

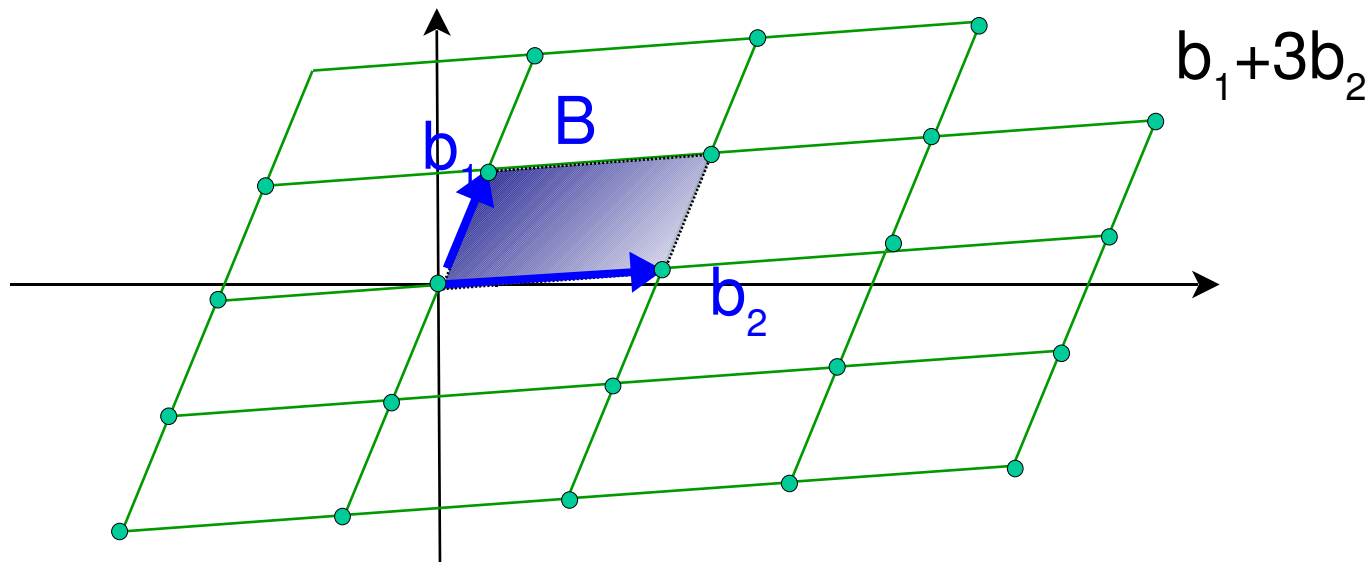
Everything you always wanted to know  
about lattices, but were afraid to ask

Daniele Micciancio

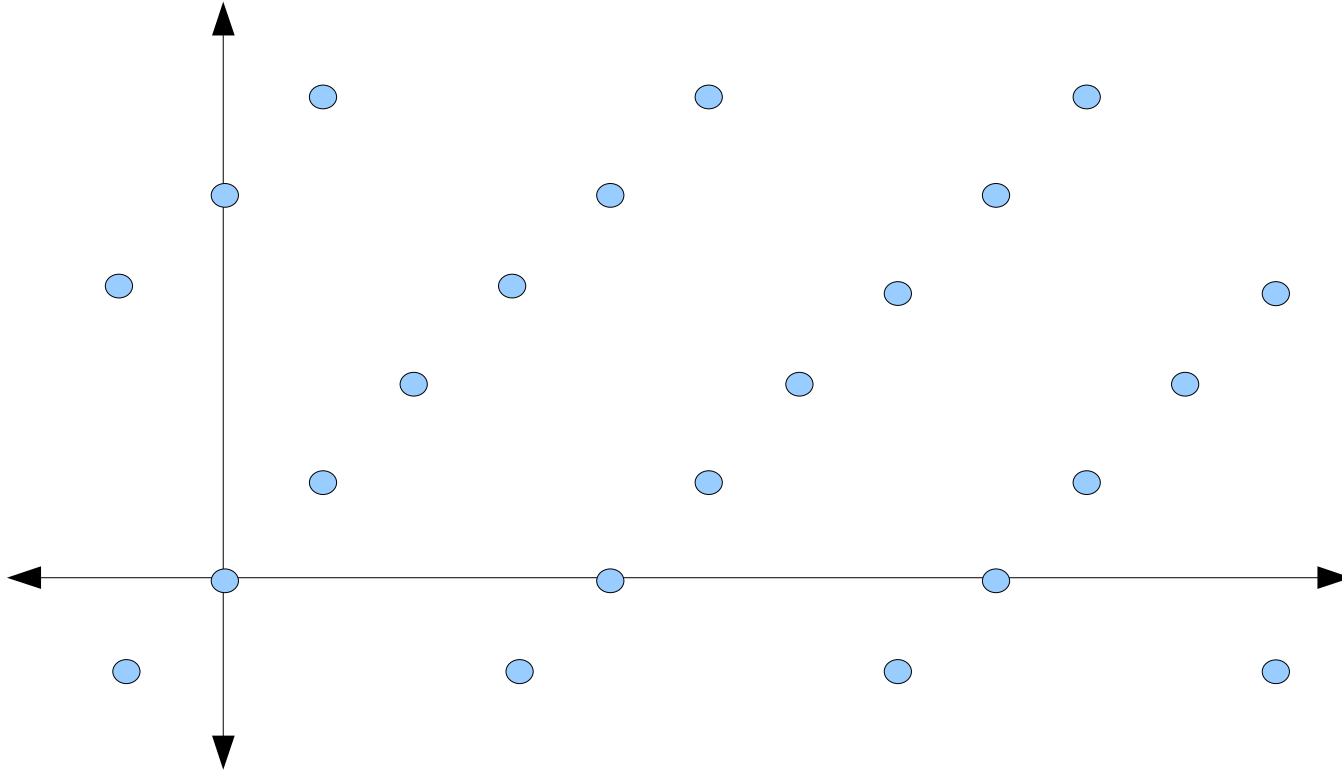
based in part on joint work with Vadim Lyubashevsky  
“On Bounded Distance Decoding, unique Shortest Vectors  
and the Minimum Distance Problem”

# Point Lattices

- Set of all integer linear combinations of basis vectors  $B = \{b_1, \dots, b_n\} \subset \mathbb{R}^n$
- $L(B) = \{Bx : x \in \mathbb{Z}^n\} \subset \text{span}(B) = \{Bx : x \in \mathbb{R}^n\}$



# Lattices

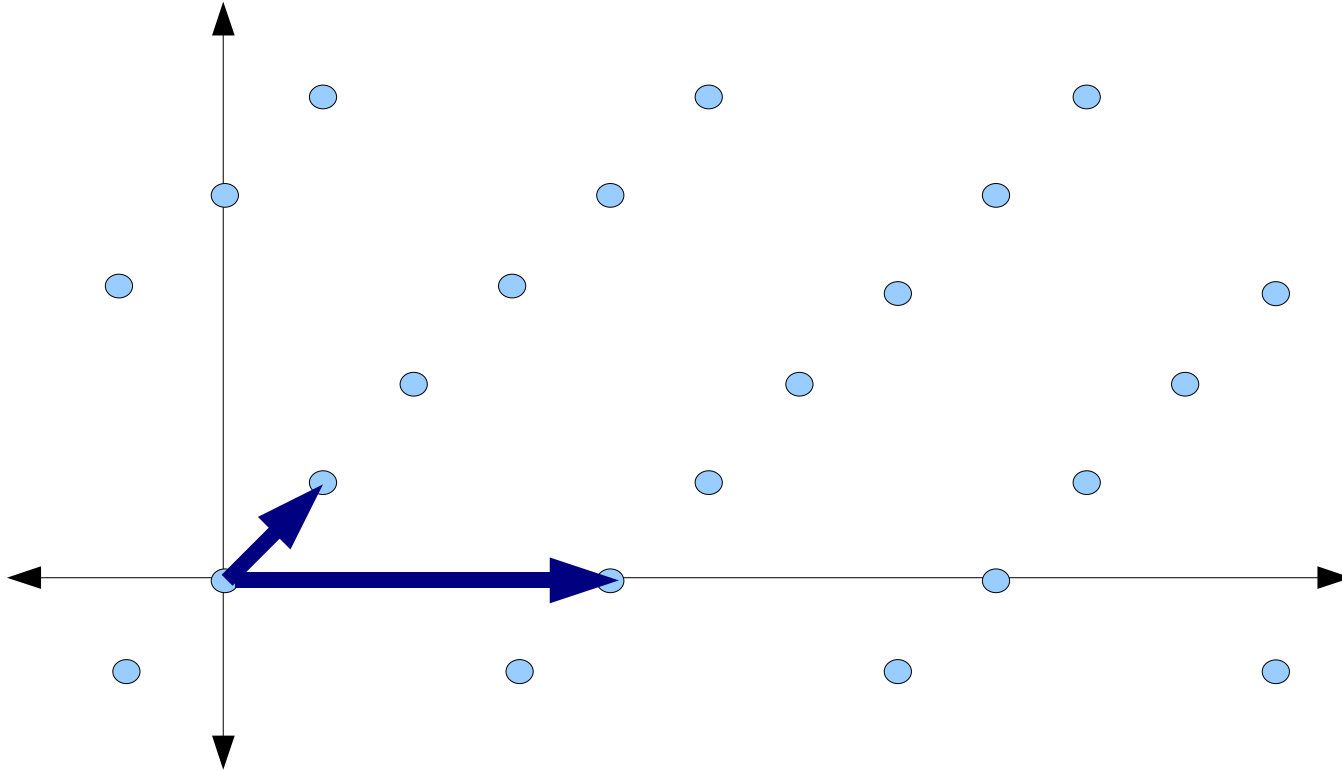


**Lattice: A discrete subgroup of  $\mathbb{R}^n$**

Group elements are vectors.

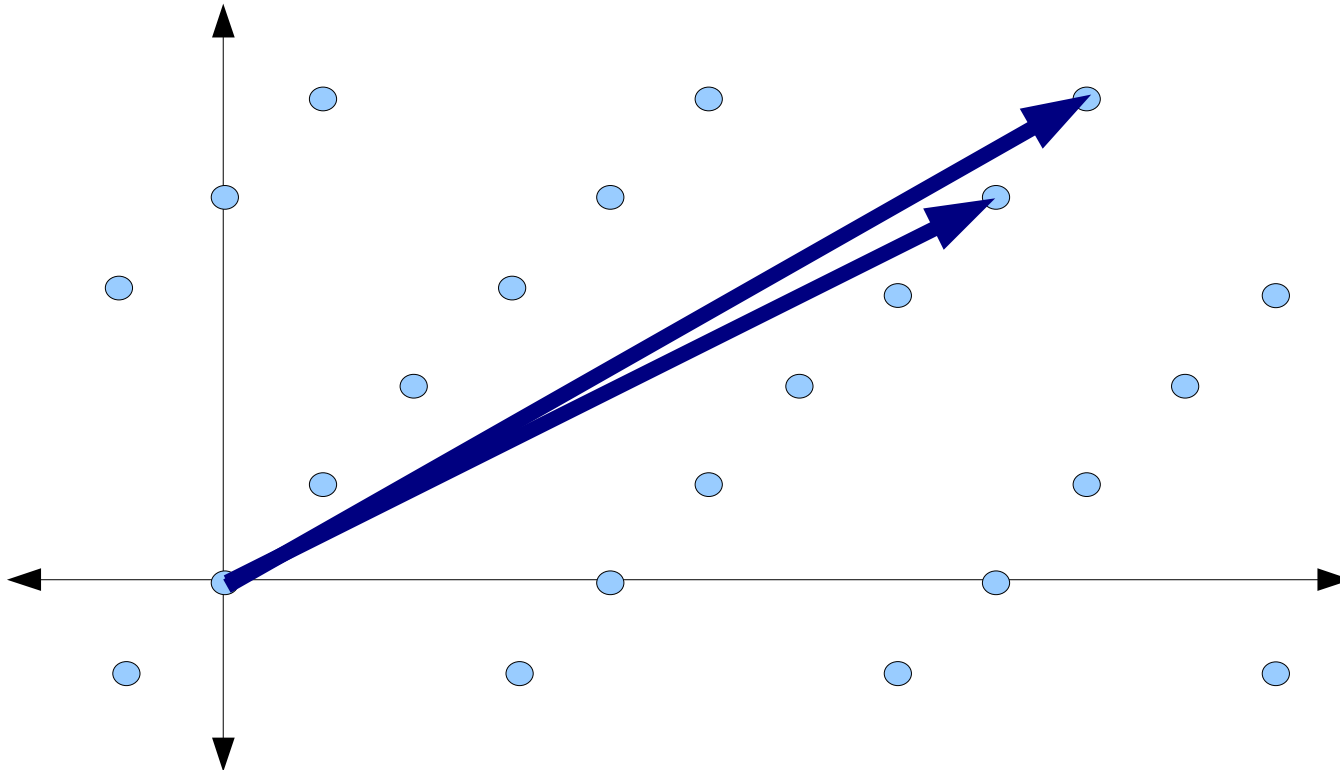
Group operation is the usual vector addition.

# Basis



Basis: A set of linearly independent vectors that generate the lattice.

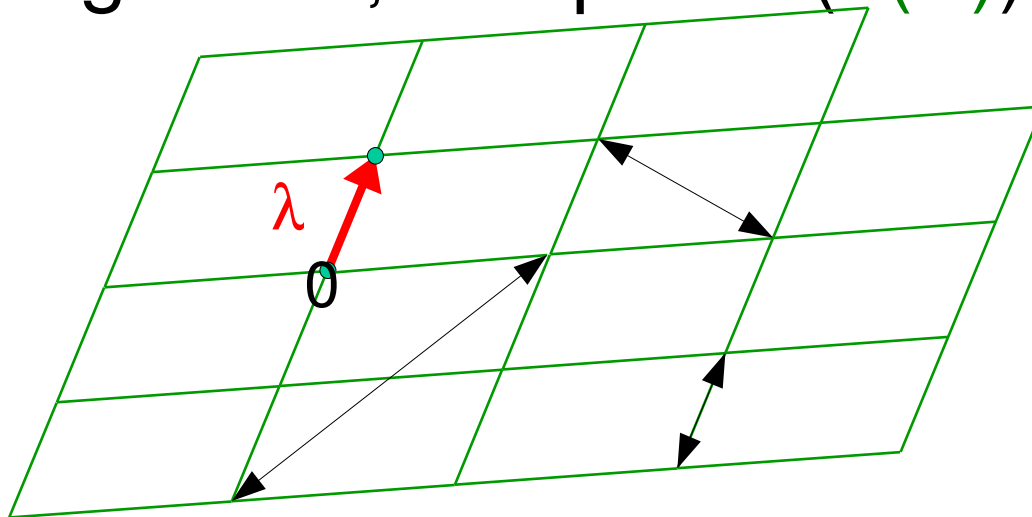
# Basis



Basis: A set of linearly independent vectors that generate the lattice.

