

# Dual lattice attacks for closest vector problems (with preprocessing)

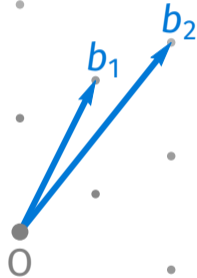
Thijs Laarhoven, Michael Walter

mail@thijs.com  
<https://www.thijs.com/>

Simons Reunion, virtual  
(June 15, 2021)

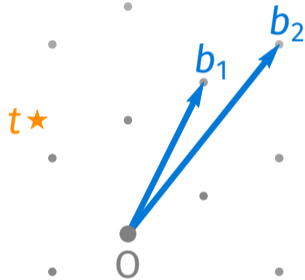
## Lattices

## Basics



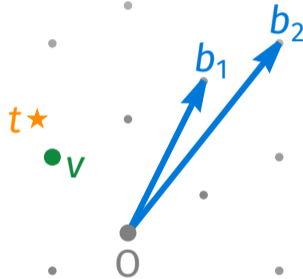
# Lattices

## Closest Vector Problem (CVP)



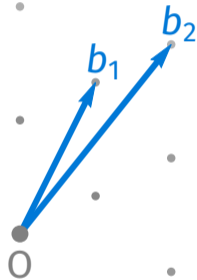
# Lattices

## Closest Vector Problem (CVP)



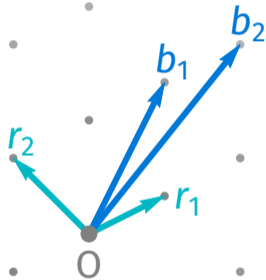
# Lattices

CVP with Preprocessing (CVPP)



# Lattices

CVP with Preprocessing (CVPP)



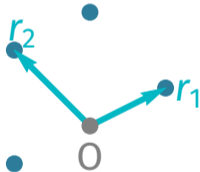
# Lattices

CVP with Preprocessing (CVPP)



# Lattices

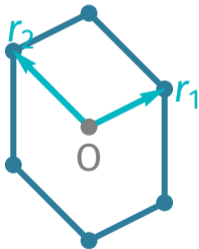
CVP with Preprocessing (CVPP)





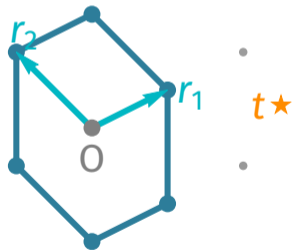
# Lattices

CVP with Preprocessing (CVPP)



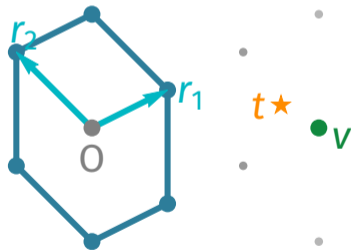
# Lattices

CVP with Preprocessing (CVPP)



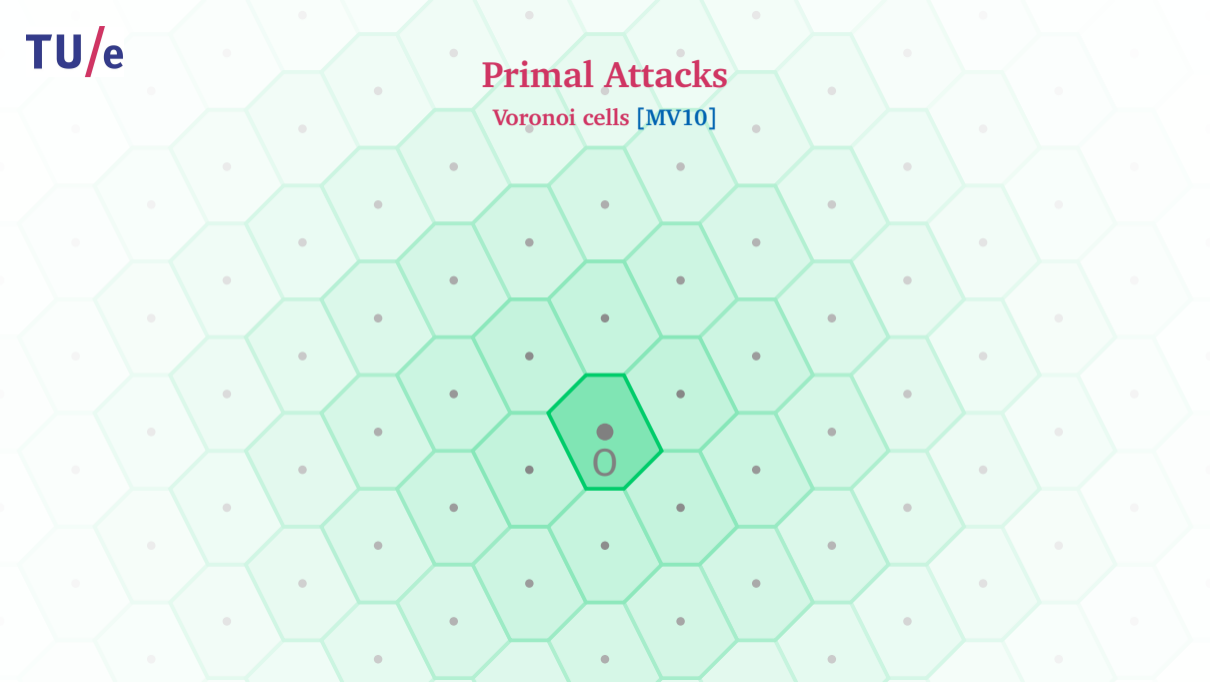
# Lattices

CVP with Preprocessing (CVPP)



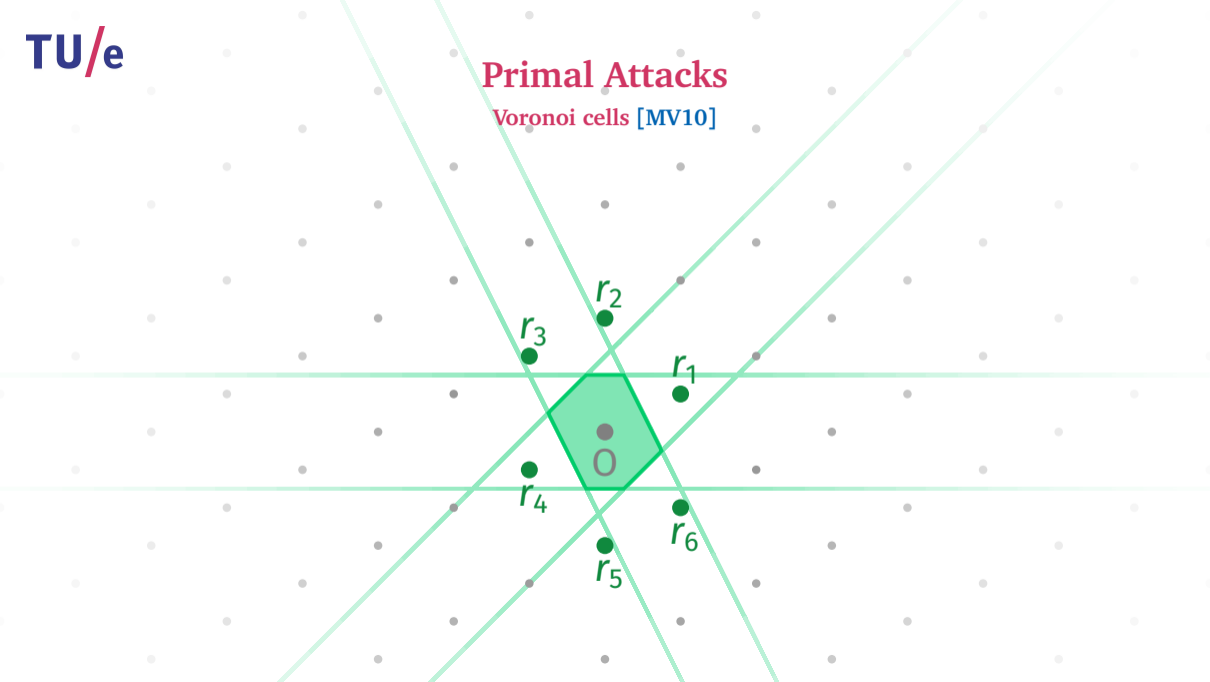
# Primal Attacks

Voronoi cells [MV10]



