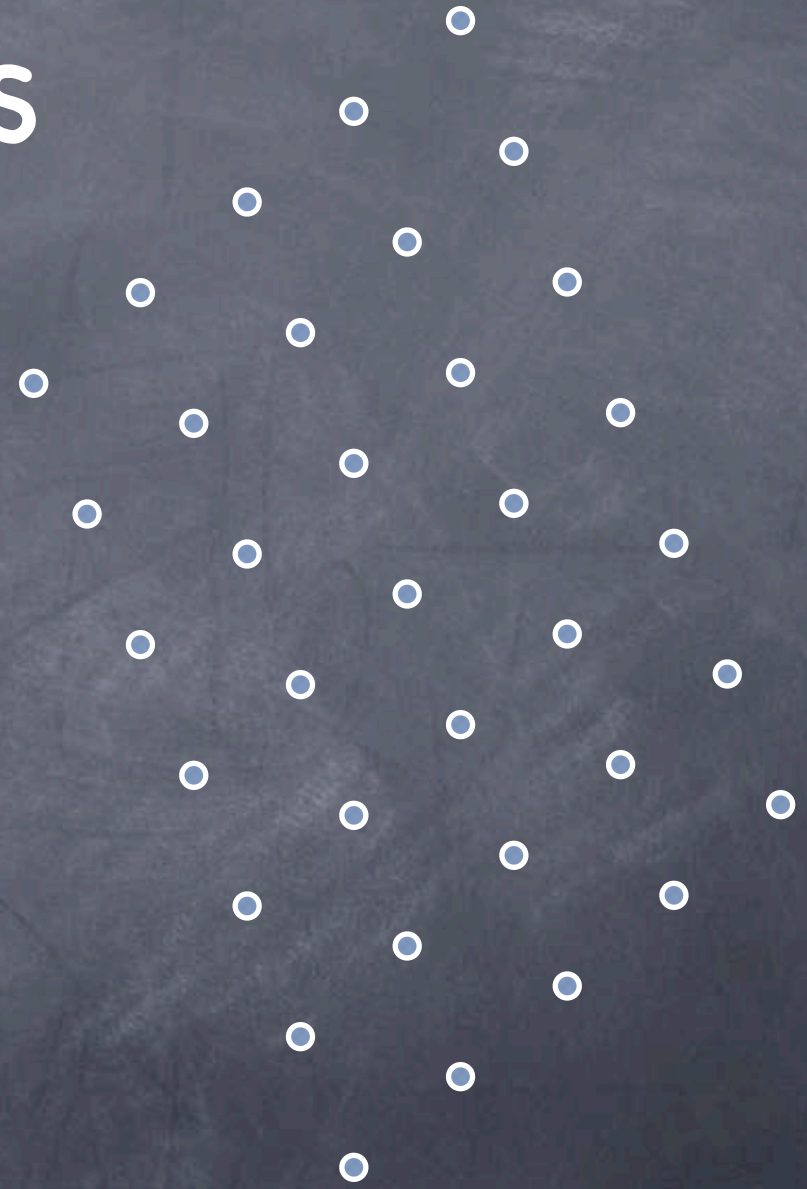


# Lattice Cryptography

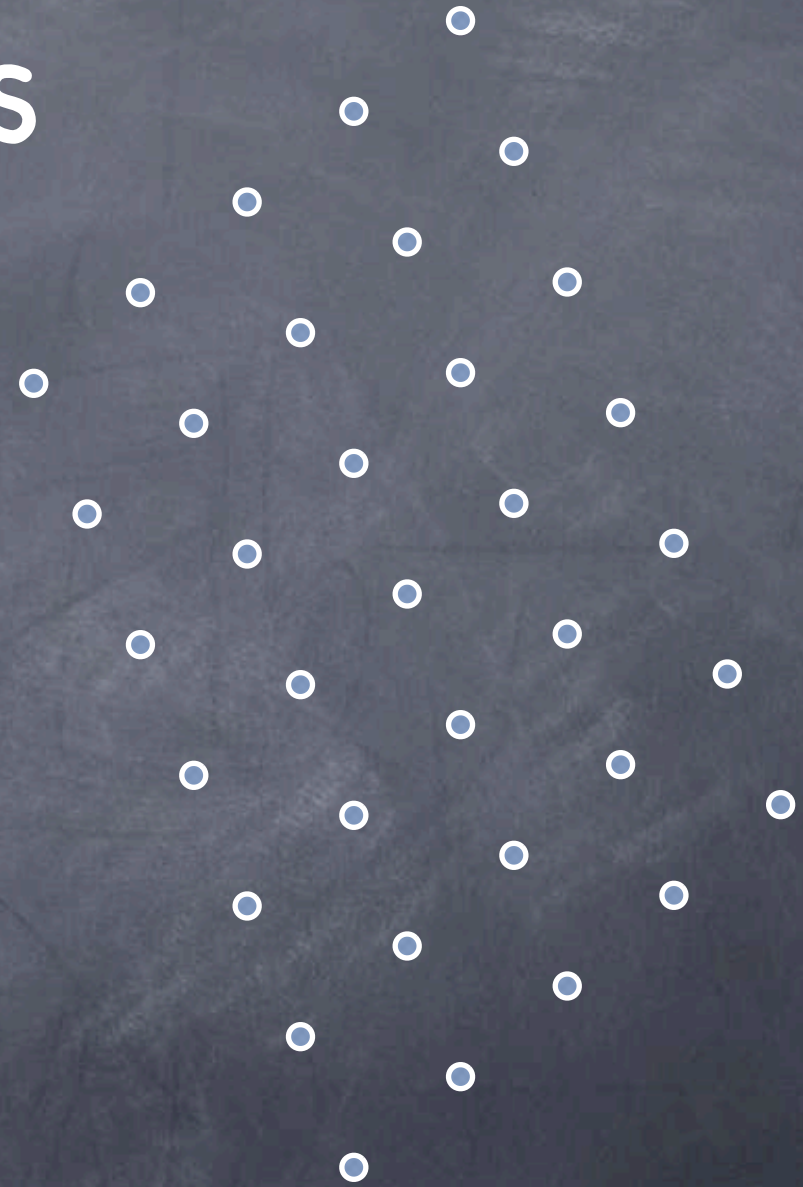
Lecture 24

# Lattices



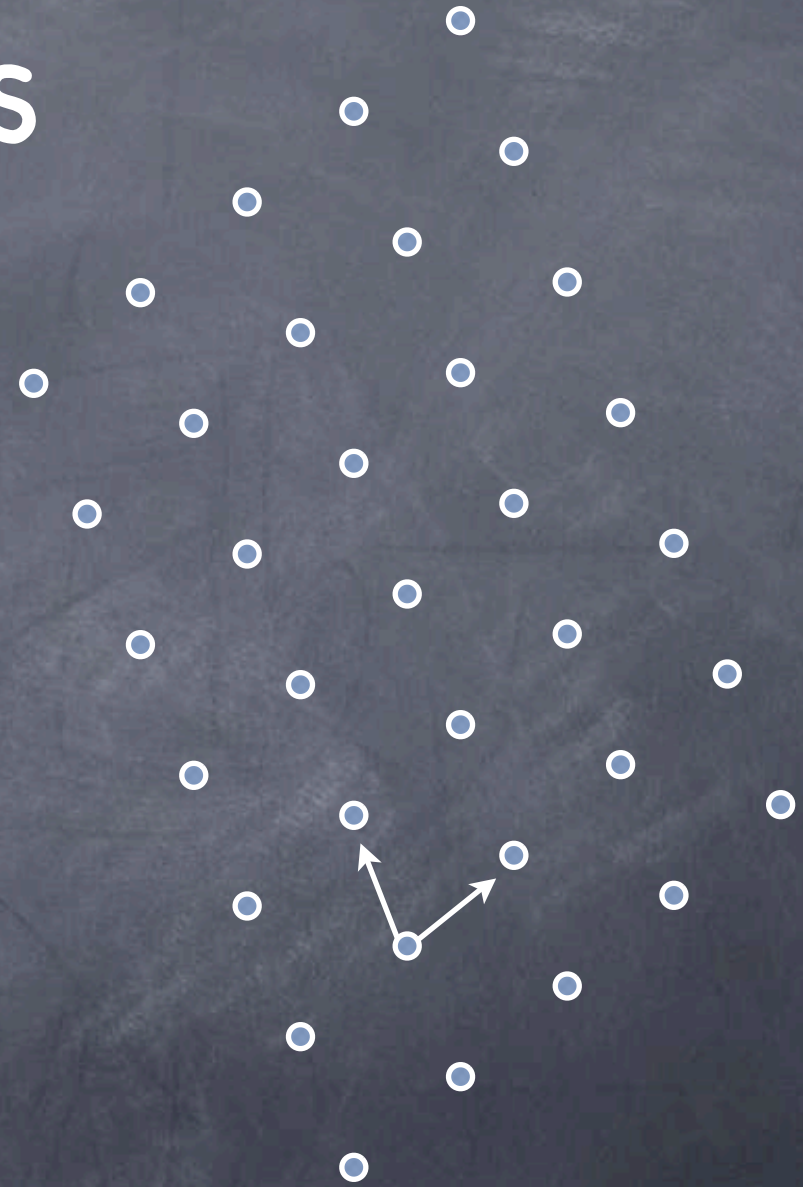
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"



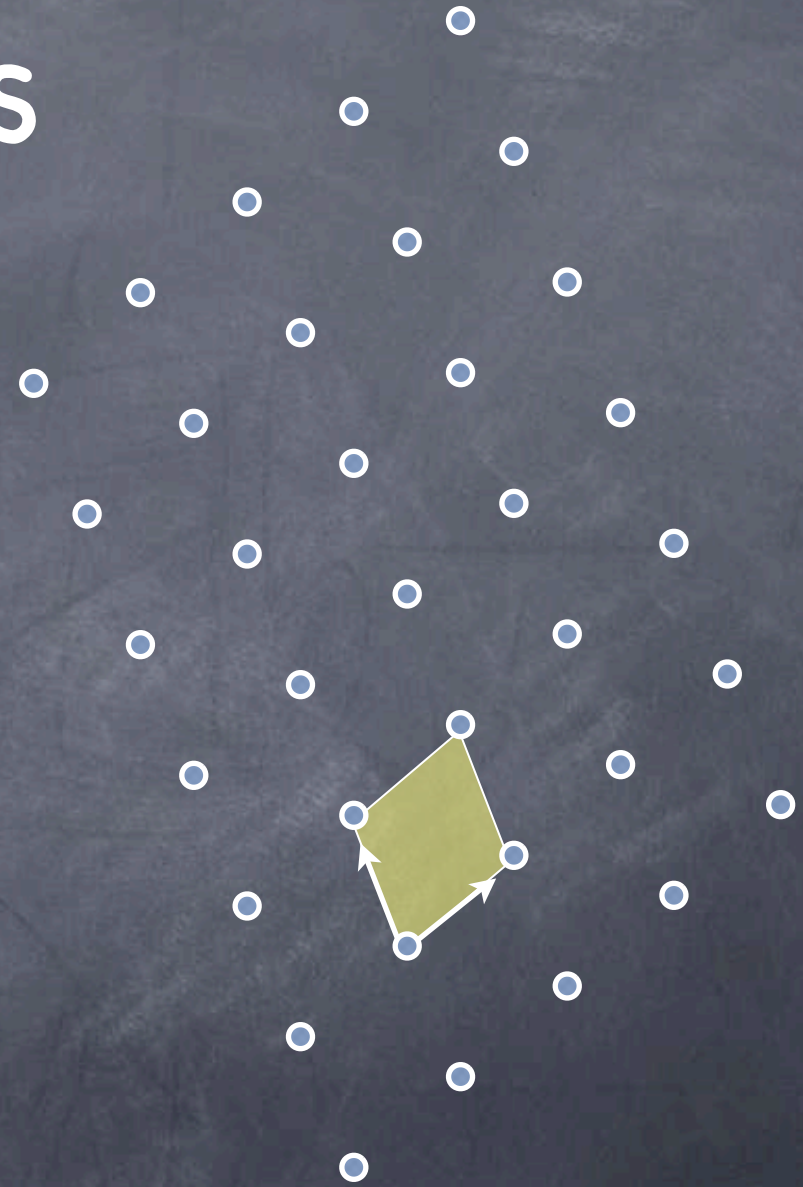
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"



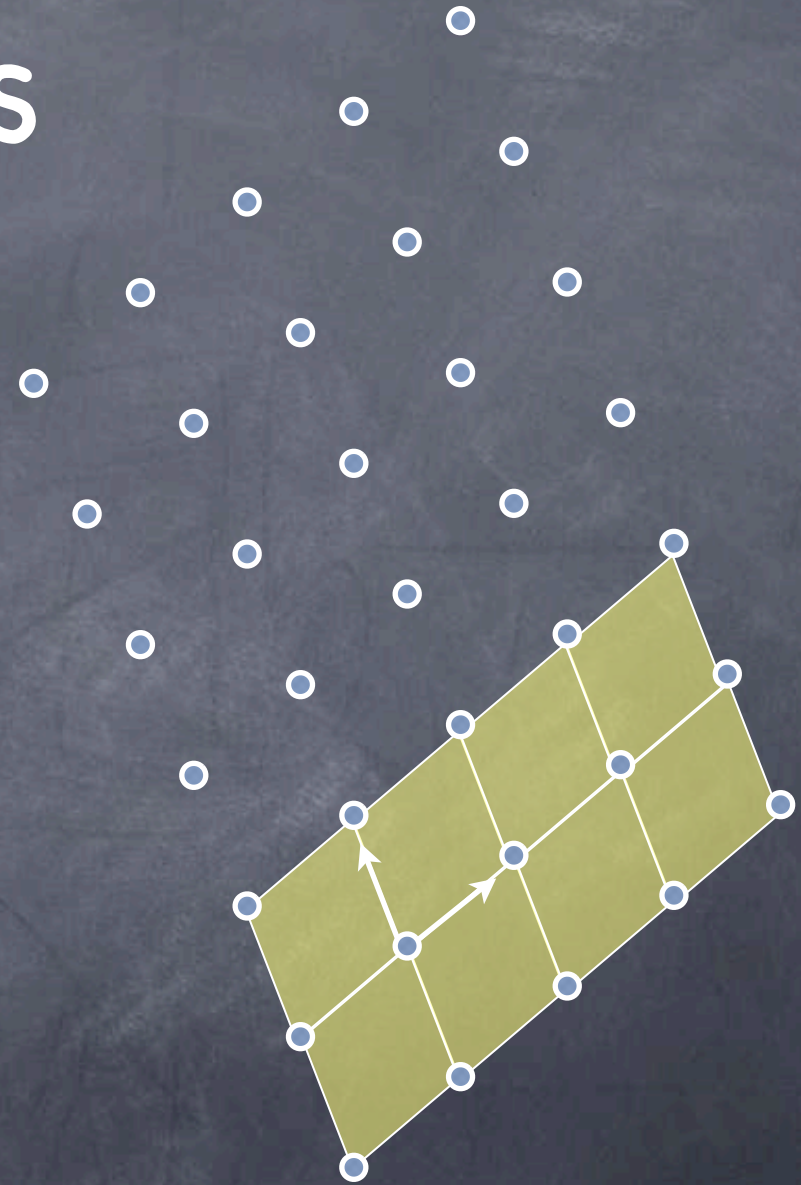
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"