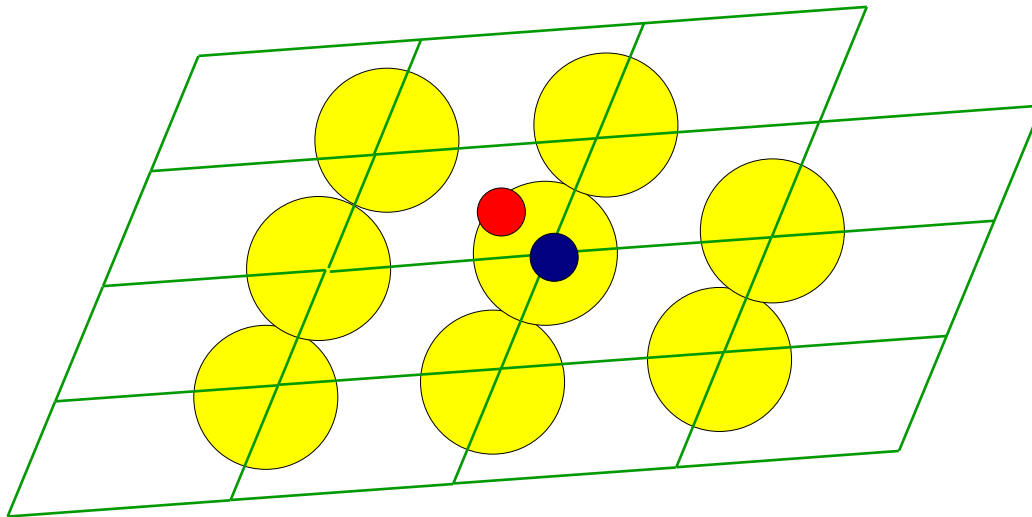
# Minimum Distance Problem

- $\lambda(L(B)) = \min\{\|v-w\|: v \neq w \in L(B)\}$   
=  $\min\{\|v\|: v \in L(B) \setminus \{0\}\}$
- Length of shortest nonzero lattice vector
- **Problem:** given  $B$ , compute  $\lambda(L(B))$



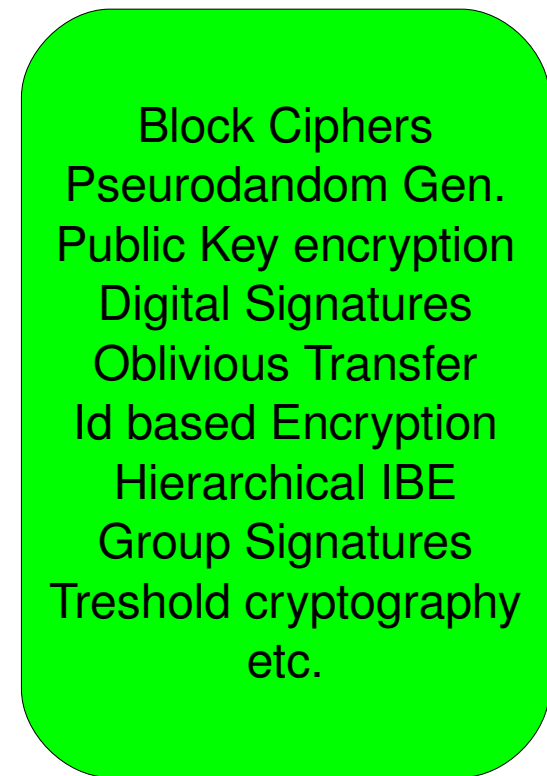
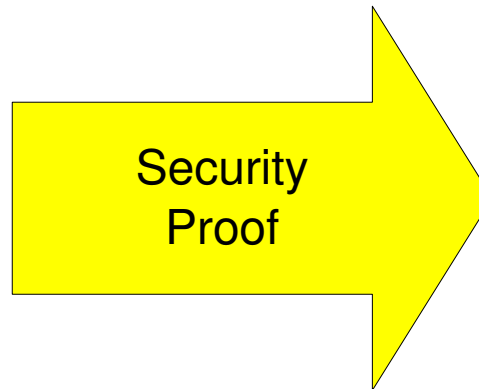
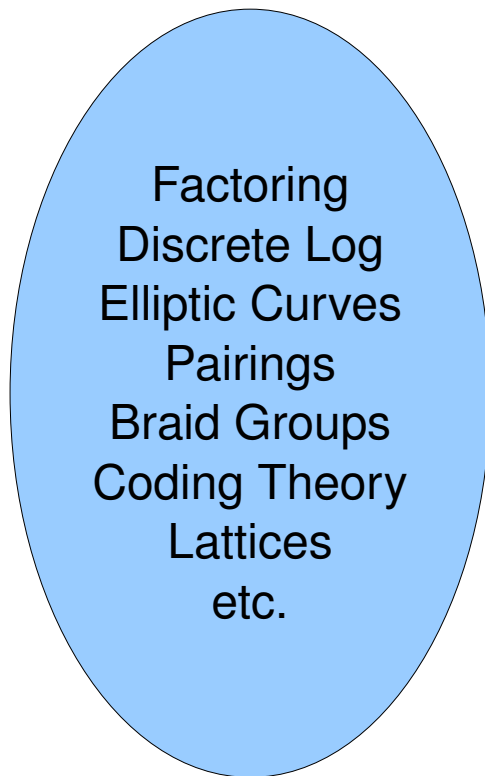
# Bounded Distance Decoding

- Input: Lattice  $\mathbf{B}$  and target  $\mathbf{t} = \mathbf{B}\mathbf{x} + \mathbf{e}$
- Goal: Recover lattice point  $\mathbf{B}\mathbf{x}$  closest to  $\mathbf{t}$
- Application: Error correcting codes in  $\mathbb{R}^n$



# Cryptography

- Find hard **mathematical problems**
- Use them to build hard to break **crypto functions**



# Why are lattices interesting for cryptography?

- Exhibit enough structure to build rich set of cryptographic primitives
  - Public Key Encryption, ..., Oblivious Transfer, (Hierarchical) Identity Based Encryption, Fully Homomorphic Encryption!
- Supported but strong theoretical security proofs
  - Cryptographic security is proved based on worst-case hardness assumptions

# Private Key Encryption

- Bounded Distance Decoding:
  - Given  $B$  and  $t=Bx+e$ , it is hard to recover  $Bx$
- Private key encryption
  - Key: random  $x$
  - $\text{Encrypt}(m;B,e) = (B, Bx + e + \text{Encode}(m)) = (B,u)$
  - $\text{Decrypt}(B,u) = \text{Decode}(u - Bx)$

Randomness used to encrypt

Error correcting code

- It works:

$$u - Bx = Bx + e + \text{Encode}(m) - Bx = e + \text{Encode}(m)$$

$$\text{Decode}(u - Bx) = \text{Decode}(e + \text{Encode}(m)) = m$$

# Security of private key encryption

- Consider a BDD instance:  $(B, Bx+e)$
- Let  $H$  be the dual lattice of  $B$ 
  - Knowing  $Bx+e$  for random  $x$  is equivalent to knowing the “syndrome”  $H(Bx+e) = He$
- (dual) BDD: Given  $(H, He)$ , find  $e$ 
  - This is a subsetsum or knapsack problem
  - [Impagliazzo, Naor]:  $e \Rightarrow He$  is a good PRG
  - If  $He$  looks random, then  $Bx+e$  looks random

# Security of private key encryption

- Consider a BDD instance:  $(B, Bx+e + m)$
- Let  $H$  be the dual lattice of  $B$ 
  - Knowing  $Bx+e$  for random  $x$  is equivalent to knowing the “syndrome”  $H(Bx+e) = He$
- (dual) BDD: Given  $(H, He)$ , find  $e$ 
  - This is a subsetsum or knapsack problem
  - [Impagliazzo, Naor]:  $e \Rightarrow He$  is a good PRG
  - If  $He$  looks random, then  $Bx+e$  looks random, and can be used as a **one-time pad**

# Homomorphic properties

- $\text{Enc}_x(m; B, e) = (B, Bx + e + m)$ 
  - $\text{Enc}_x(0; B, e) = (B, Bx + e)$

# Homomorphic properties

- $\text{Enc}_x(m; B, e) = (B, Bx + e + m)$ 
  - $\text{Enc}_x(0; B, e) = (B, Bx + e)$
- $\text{Enc}_x(0; B, e) + \text{Enc}(0; B', e') = ???$

# Homomorphic properties

- $\text{Enc}_x(m; B, e) = (B, Bx + e + m)$ 
  - $\text{Enc}_x(0; B, e) = (B, Bx + e)$
- $\text{Enc}_x(0; B, e) + \text{Enc}_x(0; B', e') = ???$ 
  - $(B+B', Bx+e + B'x+e') = \text{Enc}_x(0; B+B', e+e')$

# Homomorphic properties

- $\text{Enc}_x(m; B, e) = (B, Bx + e + m)$ 
  - $\text{Enc}_x(0; B, e) = (B, Bx + e)$
- $\text{Enc}_x(0; B, e) + \text{Enc}(0; B', e') = ???$   
 $(B+B', Bx+e + B'x+e') = \text{Enc}_x(0; B+B', e+e')$
- $\text{Enc}_x(0; B, e) + (O, m) = ???$

# Homomorphic properties

- $\text{Enc}_x(m; B, e) = (B, Bx + e + m)$ 
  - $\text{Enc}_x(0; B, e) = (B, Bx + e)$
- $\text{Enc}_x(0; B, e) + \text{Enc}(0; B', e') = ???$ 
  - $(B+B', Bx+e + B'x+e') = \text{Enc}_x(0; B+B', e+e')$
- $\text{Enc}_x(0; B, e) + (O, m) = ???$ 
  - $(B+O, Bx+e+m) = \text{Enc}_x(m; B, e)$

# Public Key Encryption

- Secret key:  $x$
- Public Key:  $c_i = \text{Enc}(0)$ , for  $i=1, \dots, k$
- Encryption of  $m$  with randomness  $r_i$   
 $(\sum r_i c_i) + (0, m) = \text{Enc}(0) + \dots + \text{Enc}(0) + \text{Enc}(m) = \text{Enc}(m)$
- Decrypt using secret key  $x$
- Notice:  $x$  is not needed to encrypt!
- [Ajtai, Dwork] and [Regev] encryption schemes

# Hashing and Signing

- Let  $H$  be a random matrix
  - $e \Rightarrow He$  is a good hash function
  - Collisions  $Hx = Hy$  give short  $(x-y)$  in  $\text{Kernel}(H)$
  - Homomorphic property:  $Hx + Hy = H(x+y)$
- One-time signatures [M., Lyubashevsky]
  - Secret Key:  $x, y$
  - Public Key:  $X = Hx, Y = Hy$
  - $\text{Sign}(m) = xm+y$
  - $\text{Verify}(X, Y, m, s)$ : Check if  $Hs = Xm+Y$

# Advantages of Lattice cryptography

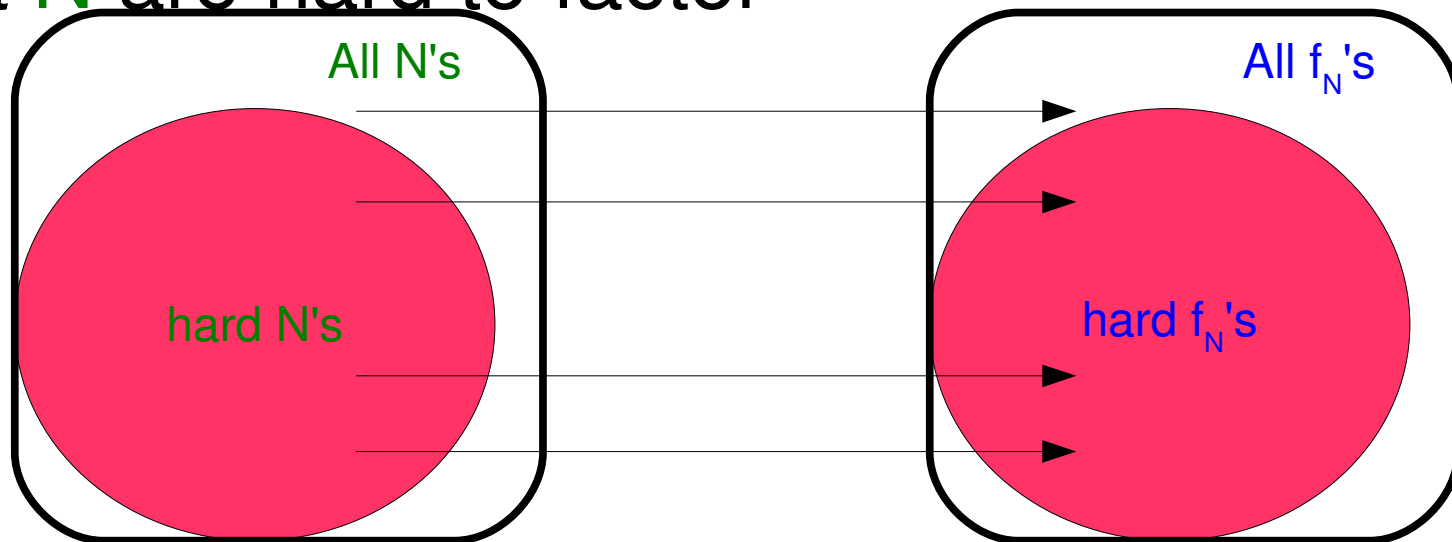
- Strong theoretical foundation
  - Connection between worst and average case complexity
- Potentially very efficient
  - Only simple operations (no big int arithmetic)
- Many applications
  - CPA and CCA Encryption, Digital Signatures, Hash functions, Oblivious Transfer, (Hierarchical) Id based encryption, ..., Fully homomorphic encryption!