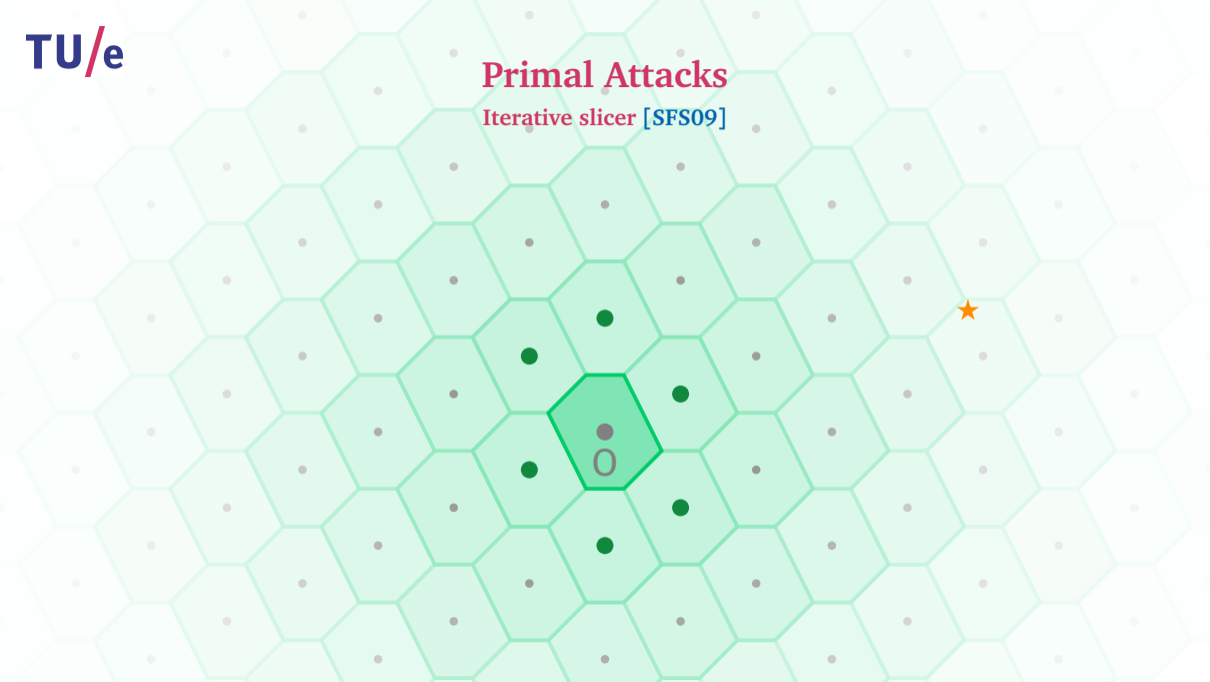
# Primal Attacks

Voronoi cells [MV10]



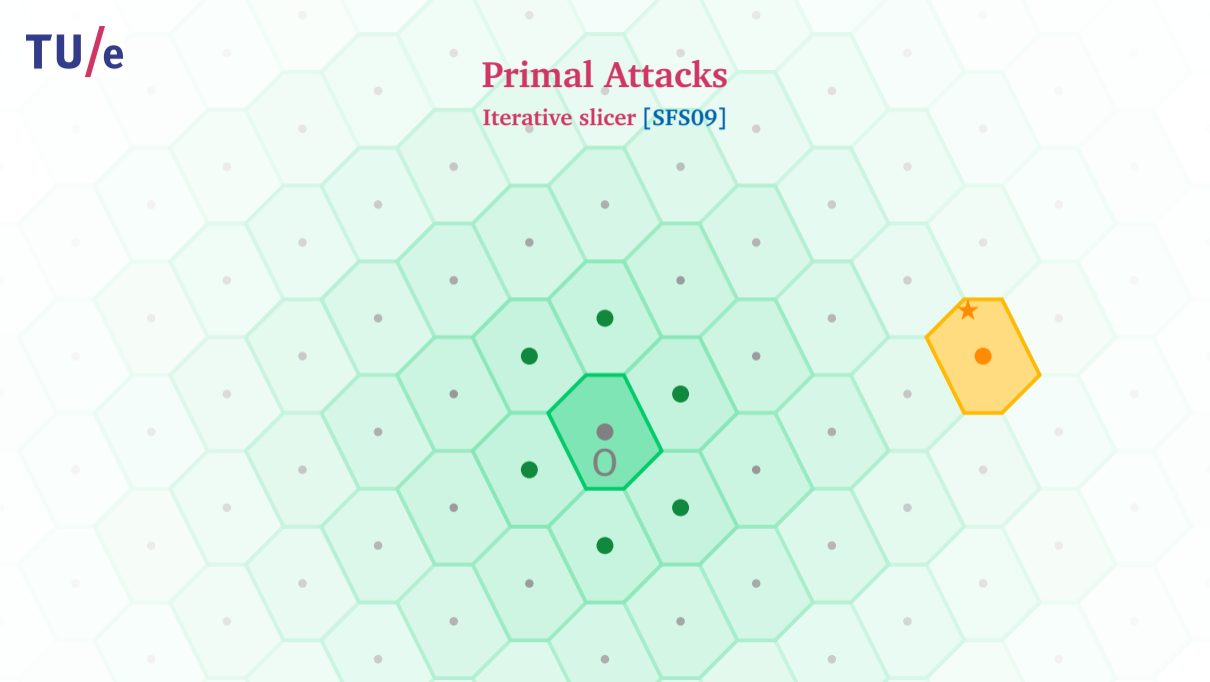
# Primal Attacks

Iterative slicer [SFS09]