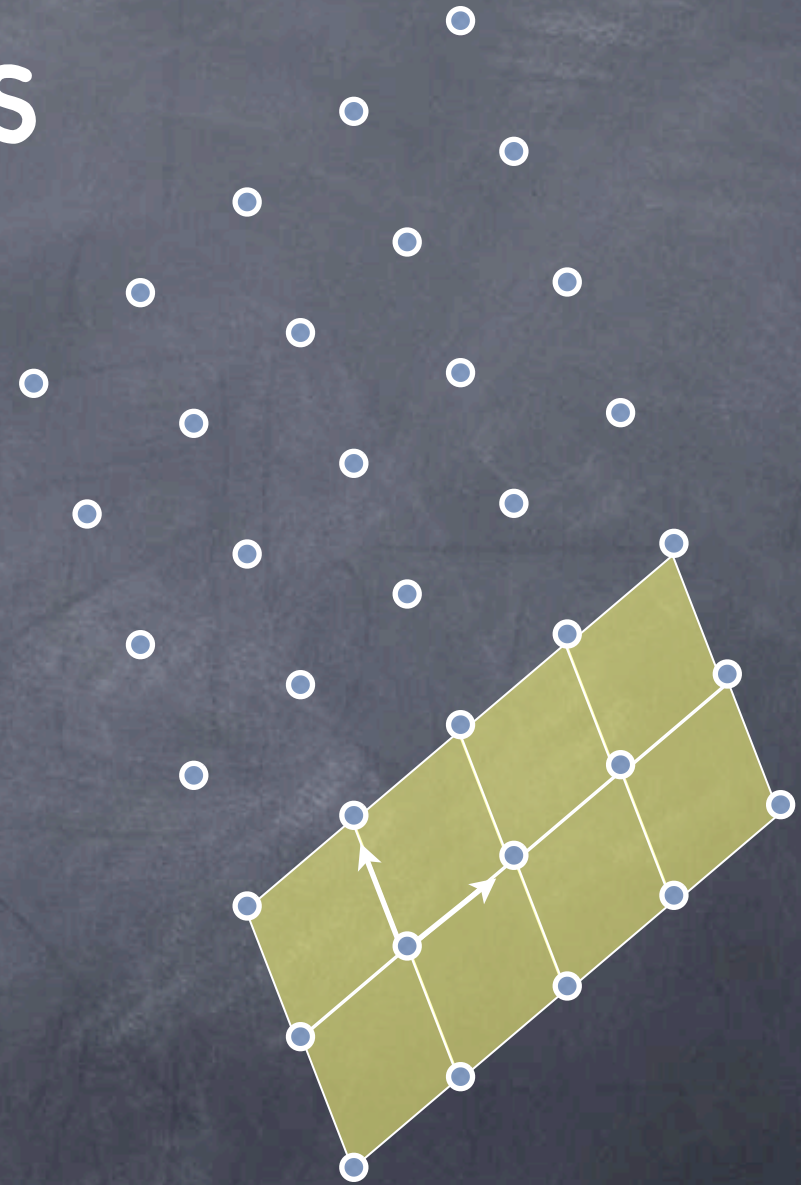
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"



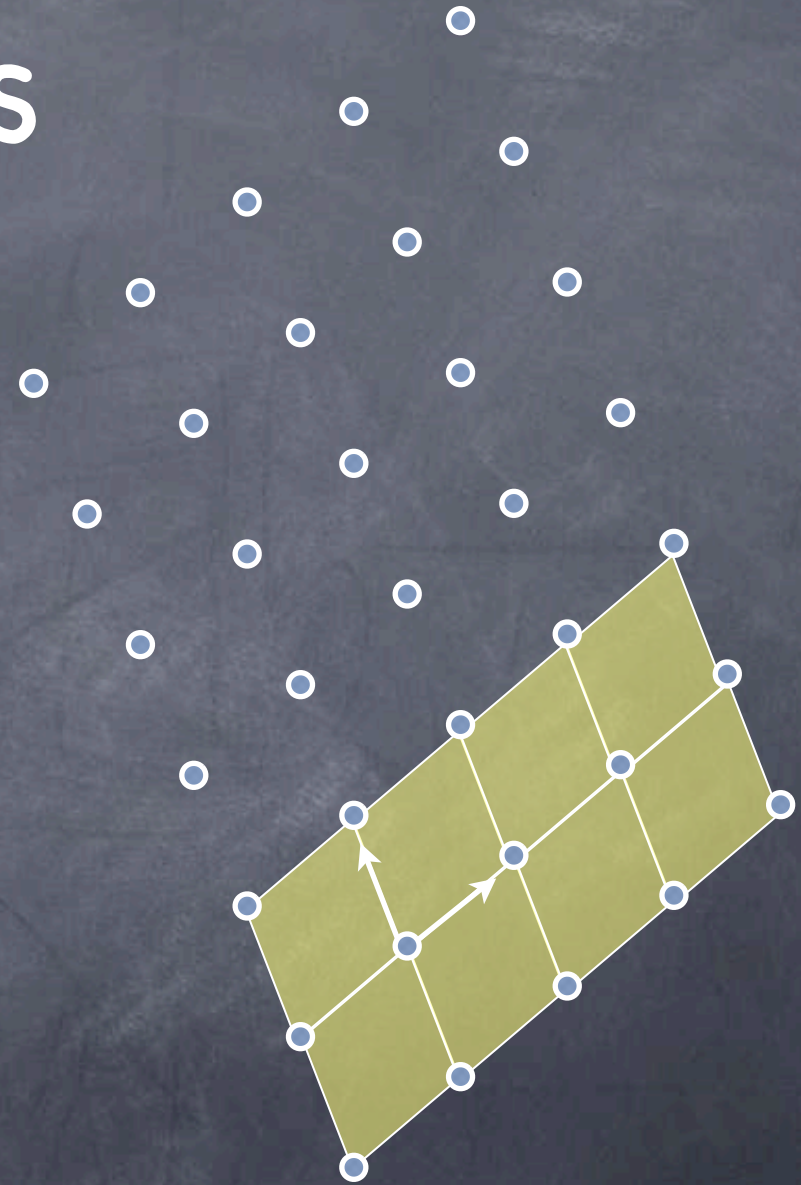
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
- Formally,  $\{ \sum_i x_i \underline{\mathbf{b}}_i \mid x_i \text{ integers} \}$



# Lattices

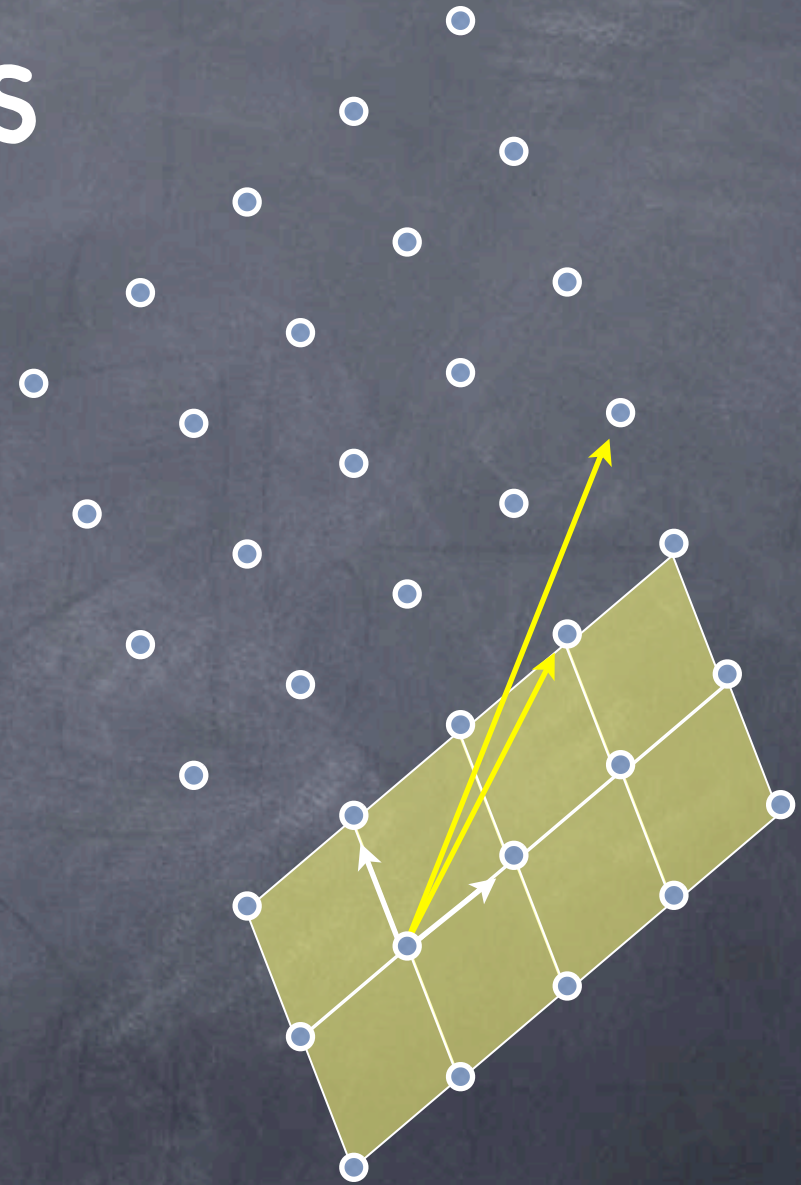
- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
- Formally,  $\{ \sum_i x_i \underline{\mathbf{b}}_i \mid x_i \text{ integers} \}$
- Basis is not unique





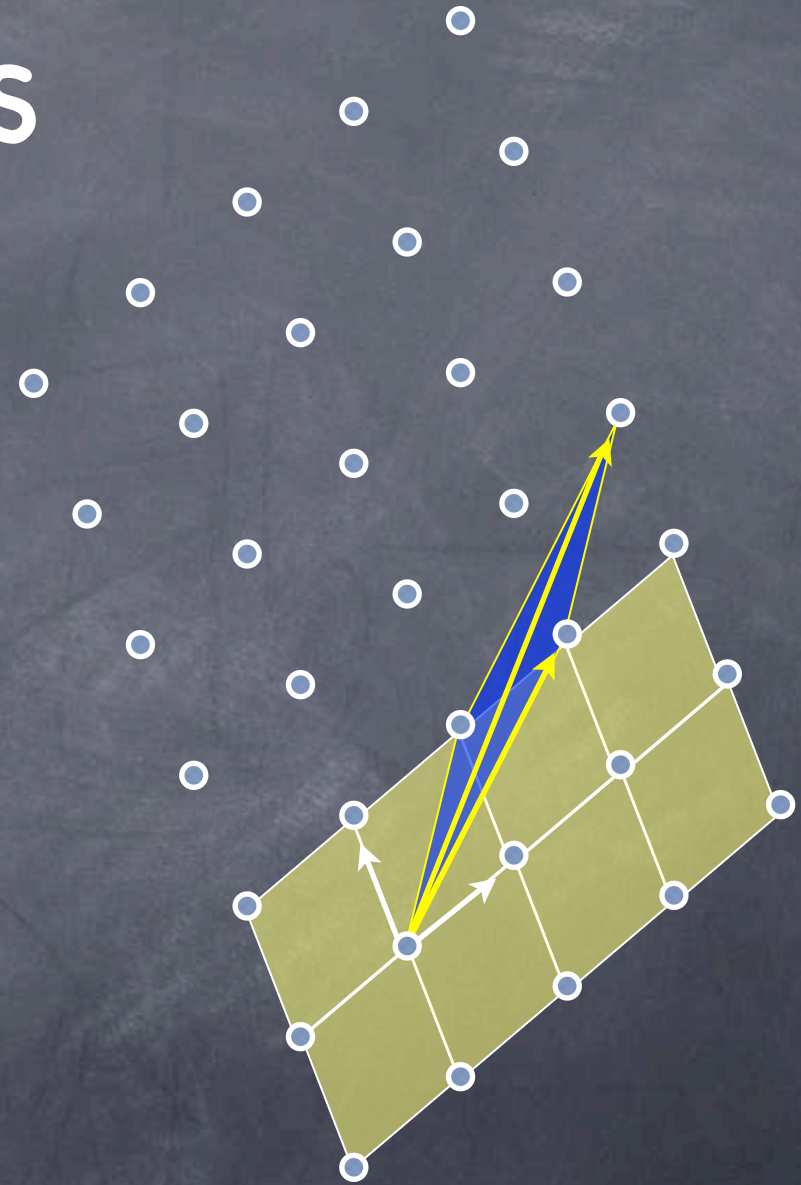
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
- Formally,  $\{ \sum_i x_i \underline{\mathbf{b}}_i \mid x_i \text{ integers} \}$
- Basis is not unique



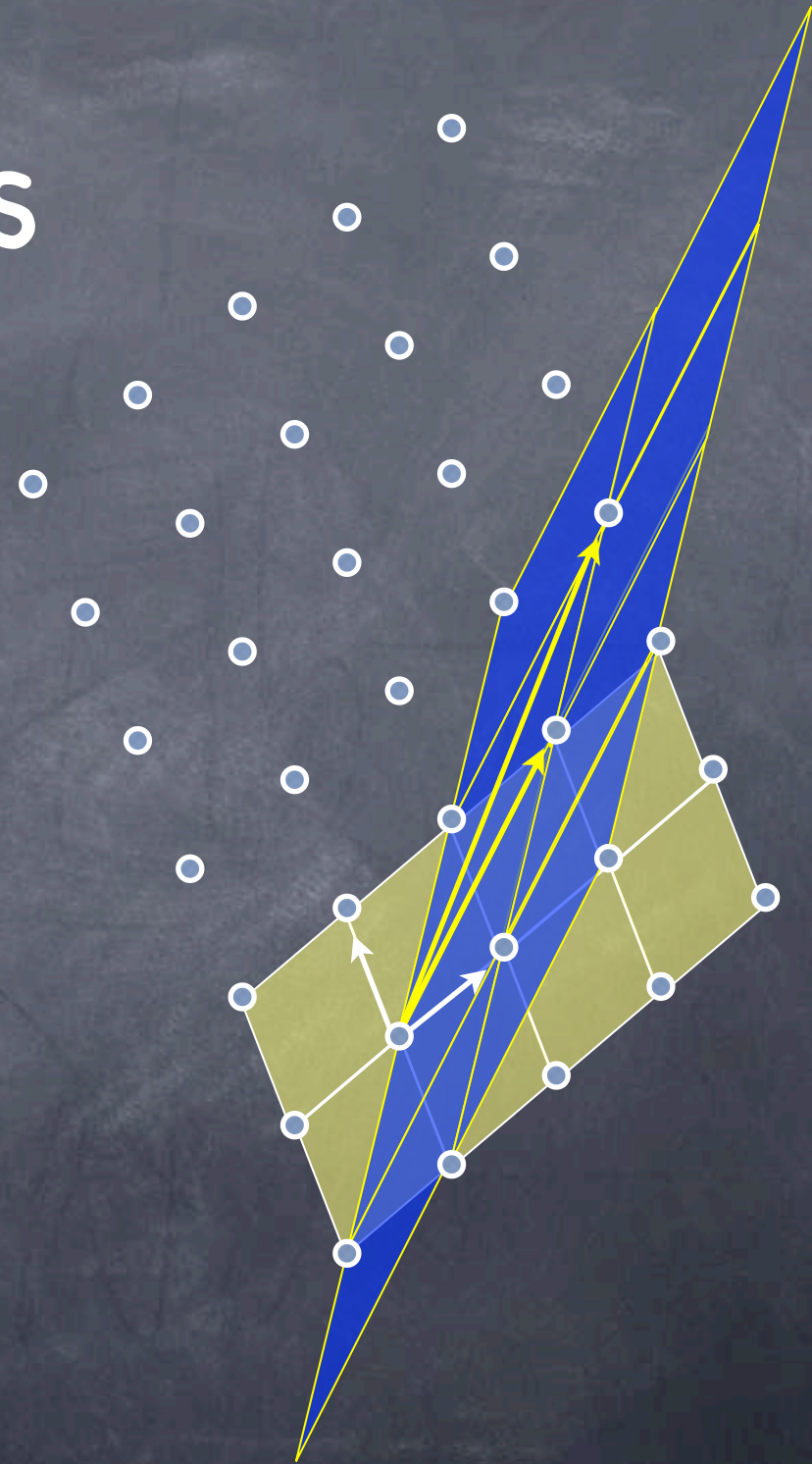
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
- Formally,  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- Basis is not unique



