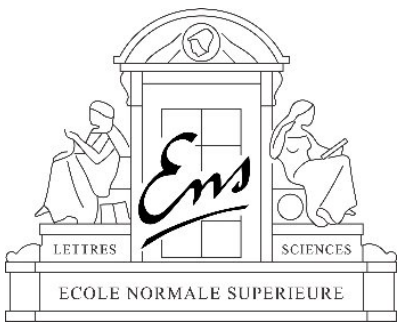
Quadratic Time, Linear Space Algorithms for Gram-Schmidt Orthogonalization and Gaussian Sampling in Structured Lattices

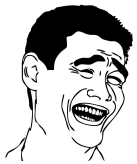
Vadim Lyubashevsky and Thomas Prest



THALES

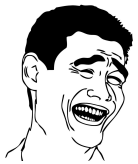
Key Sizes in Cryptography

Key Sizes in Cryptography



RSA

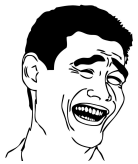
Key Sizes in Cryptography



RSA

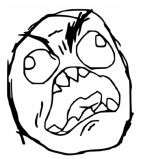
Elliptic Curves

Key Sizes in Cryptography

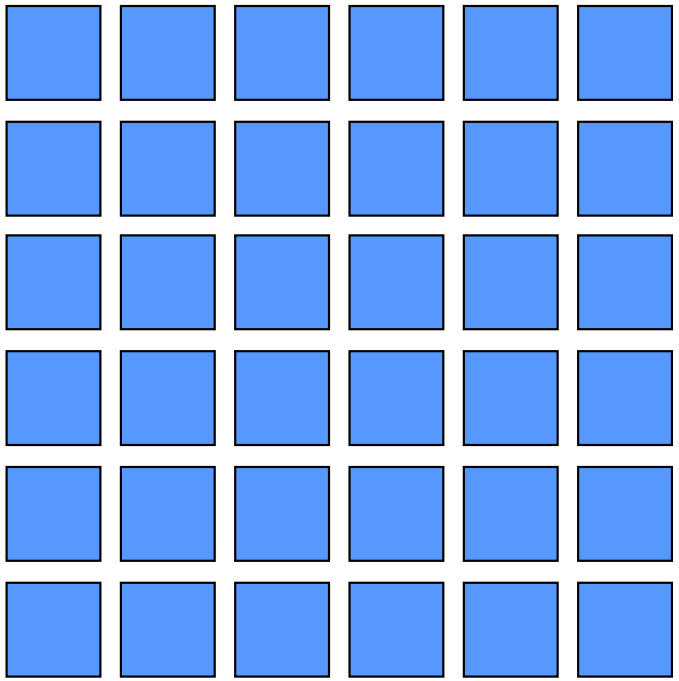


RSA

Elliptic Curves

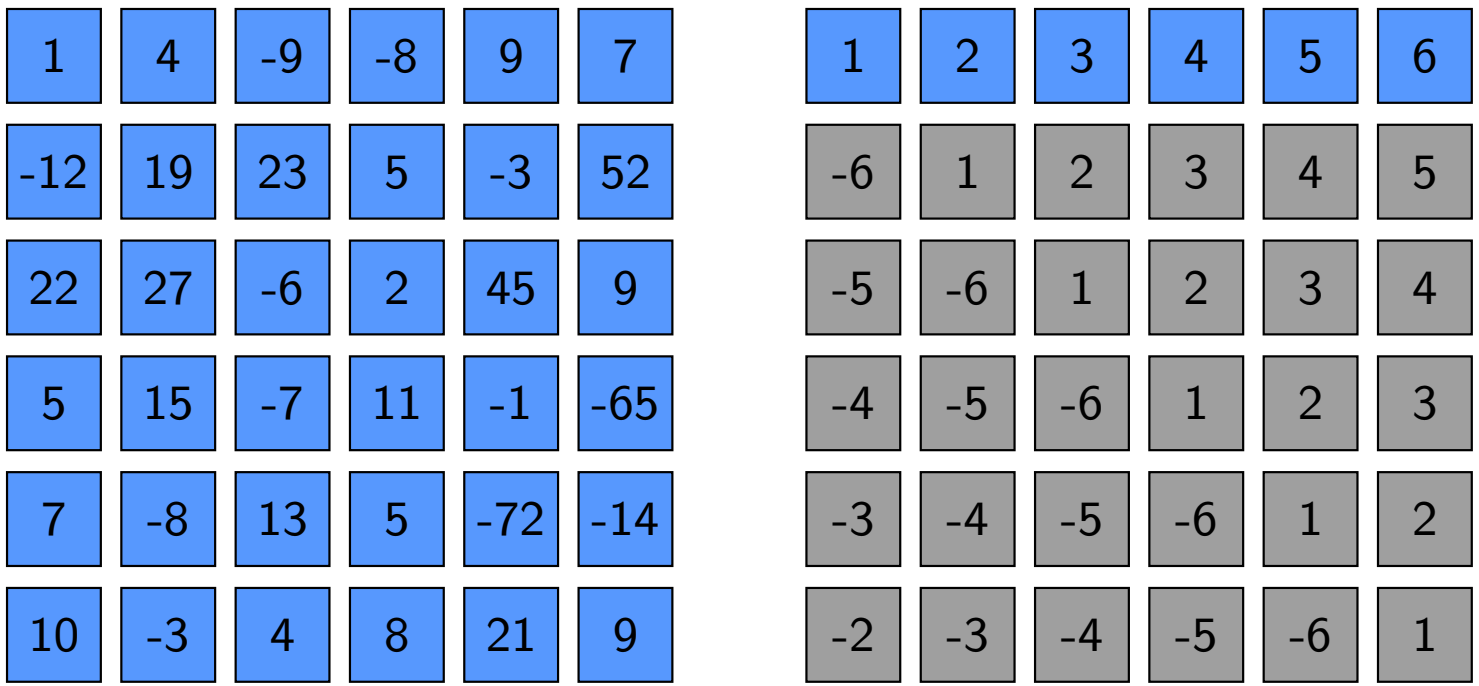


Lattices



Ideal lattices [LM06]

Figure: Size of the keys in lattice-based cryptography



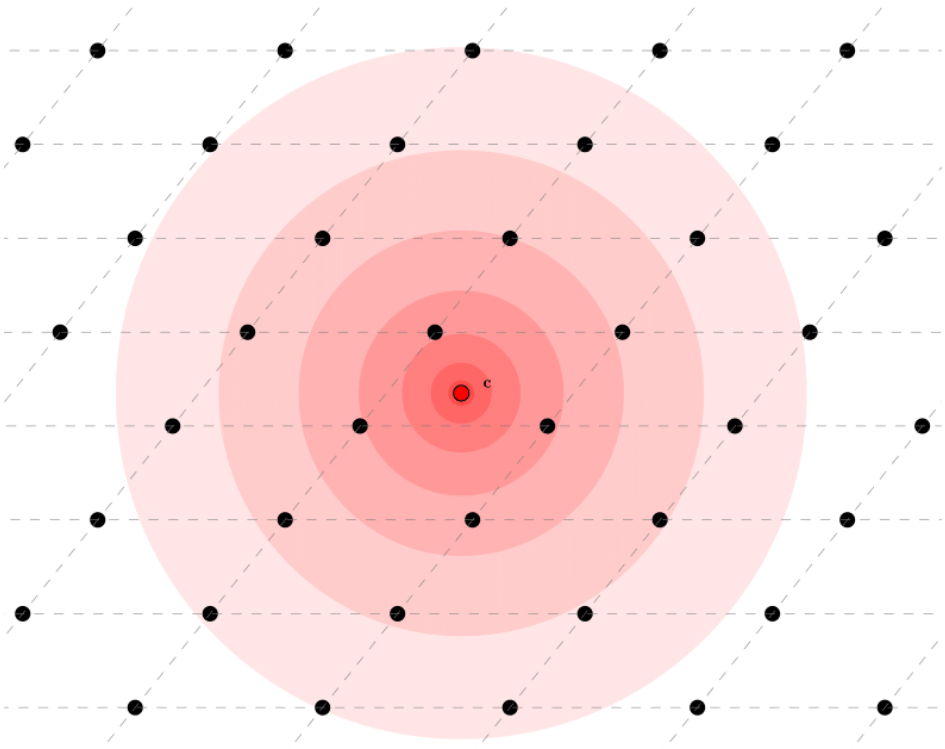
Space requirement goes from $O(n^2)$ to $O(n)$, where n is the dimension.