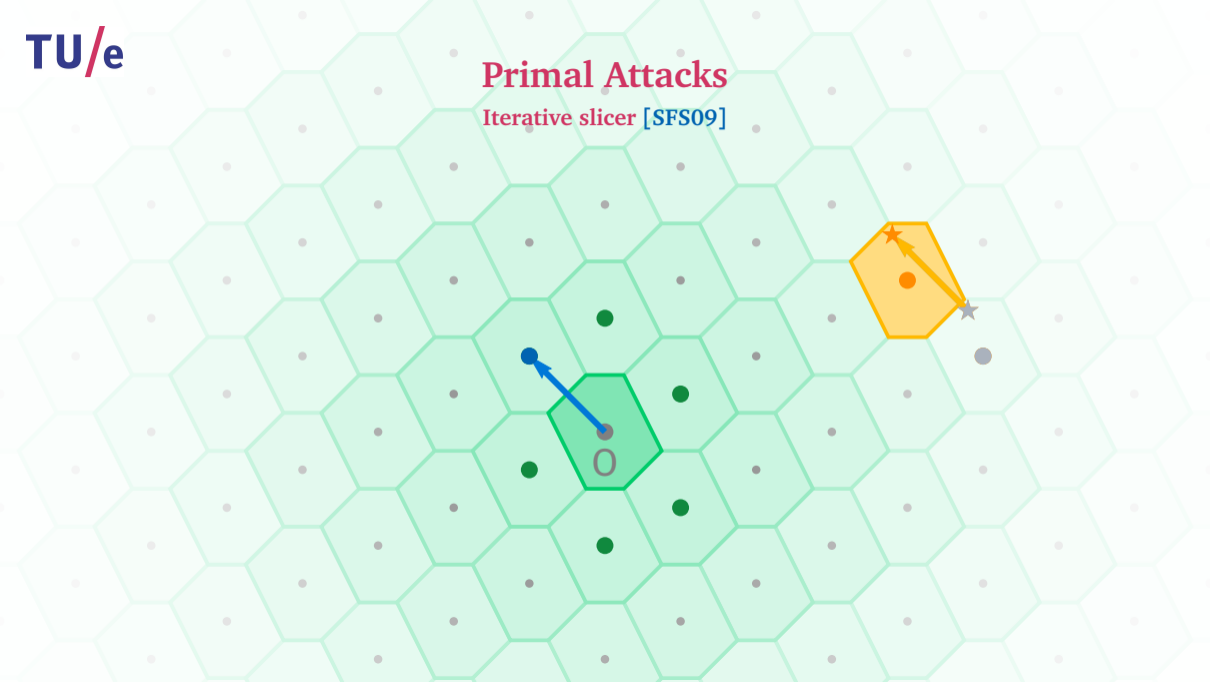
# Primal Attacks

Iterative slicer [SFS09]



# Primal Attacks

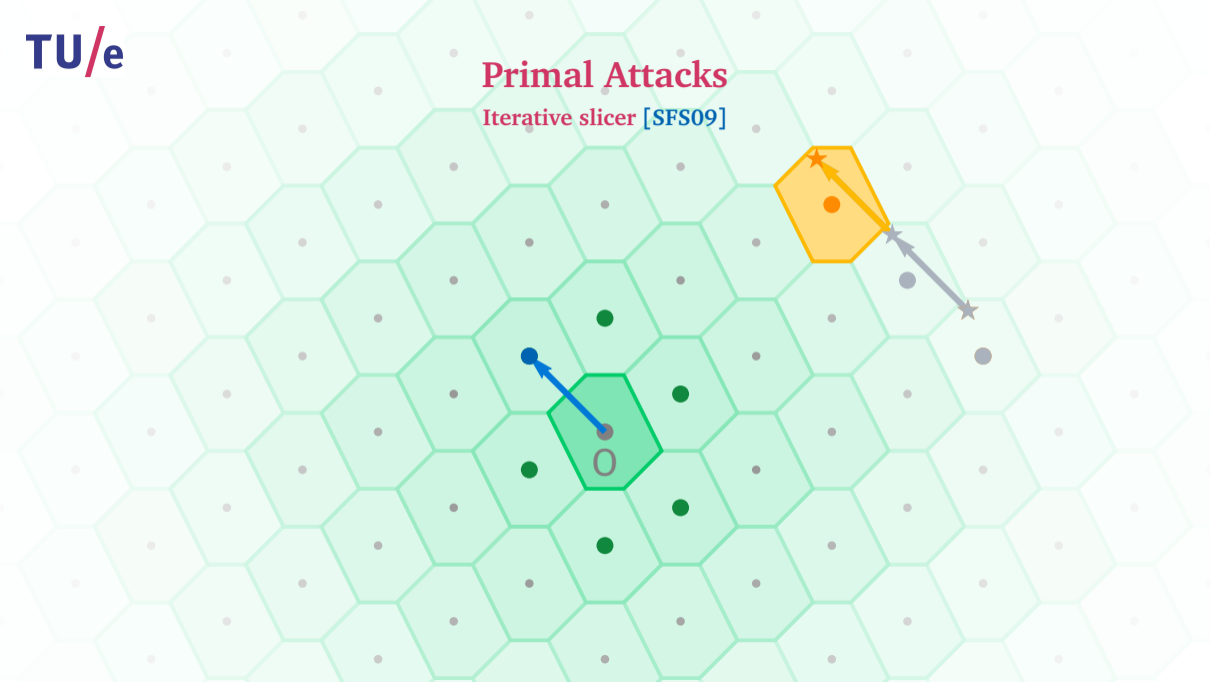
Iterative slicer [SFS09]





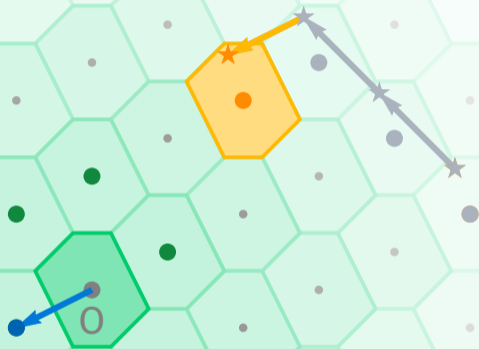
# Primal Attacks

Iterative slicer [SFS09]



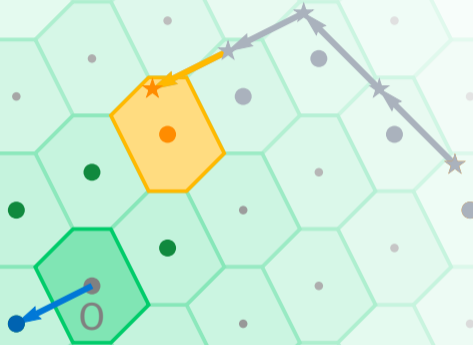
# Primal Attacks

Iterative slicer [SFS09]



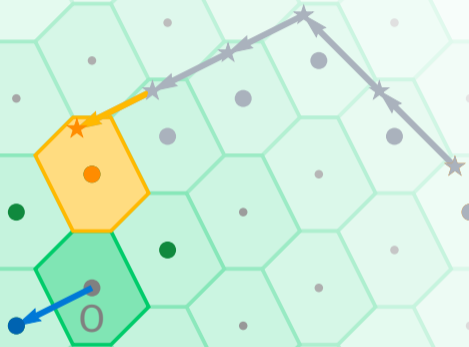
# Primal Attacks

Iterative slicer [SFS09]



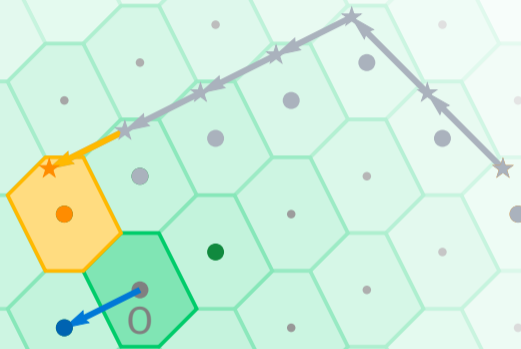
# Primal Attacks

Iterative slicer [SFS09]