# Worst-case vs. Average-case

- Worst-case complexity
  - A problem can be solved in time  $T(n)$  if there is an algorithm with running time  $T(n)$  that solves **all** problems instances of size  $n$
  - Used in algorithms and complexity: P, NP, etc.
- Average-case complexity
  - There is an algorithm that solves a **large fraction** of the instances of size  $n$
  - Used in cryptography: assume there is no such algorithm

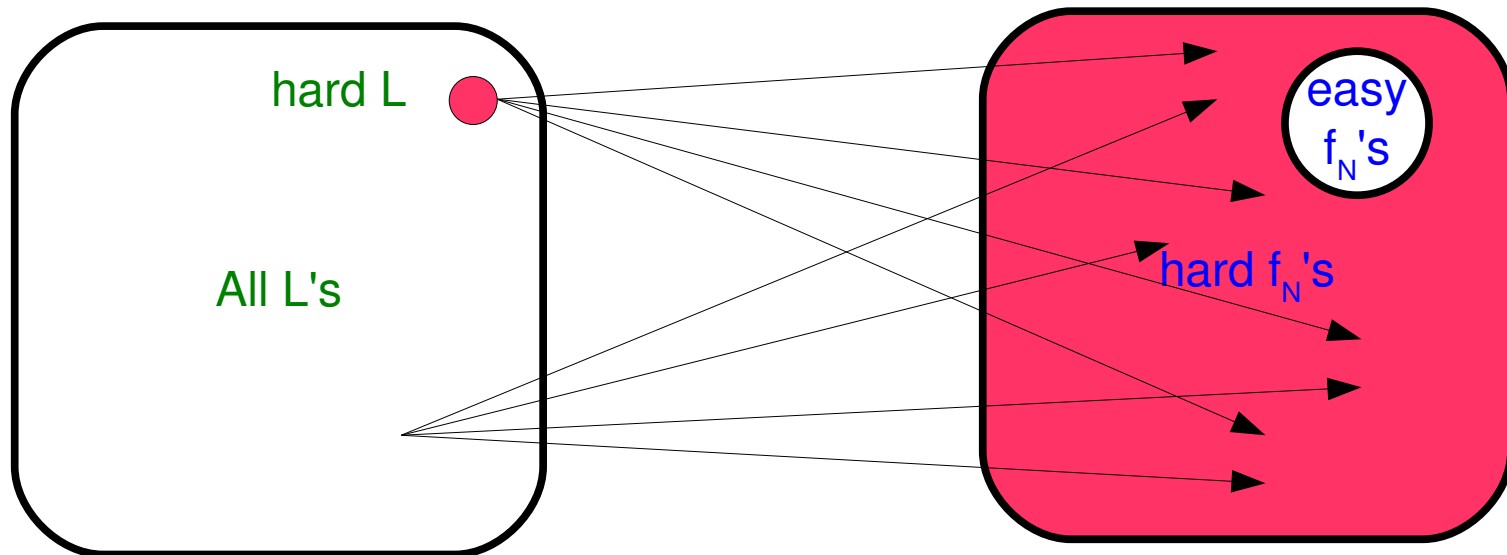
# Provable security from average case hardness

- Example: (Rabin) modular squaring
  - $f_N(x) = x^2 \bmod N$ , where  $N=pq$ , ...
  - Inverting  $f_N$  is as hard as factoring  $N$
- $f_N$  is cryptographically hard to invert, provided *most*  $N$  are hard to factor



# Provable security from worst case hardness

- Any fixed  $L$  is mapped to random  $f_N$
- $f_N$  is cryptographically hard to invert if lattice problem  $L$  is hard in the worst case



# Factoring vs Lattices

	Factoring	Lattices
Math Problem	Factor $N=PQ$	Approximate $\lambda(L)$
Crypto Function	$f_N(x) = x^2 \bmod N$	$f_A(x) = Ax$
Security Assumption	<b>Random</b> $N$ is hard to factor	<b>Some</b> $\lambda(L)$ is hard to approximate
Note	$N$ must follow hard distribution	

# Picking a Hard-to-Factor $N$

How do you pick a “good”  $N$ ?

Just pick  $p, q$  as random large primes and set  $N=pq$ ?

(1978) Largest prime factors of  $p-1, q-1$  should be large

(1981)  $p+1$  and  $q+1$  should have a large prime factor

(1982) If the largest prime factor of  $p-1$  and  $q-1$  is  $p'$  and  $q'$ , then  $p'-1$  and  $q'-1$  should have large prime factors

(1984) If the largest prime factor of  $p+1$  and  $q+1$  is  $p'$  and  $q'$ , then  $p'-1$  and  $q'-1$  should have large prime factors

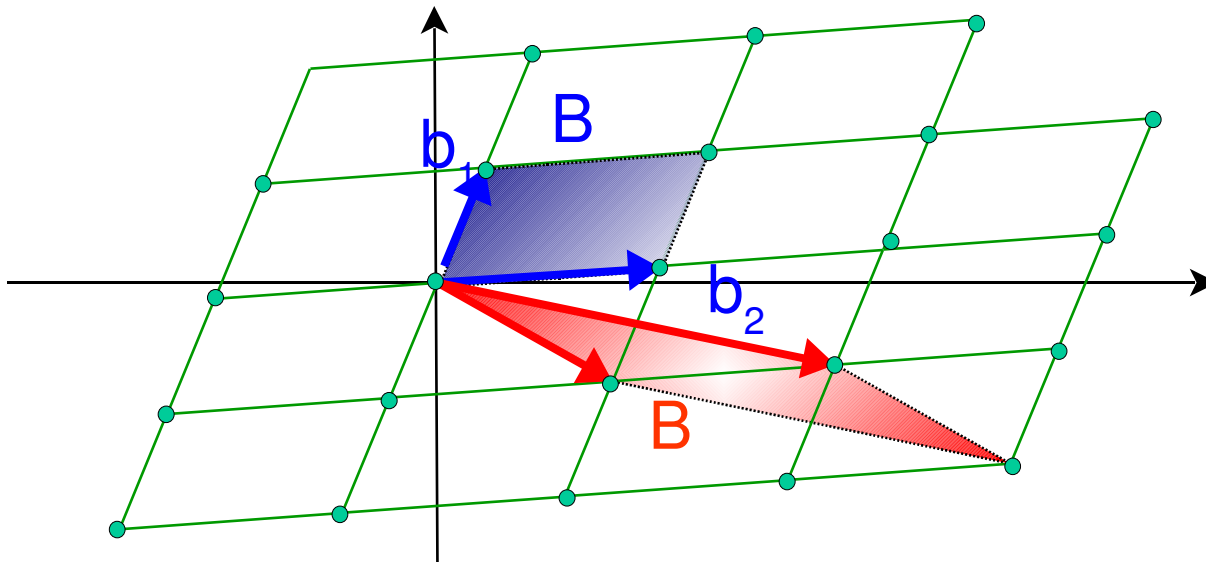
...

# Picking a hard matrix $A$

- Just choose  $A$  uniformly at random in  $Z_q^{n \times m}$
- Security [Ajtai]
  - Assume some lattice problem is hard in worst case
  - Then inverting  $f_A(x)$  is hard with very high probability
- [1997, 1998, ..., 2009, ...]
  - Still choose  $A$  uniformly at random
  - Worst-case/Average-case proof tell us that's the right distribution

# Lattices and Bases

- Every lattice has infinitely many bases
  - $B = \{b_2 - b_1, 2b_2 - b_1\} = BU$  where  $|\det(U)|=1$
- $B$  and  $B$  define different “grids”, but with the same set of intersection points.



# GapSVP

- Shortest Vector Problem:

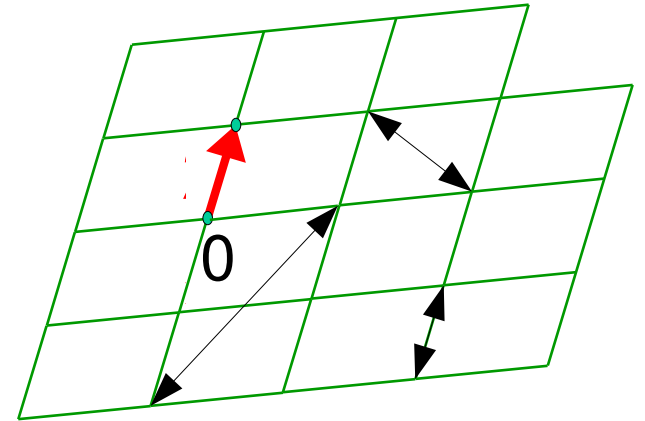
- Given  $B$ , compute  $\lambda(L(B))$

- GapSVP:

- Given  $B$ , **approximate**  $\lambda(L(B))$  within factor  $g$

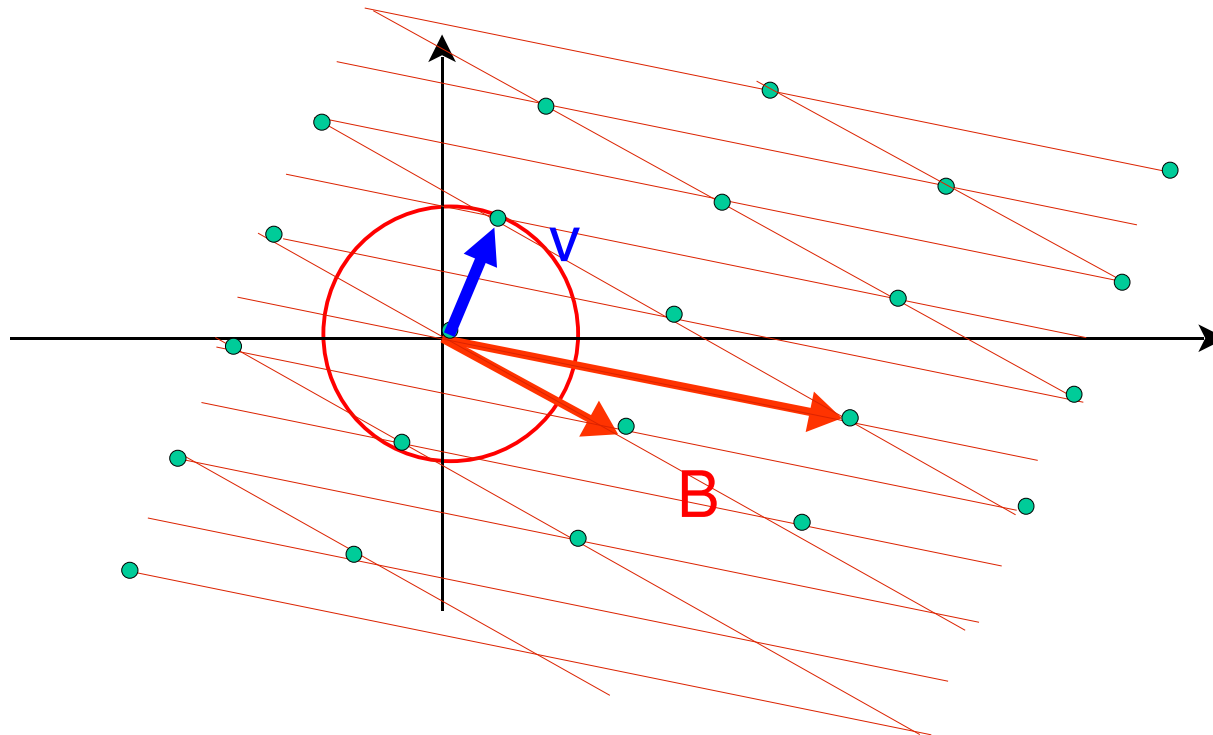
- Complexity:

- NP hard for any constant  $g=O(1)$  [Khot'04]
- CoAM for  $g=O(n/\log n)^{1/2}$  [Goldreich,Wasser'97]
- Polytime for  $g=\exp(n \log \log n / \log n)$  [LLL,AKS]



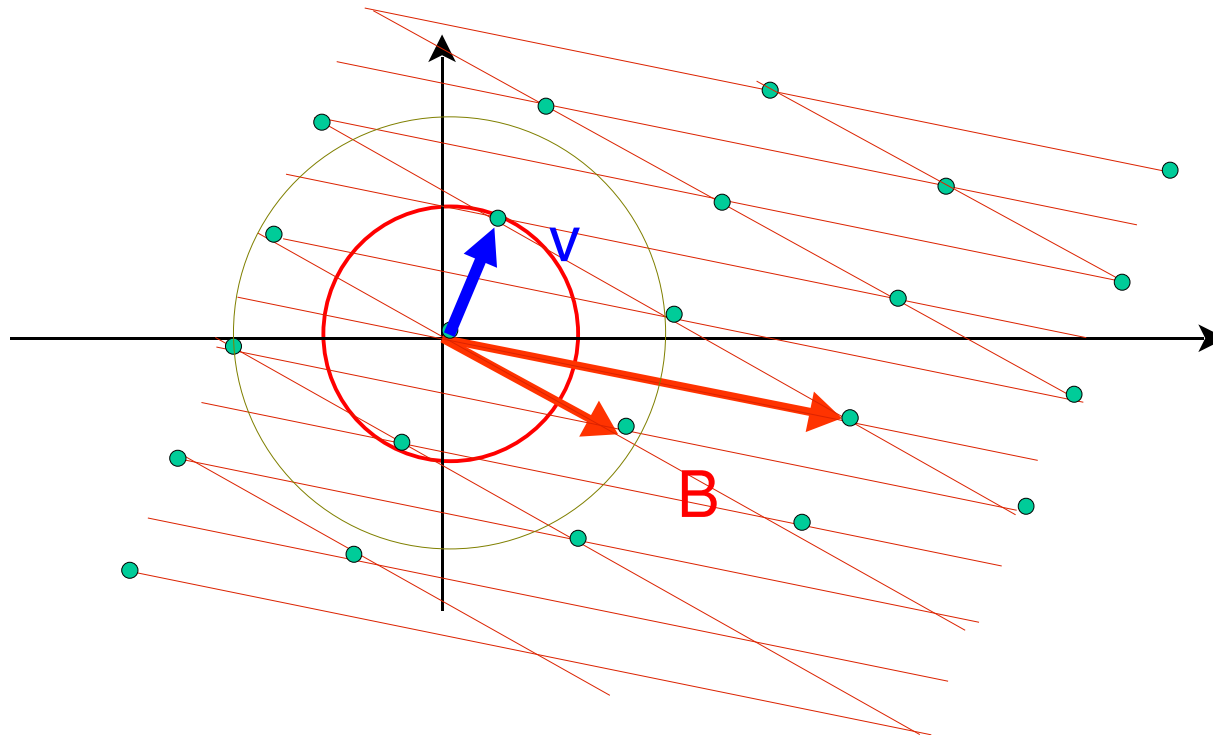
# Shortest Vector Problem (SVP)

- Given a lattice  $B$ , find nonzero lattice vector  $v$  closest to the origin ( $\|v\| = \lambda$ )



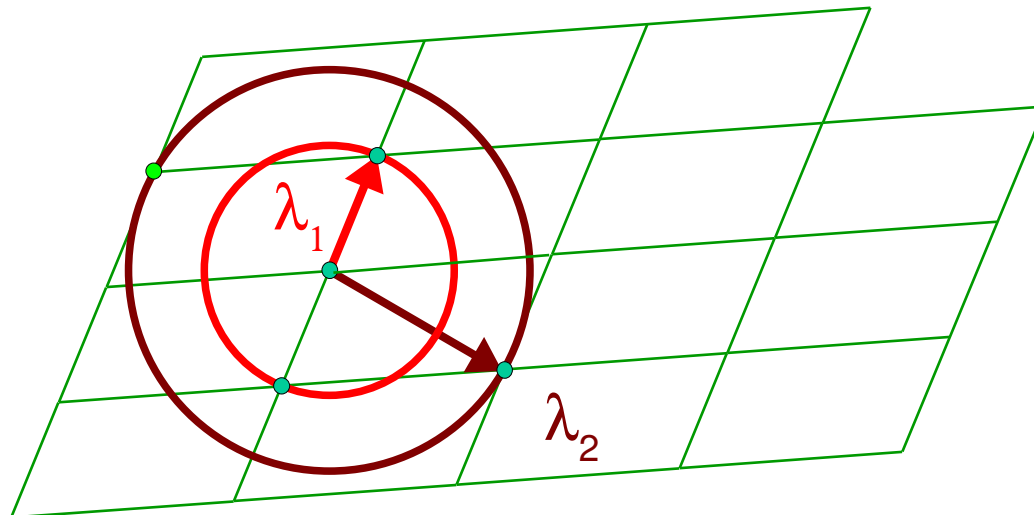
# Approximate SVP

- Given a lattice  $B$ , find nonzero lattice vector  $v$  close to the origin ( $\|v\| < g \lambda_1$ )