Gaussian Sampling [Kle00, GPV08]

Very important primitive in lattice-based cryptography:

- Hash-and-sign signatures [GPV08]
- (H)IBE [GPV08, CHKP10, ABB10]
- Standard-model signatures [ABB10]
- Attribute-based encryption [BGG$^+$14]
- ...

Current "best" one: variation [BLP$^+$13] of Gaussian Sampler [GPV08]



The Gaussian Sampler of [Kle00, GPV08]

What is the data required by the Gaussian Sampler for ideal lattices?

Basis $\mathbf{B}$

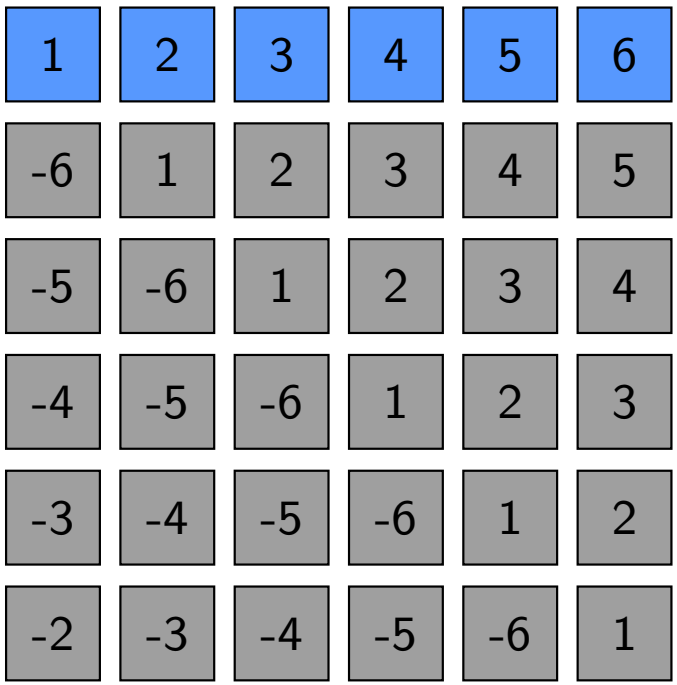
1	2	3	4	5	6
-6	1	2	3	4	5
-5	-6	1	2	3	4
-4	-5	-6	1	2	3
-3	-4	-5	-6	1	2
-2	-3	-4	-5	-6	1

Highly structured!

The Gaussian Sampler of [Kle00, GPV08]

What is the data required by the Gaussian Sampler for ideal lattices?

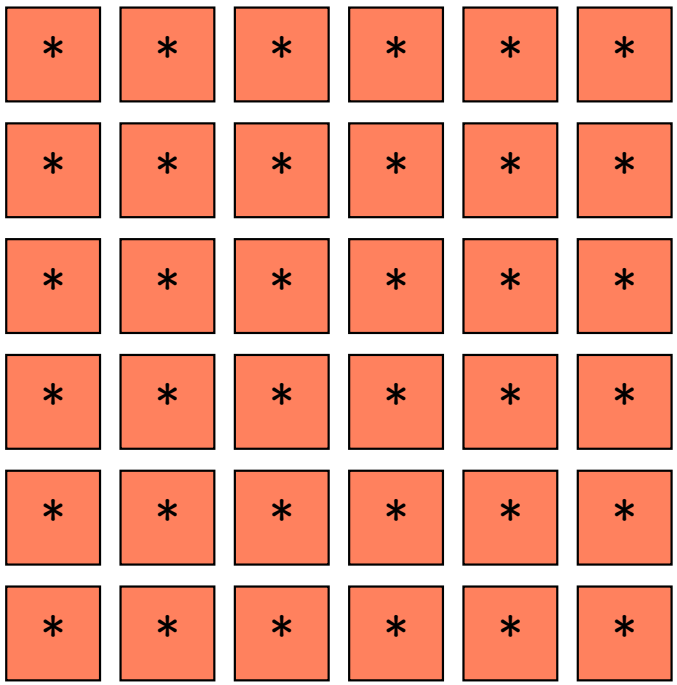
Basis $\mathbf{B}$



Highly structured!