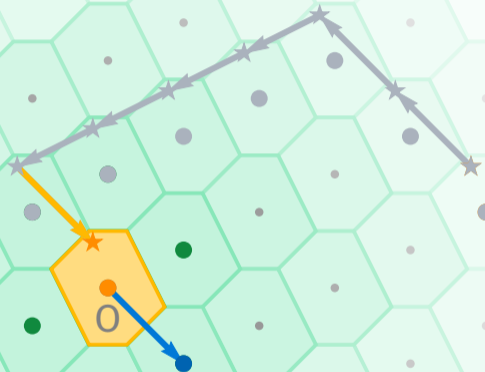
# Primal Attacks

Iterative slicer [SFS09]



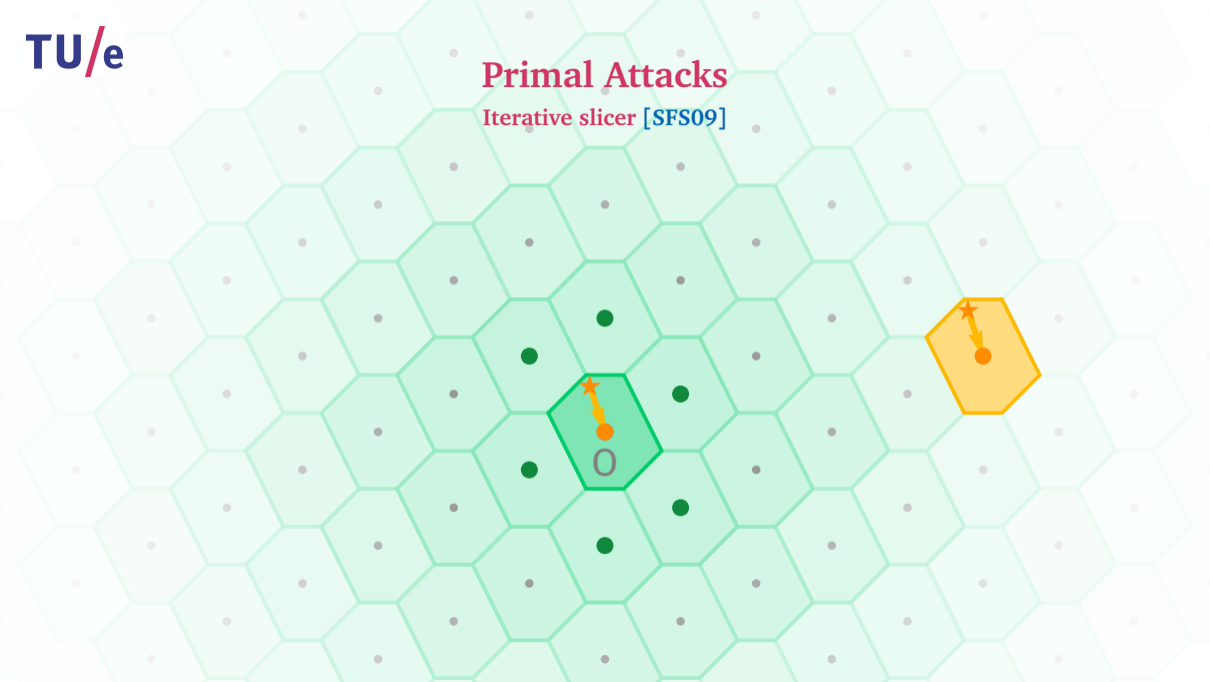
# Primal Attacks

Iterative slicer [SFS09]



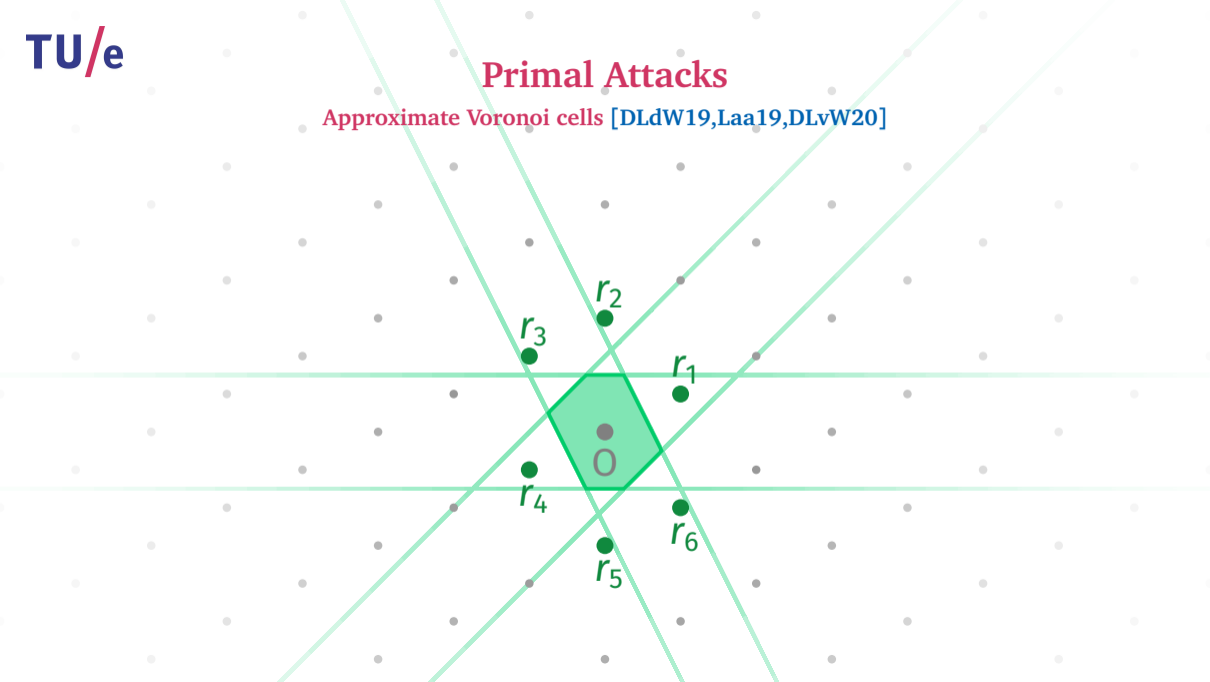
# Primal Attacks

Iterative slicer [SFS09]



# Primal Attacks

Approximate Voronoi cells [DLdW19,Laa19,DLvW20]





## Primal Attacks

Approximate Voronoi cells [DLdW19,Laa19,DLvW20]