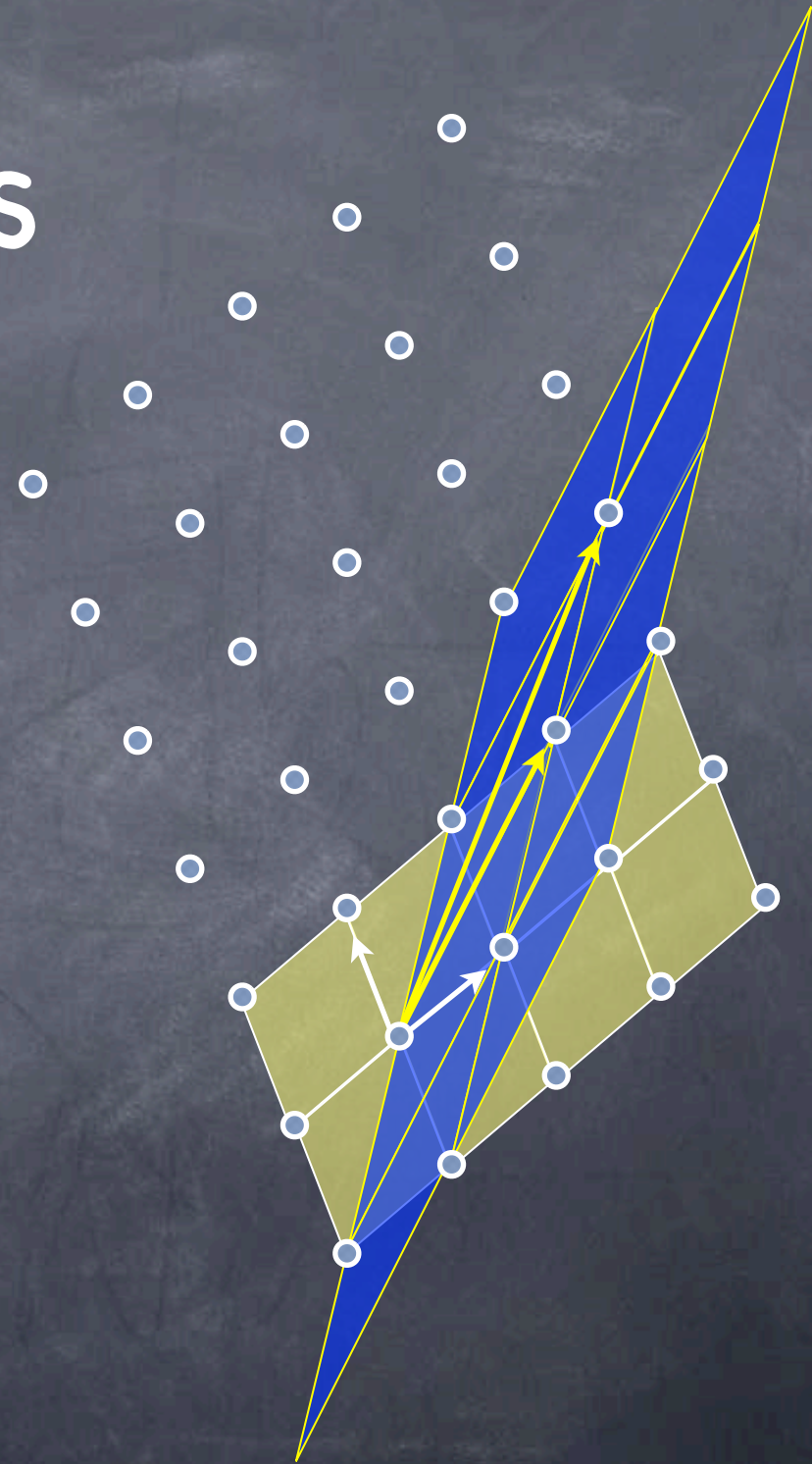
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
- Formally,  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- Basis is not unique



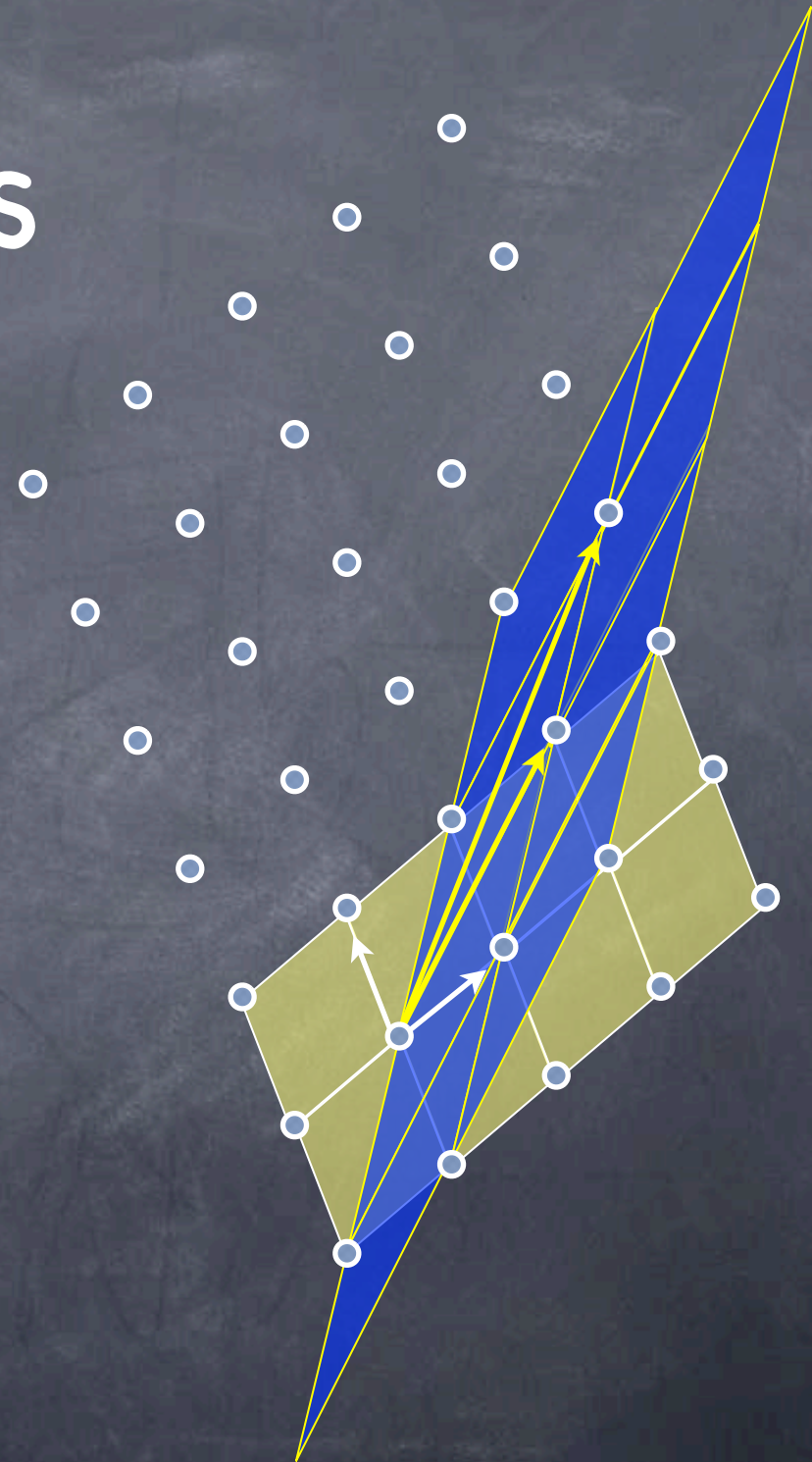
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
  - Formally,  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- Basis is not unique
- Several problems related to high-dimensional lattices are believed to be hard, with cryptographic applications



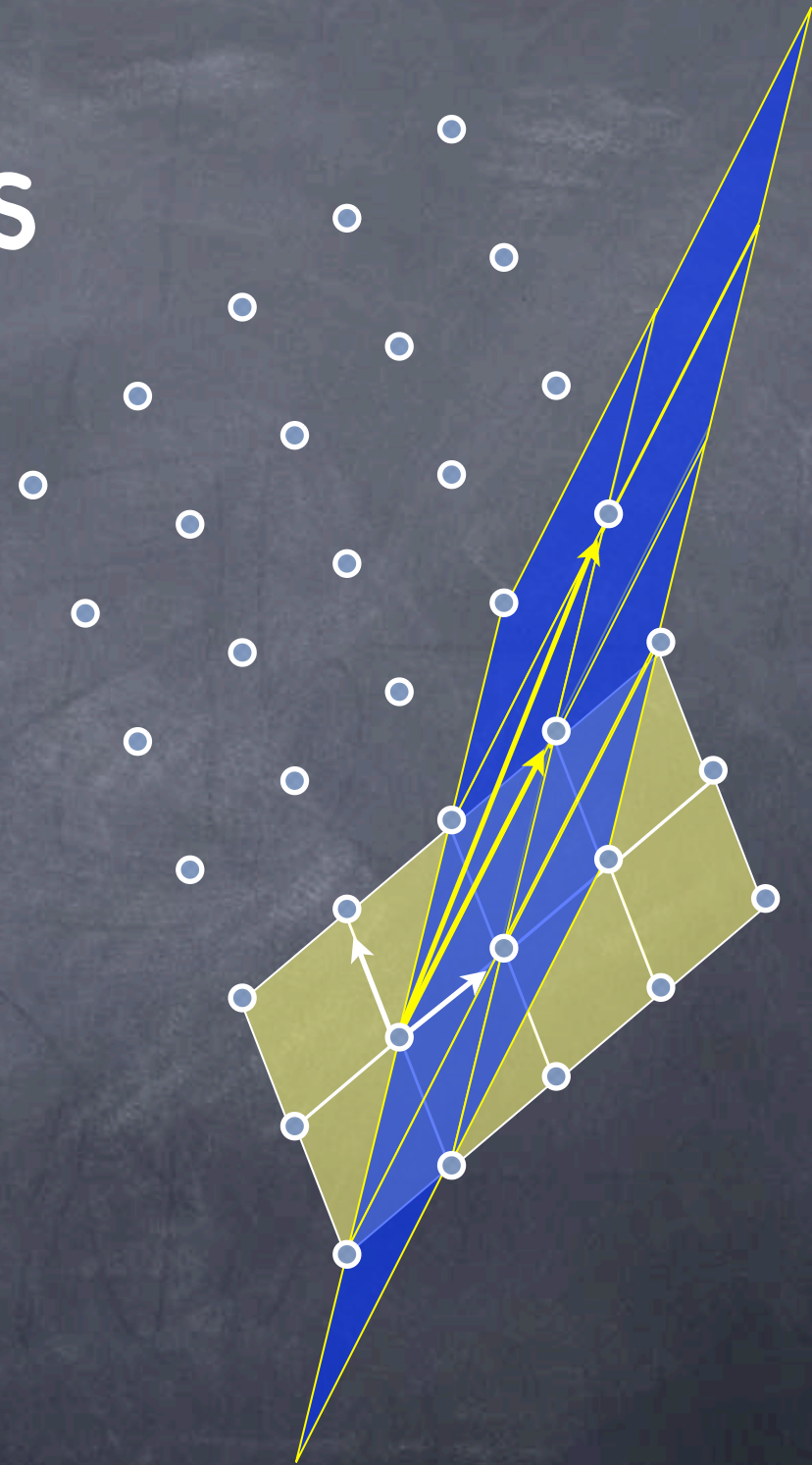
# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a "basis"
  - Formally,  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- Basis is not unique
- Several problems related to high-dimensional lattices are believed to be hard, with cryptographic applications
  - Hardness assumptions are "milder" (worst-case hardness)

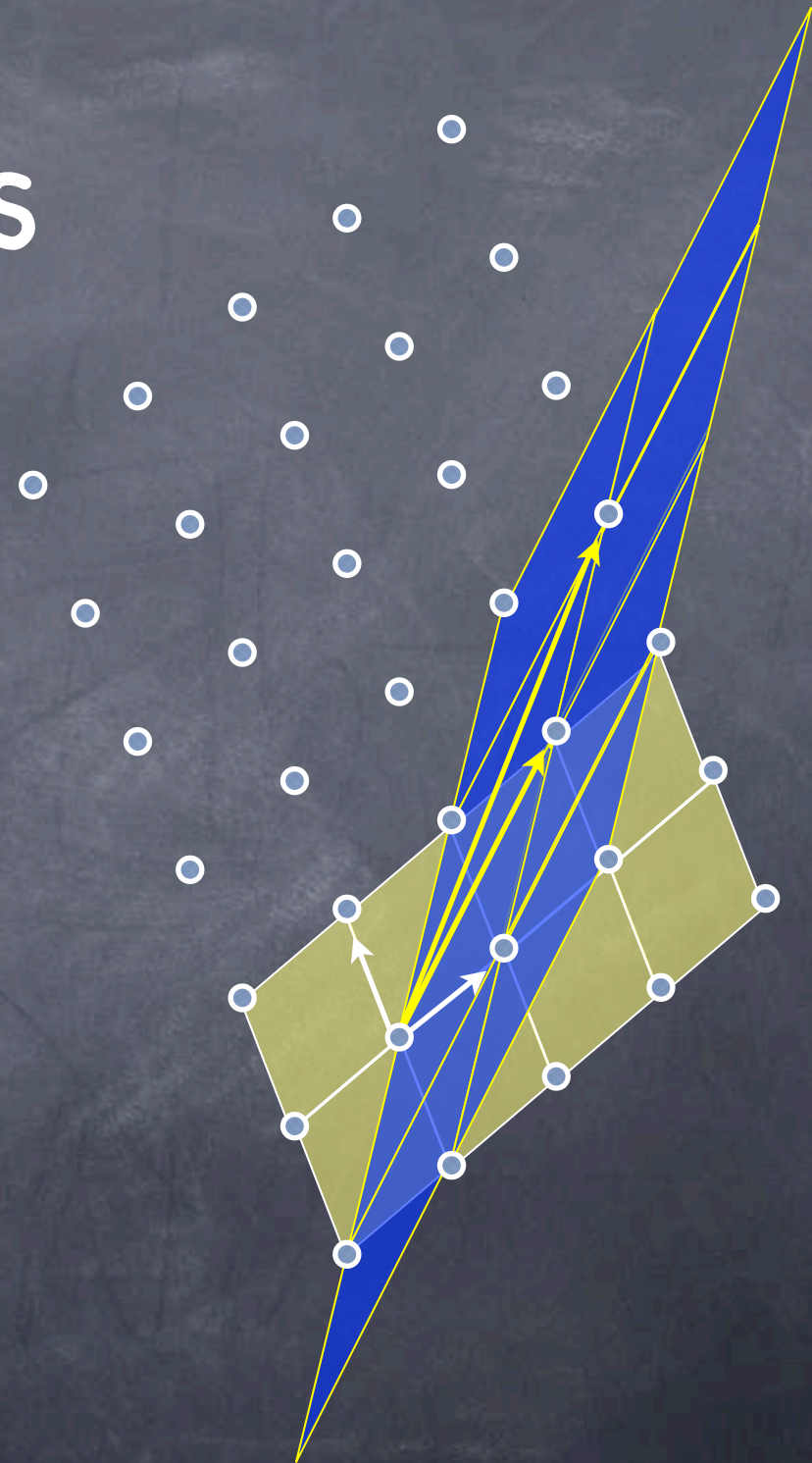


# Lattices

- A infinite set of points in  $\mathbb{R}^n$  obtained by tiling with a “basis”
  - Formally,  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- Basis is not unique
- Several problems related to high-dimensional lattices are believed to be hard, with cryptographic applications
  - Hardness assumptions are “milder” (worst-case hardness)
  - Believed to hold even against quantum computation: “Post-Quantum Cryptography”