$+$

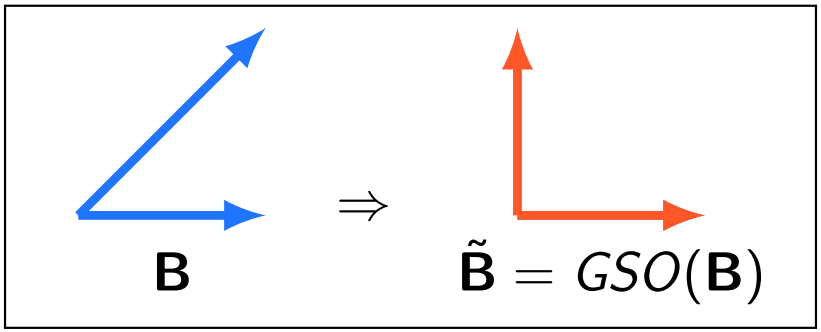
$\tilde{\mathbf{B}} = \texttt{GramSchmidt}(\mathbf{B})$



No space-saving structure!

Gram-Schmidt Orthogonalization (GSO)

What is the Gram-Schmidt Orthogonalization (GSO)?



The GSO for two vectors

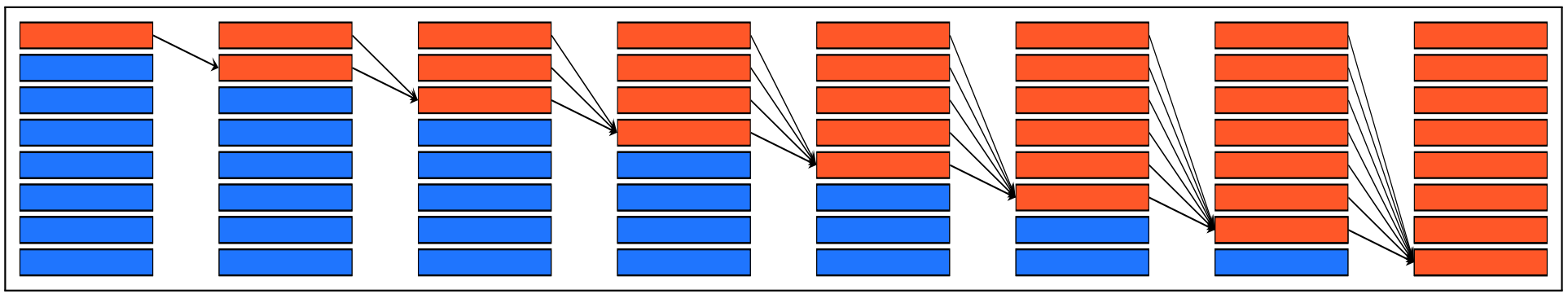


Figure: Computing the GSO $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \tilde{\mathbf{b}}_2, ..., \tilde{\mathbf{b}}_n\}$ for n vectors.
One $\longrightarrow$ arrow takes time $O(n)$. Total time complexity $= O(n^3)$

The Geometry of Ideal Lattices

For the bases we study, we have $\forall k, \mathbf{b}_k = r(\mathbf{b}_{k-1})$:

Where r is an easily computable isometry: $\|r(\mathbf{v})\| = \|\mathbf{v}\|$.