$V_2$

$V_1$

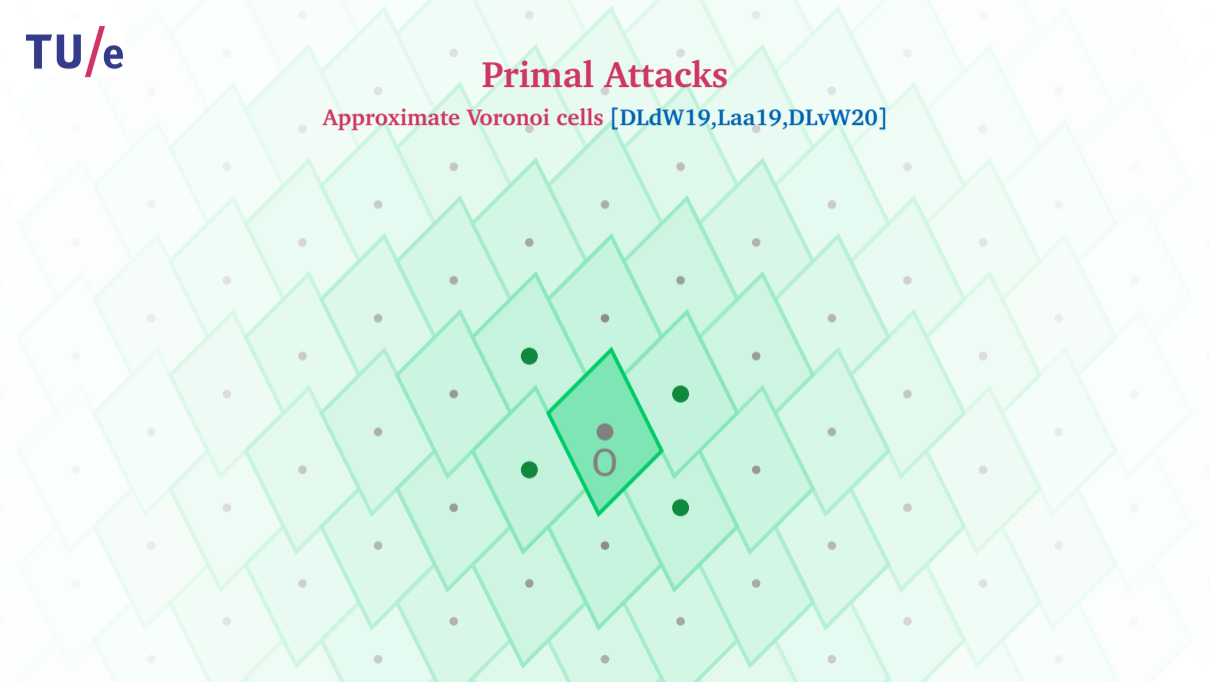
0

$V_3$

$V_4$

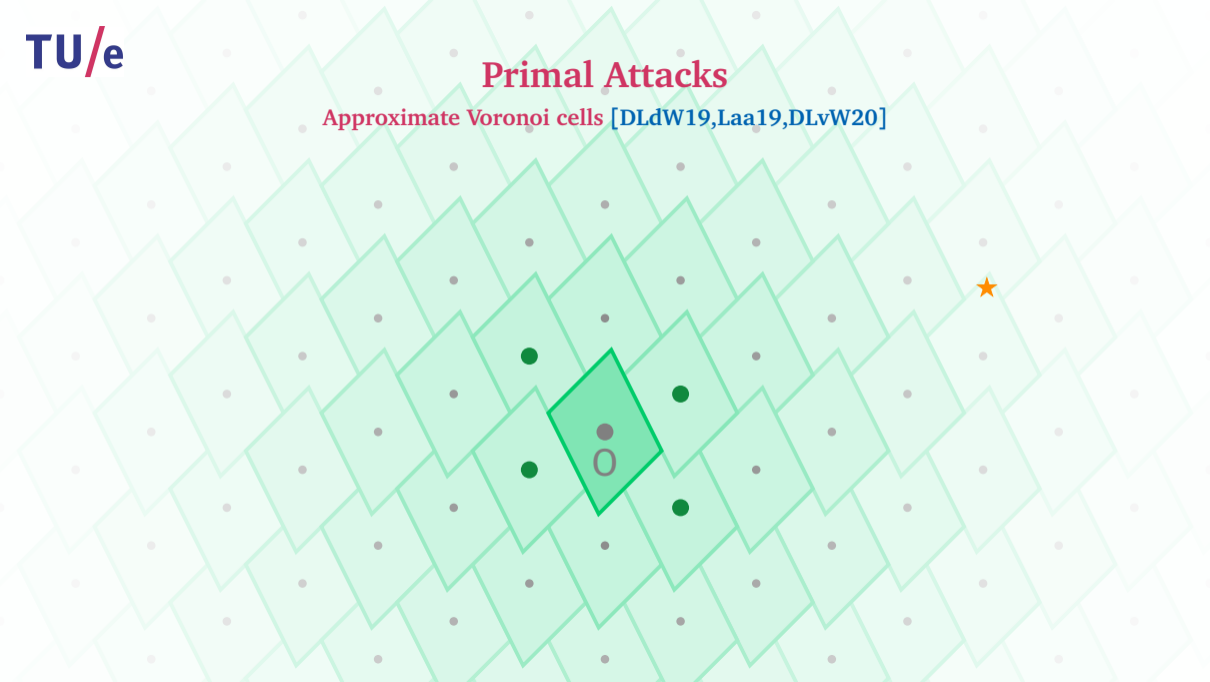
# Primal Attacks

Approximate Voronoi cells [DLdW19,Laa19,DLvW20]



# Primal Attacks

Approximate Voronoi cells [DLdW19,Laa19,DLvW20]



# Primal Attacks

Approximate Voronoi cells [DLdW19,Laa19,DLvW20]



# Primal Attacks

Approximate Voronoi cells [DLdW19,Laa19,DLvW20]



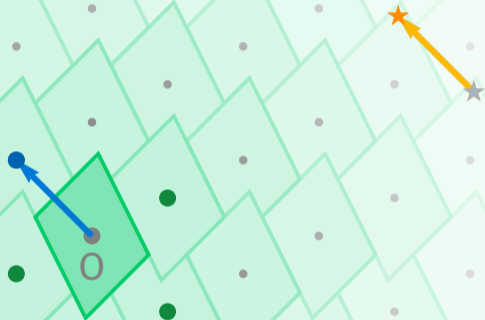
# Primal Attacks

Randomized slicer [DLdW19]



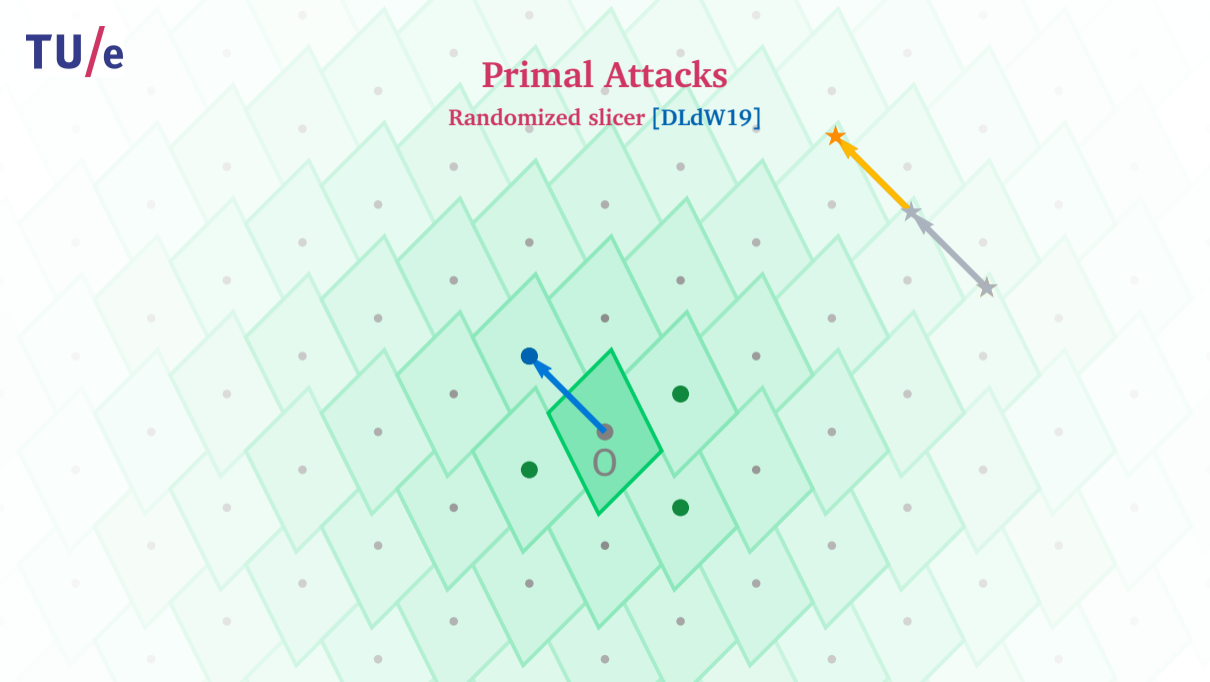
# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

