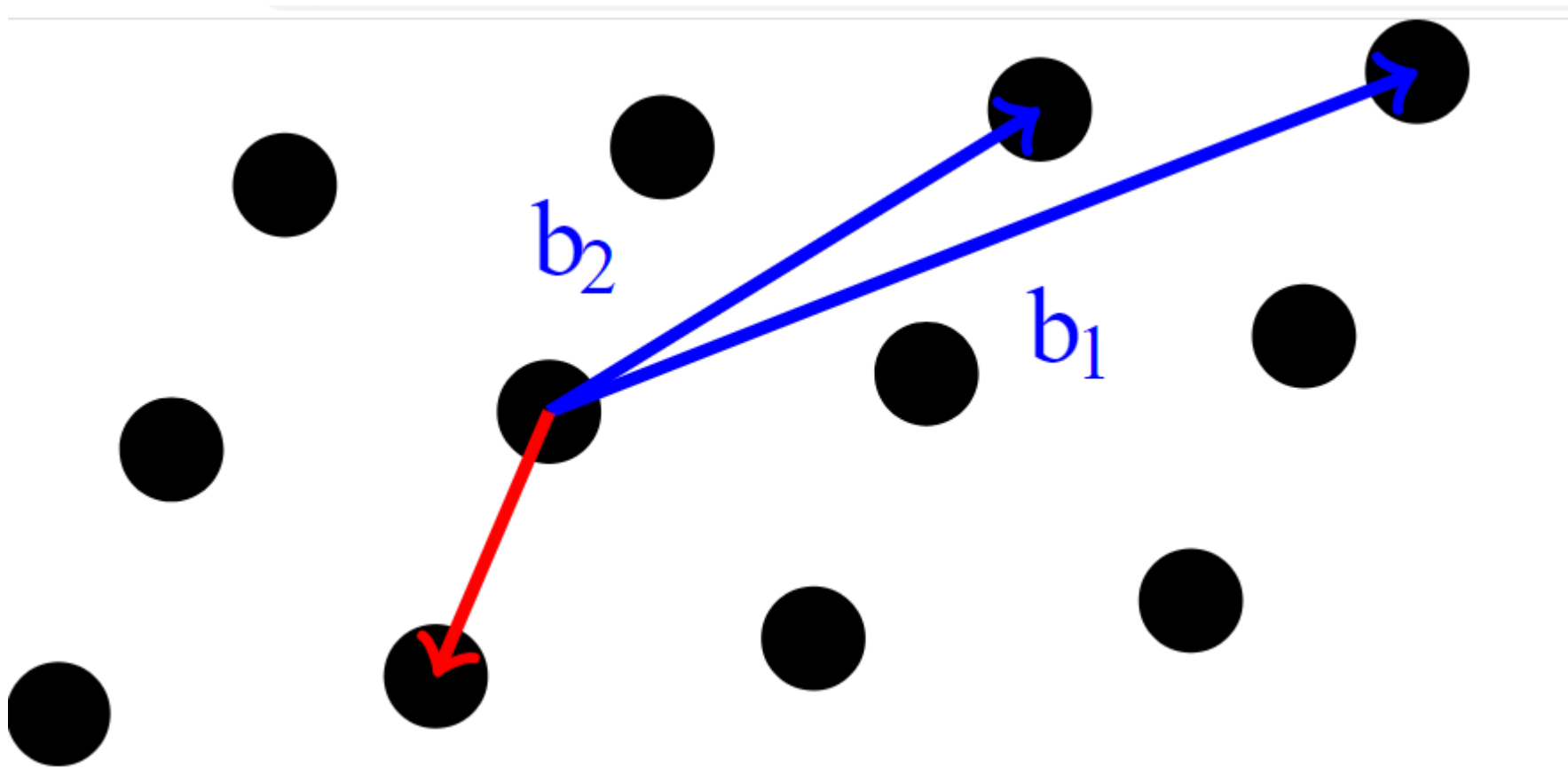
```
✓ 0s
▶ from scipy import spatial
A = [[0,1,2,3,4], [4,3,2,1,0], [2,5,3,7,1], [1,0,1,0,1]]
tree = spatial.KDTree(A)
print(A[tree.query([0.5,0.5,0.5,0.5,0.5])[1]])

[1, 0, 1, 0, 1]
```



CVP: closest vector problem (basis vectors in blue, external vector in green, closest vector in red).