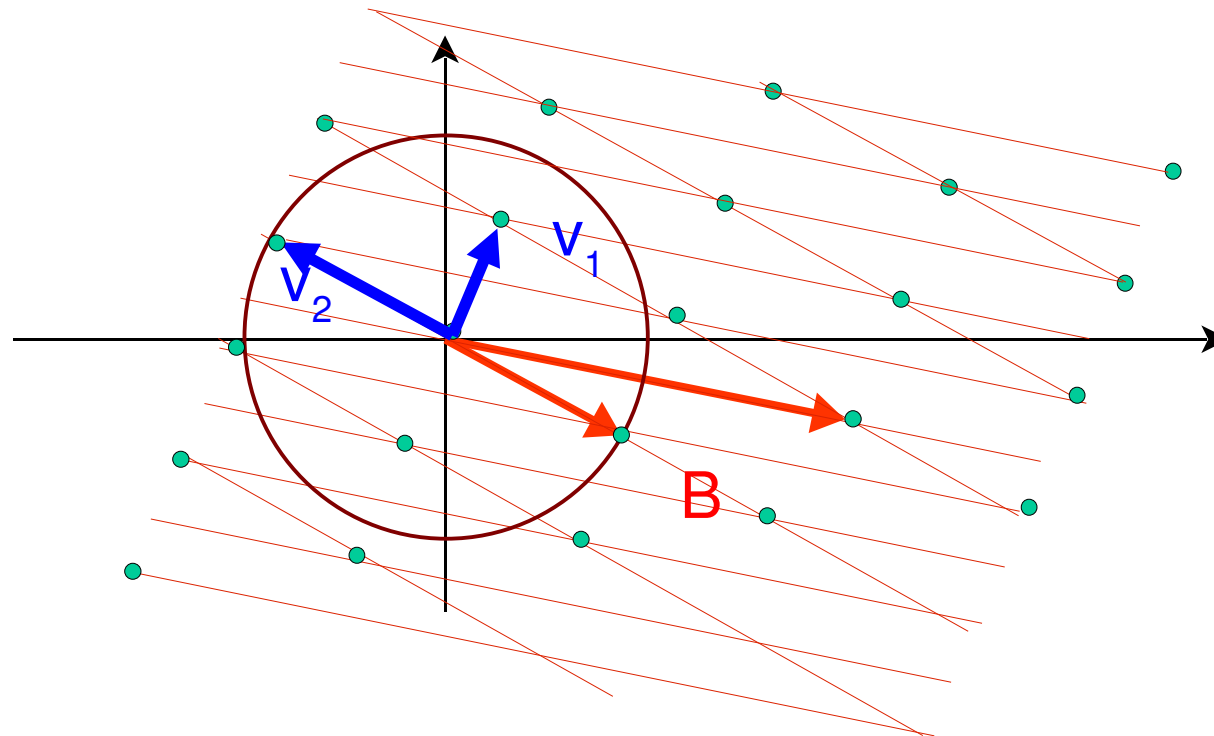
# Successive Minima

- For every  $n$ -dimensional lattice  $B$ , and  $i=1,\dots,n$ , the  $i$ th successive minimum  $\lambda_i(B)$  is the smallest radius  $r$  such that  $B(0,r)$  contains  $i$  linearly independent lattice vectors



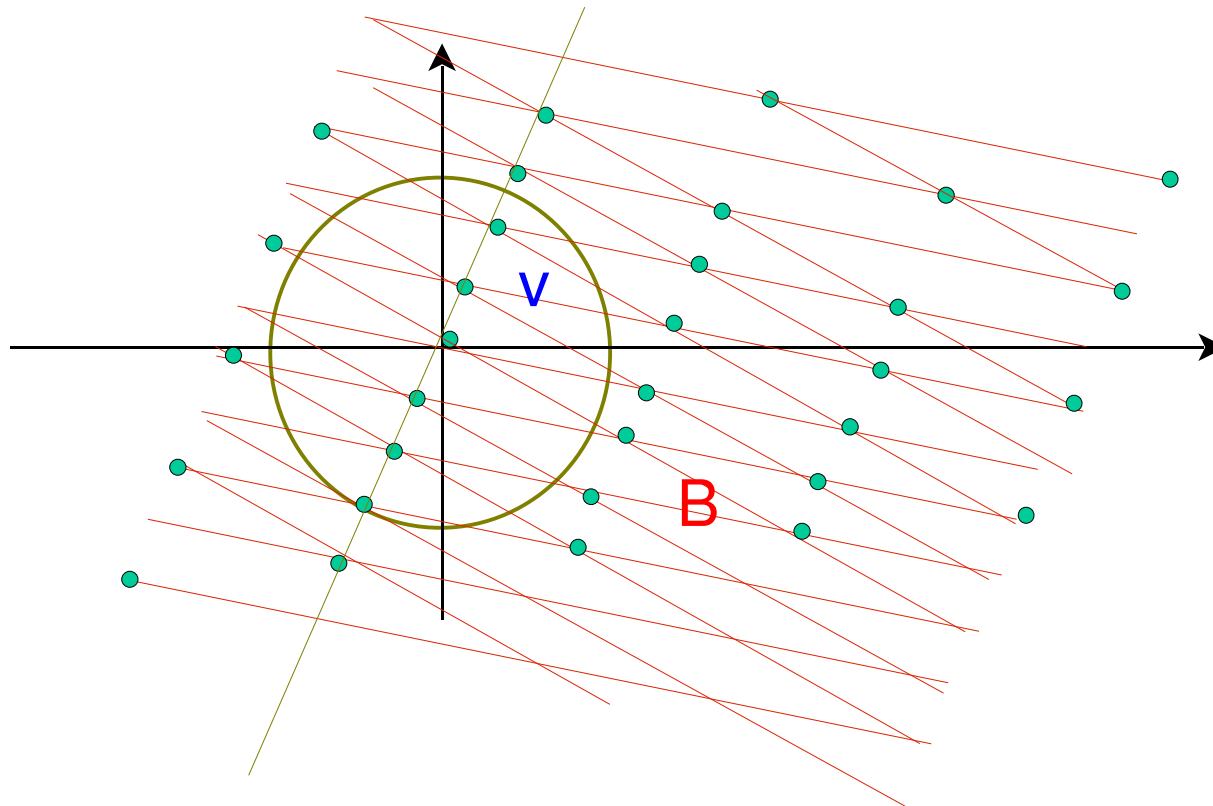
# Shortest Independent Vector Problem (SIVP)

- Given a lattice  $B$ , find  $n$  linearly independent vectors  $v_1, \dots, v_n$  of length  $\max_i \|v_i\| \leq g \lambda_n$



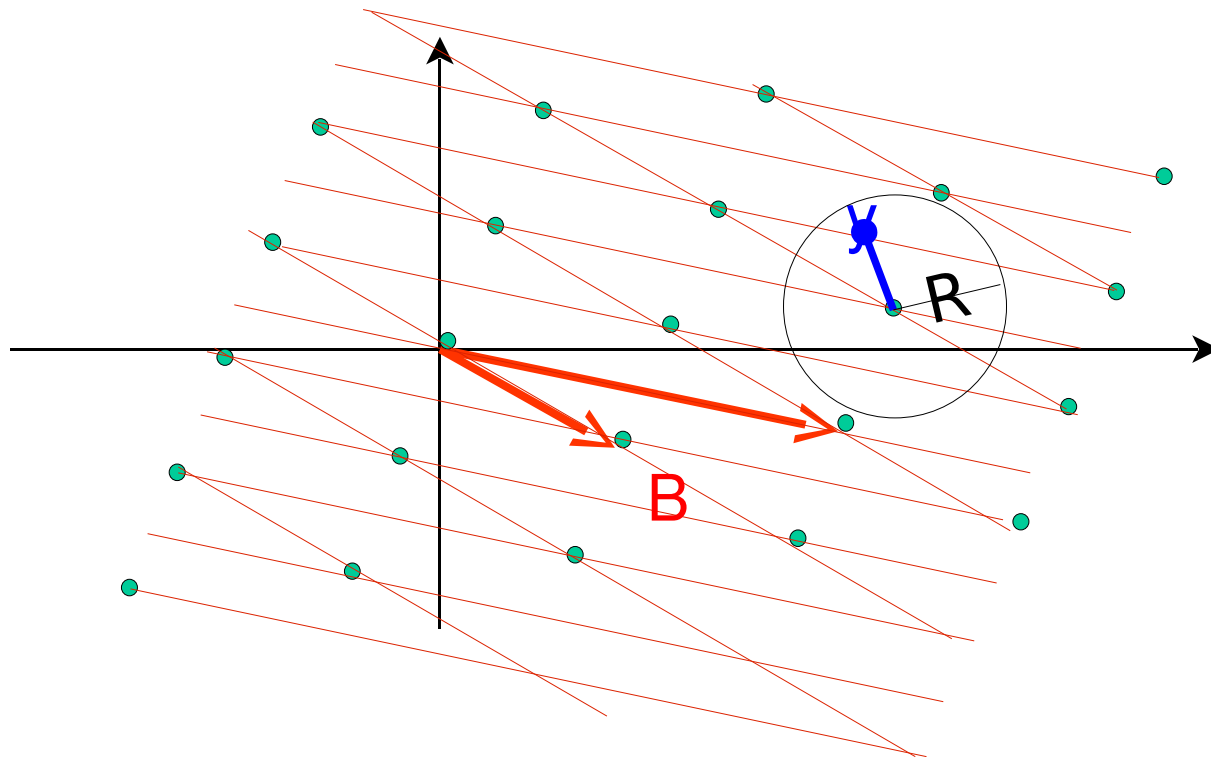
# Unique SVP (uSVP)

- Given a lattice  $B$  such that  $\lambda_1 \ll \lambda_2$ , find shortest vector  $\|v\| = \lambda_1$



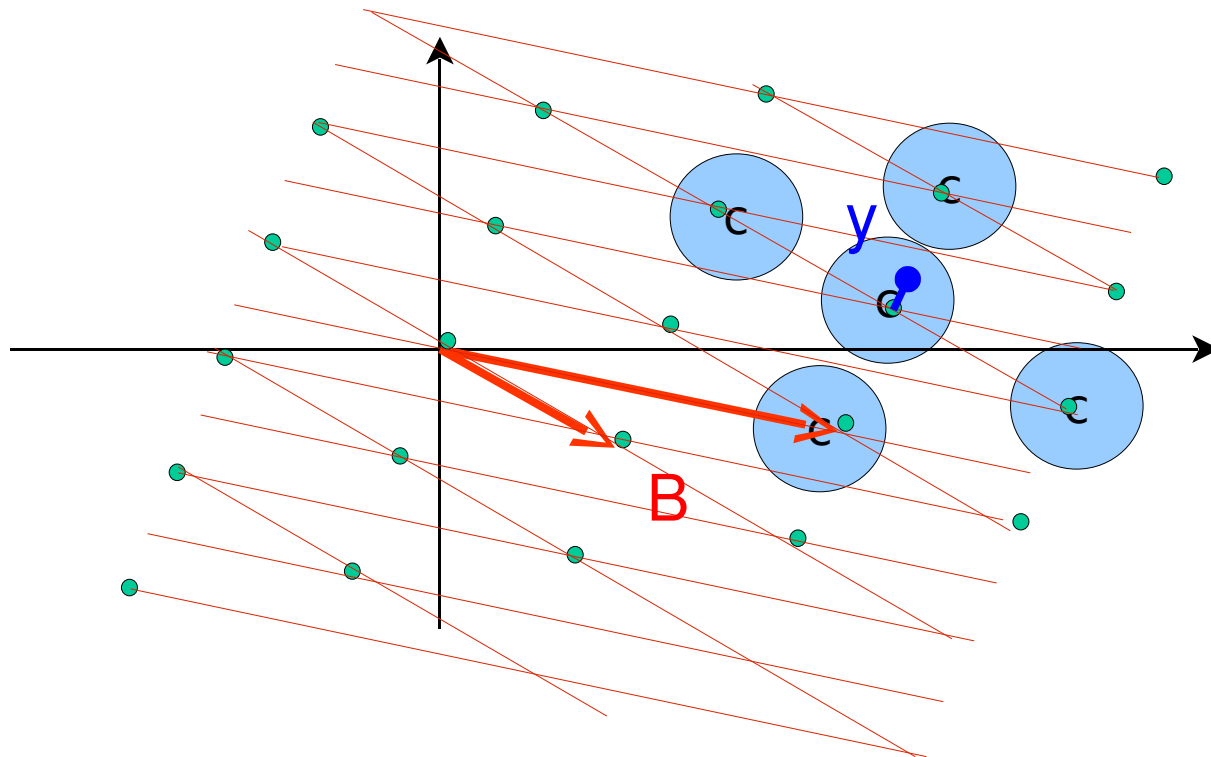
# Closest vector problem

- Given a lattice  $B$  and a point  $y$ , find the lattice point closest to  $y$



# Bounded Distance Decoding

- Given a lattice  $B$  and a point  $y$ , find the lattice point closest to  $y$ , assuming  $\text{dist}(y, B) < \lambda/2$