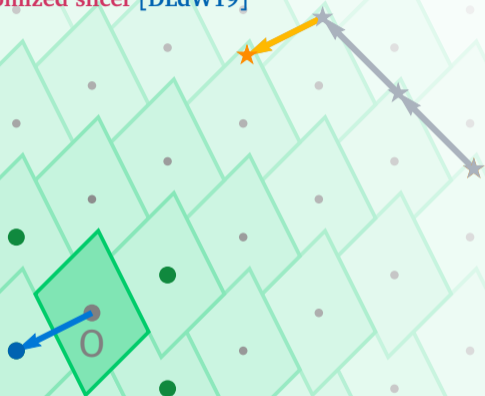
Randomized slicer [DLdW19]





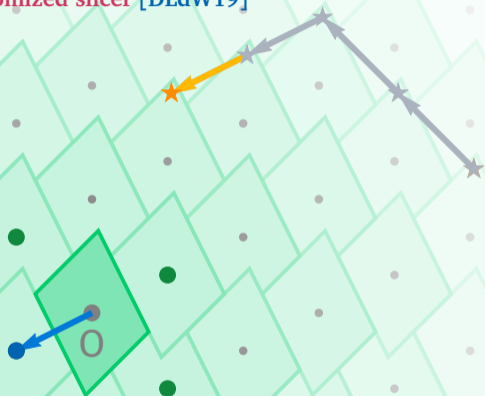
# Primal Attacks

Randomized slicer [DLdW19]



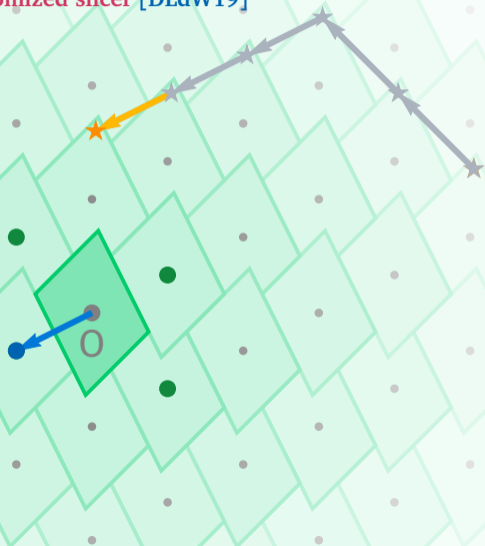
# Primal Attacks

Randomized slicer [DLdW19]



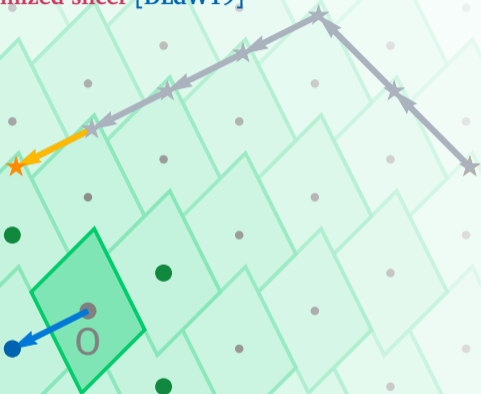
# Primal Attacks

Randomized slicer [DLdW19]



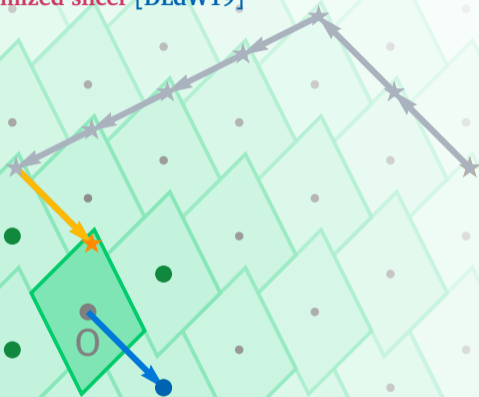
# Primal Attacks

Randomized slicer [DLdW19]



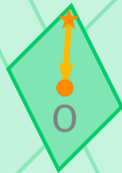
# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

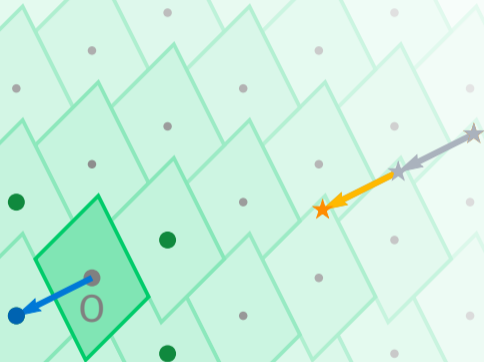
Randomized slicer [DLdW19]





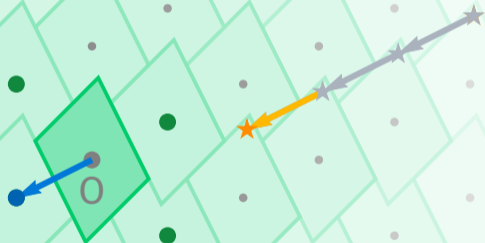
# Primal Attacks

Randomized slicer [DLdW19]



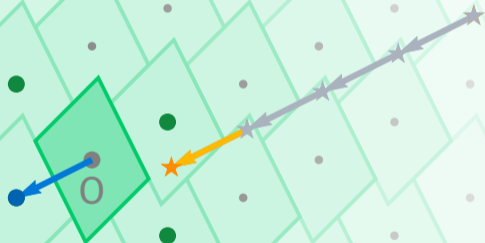
# Primal Attacks

Randomized slicer [DLdW19]



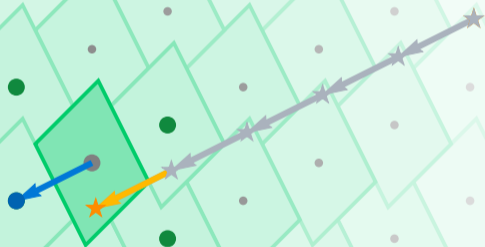
# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