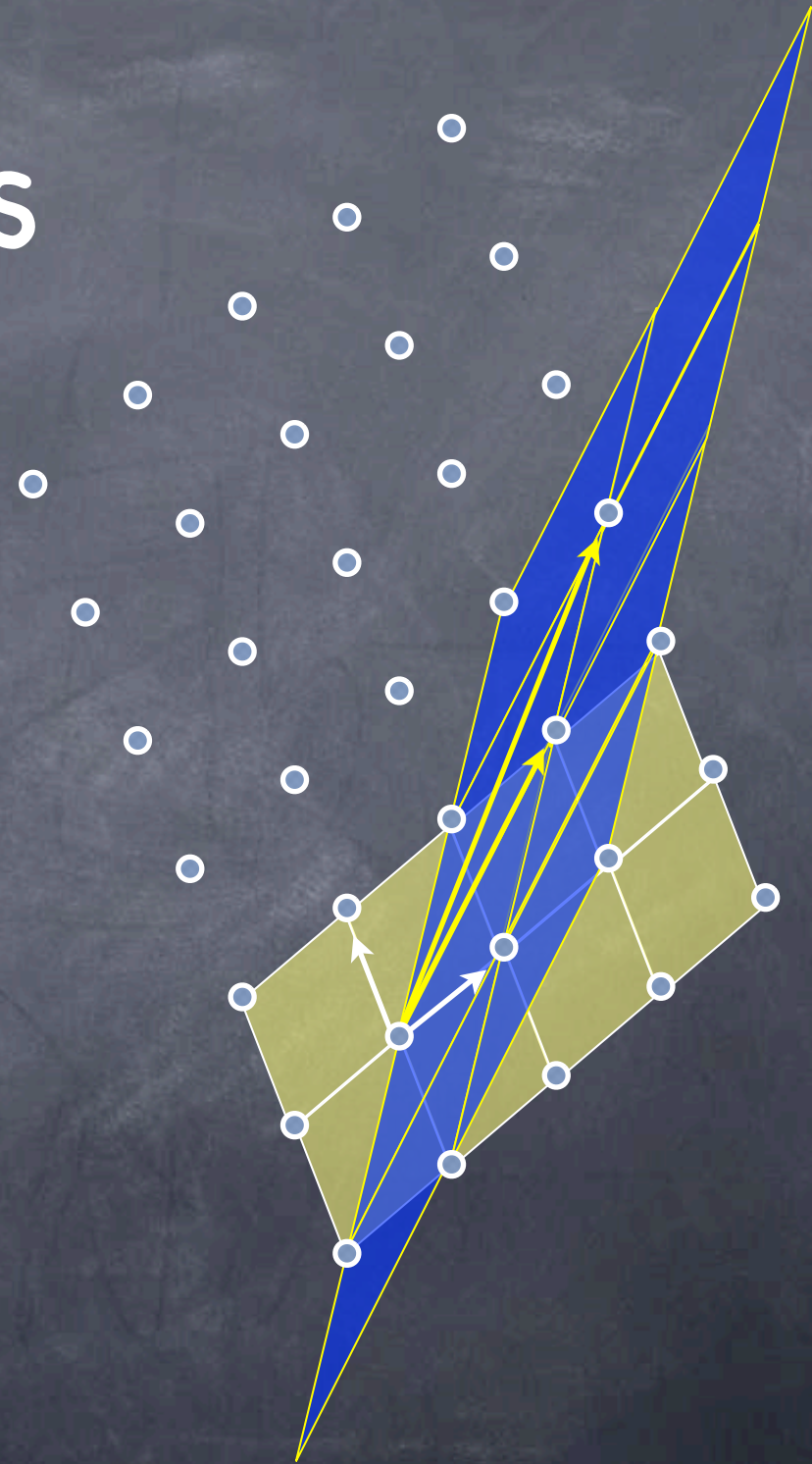


# Lattices



# Lattices

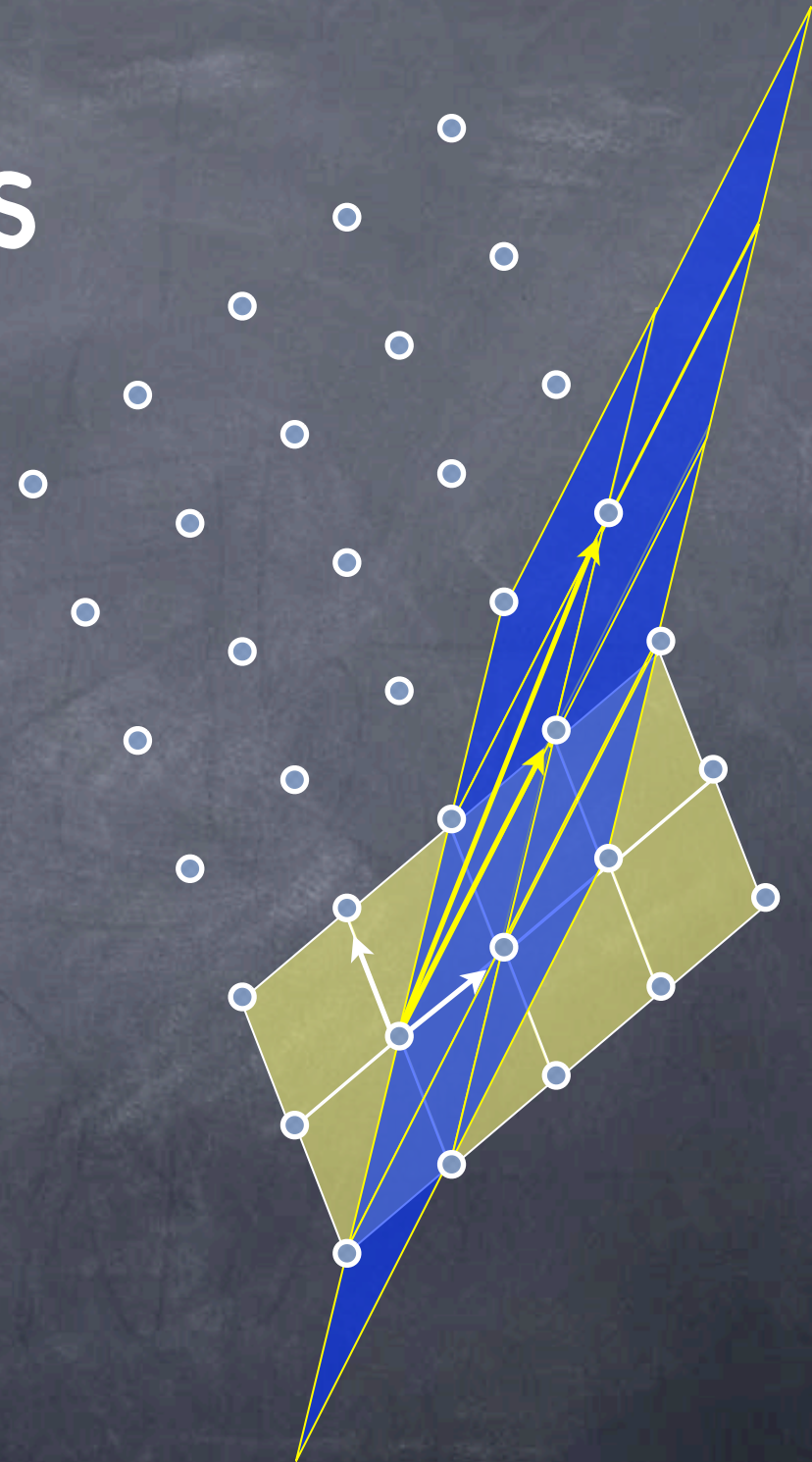
- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$





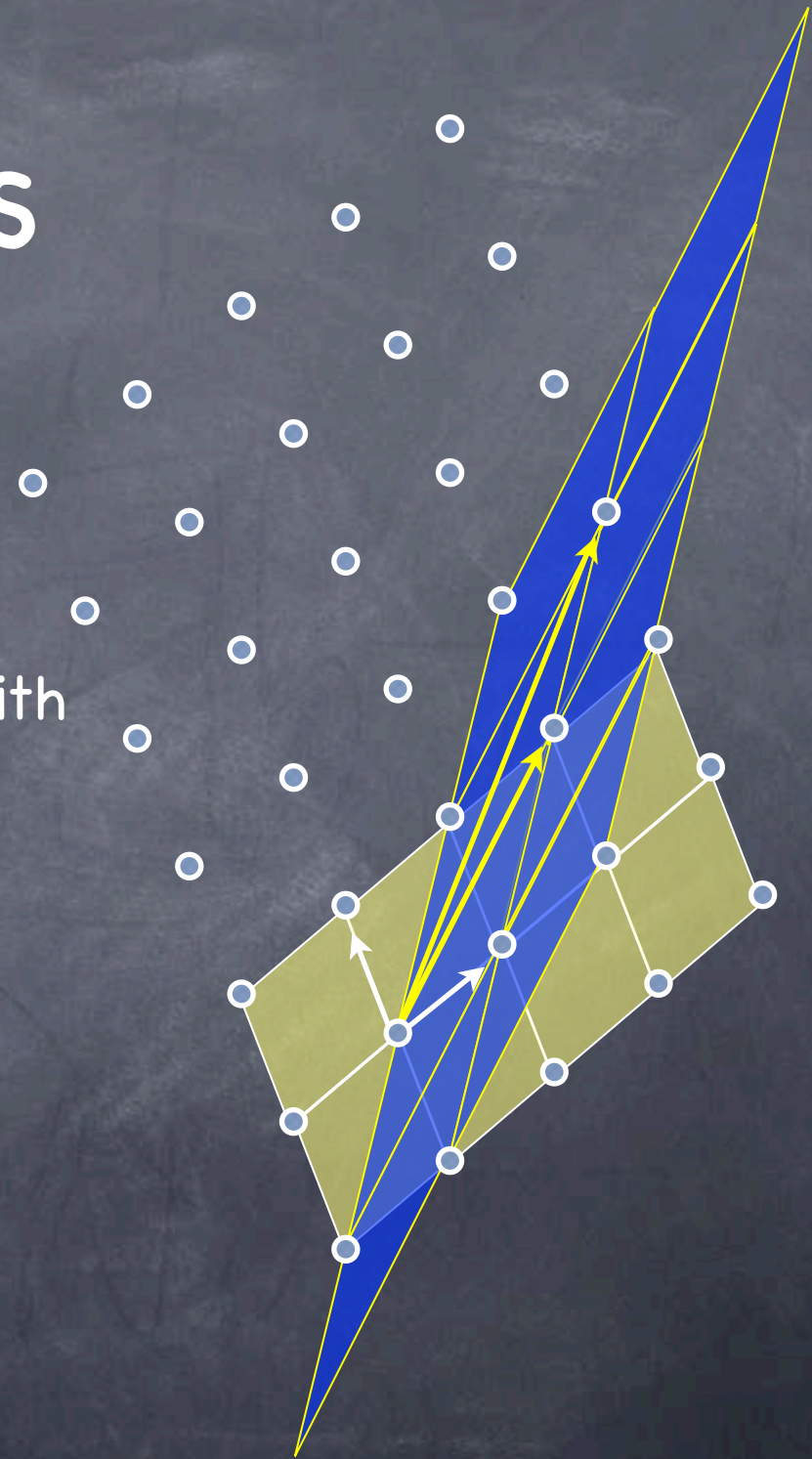
# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$



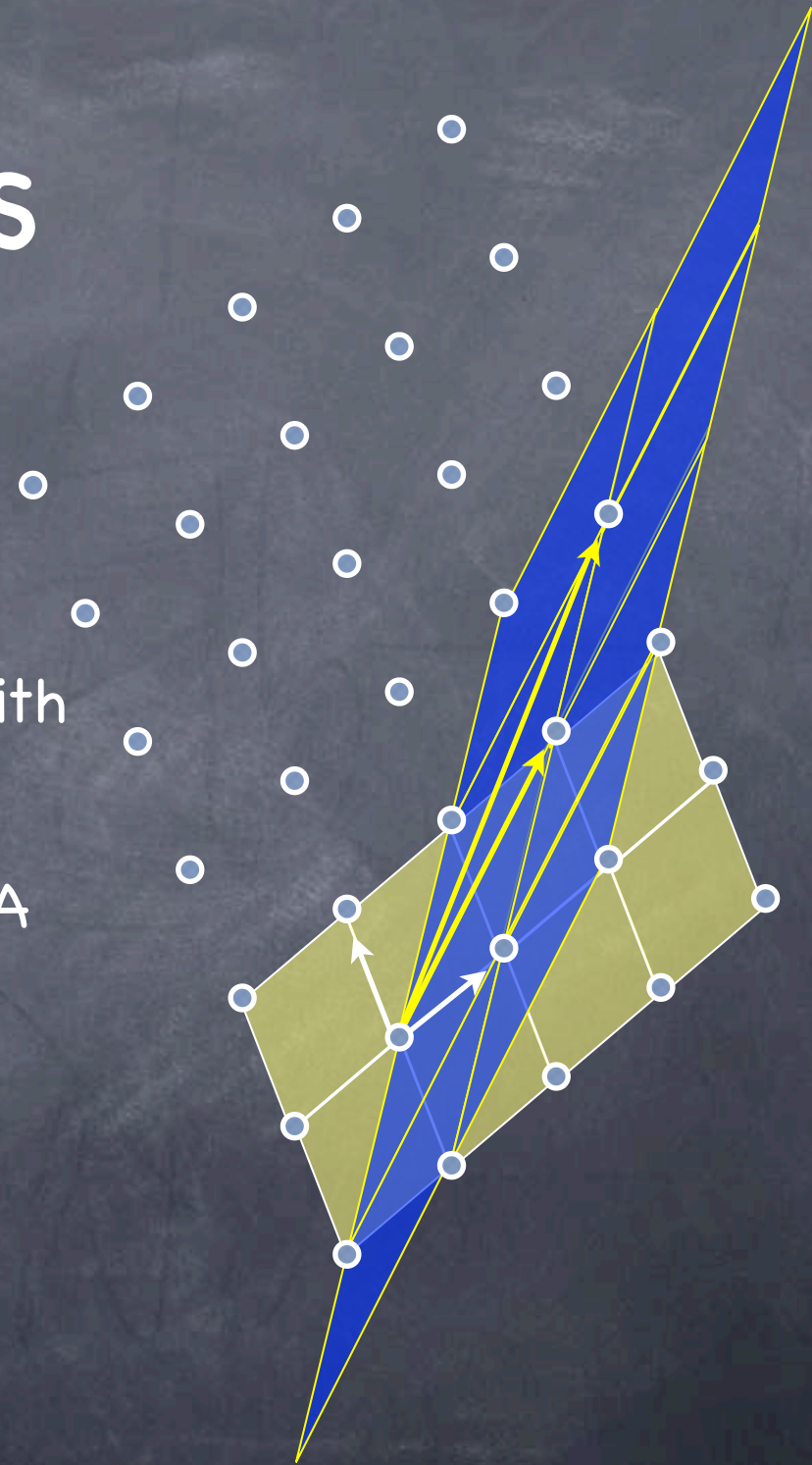
# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$ 
  - Two  $n$ -dim lattices in  $\mathbb{Z}^n$  associated with an  $m \times n$  matrix  $A$  over  $\mathbb{Z}_q$