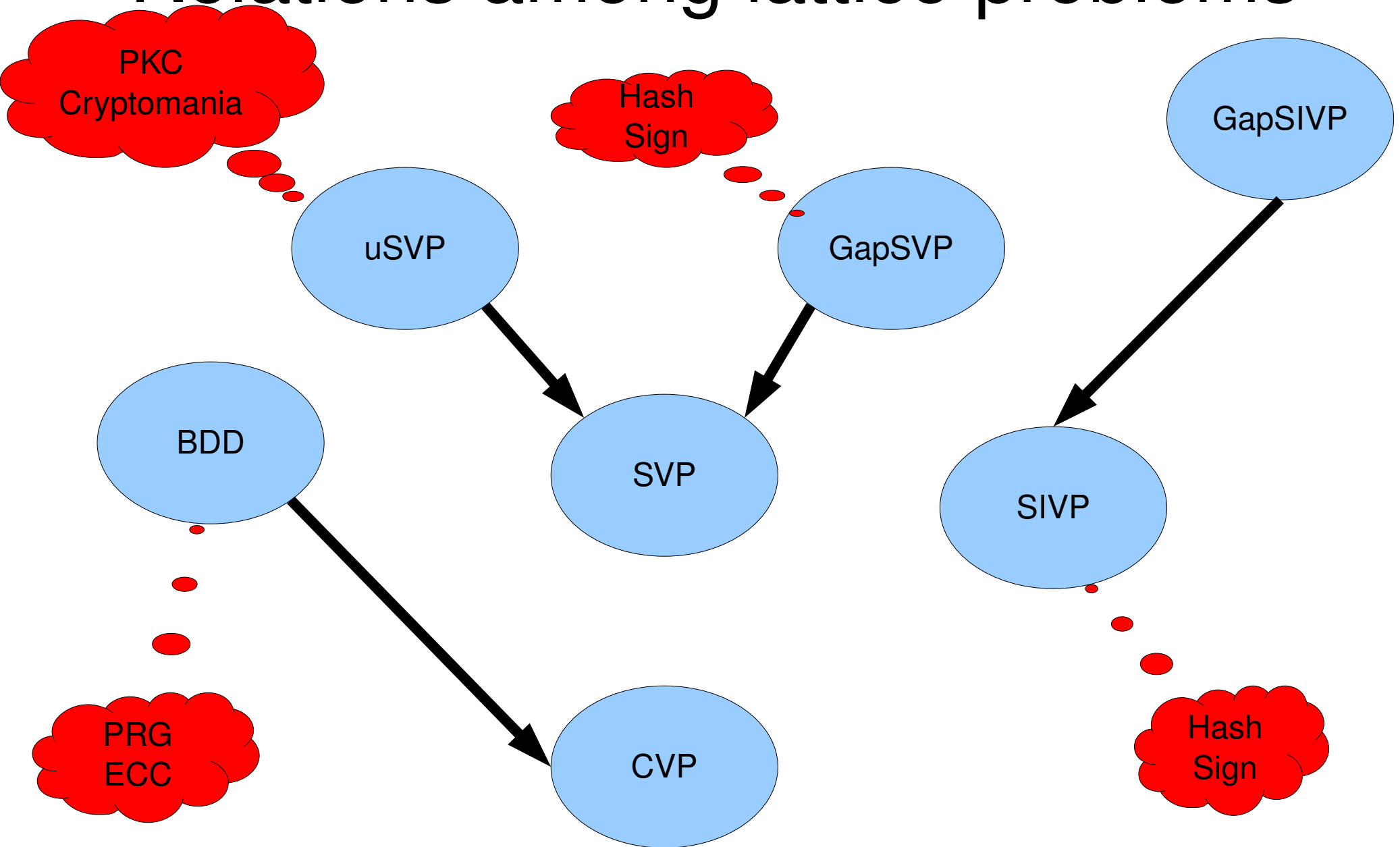
# Lattice problems

- Many problems:
  - SVP, CVP, SIVP, GapSVP, GapSIVP, GapCVP, BDD, uSVP, SAP, GCVP, GDD, CRP, SVP', ...
- Many applications:
  - uSVP: Public key encryption
  - BDD: Error correcting codes, pseudorandom gen.
  - SIVP: cryptographic hash functions, signatures
  - GapSVP: Complexity theory, etc.
- Question: how can we compare problems?

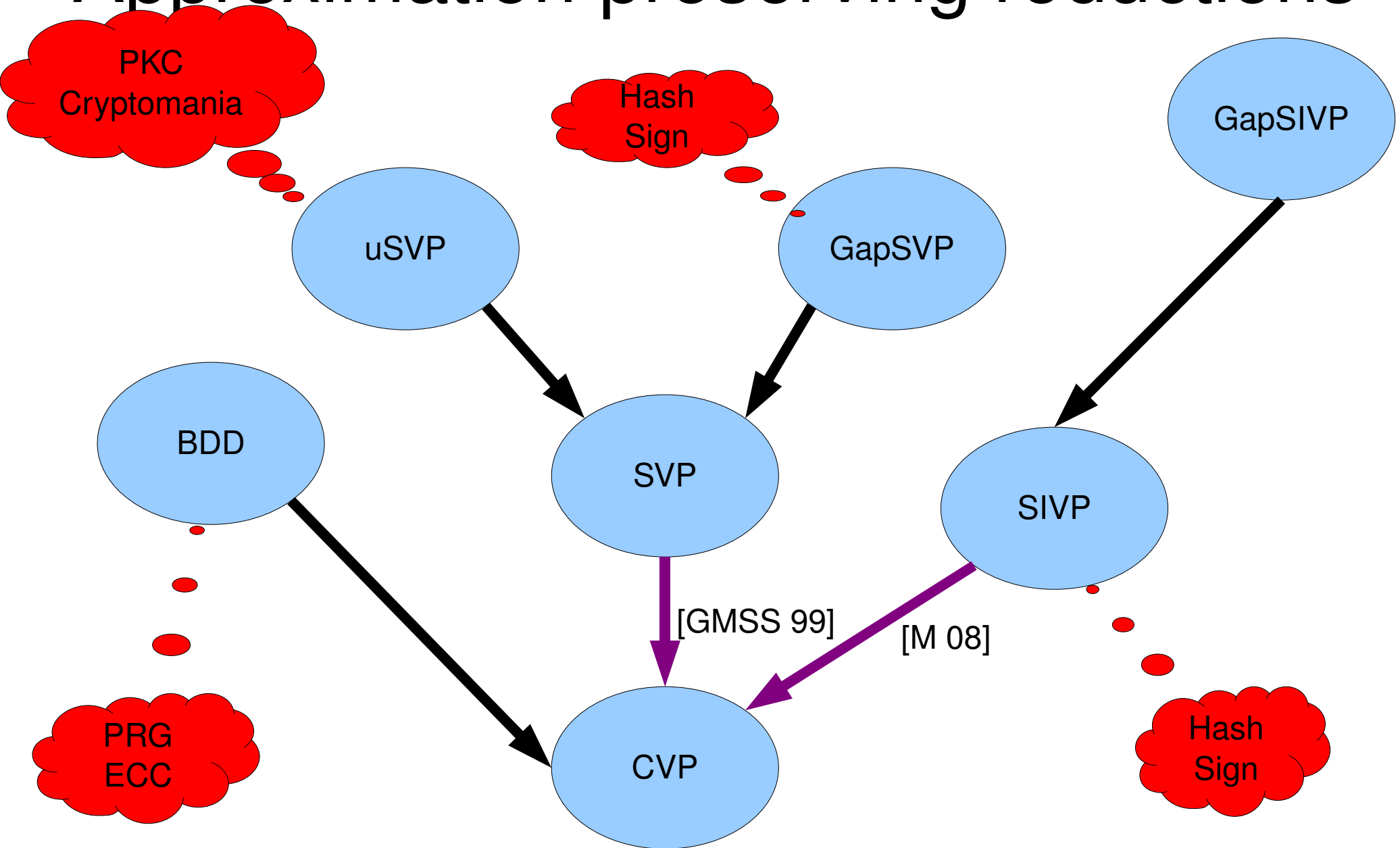
# Comparing lattice problems

- Approximation preserving reductions:
  - Approximating problem A within  $\epsilon$ , allows to approximate problem B within  $O(\epsilon)$
  - A is at least as hard as B
- Polynomial reductions
  - Approximating A within  $\epsilon$ , allows to approximate B within  $\text{poly}(n, \epsilon)$
  - Already interesting in cryptography