Randomized slicer [DLdW19]



# Primal Attacks

## Overview

- *Preprocessing*: Find short primal lattice vectors ( $2^{O(d)}$  time, space)



# Primal Attacks

## Overview

- *Preprocessing*: Find short primal lattice vectors ( $2^{O(d)}$  time, space)
- *Querying*: Reduce to shortest representative in coset  $\mathbf{t} + \mathcal{L}$  ( $2^{O(d)}$  time, space)



# Primal Attacks

## Overview

- *Preprocessing*: Find short primal lattice vectors ( $2^{O(d)}$  time, space)
- *Querying*: Reduce to shortest representative in coset  $\mathbf{t} + \mathcal{L}$  ( $2^{O(d)}$  time, space)
- *Strengths*: Works well for approximate CVPP





# Primal Attacks

## Overview

- *Preprocessing*: Find short primal lattice vectors ( $2^{O(d)}$  time, space)
- *Querying*: Reduce to shortest representative in coset  $\mathbf{t} + \mathcal{L}$  ( $2^{O(d)}$  time, space)
- *Strengths*: Works well for approximate CVPP
- *Limitations*: Does not scale well for BDDP



## Dual Attacks

$$\text{Dual: } \mathcal{L}^* = \{w \in \mathbb{R}^d : \langle v, w \rangle \in \mathbb{Z}, \forall v \in \mathcal{L}\}$$


# Dual Attacks

Dual:  $\mathcal{L}^* = \{w \in \mathbb{R}^d : \langle v, w \rangle \in \mathbb{Z}, \forall v \in \mathcal{L}\}$

A 2D lattice of points is shown, with the origin (0,0) highlighted by a larger black dot and the number '0' below it. The lattice consists of a grid of points, with some points colored in shades of pink and purple, and others in grey. The points are arranged in a regular grid pattern, representing a lattice structure in a 2D space.

0

# Dual Attacks

Primal:  $\mathcal{L} = \{v \in \mathbb{R}^d : \langle w, v \rangle \in \mathbb{Z}, \forall w \in \mathcal{L}^*\}$

A 2D lattice of points is shown, with the origin (0,0) highlighted by a larger black dot and labeled with the number 0. The lattice consists of a grid of smaller dots, with some dots colored in light blue and others in light red. The origin is the central point where the grid lines intersect.

0

## Dual Attacks

Primal:  $\mathcal{L} = \{v \in \mathbb{R}^d : \langle w, v \rangle \in \mathbb{Z}, \forall w \in \mathcal{L}^*\}$



## Dual Attacks

Primal:  $\mathcal{L} = \{v \in \mathbb{R}^d : \langle w, v \rangle \in \mathbb{Z}, \forall w \in \mathcal{L}^*\}$

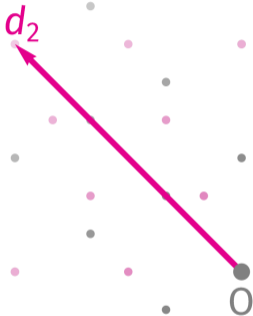


$d_1$

0

## Dual Attacks

Primal:  $\mathcal{L} = \{v \in \mathbb{R}^d : \langle w, v \rangle \in \mathbb{Z}, \forall w \in \mathcal{L}^*\}$



## Dual Attacks

Primal:  $\mathcal{L} = \{v \in \mathbb{R}^d : \langle w, v \rangle \in \mathbb{Z}, \forall w \in \mathcal{L}^*\}$

$d_2$

0

A 2D lattice of points is shown, with a red arrow pointing from the origin (labeled 0) to a point labeled  $d_2$ . The lattice points are arranged in a grid pattern, and the red arrow is the longest vector starting from the origin and ending at a lattice point.



## Dual Attacks

Distinguisher:  $f(t) = \sum_{w \in \mathcal{L}^*} \cos(2\pi \langle w, t \rangle)$

A grid of points is shown on a white background. The points are arranged in a regular grid. The central point is highlighted with a larger, darker grey circle. Below this central point is the letter '0'.

0

## Dual Attacks

Approximate distinguisher:  $\hat{f}(t) = \sum_{w \in L} \cos(2\pi \langle w, t \rangle)$



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$

$t$   
☆



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



$t$   
★

## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$

$t'$   
★★



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



## Dual Attacks

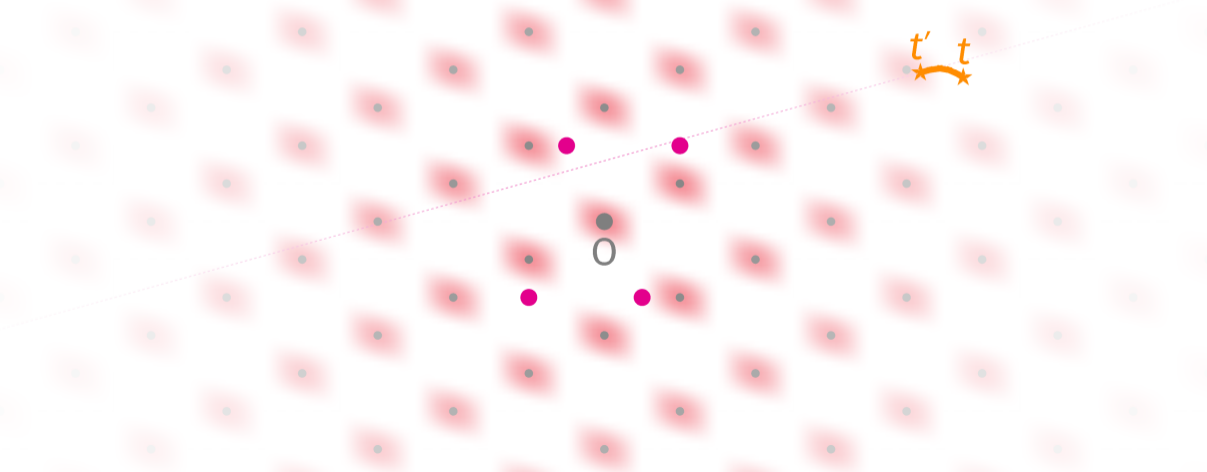
Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$





## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



0

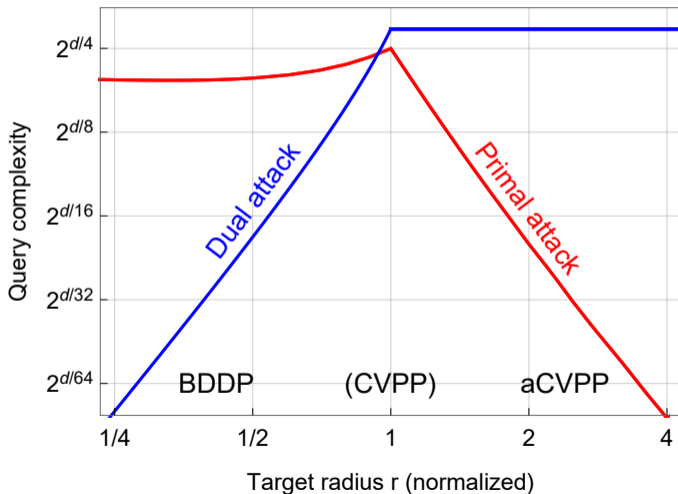
## Dual Attacks

Gradient ascent:  $t' = t - \alpha \cdot \sum_{w \in L} \sin(2\pi \langle w, t \rangle) \cdot w$



# Dual Attacks

Asymptotics (with preprocessing)



## Dual Attacks

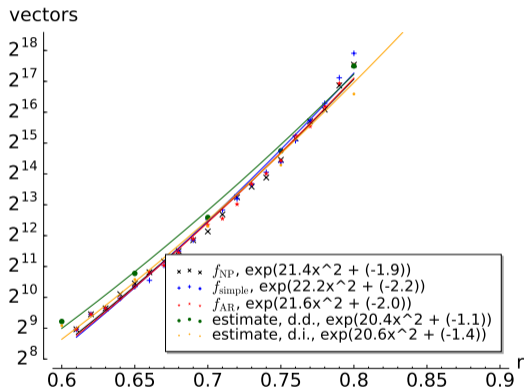
Experiments ( $d = 80$ )

Figure: Complexity of distinguishing from random at radius  $r$  ( $p = 0.90$ ).