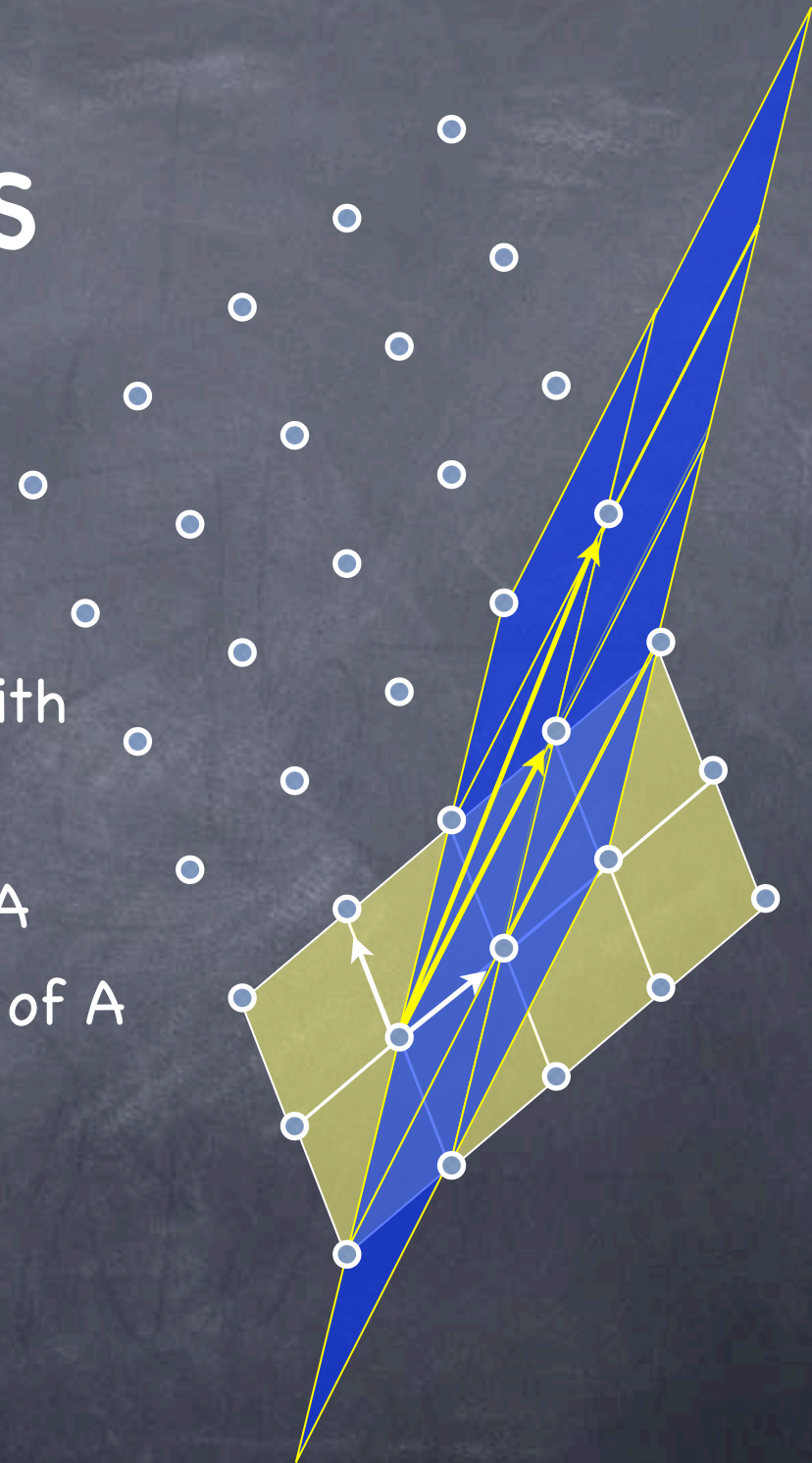
# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$ 
  - Two  $n$ -dim lattices in  $\mathbb{Z}^n$  associated with an  $m \times n$  matrix  $A$  over  $\mathbb{Z}_q$ 
    - $L_A$ : Vectors "spanned" by rows of  $A$



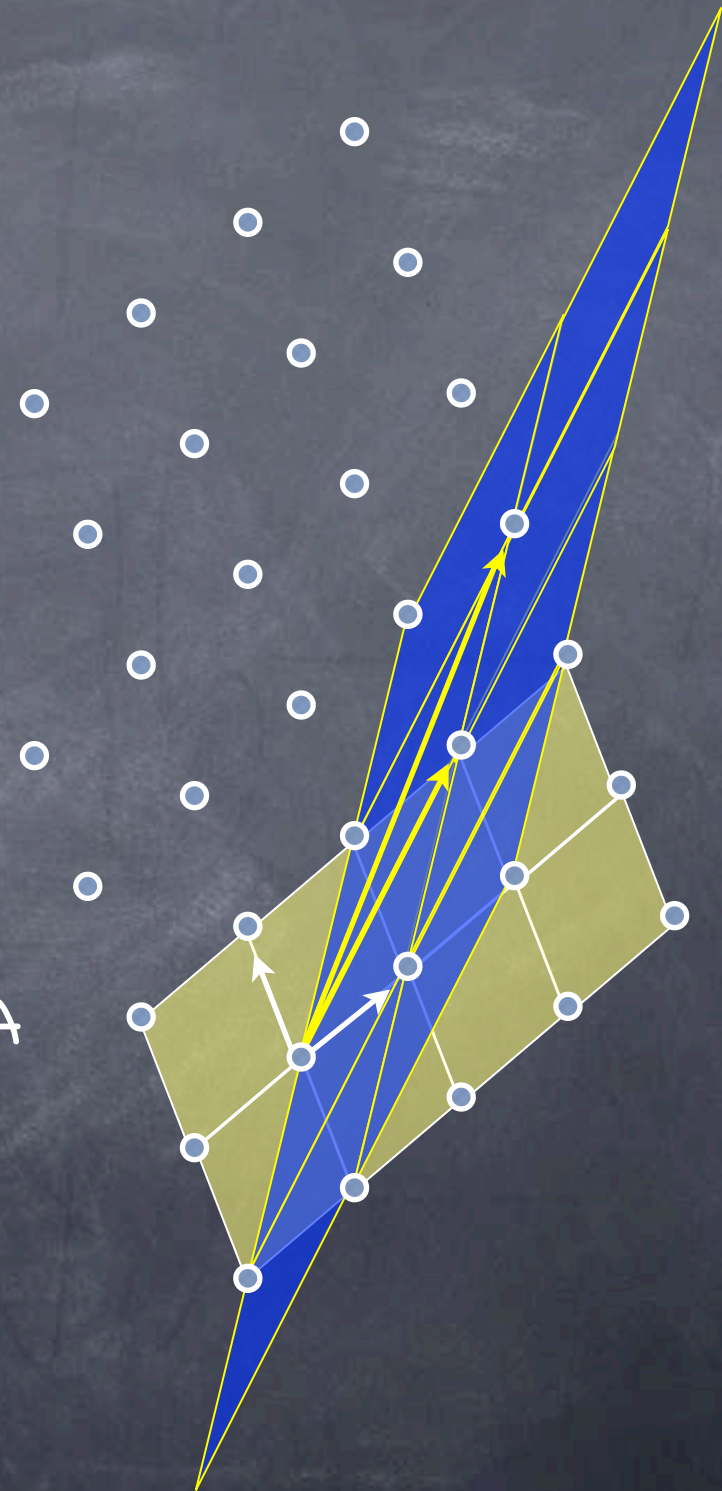
# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$ 
  - Two  $n$ -dim lattices in  $\mathbb{Z}^n$  associated with an  $m \times n$  matrix  $A$  over  $\mathbb{Z}_q$ 
    - $L_A$ : Vectors "spanned" by rows of  $A$
    - $L_A^\perp$ : Vectors "orthogonal" to rows of  $A$



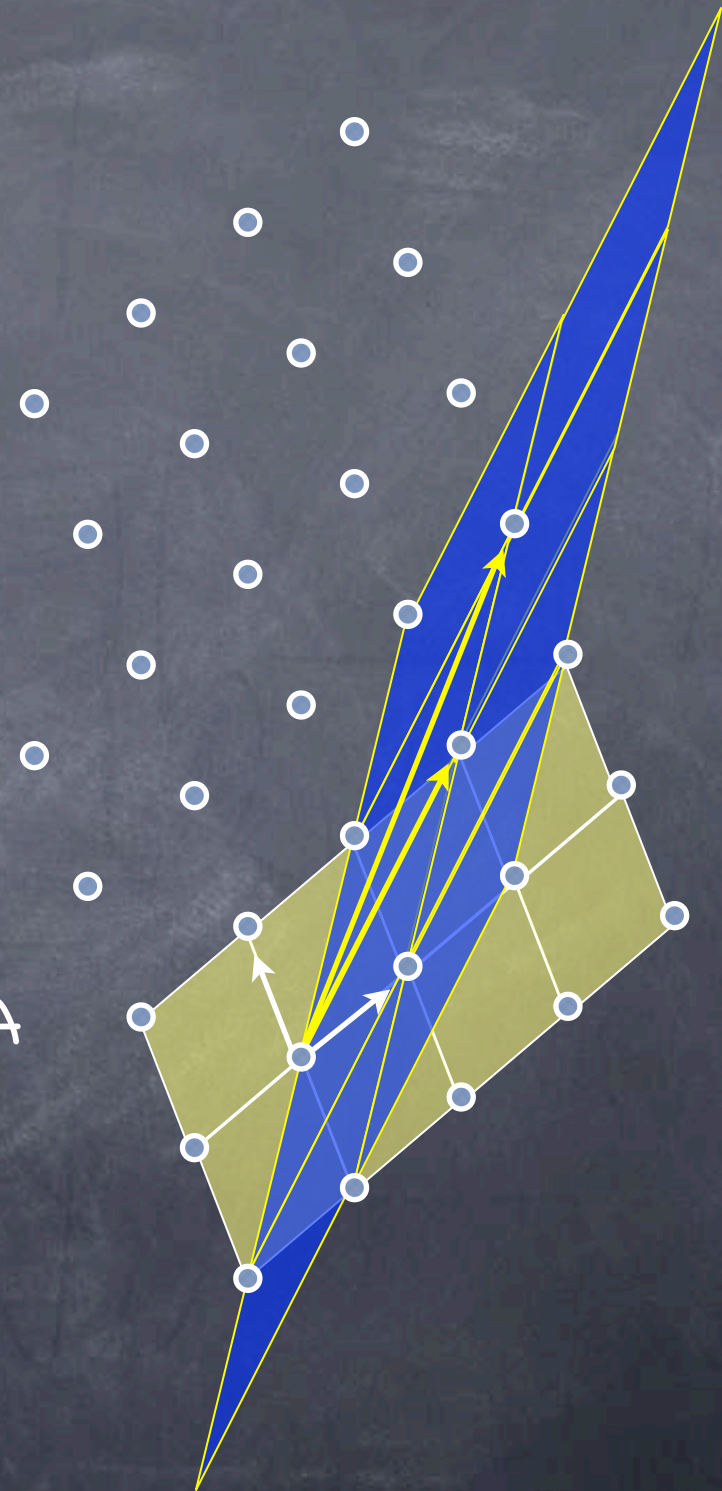
# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$ 
  - Two  $n$ -dim lattices in  $\mathbb{Z}^n$  associated with an  $m \times n$  matrix  $A$  over  $\mathbb{Z}_q$ 
    - $L_A$ : Vectors "spanned" by rows of  $A$
    - $L_A^\perp$ : Vectors "orthogonal" to rows of  $A$
    - Here,  $L_A, L_A^\perp$  in  $\mathbb{Z}^n$ , but above operations mod  $q$  (i.e., over  $\mathbb{Z}_q$ )



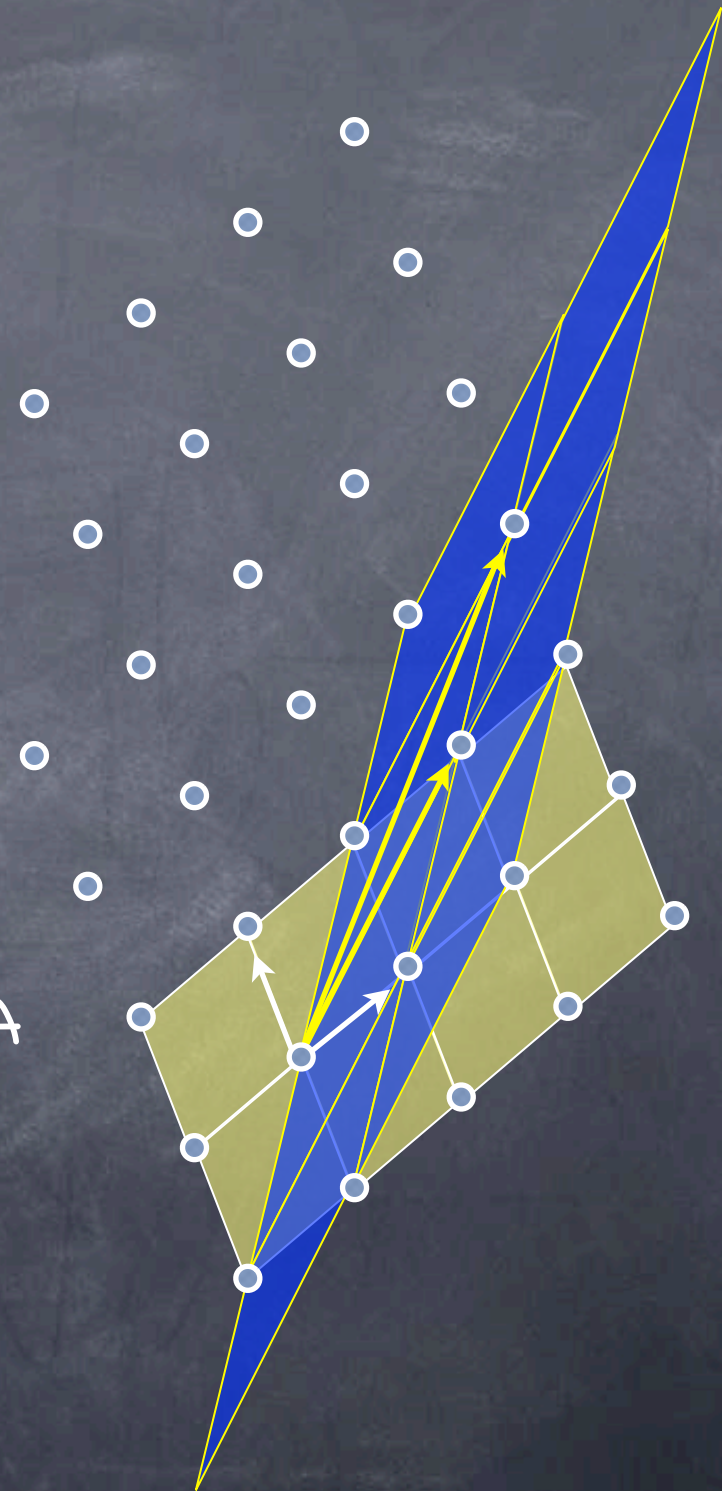
# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$ 
  - Two  $n$ -dim lattices in  $\mathbb{Z}^n$  associated with an  $m \times n$  matrix  $A$  over  $\mathbb{Z}_q$ 
    - $L_A$ : Vectors "spanned" by rows of  $A$
    - $L_A^\perp$ : Vectors "orthogonal" to rows of  $A$
    - Here,  $L_A, L_A^\perp$  in  $\mathbb{Z}^n$ , but above operations mod  $q$  (i.e., over  $\mathbb{Z}_q$ )
- Dual lattice  $L^*$ :  $\{ \underline{v} \mid \langle \underline{v}, \underline{u} \rangle \text{ is an integer} \}$



# Lattices

- Given a basis  $\{\underline{b}_1, \dots, \underline{b}_m\}$  in  $\mathbb{R}^n$ , lattice has points  $\{ \sum_i x_i \underline{b}_i \mid x_i \text{ integers} \}$
- An interesting case: lattices in  $\mathbb{Z}^n$ 
  - Two  $n$ -dim lattices in  $\mathbb{Z}^n$  associated with an  $m \times n$  matrix  $A$  over  $\mathbb{Z}_q$ 
    - $L_A$ : Vectors "spanned" by rows of  $A$
    - $L_A^\perp$ : Vectors "orthogonal" to rows of  $A$
    - Here,  $L_A, L_A^\perp$  in  $\mathbb{Z}^n$ , but above operations mod  $q$  (i.e., over  $\mathbb{Z}_q$ )
- Dual lattice  $L^*$ :  $\{ \underline{v} \mid \langle \underline{v}, \underline{u} \rangle \text{ is an integer} \}$ 
  - e.g.  $(L_A)^* = 1/q L_A^\perp$  and  $(L_A^\perp)^* = 1/q L_A$



# Lattices in Cryptography



# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions

# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions
- **Closest Vector Problem (CVP)**: Given a point in  $\mathbb{R}^n$ , find the point closest to it in the lattice

# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions
- **Closest Vector Problem (CVP)**: Given a point in  $\mathbb{R}^n$ , find the point closest to it in the lattice
- **Shortest Vector Problem (SVP)**: Find the shortest non-zero vector in the lattice

# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions
- **Closest Vector Problem (CVP)**: Given a point in  $\mathbb{R}^n$ , find the point closest to it in the lattice
- **Shortest Vector Problem (SVP)**: Find the shortest non-zero vector in the lattice
  - **SVP $_{\gamma}$** : find one within a factor  $\gamma$  of the shortest

# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions
- **Closest Vector Problem (CVP)**: Given a point in  $\mathbb{R}^n$ , find the point closest to it in the lattice
- **Shortest Vector Problem (SVP)**: Find the shortest non-zero vector in the lattice
  - **SVP $_{\gamma}$** : find one within a factor  $\gamma$  of the shortest
  - **GapSVP $_{\gamma}$** : decide if the length of the shortest vector is < 1 or >  $\gamma$  (promised to be one of the two)