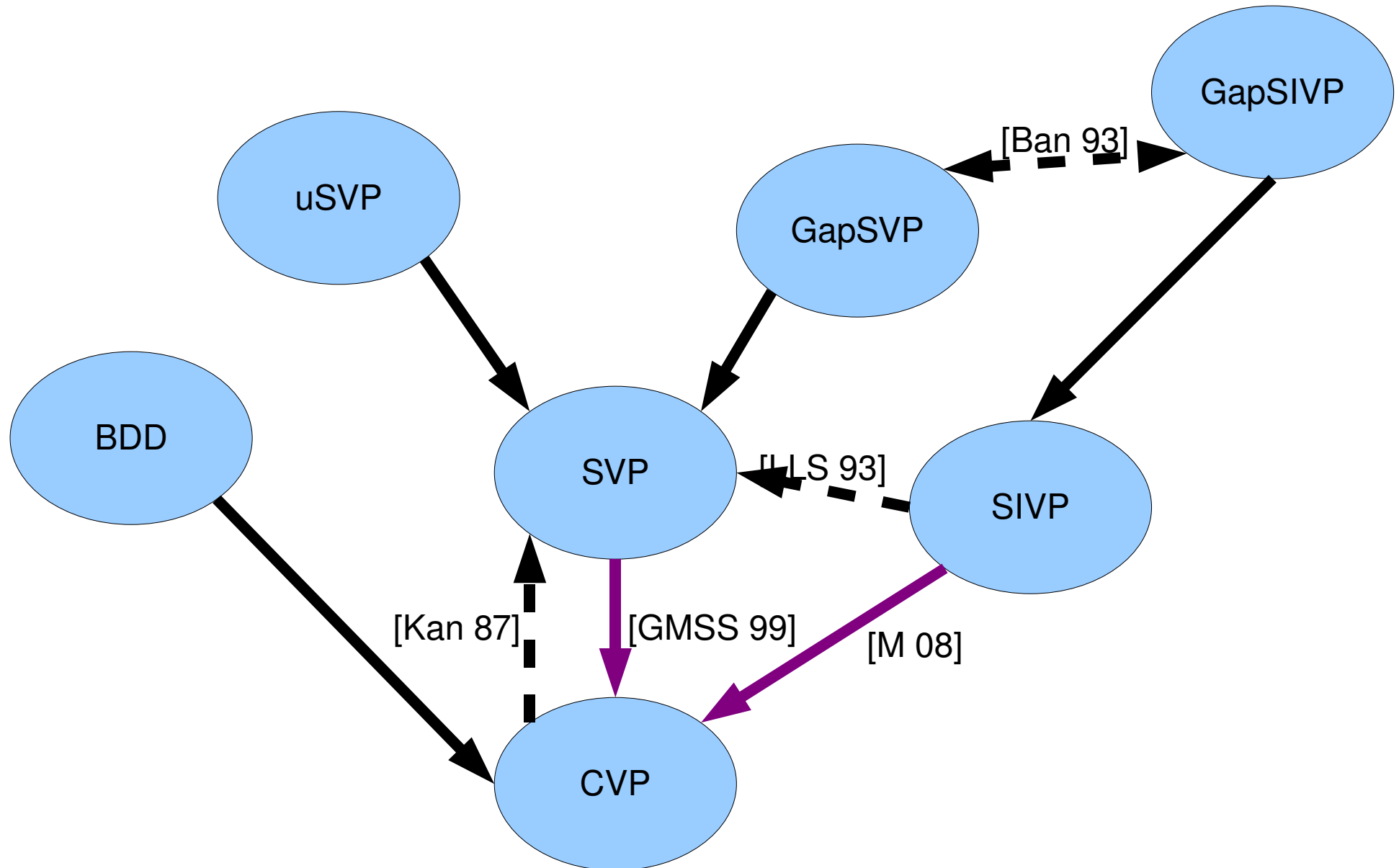
# Relations among lattice problems



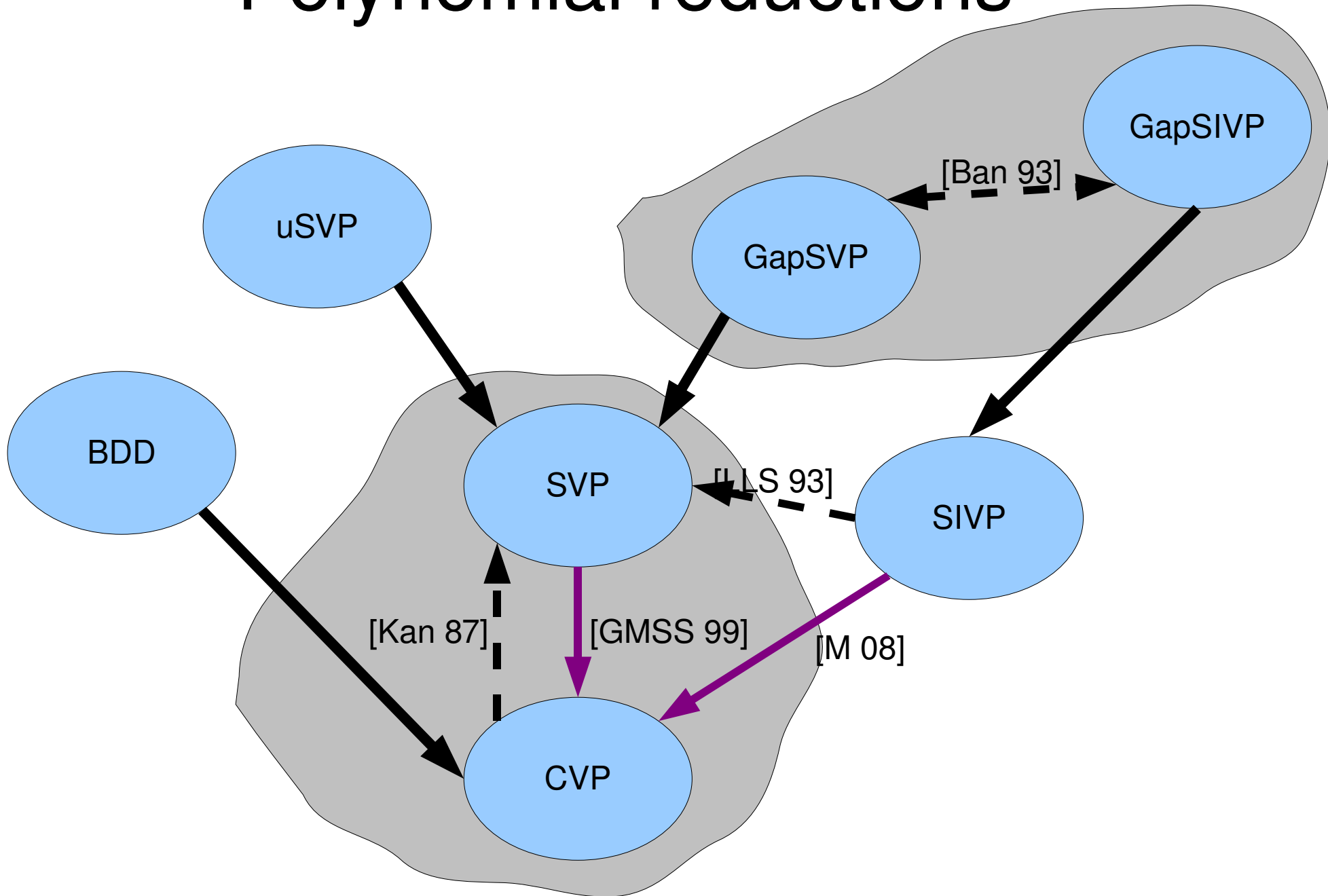
# Approximation preserving reductions



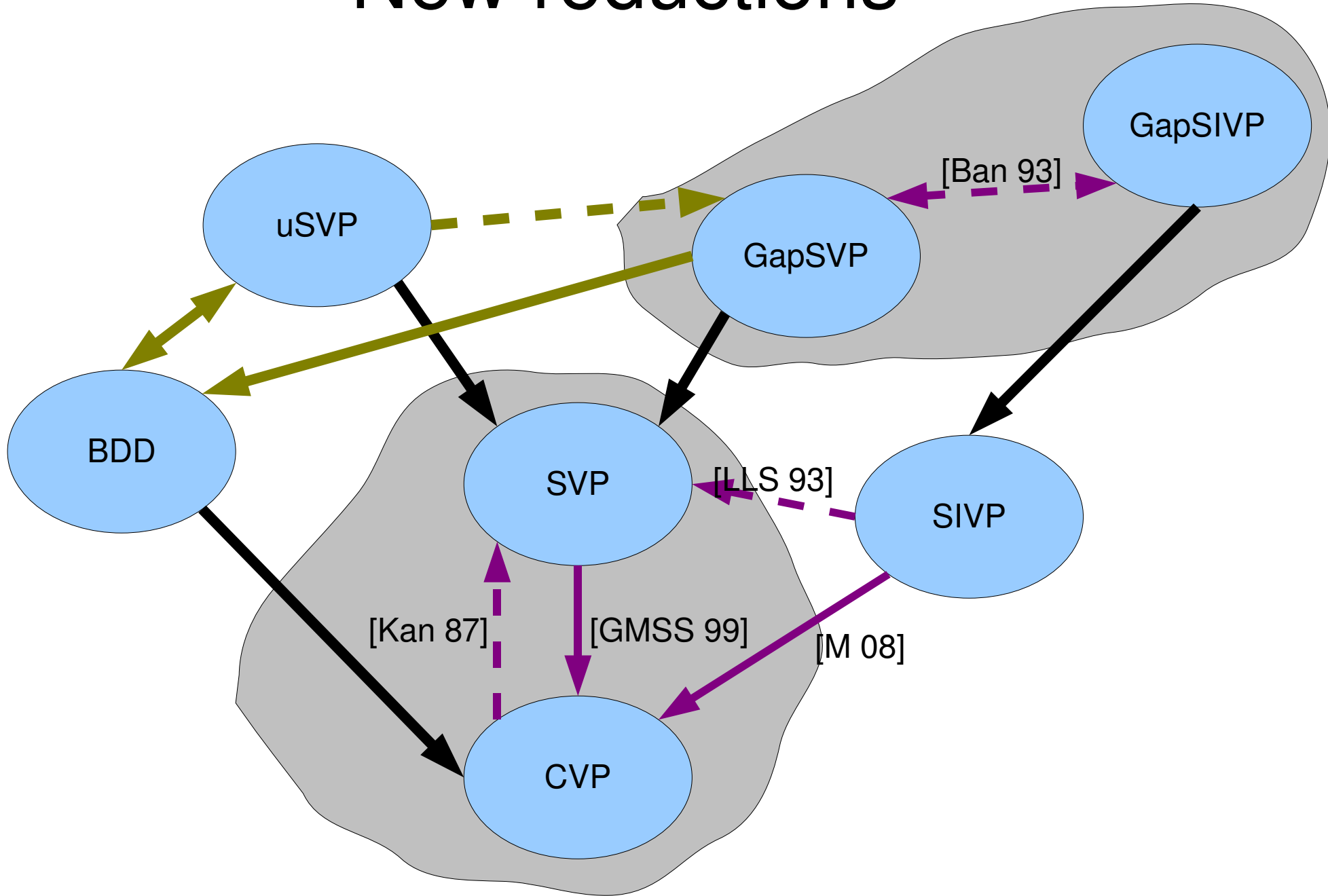
# Polynomial reductions



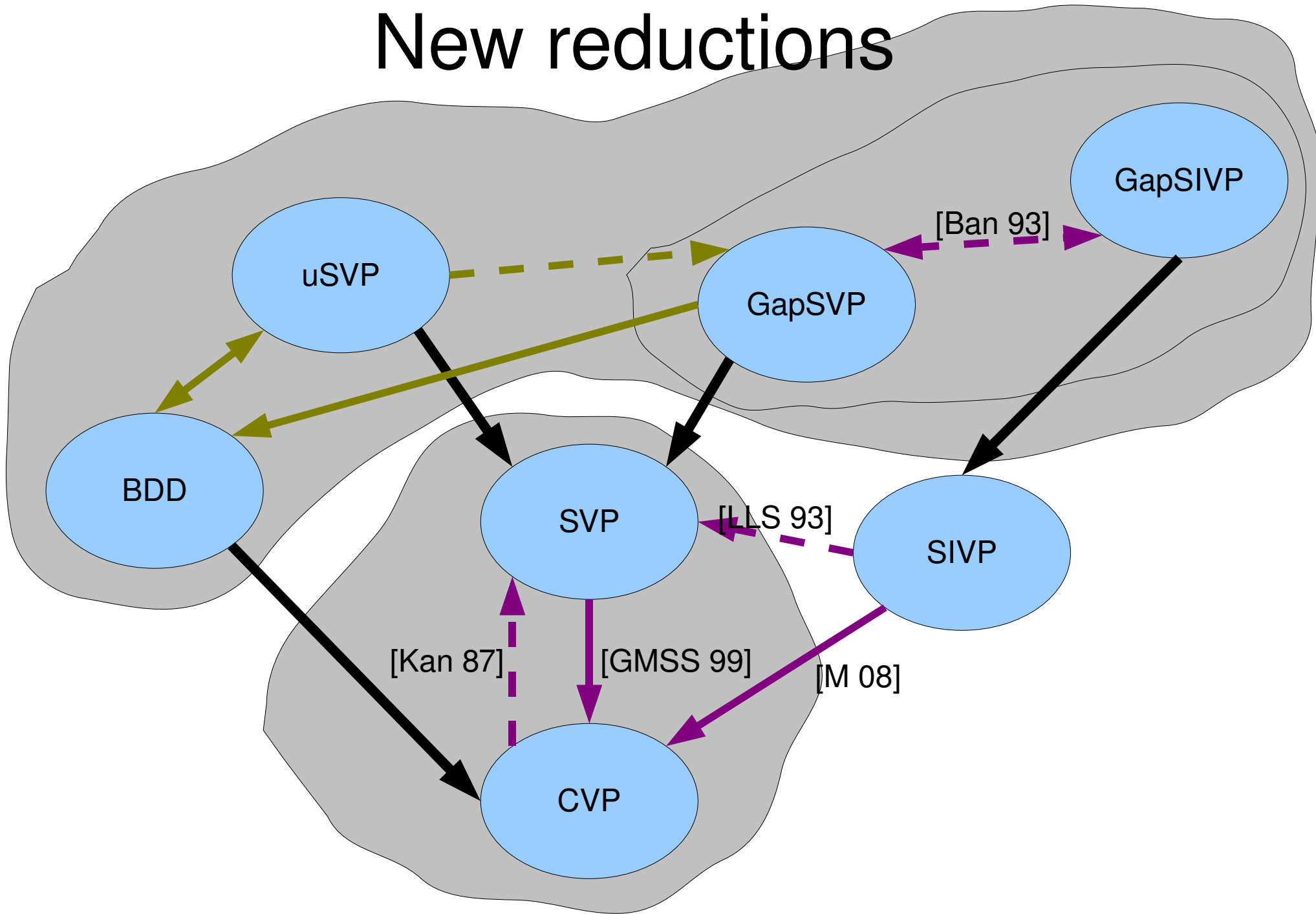
# Polynomial reductions



# New reductions



# New reductions





# Summary

- GapSVP, BDD,  $\text{uSVP}$ 
  - Give Public Key Encryption
  - Qualitatively equivalent under classic reductions
- SIVP
  - Gives Hash functions and Digital Signatures
  - Classic: Harder than GapSVP, BDD,  $\text{uSVP}$
  - Quantum: Equivalent to GapSVP, BDD,  $\text{uSVP}$
- SVP, CVP: hardest problems, no crypto yet

# Open problems

- Prove equivalence under approximation preserving reductions
- Prove equivalence of SIVP and BDD under classic reductions
- Base cryptography on hardness of SVP, CVP
- Search to Decision reduction for SVP:
  - Give reduction from SVP to GapSVP!
  - It would show all lattice problems are equivalent

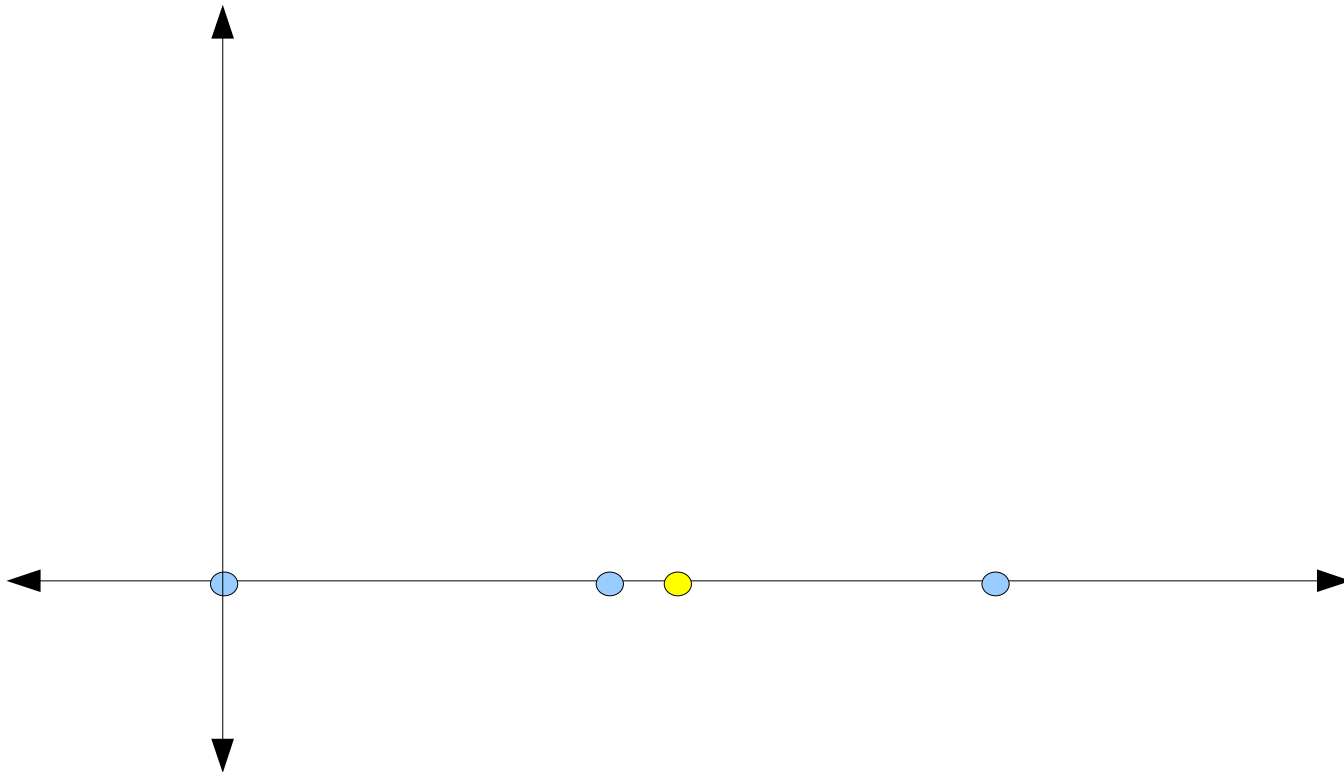
# More Open Problems

- Relating Lattices to Factoring, ECC, etc.
  - Is there a reduction from factoring (random) integers to approximate (Gap)SVP within factor  $n^c$ ?
- Previous work:
  - [Adleman], [Schnorr]: use exact SVP and heuristics
  - Inspired NP hardness proofs for SVP in [Ajtai], [M]
- Implications:
  - Evidence that approximate SVP is hard
  - Lattice based crypto is as secure as factoring

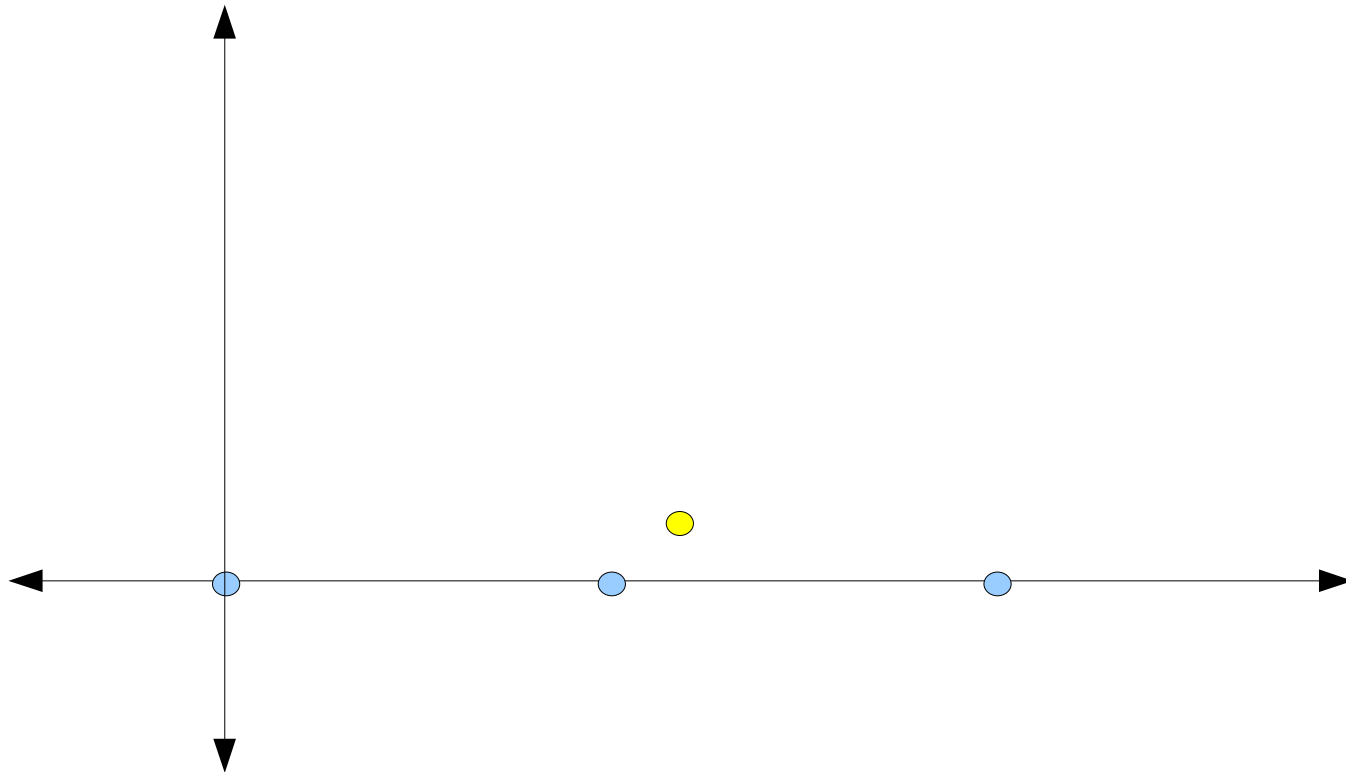
# Rest of this talk

- Reduction from GapSVP to BDD
- Reduction from BDD to uSVP
- Reduction from uSVP to GapSVP
- Notes:
  - Only sketch the main ideas
  - Can reduce uSVP directly to BDD, rather than using  $uSVP < GapSVP < BDD$ .