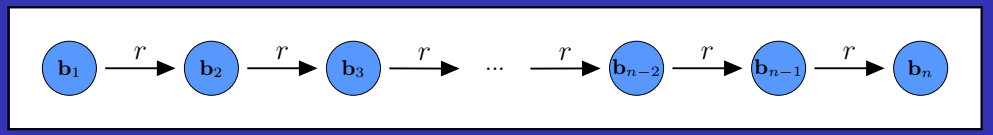
Example:

1	2	3	4
-4	1	2	2
-3	-4	1	2
-2	-3	-4	1

$$f \mod x^4 + 1$$
$$xf \mod x^4 + 1$$
$$x^2 f \mod x^4 + 1$$
$$x^3 f \mod x^4 + 1$$

(It still is the case when replacing $x^4 + 1$ with any cyclotomic polynomial)

Exploiting the Relationship



The idea: compute $\tilde{\mathbf{b}}_{k+1}$ from $\tilde{\mathbf{b}}_k$

$\tilde{\mathbf{b}}_k$ is the reduction of $\mathbf{b}_k$ w.r.t. all the previous vectors

$\Downarrow$ $\qquad$ $\Downarrow$

$$\begin{aligned} & \tilde{\mathbf{b}}_k \perp \mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{k-1} \\ \Rightarrow\ & r(\tilde{\mathbf{b}}_k) \perp r(\mathbf{b}_1), r(\mathbf{b}_2), ..., r(\mathbf{b}_{k-1}) \\ \Rightarrow\ & r(\tilde{\mathbf{b}}_k) \perp \cancel{\mathbf{b}_1}, \mathbf{b}_2, \mathbf{b}_3, ..., \mathbf{b}_k \end{aligned}$$

$$\begin{aligned} & \mathbf{b}_k - \tilde{\mathbf{b}}_k \in \mathit{Span}(\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{k-1}) \\ \Rightarrow\ & r(\mathbf{b}_k) - r(\tilde{\mathbf{b}}_k) \in \mathit{Span}(r(\mathbf{b}_1), r(\mathbf{b}_2), ..., r(\mathbf{b}_{k-1})) \\ \Rightarrow\ & \mathbf{b}_{k+1} - r(\tilde{\mathbf{b}}_k) \in \mathit{Span}(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, ..., \mathbf{b}_k) \end{aligned}$$

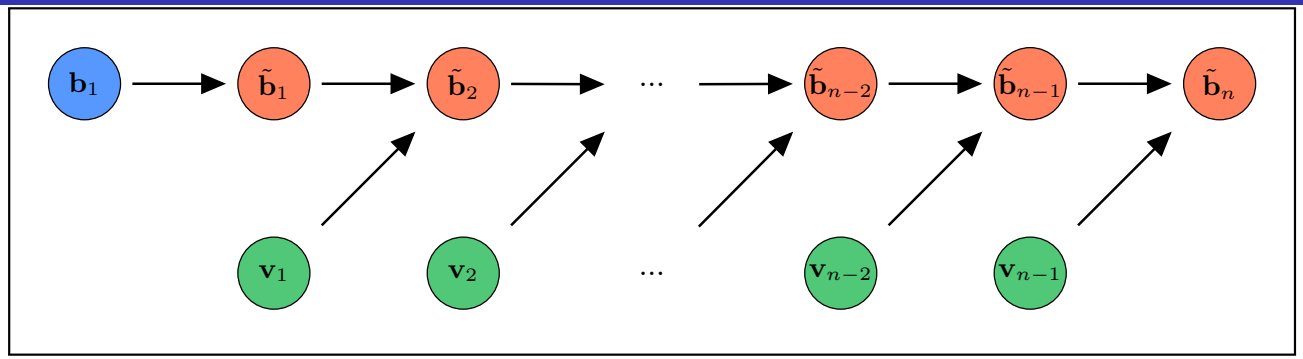
$\Downarrow$ $\qquad$ $\Downarrow$

$r(\tilde{\mathbf{b}}_k)$ is "almost" the reduction of $\mathbf{b}_{k+1}$ w.r.t. all the previous vectors