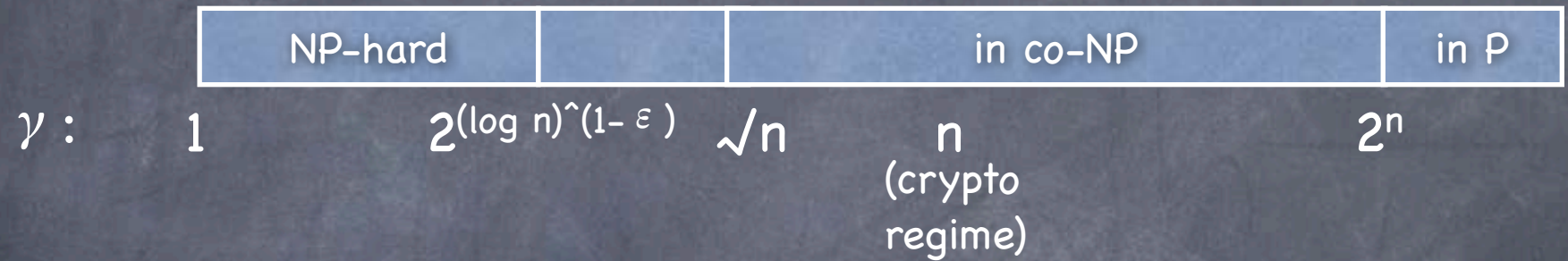
# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions
- **Closest Vector Problem (CVP)**: Given a point in  $\mathbb{R}^n$ , find the point closest to it in the lattice
- **Shortest Vector Problem (SVP)**: Find the shortest non-zero vector in the lattice
  - **SVP $_{\gamma}$** : find one within a factor  $\gamma$  of the shortest
  - **GapSVP $_{\gamma}$** : decide if the length of the shortest vector is < 1 or >  $\gamma$  (promised to be one of the two)
  - **uniqueSVP $_{\gamma}$** : SVP, when guaranteed that the next (non-parallel) shortest vector is longer by a factor  $\gamma$  or more

# Lattices in Cryptography

- Several problems related to lattices (lattice given as a basis) are believed to be computationally hard in high dimensions
- **Closest Vector Problem (CVP)**: Given a point in  $\mathbb{R}^n$ , find the point closest to it in the lattice
- **Shortest Vector Problem (SVP)**: Find the shortest non-zero vector in the lattice
  - **SVP $_{\gamma}$** : find one within a factor  $\gamma$  of the shortest
  - **GapSVP $_{\gamma}$** : decide if the length of the shortest vector is < 1 or >  $\gamma$  (promised to be one of the two)
  - **uniqueSVP $_{\gamma}$** : SVP, when guaranteed that the next (non-parallel) shortest vector is longer by a factor  $\gamma$  or more
- **Shortest Independent Vector Problem (SIVP)**: Find  $n$  independent vectors minimizing the longest of them

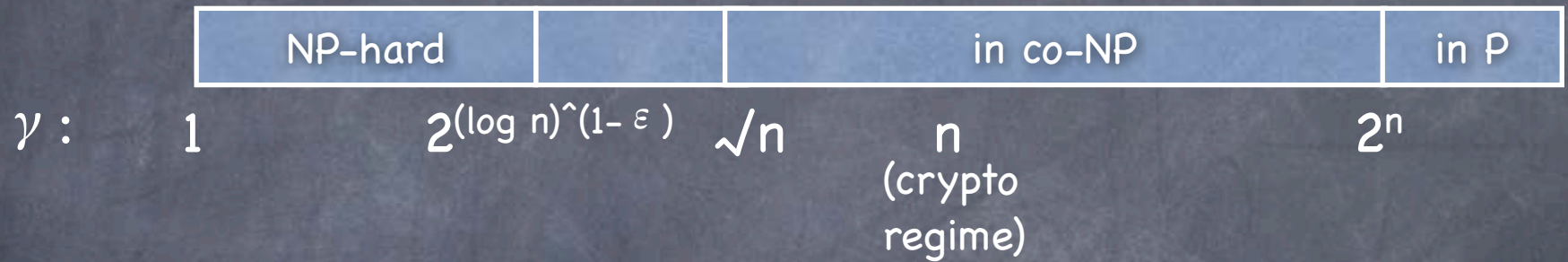
# Lattices in Cryptography





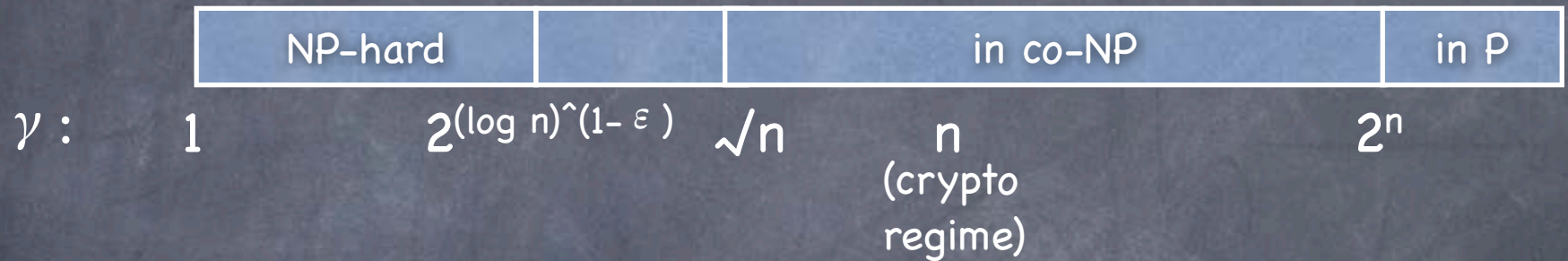
# Lattices in Cryptography

- Worst-case hardness of lattice problems (e.g. GapSVP)



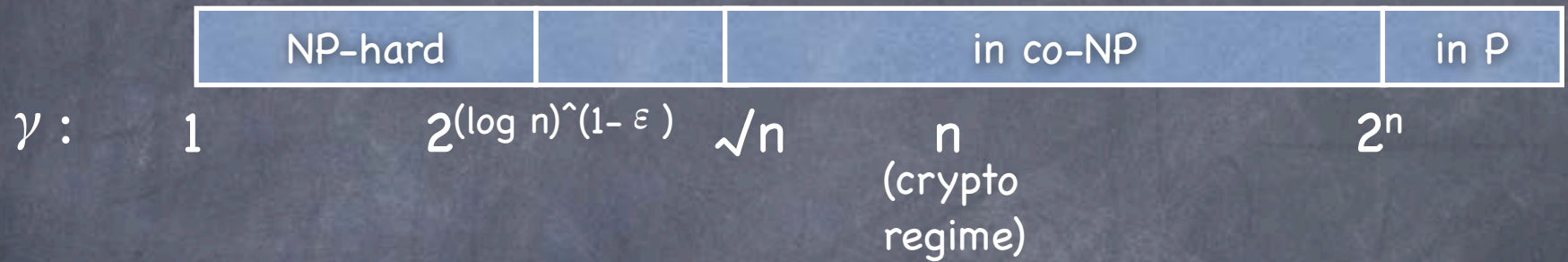
# Lattices in Cryptography

- Worst-case hardness of lattice problems (e.g. GapSVP)



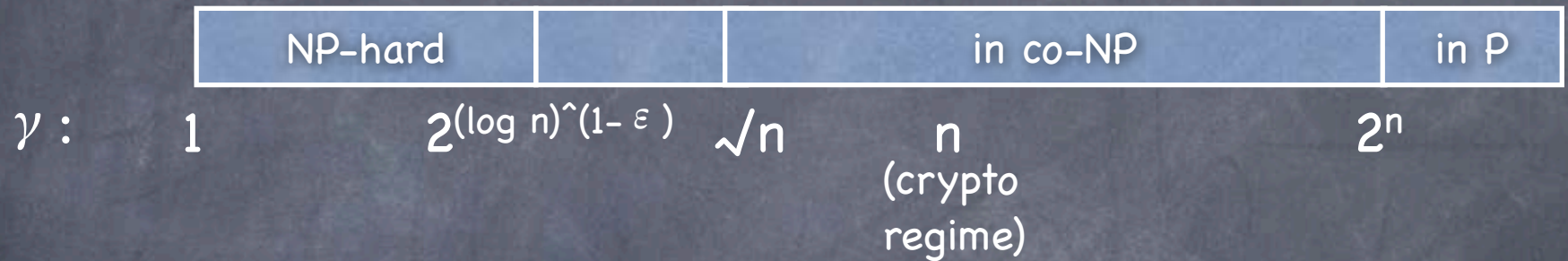
# Lattices in Cryptography

- Worst-case hardness of lattice problems (e.g. GapSVP)



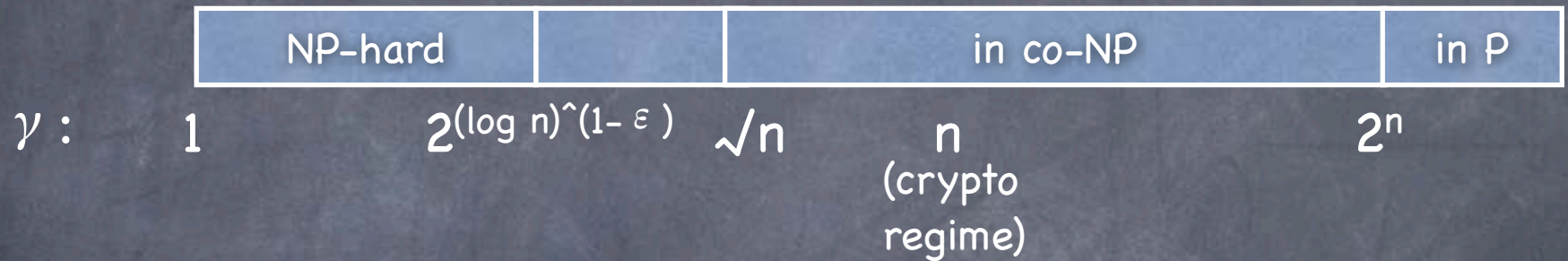
# Lattices in Cryptography

- Worst-case hardness of lattice problems (e.g. GapSVP)



# Lattices in Cryptography

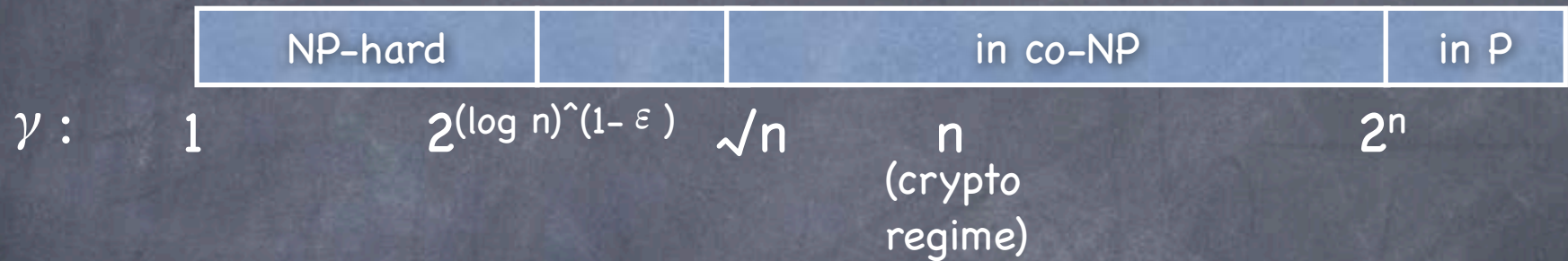
- Worst-case hardness of lattice problems (e.g. GapSVP)



- Assumptions about worst-case hardness (e.g.  $P \neq NP$ ) are qualitatively simpler than that of average-case hardness

# Lattices in Cryptography

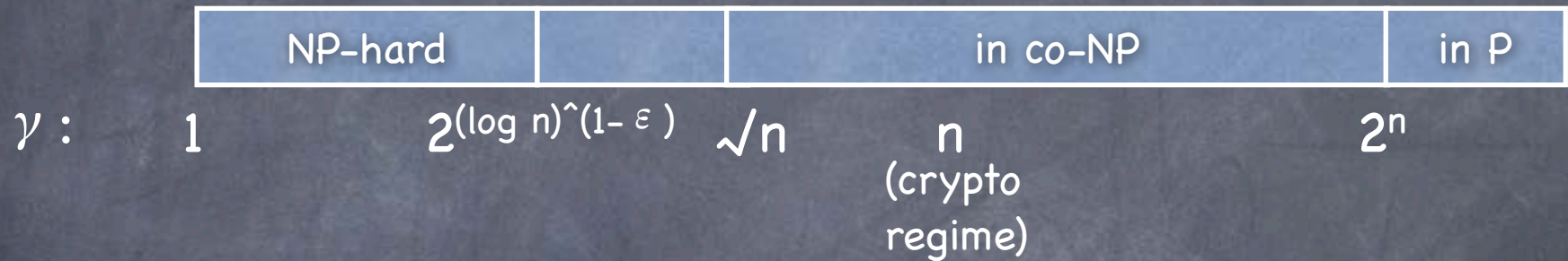
- Worst-case hardness of lattice problems (e.g. GapSVP)



- Assumptions about worst-case hardness (e.g.  $P \neq NP$ ) are qualitatively simpler than that of average-case hardness
  - Crypto requires average-case hardness

# Lattices in Cryptography

- Worst-case hardness of lattice problems (e.g. GapSVP)



- Assumptions about worst-case hardness (e.g.  $P \neq NP$ ) are qualitatively simpler than that of average-case hardness
  - Crypto requires average-case hardness
  - For many lattice problems average-case hardness assumptions are implied by worst-case hardness of related problems (but at regimes not known to be NP-hard)

# Learning With Errors



# Learning With Errors

- LWE: given noisy inner-products of random vectors with a hidden vector, find the hidden vector

# Learning With Errors

- LWE: given noisy inner-products of random vectors with a hidden vector, find the hidden vector
  - Given  $\langle \underline{a}_1, \underline{s} \rangle + \underline{e}_1, \dots, \langle \underline{a}_m, \underline{s} \rangle + \underline{e}_m$  and  $\underline{a}_1, \dots, \underline{a}_m$  find  $\underline{s}$ .  
 $\underline{a}_i$  uniform,  $\underline{e}_i$  Gaussian noise

# Learning With Errors

- LWE: given noisy inner-products of random vectors with a hidden vector, find the hidden vector
  - Given  $\langle \underline{a}_1, \underline{s} \rangle + \underline{e}_1, \dots, \langle \underline{a}_m, \underline{s} \rangle + \underline{e}_m$  and  $\underline{a}_1, \dots, \underline{a}_m$  find  $\underline{s}$ .  
 $\underline{a}_i$  uniform,  $\underline{e}_i$  Gaussian noise
- LWE-Decision version: distinguish between such an input and a random input

# Learning With Errors

- LWE: given noisy inner-products of random vectors with a hidden vector, find the hidden vector
  - Given  $\langle \underline{a}_1, \underline{s} \rangle + \underline{e}_1, \dots, \langle \underline{a}_m, \underline{s} \rangle + \underline{e}_m$  and  $\underline{a}_1, \dots, \underline{a}_m$  find  $\underline{s}$ .  
 $\underline{a}_i$  uniform,  $\underline{e}_i$  Gaussian noise
- LWE-Decision version: distinguish between such an input and a random input
- Assumed to be hard (note: average-case hardness). Has been connected with worst-case hardness of GapSVP

# Learning With Errors

- LWE: given noisy inner-products of random vectors with a hidden vector, find the hidden vector
  - Given  $\langle \underline{a}_1, \underline{s} \rangle + \underline{e}_1, \dots, \langle \underline{a}_m, \underline{s} \rangle + \underline{e}_m$  and  $\underline{a}_1, \dots, \underline{a}_m$  find  $\underline{s}$ .  
 $\underline{a}_i$  uniform,  $\underline{e}_i$  Gaussian noise
- LWE-Decision version: distinguish between such an input and a random input
- Assumed to be hard (note: average-case hardness). Has been connected with worst-case hardness of GapSVP
  - Turns out to be a very useful assumption