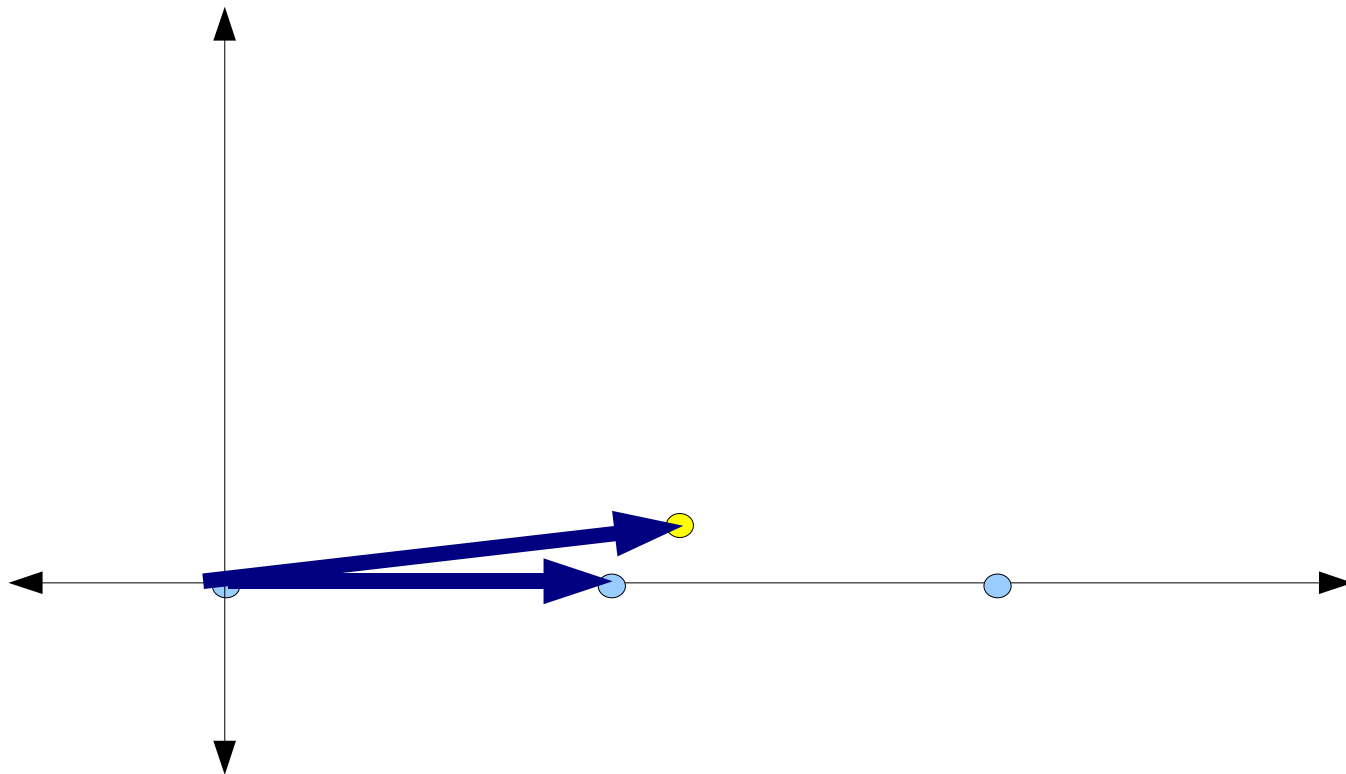
# Proof Sketch (BDD < uSVP)



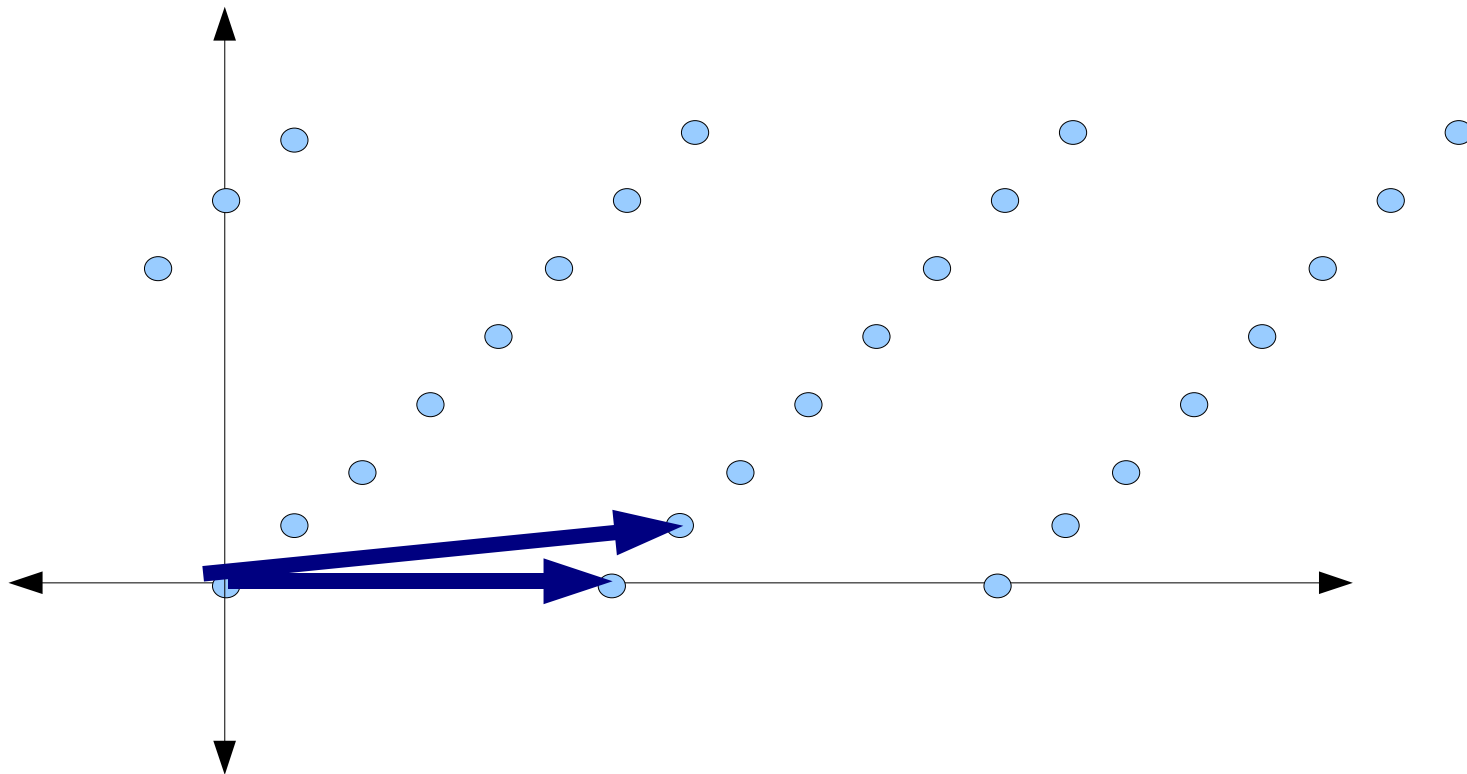
# Proof Sketch (BDD < uSVP)



# Proof Sketch (BDD < uSVP)

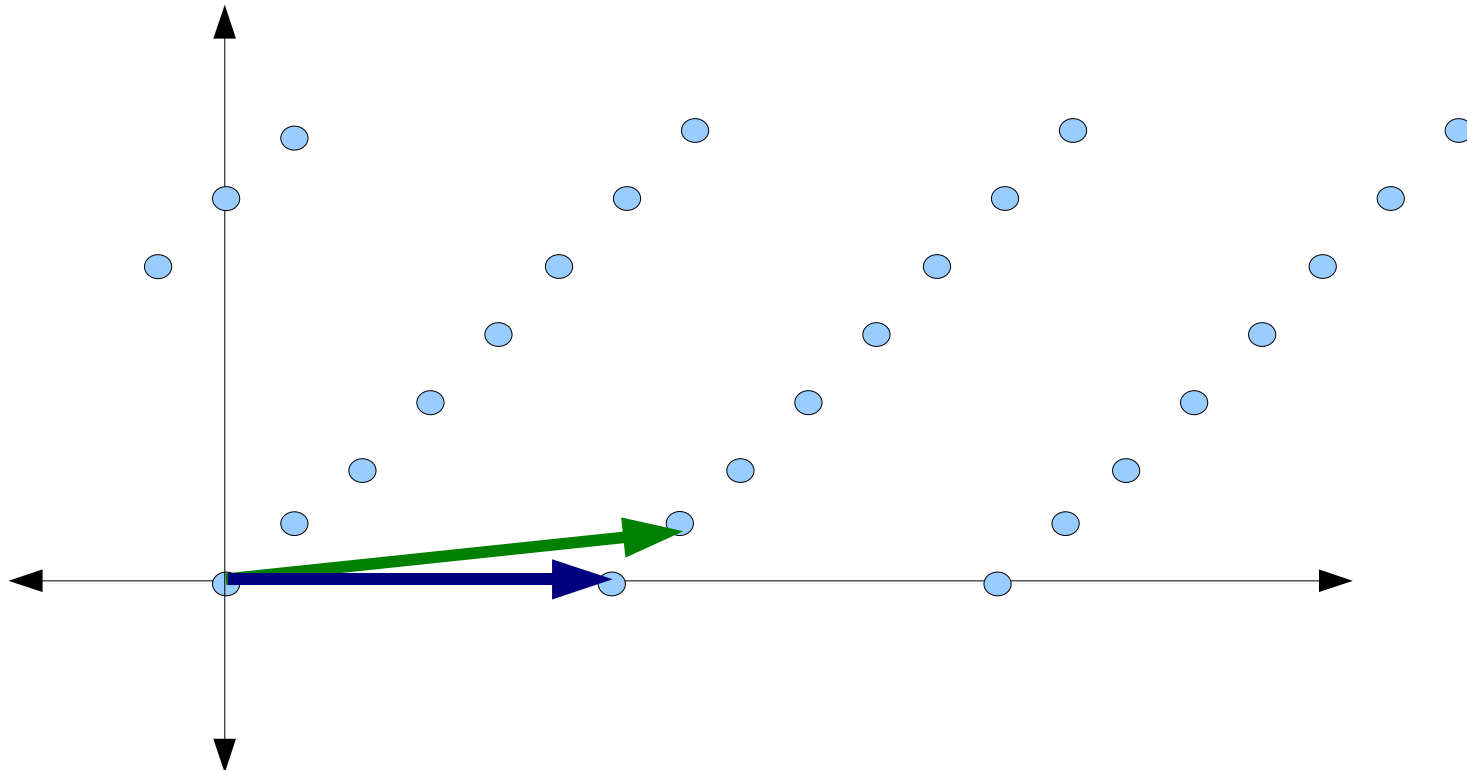


# Proof Sketch (BDD < uSVP)



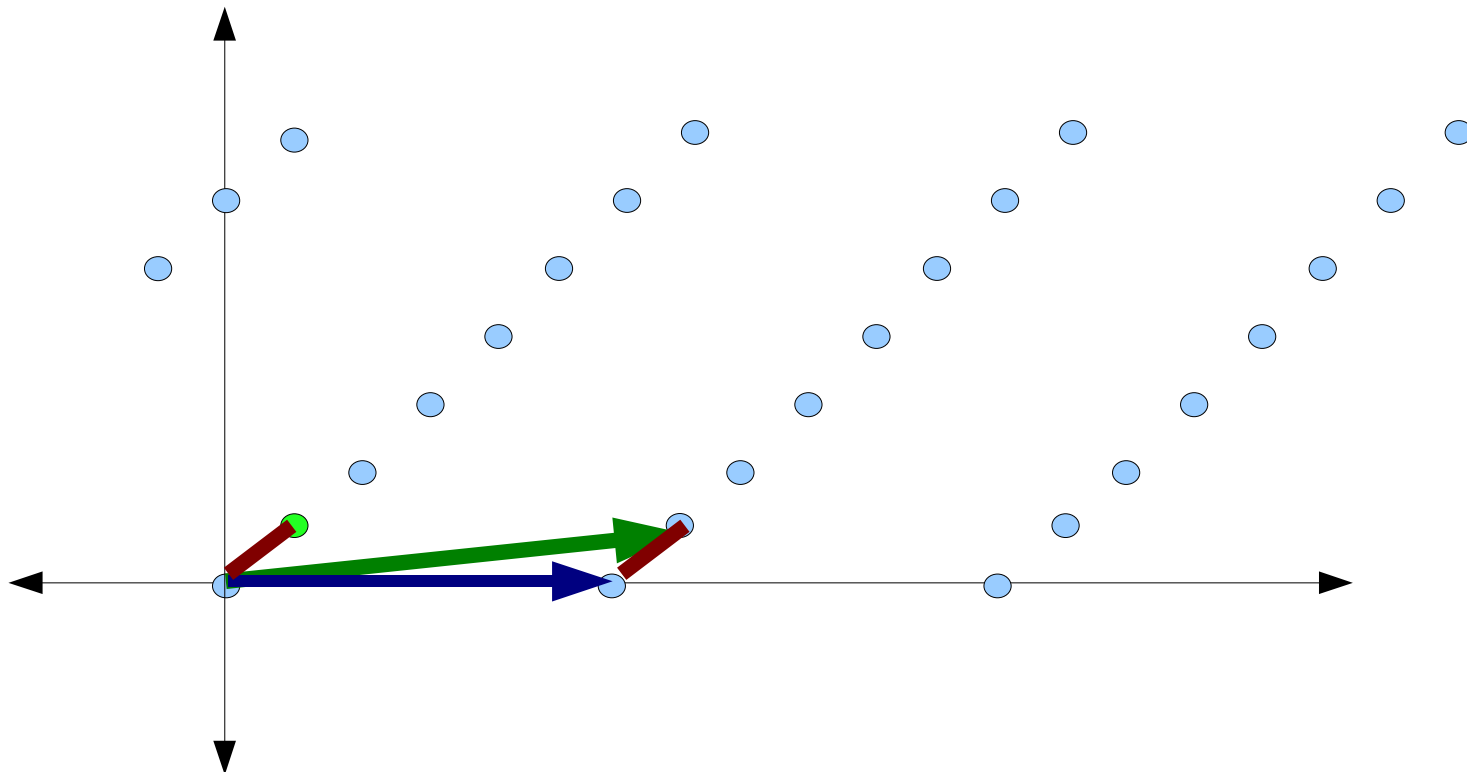
# Proof Sketch (BDD < uSVP)