## Dual Attacks

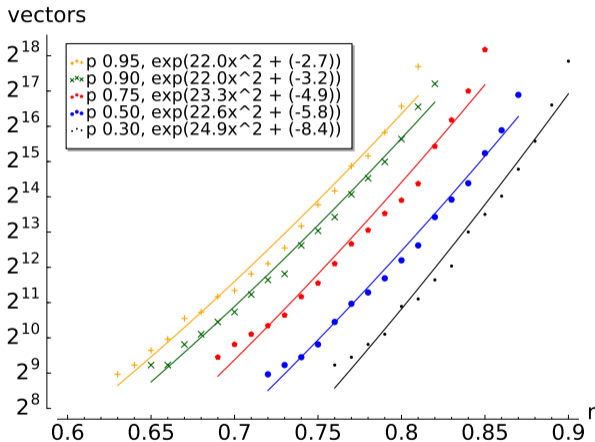
Experiments ( $d = 80$ )

Figure: Complexity of decoding a target at distance  $r$  with probability  $p$  in dimension 80.



## Dual Attacks

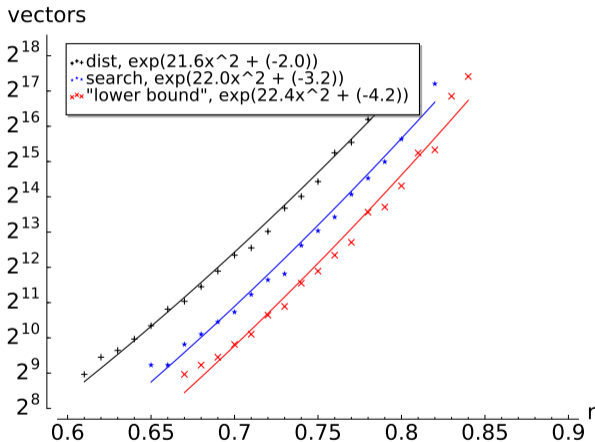
Experiments ( $d = 80$ )

Figure: Experimental complexities for distinguishing/searching and a heuristic lower bound.

## Dual Attacks

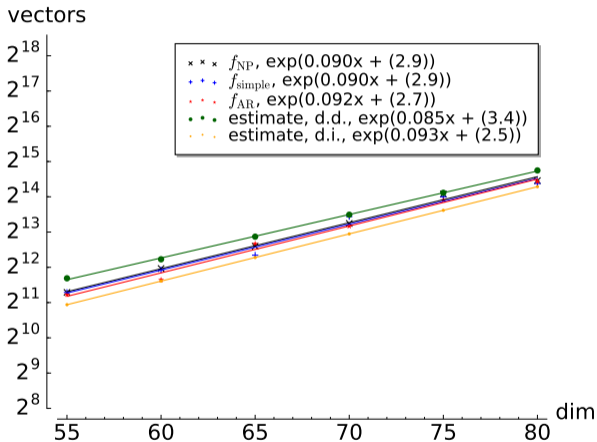
Experiments (variable  $d$ )

Figure: Complexity of distinguishing a planted target at radius 0.75 from random ( $p = 0.9$ ).

## Dual Attacks

Experiments ( $d = 80$ )

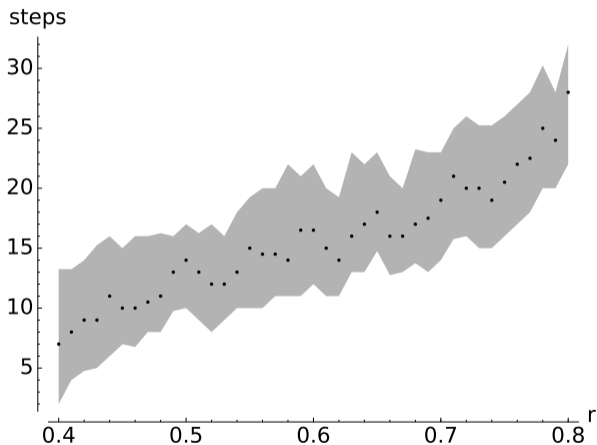


Figure: Steps required to decode target at radius  $r$  using  $2^{14}$  vectors ( $p = 0.9$ ).

# Dual Attacks

## Overview

- *Preprocessing*: Find short dual vectors ( $2^{O(d)}$  time, space)



# Dual Attacks

## Overview

- *Preprocessing*: Find short dual vectors ( $2^{O(d)}$  time, space)
- *Querying*: Gradient ascent using dot products modulo 1 ( $2^{O(d)}$  time, space)



0

# Dual Attacks

## Overview

- *Preprocessing*: Find short dual vectors ( $2^{O(d)}$  time, space)
- *Querying*: Gradient ascent using dot products modulo 1 ( $2^{O(d)}$  time, space)
- *Strengths*: Works well for BDDP, predictable



# Dual Attacks

## Overview

- *Preprocessing*: Find short dual vectors ( $2^{O(d)}$  time, space)
- *Querying*: Gradient ascent using dot products modulo 1 ( $2^{O(d)}$  time, space)
- *Strengths*: Works well for BDDP, predictable
- *Limitations*: Does not scale well for approximate CVPP



# Conclusion

## Summary

### Primal Attacks

- Using list of short primal lattice vectors
- Works well for approximate CVP(P), not for BDD(P)



## Conclusion

### Summary

#### Primal Attacks

- Using list of short primal lattice vectors
- Works well for approximate CVP(P), not for BDD(P)

#### Dual Attacks

- Using list of short dual lattice vectors
- Works well for BDD(P), not for approximate CVP(P)
- *Contribution*: Complete heuristic average-case analysis
- *Contribution*: Experiments, closely matching heuristic predictions

## Conclusion

### Open problems

#### Combining both approaches?

- Short primal vectors  $\rightarrow$  Efficient approximate CVPP algorithm
- Short dual vectors  $\rightarrow$  Efficient BDDP algorithm
- Short primal *and* dual vectors  $\rightarrow$  ???

## Conclusion

### Open problems

#### Combining both approaches?

- Short primal vectors → Efficient approximate CVPP algorithm
- Short dual vectors → Efficient BDDP algorithm
- Short primal *and* dual vectors → ???

#### Applications?

- Dual attack: Faster algorithm for huge BDD batches
- Sieving-enumeration hybrid [DLdW20]: not so promising
- Other applications?