# Hash Functions and OWF

# Hash Functions and OWF

- CRHF:  $f(\underline{x}) = A\underline{x} \pmod{q}$

# Hash Functions and OWF

- CRHF:  $f(\underline{x}) = A\underline{x} \pmod{q}$
- $\underline{x}$  required to be a "short" vector (i.e., each co-ordinate in the range  $[0, d-1]$  for some small  $d$ )



# Hash Functions and OWF

- CRHF:  $f(\underline{x}) = A\underline{x} \pmod{q}$
- $\underline{x}$  required to be a "short" vector (i.e., each co-ordinate in the range  $[0, d-1]$  for some small  $d$ )
- $A$  is an  $n \times m$  matrix: maps  $m \log d$  bits to  $n \log q$  bits (for compression we require  $m > n \log_d q$ )

# Hash Functions and OWF

- CRHF:  $f(\underline{x}) = A\underline{x} \pmod{q}$
- $\underline{x}$  required to be a "short" vector (i.e., each co-ordinate in the range  $[0, d-1]$  for some small  $d$ )
  - $A$  is an  $n \times m$  matrix: maps  $m \log d$  bits to  $n \log q$  bits (for compression we require  $m > n \log_d q$ )
  - Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$

# Hash Functions and OWF

- CRHF:  $f(\underline{x}) = A\underline{x} \pmod{q}$
- $\underline{x}$  required to be a "short" vector (i.e., each co-ordinate in the range  $[0, d-1]$  for some small  $d$ )
  - $A$  is an  $n \times m$  matrix: maps  $m \log d$  bits to  $n \log q$  bits (for compression we require  $m > n \log_d q$ )
  - Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  $\underline{z}$  s.t  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$
  - Simple to compute: if  $d$  small (say,  $d=2$ , i.e.,  $\underline{x}$  binary),  $f(\underline{x})$  can be computed using  $O(n m)$  additions mod  $q$

# Hash Functions and OWF

- CRHF:  $f(\underline{x}) = A\underline{x} \pmod{q}$ 
  - $\underline{x}$  required to be a "short" vector (i.e., each co-ordinate in the range  $[0, d-1]$  for some small  $d$ )
    - $A$  is an  $n \times m$  matrix: maps  $m \log d$  bits to  $n \log q$  bits (for compression we require  $m > n \log_d q$ )
    - Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  $\underline{z}$  s.t  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$
    - Simple to compute: if  $d$  small (say,  $d=2$ , i.e.,  $\underline{x}$  binary),  $f(\underline{x})$  can be computed using  $O(n m)$  additions mod  $q$
- If sufficiently compressing (say by half), a CRHF is also a OWF

# Average-Case/Worst-Case Connection

# Average-Case/Worst-Case Connection

- Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t.  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$

# Average-Case/Worst-Case Connection

- Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t.  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$
- Considered hard when  $A$  is chosen uniformly at random

# Average-Case/Worst-Case Connection

- Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t.  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$ 
  - Considered hard when  $A$  is chosen uniformly at random
  - This is as hard as solving certain lattice problems in the worst case (i.e., with good success probability for every instance of the problem)



# Average-Case/Worst-Case Connection

- Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t.  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$ 
  - Considered hard when  $A$  is chosen uniformly at random
  - This is as hard as solving certain lattice problems in the worst case (i.e., with good success probability for every instance of the problem)
- In general average case assumptions may be risky: there will be many easy instances

# Average-Case/Worst-Case Connection

- Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t.  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$ 
  - Considered hard when  $A$  is chosen uniformly at random
  - This is as hard as solving certain lattice problems in the worst case (i.e., with good success probability for every instance of the problem)
- In general average case assumptions may be risky: there will be many easy instances
- Worst case assumptions are OK even if most instances are easy

# Average-Case/Worst-Case Connection

- Collision yields a short vector (co-ordinates in  $[-(d-1), d-1]$ )  
 $\underline{z}$  s.t.  $A\underline{z} = 0$ : i.e., a short vector in the lattice  $L_A^\perp$ 
  - Considered hard when  $A$  is chosen uniformly at random
  - This is as hard as solving certain lattice problems in the worst case (i.e., with good success probability for every instance of the problem)
- In general average case assumptions may be risky: there will be many easy instances
- Worst case assumptions are OK even if most instances are easy
- Connection shows that if a few instances hard, most instances are

# Succinct Keys

# Succinct Keys

- The hash function is described by an  $n \times m$  matrix over  $\mathbb{Z}_q$ , where  $n$  is the security parameter and  $m > n$

# Succinct Keys

- The hash function is described by an  $n \times m$  matrix over  $\mathbb{Z}_q$ , where  $n$  is the security parameter and  $m > n$ 
  - Large key and correspondingly large number of operations

# Succinct Keys

- The hash function is described by an  $n \times m$  matrix over  $\mathbb{Z}_q$ , where  $n$  is the security parameter and  $m > n$ 
  - Large key and correspondingly large number of operations
- Using "ideal lattices"

# Succinct Keys

- The hash function is described by an  $n \times m$  matrix over  $\mathbb{Z}_q$ , where  $n$  is the security parameter and  $m > n$ 
  - Large key and correspondingly large number of operations
- Using "ideal lattices"
  - Have more structure: a random basis for such a lattice can be represented using just  $m$  elements of  $\mathbb{Z}_q$  (instead of  $mn$ )



# Succinct Keys

- The hash function is described by an  $n \times m$  matrix over  $\mathbb{Z}_q$ , where  $n$  is the security parameter and  $m > n$ 
  - Large key and correspondingly large number of operations
- Using "ideal lattices"
  - Have more structure: a random basis for such a lattice can be represented using just  $m$  elements of  $\mathbb{Z}_q$  (instead of  $mn$ )
  - Matrix multiplication can be carried out faster (using FFT) with  $\tilde{O}(m)$  operations over  $\mathbb{Z}_q$  (instead of  $O(mn)$ )

# Succinct Keys

- The hash function is described by an  $n \times m$  matrix over  $\mathbb{Z}_q$ , where  $n$  is the security parameter and  $m > n$ 
  - Large key and correspondingly large number of operations
- Using “ideal lattices”
  - Have more structure: a random basis for such a lattice can be represented using just  $m$  elements of  $\mathbb{Z}_q$  (instead of  $mn$ )
  - Matrix multiplication can be carried out faster (using FFT) with  $\tilde{O}(m)$  operations over  $\mathbb{Z}_q$  (instead of  $O(mn)$ )
- Security depends on worst-case hardness of same problems as before, but when restricted to ideal lattices

# Public-Key Encryption

# Public-Key Encryption

- NTRU/GGH approach: Private key is a “good” basis, and the public key is a “bad basis”

# Public-Key Encryption

- NTRU/GGH approach: Private key is a “good” basis, and the public key is a “bad basis”
  - Worst basis (one that can be efficiently computed from any basis): Hermite Normal Form (HNF) basis

# Public-Key Encryption

- NTRU/GGH approach: Private key is a “good” basis, and the public key is a “bad basis”
  - Worst basis (one that can be efficiently computed from any basis): Hermite Normal Form (HNF) basis
- To encrypt a message, encode it (randomized) as a short “noise vector”  $u$ . Output  $c = v + u$  for a lattice point  $v$  that is chosen using the public basis

# Public-Key Encryption

- NTRU/GGH approach: Private key is a “good” basis, and the public key is a “bad basis”
  - Worst basis (one that can be efficiently computed from any basis): Hermite Normal Form (HNF) basis
- To encrypt a message, encode it (randomized) as a short “noise vector”  $u$ . Output  $c = v + u$  for a lattice point  $v$  that is chosen using the public basis
  - To decrypt, use the good basis to find  $v$  as the closest lattice vector to  $c$ , and recover  $u = c - v$

# Public-Key Encryption

- NTRU/GGH approach: Private key is a “good” basis, and the public key is a “bad basis”
  - Worst basis (one that can be efficiently computed from any basis): Hermite Normal Form (HNF) basis
- To encrypt a message, encode it (randomized) as a short “noise vector”  $u$ . Output  $c = v + u$  for a lattice point  $v$  that is chosen using the public basis
  - To decrypt, use the good basis to find  $v$  as the closest lattice vector to  $c$ , and recover  $u = c - v$
- NTRU Encryption: use lattices with succinct basis



# Public-Key Encryption

- NTRU/GGH approach: Private key is a “good” basis, and the public key is a “bad basis”
  - Worst basis (one that can be efficiently computed from any basis): Hermite Normal Form (HNF) basis
- To encrypt a message, encode it (randomized) as a short “noise vector”  $u$ . Output  $c = v + u$  for a lattice point  $v$  that is chosen using the public basis
  - To decrypt, use the good basis to find  $v$  as the closest lattice vector to  $c$ , and recover  $u = c - v$
- NTRU Encryption: use lattices with succinct basis
- Conjectured to be CPA secure for appropriate lattices. No security reduction known to simple lattice problems

# Public-Key Encryption

# Public-Key Encryption

- A subset-sum approach:

# Public-Key Encryption

- A subset-sum approach:
  - Encryption of bit 0 is a point from a uniform distribution (over an interval of integers); encryption of 1 comes from a “wavy” distribution of secret period

# Public-Key Encryption

- A subset-sum approach:
  - Encryption of bit 0 is a point from a uniform distribution (over an interval of integers); encryption of 1 comes from a “wavy” distribution of secret period
  - Public-key gives several points from the wavy distribution that can be combined (subset sum) to get more points from the wavy distribution

# Public-Key Encryption

- A subset-sum approach:
  - Encryption of bit 0 is a point from a uniform distribution (over an interval of integers); encryption of 1 comes from a “wavy” distribution of secret period
  - Public-key gives several points from the wavy distribution that can be combined (subset sum) to get more points from the wavy distribution
  - Secret-key consists of the period: enough for a statistical test to distinguish the two distributions

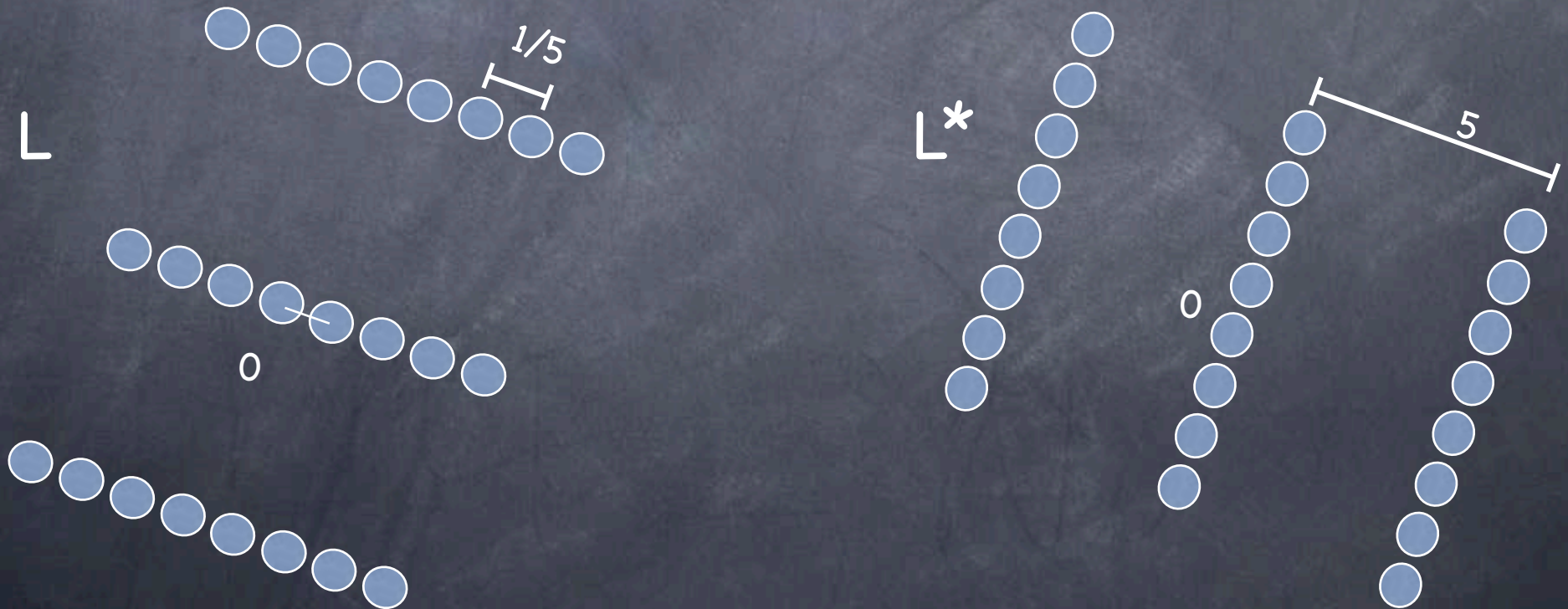
# Public-Key Encryption

- A subset-sum approach:
  - Encryption of bit 0 is a point from a uniform distribution (over an interval of integers); encryption of 1 comes from a “wavy” distribution of secret period
    - Public-key gives several points from the wavy distribution that can be combined (subset sum) to get more points from the wavy distribution
    - Secret-key consists of the period: enough for a statistical test to distinguish the two distributions
  - CPA Security: distinguishing the uniform and wavy distributions can be used to distinguish between noise added to lattices obtained as duals of lattices either with no short vector or with a unique short vector

# Dual Lattice

Given a lattice  $L$ , the dual lattice is

$$L^* = \{ x \mid \text{for all } y \in L, \langle x, y \rangle \in \mathbb{Z} \}$$



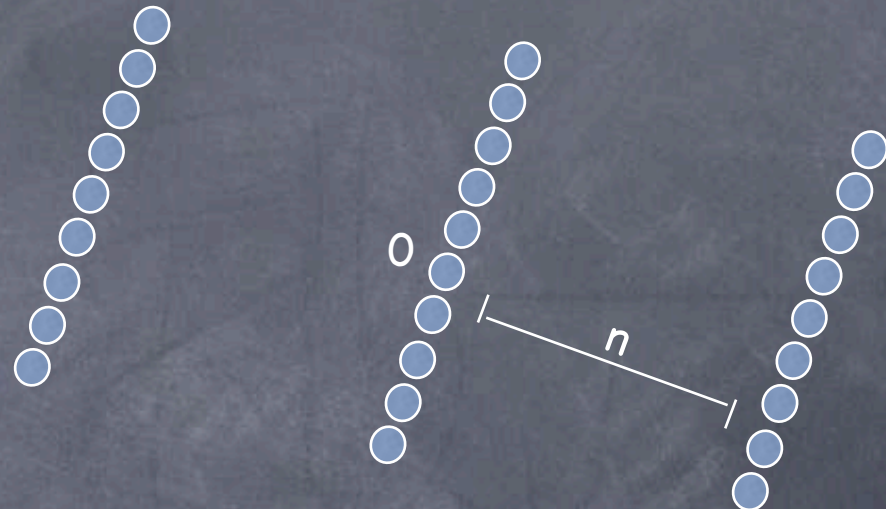
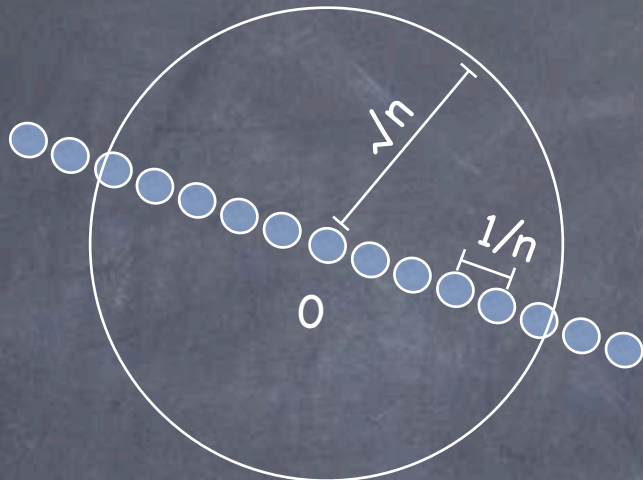