SVP: shortest vector problem (basis vectors in blue, shortest vector in red)

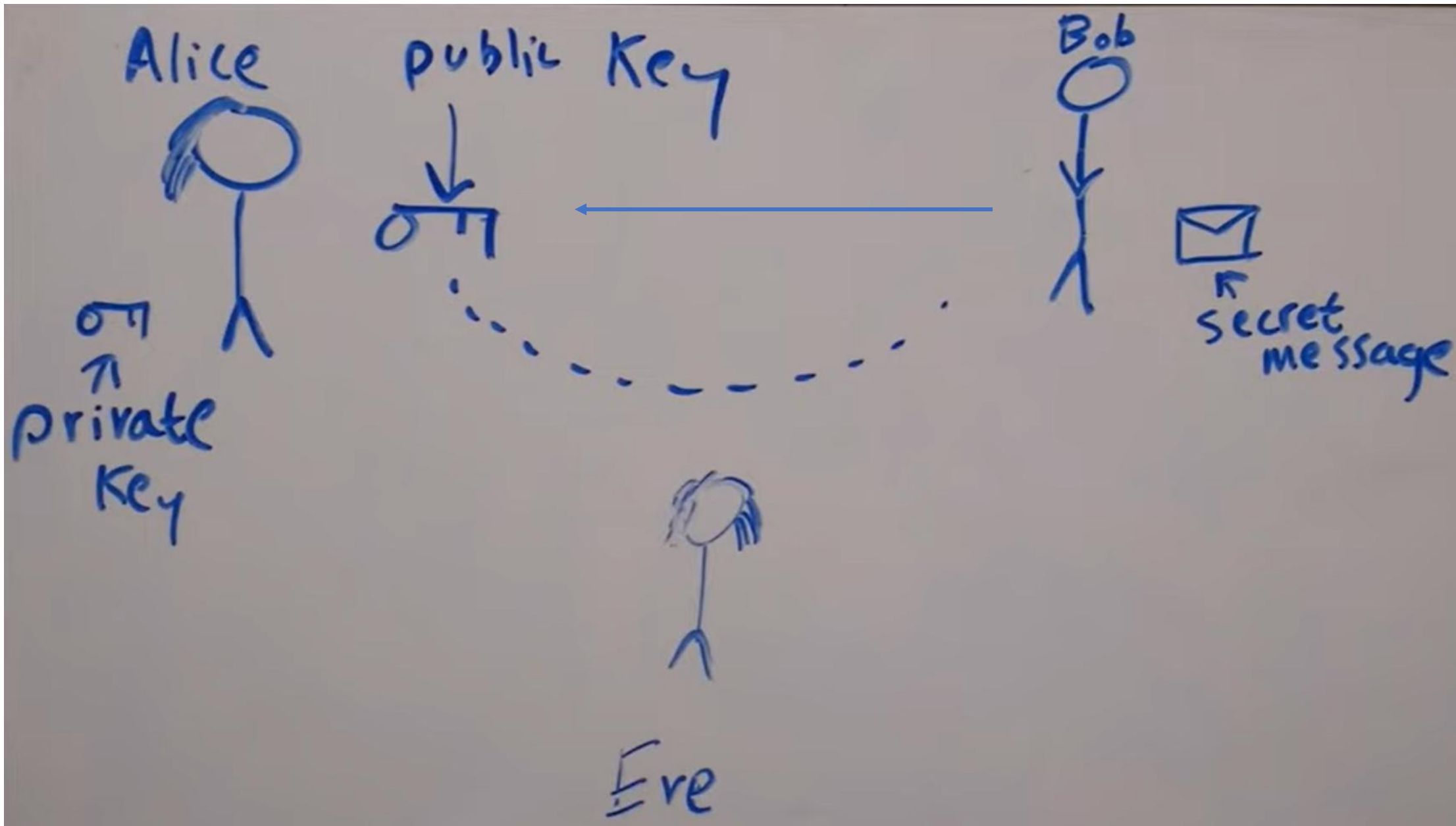
The closest vector problem is a generalization of the shortest vector problem. It is easy to show that given an oracle for CVP (defined below), one can solve SVP by making some queries to the oracle. The naive method to find the shortest vector by calling the CVP oracle to find the closest vector to 0 does not work because 0 is itself a lattice vector and the algorithm could potentially output 0.