New basis vector used exactly once in constructing the unique shortest vector



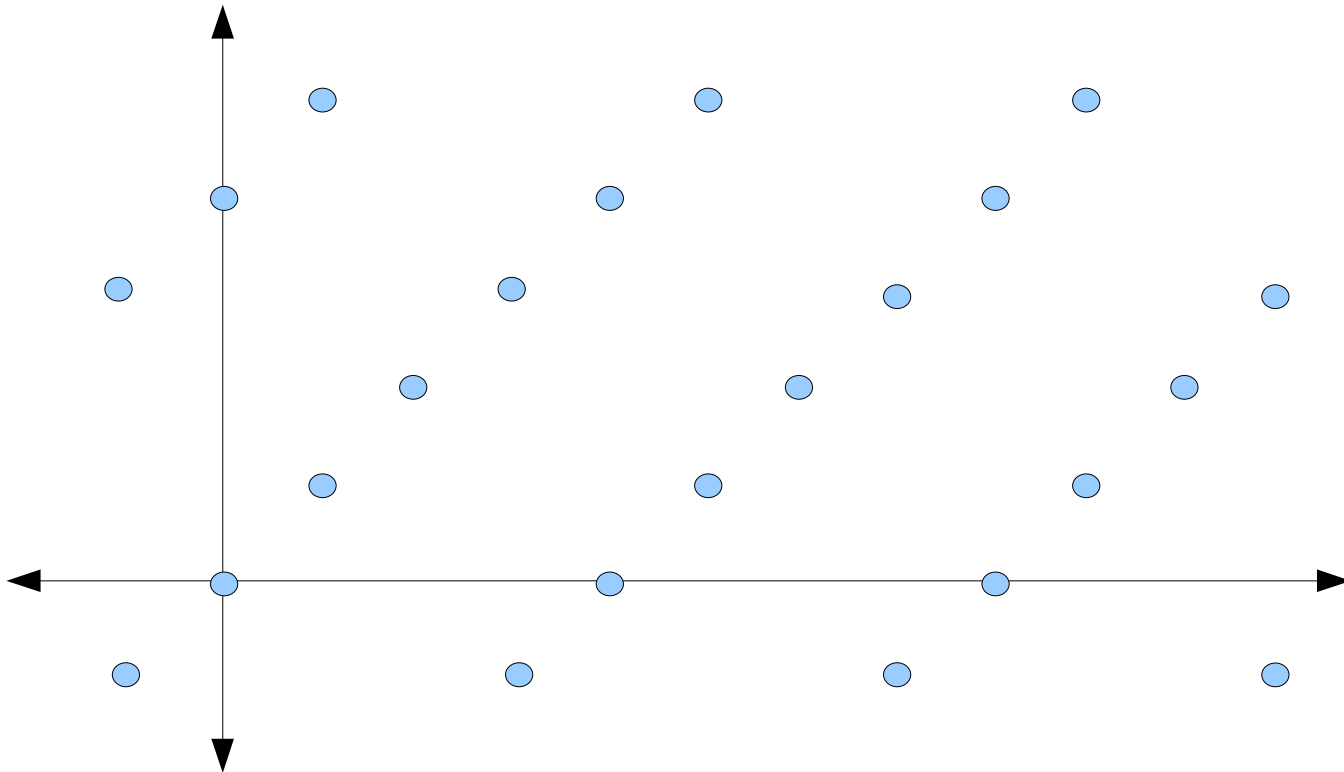
# Proof Sketch (BDD $<$ uSVP)

- Find **unique shortest vector**
- Subtract it from **new basis vector**



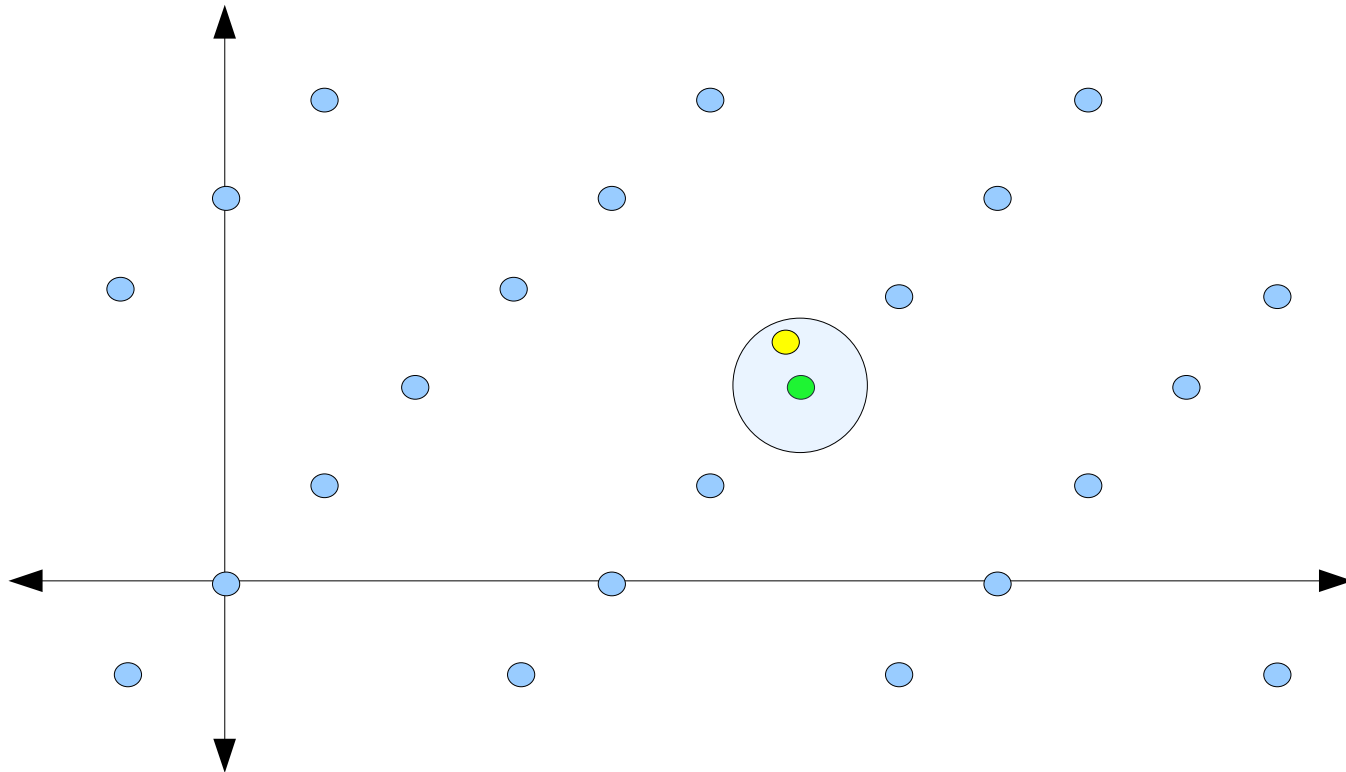
# Proof Sketch (GapSVP $\leq$ BDD)

- Assume you can solve BDD
- Question: Is  $\lambda(B) > d$ ?



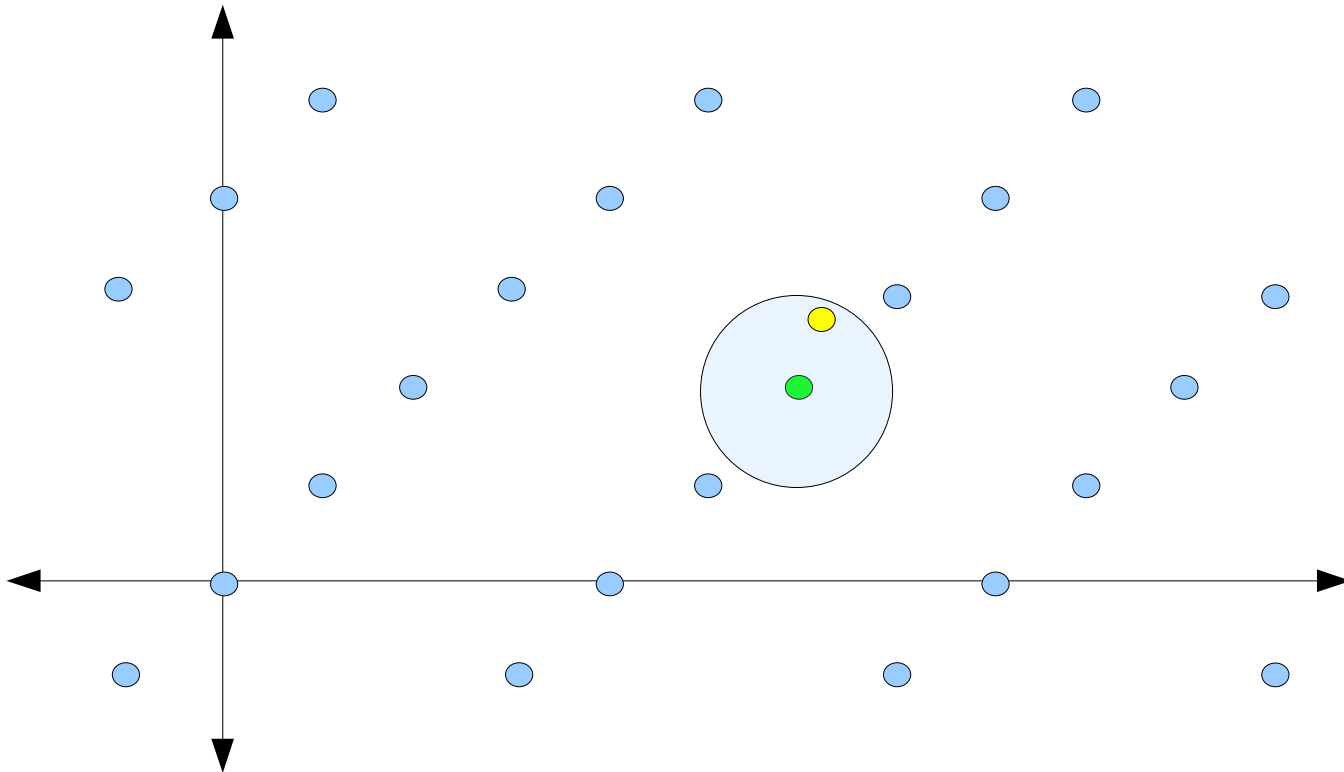
# Proof Sketch (GapSVP $\leq$ BDD)

- Pick a **lattice point**, and perturb by  $(d/2)$
- Use BDD to find **lattice point** close to **target**



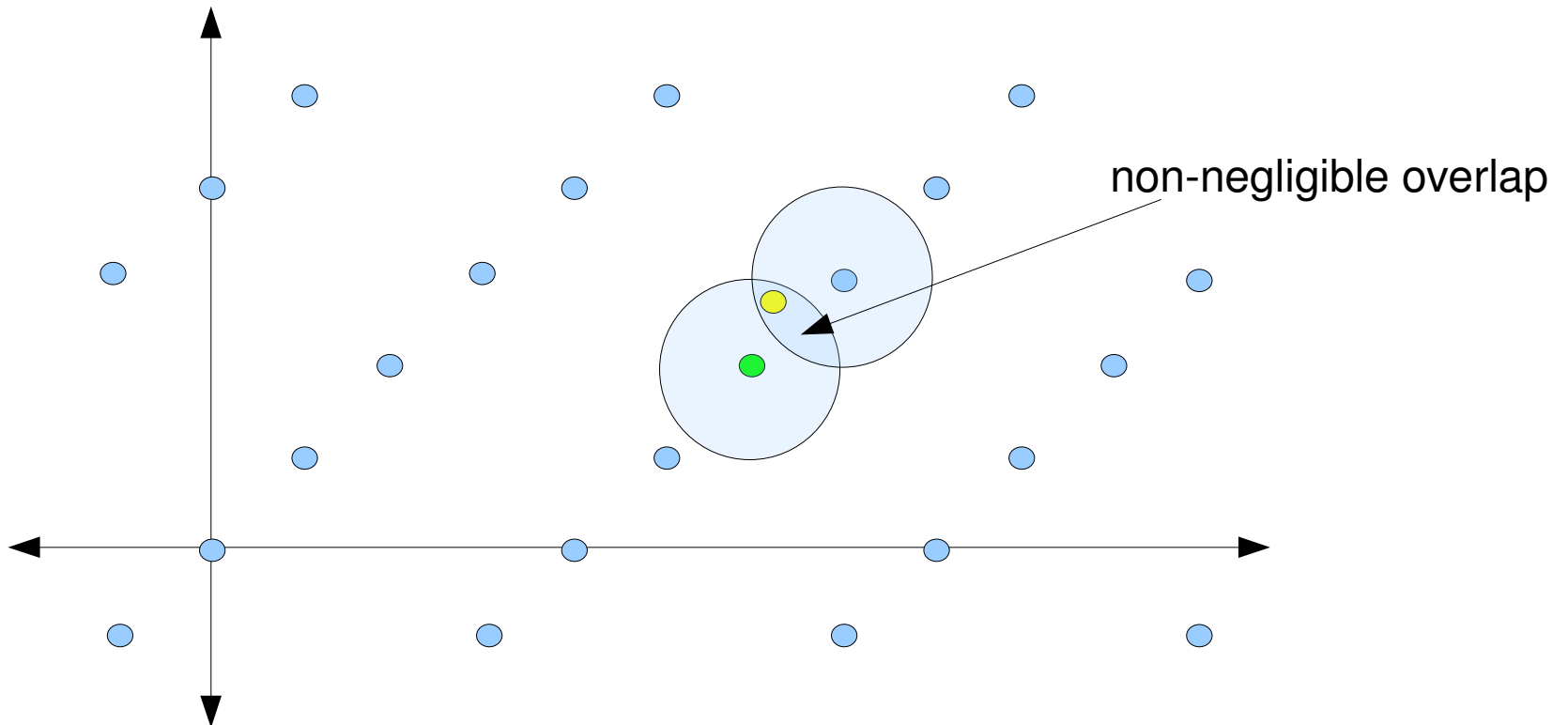
# Proof Sketch (GapSVP $<$ BDD)

- If  $\lambda < d$ , we don't get a valid BDD instance
- ... but BDD may still find nearby lattice point



# Proof Sketch (GapSVP<sub><</sub>BDD)

- If  $\lambda < d/n^{1/2}$ , there are good chances that BDD either fail or return wrong lattice point



# Conclusion

- Reductions among lattice problems are simple
  - Math is elementary
  - Ideas are not new [GG], [C], [GMSS], [R], [P]
- Reducing SVP to GapSVP
  - It looks hard to me ...
  - ... but not any harder than currently known reductions, before their discovery
- Reducing Factoring/DLOG to Approx. Lattices
  - Same story: I cannot solve it, but may you can!

# Paid Advertisements

- Pointers to Literature:
  - “Approximating shortest lattice vectors is not harder than approximating closest lattice vectors”, Goldreich, M., Safra & Seifert, Inf.Proc.Lett. (1999)
  - “Efficient reductions among lattice problems”, M. SODA (2008)
  - “On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem”, Lyubashevsky & M., CRYPTO (2009)
  - “Lattice based cryptography”, M. & Regev, in Post Quantum Cryptography (2009)
- Theory of Cryptography Conference, TCC 2010:
  - Zurich, Feb 9-11, 2010, Submission Deadline: Sep. 2, 2009.