# $L^*$ - the dual of $L$

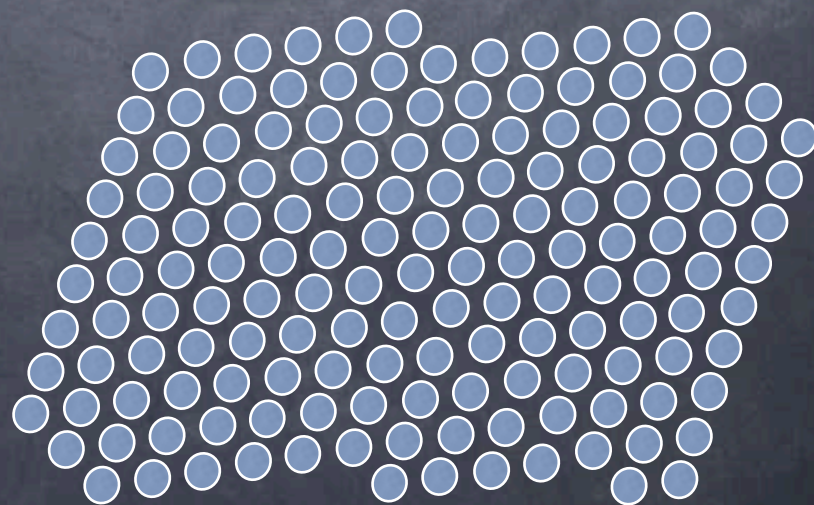
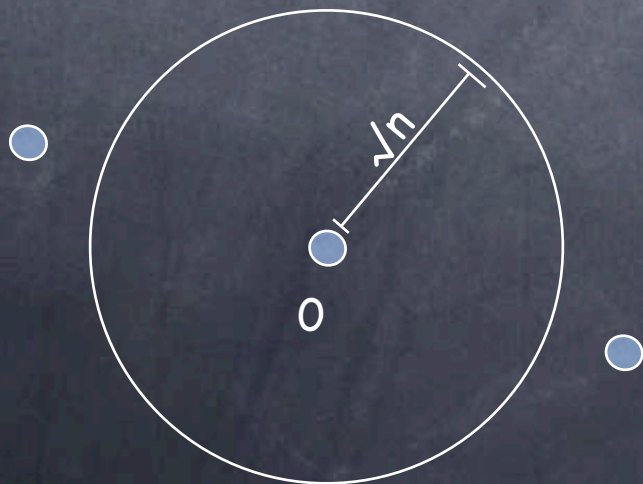
$L$

$L^*$

Case 1



Case 2



# Public-Key Encryption

# Public-Key Encryption

- An LWE based approach:

# Public-Key Encryption

- An LWE based approach:
  - Public-key is  $(A,P)$  where  $P=AS+E$ , for random matrices (of appropriate dimensions)  $A$  and  $S$ , and a noise matrix  $E$  over  $\mathbb{Z}_q$

# Public-Key Encryption

- An LWE based approach:
  - Public-key is  $(A, P)$  where  $P=AS+E$ , for random matrices (of appropriate dimensions)  $A$  and  $S$ , and a noise matrix  $E$  over  $\mathbb{Z}_q$
  - To encrypt an  $n$  bit message, first map it to a vector  $\underline{v}$  in (a sparse sub-lattice of)  $\mathbb{Z}_q^n$ ; pick a random vector  $\underline{a}$  with small coordinates; ciphertext is  $(\underline{u}, \underline{c})$  where  $\underline{u} = A^T \underline{a}$  and  $\underline{c} = P^T \underline{a} + \underline{v}$

# Public-Key Encryption

- An LWE based approach:
  - Public-key is  $(A, P)$  where  $P=AS+E$ , for random matrices (of appropriate dimensions)  $A$  and  $S$ , and a noise matrix  $E$  over  $\mathbb{Z}_q$
  - To encrypt an  $n$  bit message, first map it to a vector  $\underline{v}$  in (a sparse sub-lattice of)  $\mathbb{Z}_q^n$ ; pick a random vector  $\underline{a}$  with small coordinates; ciphertext is  $(\underline{u}, \underline{c})$  where  $\underline{u} = A^T \underline{a}$  and  $\underline{c} = P^T \underline{a} + \underline{v}$
  - Decryption using  $S$ : recover message from  $\underline{c} - S^T \underline{u} = \underline{v} + E^T \underline{a}$

# Public-Key Encryption

- An LWE based approach:
  - Public-key is  $(A,P)$  where  $P=AS+E$ , for random matrices (of appropriate dimensions)  $A$  and  $S$ , and a noise matrix  $E$  over  $\mathbb{Z}_q$
  - To encrypt an  $n$  bit message, first map it to a vector  $\underline{v}$  in (a sparse sub-lattice of)  $\mathbb{Z}_q^n$ ; pick a random vector  $\underline{a}$  with small coordinates; ciphertext is  $(\underline{u},\underline{c})$  where  $\underline{u} = A^T \underline{a}$  and  $\underline{c} = P^T \underline{a} + \underline{v}$
  - Decryption using  $S$ : recover message from  $\underline{c} - S^T \underline{u} = \underline{v} + E^T \underline{a}$ 
    - Allows a small error probability; can be made negligible by first encoding the message using an error correcting code

# Public-Key Encryption

- An LWE based approach:
  - Public-key is  $(A, P)$  where  $P=AS+E$ , for random matrices (of appropriate dimensions)  $A$  and  $S$ , and a noise matrix  $E$  over  $\mathbb{Z}_q$
  - To encrypt an  $n$  bit message, first map it to a vector  $\underline{v}$  in (a sparse sub-lattice of)  $\mathbb{Z}_q^n$ ; pick a random vector  $\underline{a}$  with small coordinates; ciphertext is  $(\underline{u}, \underline{c})$  where  $\underline{u} = A^T \underline{a}$  and  $\underline{c} = P^T \underline{a} + \underline{v}$
  - Decryption using  $S$ : recover message from  $\underline{c} - S^T \underline{u} = \underline{v} + E^T \underline{a}$ 
    - Allows a small error probability; can be made negligible by first encoding the message using an error correcting code
  - CPA security: By LWE assumption, the public-key is indistinguishable from random; and, encryption under random  $(A, P)$  loses essentially all information about the message



# Public-Key Encryption

- An LWE based approach:
  - Public-key is  $(A, P)$  where  $P=AS+E$ , for random matrices (of appropriate dimensions)  $A$  and  $S$ , and a noise matrix  $E$  over  $\mathbb{Z}_q$
  - To encrypt an  $n$  bit message, first map it to a vector  $\underline{v}$  in (a sparse sub-lattice of)  $\mathbb{Z}_q^n$ ; pick a random vector  $\underline{a}$  with small coordinates; ciphertext is  $(\underline{u}, \underline{c})$  where  $\underline{u} = A^T \underline{a}$  and  $\underline{c} = P^T \underline{a} + \underline{v}$
  - Decryption using  $S$ : recover message from  $\underline{c} - S^T \underline{u} = \underline{v} + E^T \underline{a}$ 
    - Allows a small error probability; can be made negligible by first encoding the message using an error correcting code
  - CPA security: By LWE assumption, the public-key is indistinguishable from random; and, encryption under random  $(A, P)$  loses essentially all information about the message
- LWE also used for CCA secure PKE

# Signatures

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it
    - e.g. with  $\underline{\mathbf{s}} = \mathbf{B} \lfloor \mathbf{B}^{-1} \underline{\mathbf{m}} \rceil$ , we have  $\underline{\mathbf{s}} - \underline{\mathbf{m}} = \mathbf{B} \underline{\mathbf{z}}$  for  $\underline{\mathbf{z}} \in [-\frac{1}{2}, \frac{1}{2}]^n$

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it
    - e.g. with  $\underline{\mathbf{s}} = \mathbf{B} \lfloor \mathbf{B}^{-1} \underline{\mathbf{m}} \rceil$ , we have  $\underline{\mathbf{s}} - \underline{\mathbf{m}} = \mathbf{B} \underline{\mathbf{z}}$  for  $\underline{\mathbf{z}} \in [-\frac{1}{2}, \frac{1}{2}]^n$
  - Intuitively, it is hard to find such a point using the HNF basis

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it
    - e.g. with  $\underline{\mathbf{s}} = \mathbf{B} \lfloor \mathbf{B}^{-1} \underline{\mathbf{m}} \rceil$ , we have  $\underline{\mathbf{s}} - \underline{\mathbf{m}} = \mathbf{B} \underline{\mathbf{z}}$  for  $\underline{\mathbf{z}} \in [-\frac{1}{2}, \frac{1}{2}]^n$
  - Intuitively, it is hard to find such a point using the HNF basis
  - However, multiple signatures can leak  $\mathbf{B}$

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it
    - e.g. with  $\underline{\mathbf{s}} = \mathbf{B} \lfloor \mathbf{B}^{-1} \mathbf{m} \rceil$ , we have  $\underline{\mathbf{s}} - \mathbf{m} = \mathbf{B} \underline{\mathbf{z}}$  for  $\underline{\mathbf{z}} \in [-\frac{1}{2}, \frac{1}{2}]^n$
  - Intuitively, it is hard to find such a point using the HNF basis
  - However, multiple signatures can leak  $\mathbf{B}$
  - Fix (heuristic): Perturbation, to make it harder to recover  $\mathbf{B}$



# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it
    - e.g. with  $\underline{\mathbf{s}} = B \lfloor B^{-1} \underline{\mathbf{m}} \rfloor$ , we have  $\underline{\mathbf{s}} - \underline{\mathbf{m}} = B \underline{\mathbf{z}}$  for  $\underline{\mathbf{z}} \in [-\frac{1}{2}, \frac{1}{2}]^n$
  - Intuitively, it is hard to find such a point using the HNF basis
  - However, multiple signatures can leak  $B$
  - Fix (heuristic): Perturbation, to make it harder to recover  $B$
  - Fix [GPV'08]: instead of rounding off to  $B \lfloor B^{-1} \underline{\mathbf{m}} \rfloor$ , sample from a distribution that does not leak  $B$ . Security (in ROM) reduces to worst-case hardness assumptions.

# Signatures

- GGH/NTRU approach: Secret key is a good basis, and the public key is a bad (i.e., HNF) basis
  - To sign a message, hash it (using an RO) to a random point  $\mathbf{m}$  in  $\mathbb{R}^n$  and use the good basis to find a lattice point close to it
    - e.g. with  $\underline{\mathbf{s}} = B \lfloor B^{-1} \underline{\mathbf{m}} \rceil$ , we have  $\underline{\mathbf{s}} - \underline{\mathbf{m}} = B \underline{\mathbf{z}}$  for  $\underline{\mathbf{z}} \in [1/2, -1/2]^n$
  - Intuitively, it is hard to find such a point using the HNF basis
  - However, multiple signatures can leak  $B$
  - Fix (heuristic): Perturbation, to make it harder to recover  $B$
  - Fix [GPV'08]: instead of rounding off to  $B \lfloor B^{-1} \underline{\mathbf{m}} \rceil$ , sample from a distribution that does not leak  $B$ . Security (in ROM) reduces to worst-case hardness assumptions.
    - Quadratic key size/signing complexity (unlike NTRUSign)

# Signatures

# Signatures

- Using CRHF (not in ROM)

# Signatures

- Using CRHF (not in ROM)
  - Obtaining a one-time signature from a “homomorphic” CRHF

# Signatures

- Using CRHF (not in ROM)
  - Obtaining a one-time signature from a “homomorphic” CRHF
    - $h(a.x+y)=a.h(x)+h(y)$  where  $a$  is from a ring  $\mathcal{A}$  and  $x,y$  from a module over the ring (say  $\mathcal{A}^m$ ). e.g.,  $h(\underline{x}) = A\underline{x}$ .

# Signatures

- Using CRHF (not in ROM)
  - Obtaining a one-time signature from a "homomorphic" CRHF
    - $h(a.x+y) = a.h(x) + h(y)$  where  $a$  is from a ring  $\mathcal{A}$  and  $x, y$  from a module over the ring (say  $\mathcal{A}^m$ ). e.g.,  $h(\underline{x}) = A\underline{x}$ .
    - Signing key:  $(x, y)$ . Verification key:  $(h, X, Y) = (h, h(x), h(y))$ .  
Signature: Message is mapped to an element  $a \in \mathcal{A}$ .  $s = a.x + y$   
Verification: Check  $h(s) = a.X + Y$

# Signatures

- Using CRHF (not in ROM)
  - Obtaining a one-time signature from a "homomorphic" CRHF
    - $h(a.x+y) = a.h(x) + h(y)$  where  $a$  is from a ring  $\mathcal{A}$  and  $x, y$  from a module over the ring (say  $\mathcal{A}^m$ ). e.g.,  $h(\underline{x}) = A\underline{x}$ .
    - Signing key:  $(x, y)$ . Verification key:  $(h, X, Y) = (h, h(x), h(y))$ .  
Signature: Message is mapped to an element  $a \in \mathcal{A}$ .  $s = a.x + y$   
Verification: Check  $h(s) = a.X + Y$
  - $(x, y)$  is information theoretically well-hidden after one sign; so, w.h.p., forgery yields a different signature than computed using the signing key, thereby giving a collision



# Signatures

- Using CRHF (not in ROM)
  - Obtaining a one-time signature from a "homomorphic" CRHF
    - $h(a.x+y) = a.h(x) + h(y)$  where  $a$  is from a ring  $\mathcal{A}$  and  $x, y$  from a module over the ring (say  $\mathcal{A}^m$ ). e.g.,  $h(\underline{x}) = A\underline{x}$ .
    - Signing key:  $(x, y)$ . Verification key:  $(h, X, Y) = (h, h(x), h(y))$ .  
Signature: Message is mapped to an element  $a \in \mathcal{A}$ .  $s = a.x + y$   
Verification: Check  $h(s) = a.X + Y$
  - $(x, y)$  is information theoretically well-hidden after one sign; so, w.h.p., forgery yields a different signature than computed using the signing key, thereby giving a collision
    - Trickier when using ideal lattice based hashing

# Signatures

- Using CRHF (not in ROM)
  - Obtaining a one-time signature from a “homomorphic” CRHF
    - $h(a.x+y)=a.h(x)+h(y)$  where  $a$  is from a ring  $\mathcal{A}$  and  $x,y$  from a module over the ring (say  $\mathcal{A}^m$ ). e.g.,  $h(\underline{x}) = A\underline{x}$ .
    - Signing key:  $(x,y)$ . Verification key:  $(h,X,Y) = (h,h(x),h(y))$ .  
Signature: Message is mapped to an element  $a \in \mathcal{A}$ .  $s=a.x+y$   
Verification: Check  $h(s)=a.X+Y$
    - $(x,y)$  is information theoretically well-hidden after one sign; so, w.h.p., forgery yields a different signature than computed using the signing key, thereby giving a collision
      - Trickier when using ideal lattice based hashing
  - Recall: one-time signatures can be augmented to full-fledged signatures using a CRHF (in fact, a UOWHF)

# Other Constructions

# Other Constructions

- Schemes based on LWE

# Other Constructions

- Schemes based on LWE
  - IBE, OT, Fully Homomorphic Encryption...

# Other Constructions

- Schemes based on LWE
  - IBE, OT, Fully Homomorphic Encryption...
- ZK proof systems for lattice problems

# Other Constructions

- Schemes based on LWE
  - IBE, OT, Fully Homomorphic Encryption...
- ZK proof systems for lattice problems
  - Interactive and non-interactive statistical ZK proofs of knowledge for various lattice problems

# Other Constructions

- Schemes based on LWE
  - IBE, OT, Fully Homomorphic Encryption...
- ZK proof systems for lattice problems
  - Interactive and non-interactive statistical ZK proofs of knowledge for various lattice problems
  - Useful in building “identification schemes” and potentially in other lattice-based constructions



Today

# Today

- Lattice based cryptography

# Today

- Lattice based cryptography
  - Candidate for post-quantum cryptography

# Today

- Lattice based cryptography
  - Candidate for post-quantum cryptography
  - Security typically based on worst-case hardness of problems

# Today

- Lattice based cryptography
  - Candidate for post-quantum cryptography
  - Security typically based on worst-case hardness of problems
  - Several problems: SVP and variants, LWE

# Today

- Lattice based cryptography
  - Candidate for post-quantum cryptography
  - Security typically based on worst-case hardness of problems
  - Several problems: SVP and variants, LWE
- Hash functions, PKE, Signatures, ...