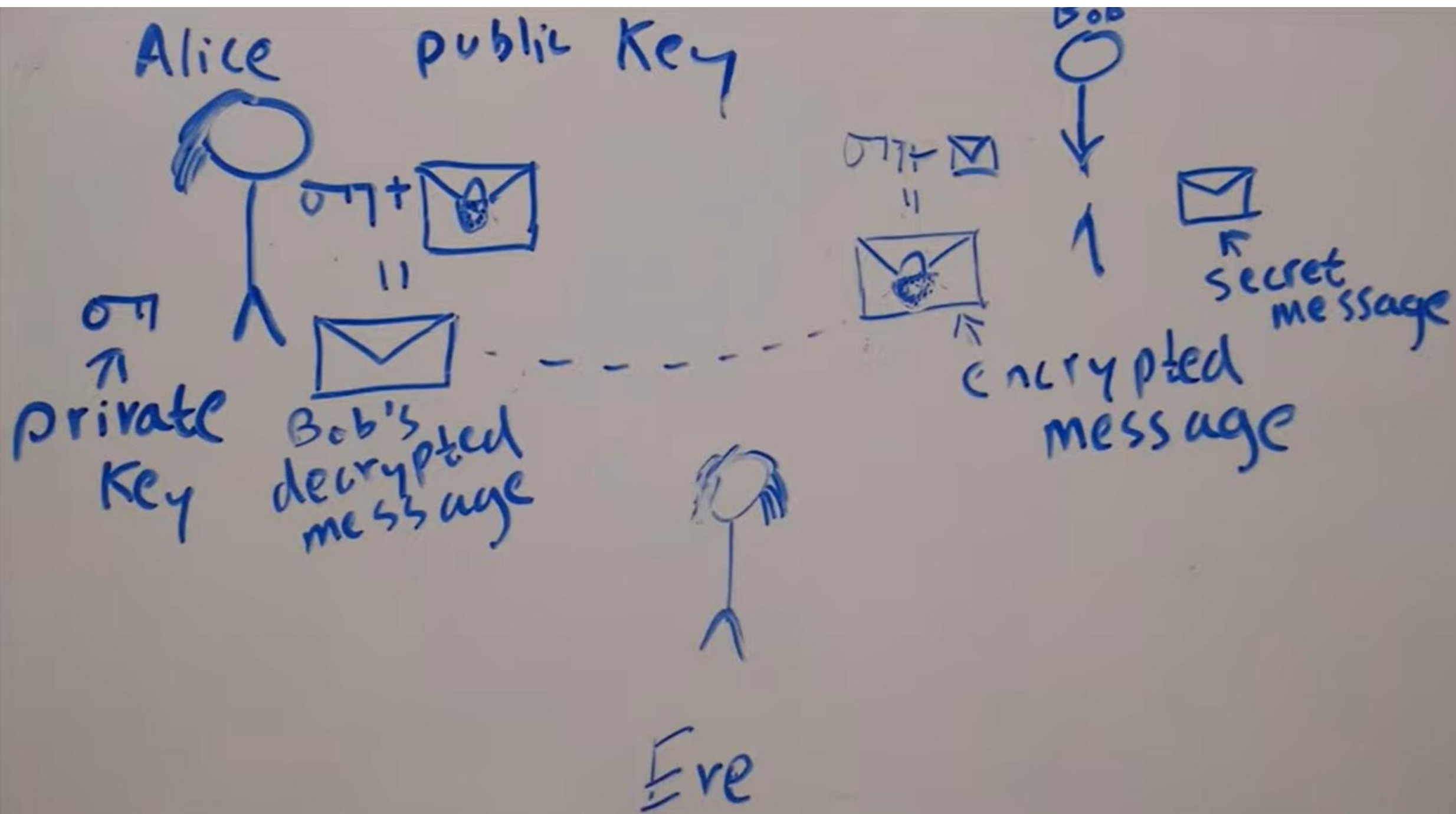
Shortest vector problem (SVP). The *SVP* is to find $x \in \Lambda(b_1, \dots, b_n) - \{0\}$ such that $\|x\|$ is smallest. Note that this can be in any metric. The only known algorithms are all in exponential time.

Shortest independent vector problem (SIVP). In the SIVP, we seek to minimize the length of the longest vector in the basis. In other words, we seek to find a new basis which yields the same lattice, but minimizes the length of the longest vector.

Dual lattice. We define the *dual lattice* Λ^* of lattice Λ , as the set of vectors whose inner product yields an integer for all points in the lattice. That is,

$$\Lambda^* = \{y \in \mathbf{R}^n \mid x^T y \in \mathbb{Z} \ \forall x \in \Lambda\}$$





public: A and q

Alice & Bob chooses a random Matrix $A^{(m \times n)}$

$$A^{m \times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \dots & \dots & \dots \\ \vdots & & & \\ a_{m1} & \dots & \dots & a_{mn} \end{bmatrix} \text{ where } a_{ij} < q \text{ and } a_{ij} \in \mathbb{Z}$$

Alice private: \vec{x} (Private Vector)

Alice public: $\vec{v} = A\vec{x}$

Alice sends to Bob

$$\vec{x} \stackrel{?}{=} A^{-1}\vec{v}$$

Collison Resistant Hash Function

Bob sends an encrypted message to Alice using PubKey

Bob
public: \vec{b}_1, b_2

$$\vec{b}_1 = A\vec{s} + \vec{e}_1$$

private: \vec{s}, \vec{e}_1, e_2 $b_2 = \vec{s} \cdot \vec{u} + e_2 + \text{bit} \cdot \frac{q}{2}$

bit = 0 or 1 $e \ll \frac{q}{4}$

Alice can find if Bob sends 0 or 1

Decryption

Ax

$$\vec{b}_1 \cdot \vec{x} = \vec{s} \cdot \vec{u} + \vec{e}_1 \cdot \vec{x}$$

$$b_2 - \vec{b}_1 \cdot \vec{x} = \vec{s} \cdot \vec{u} + e_2 + \text{bit} \cdot \frac{q}{2} - \vec{s} \cdot \vec{u} + \vec{e}_1 \cdot \vec{x}$$

$$(e_2 - \vec{e}_1 \cdot \vec{x}) + \text{bit} \cdot \frac{q}{2}$$

$$(e_2 - \vec{e}_1 \cdot \vec{x}) + \underbrace{\text{bit} \cdot \frac{a}{2}}_{\text{only small if bit}=0}$$

very small

If Bob sends 0 then result will be closed to 0

Or if he sends 1 then result will be closed to 1