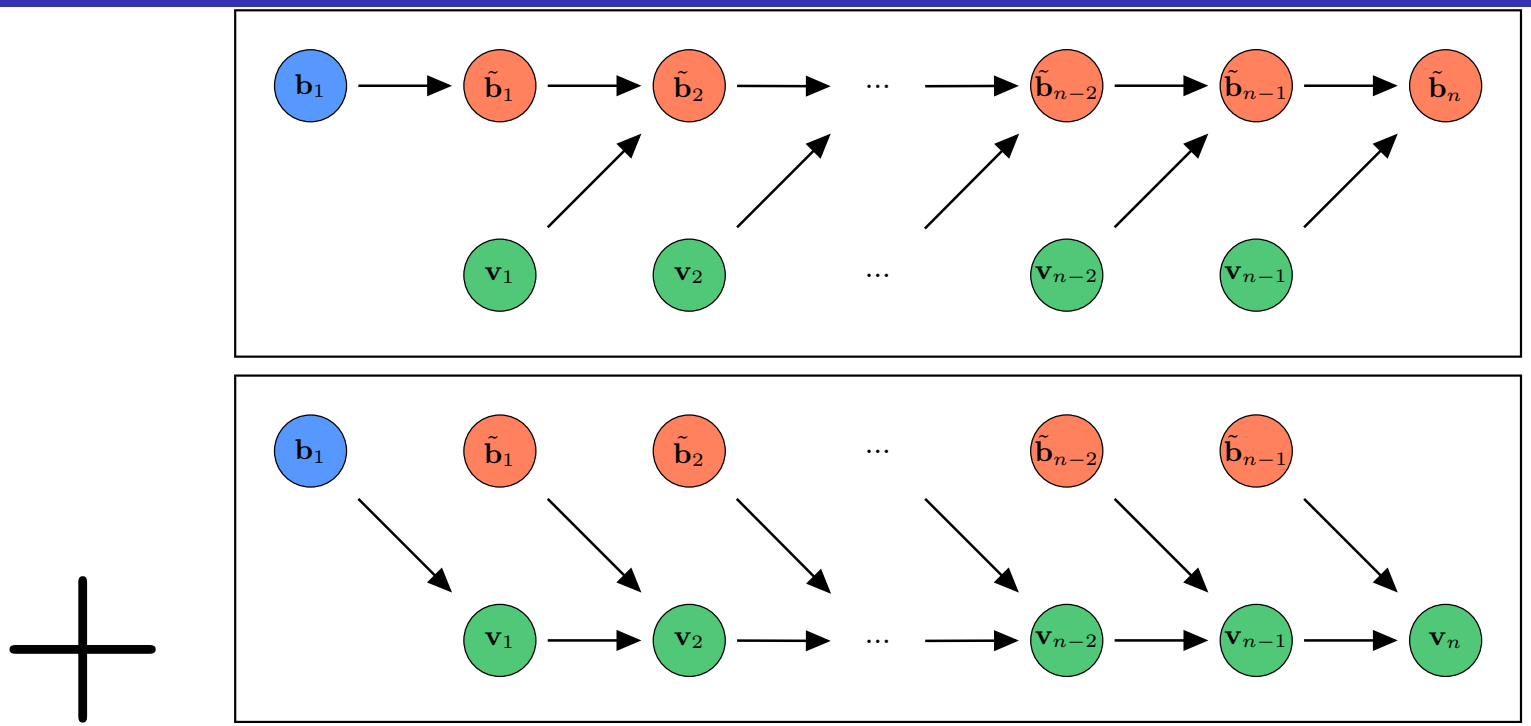
How to turn $r(\tilde{\mathbf{b}}_k)$ into a complete reduction of $\mathbf{b}_{k+1}$?

- Reduce $r(\tilde{\mathbf{b}}_k)$ w.r.t. to $\mathbf{b}_1$? Breaks orthogonality ✗
- Reduce $r(\tilde{\mathbf{b}}_k)$ w.r.t. to $\mathbf{v}_k \triangleq \mathbf{b}_1 - \mathbf{Proj}(\mathbf{b}_1, \mathit{Span}(\mathbf{b}_2, ..., \mathbf{b}_k))$? ✓

GSO for Isometric Bases



GSO for Isometric Bases



GSO for Isometric Bases

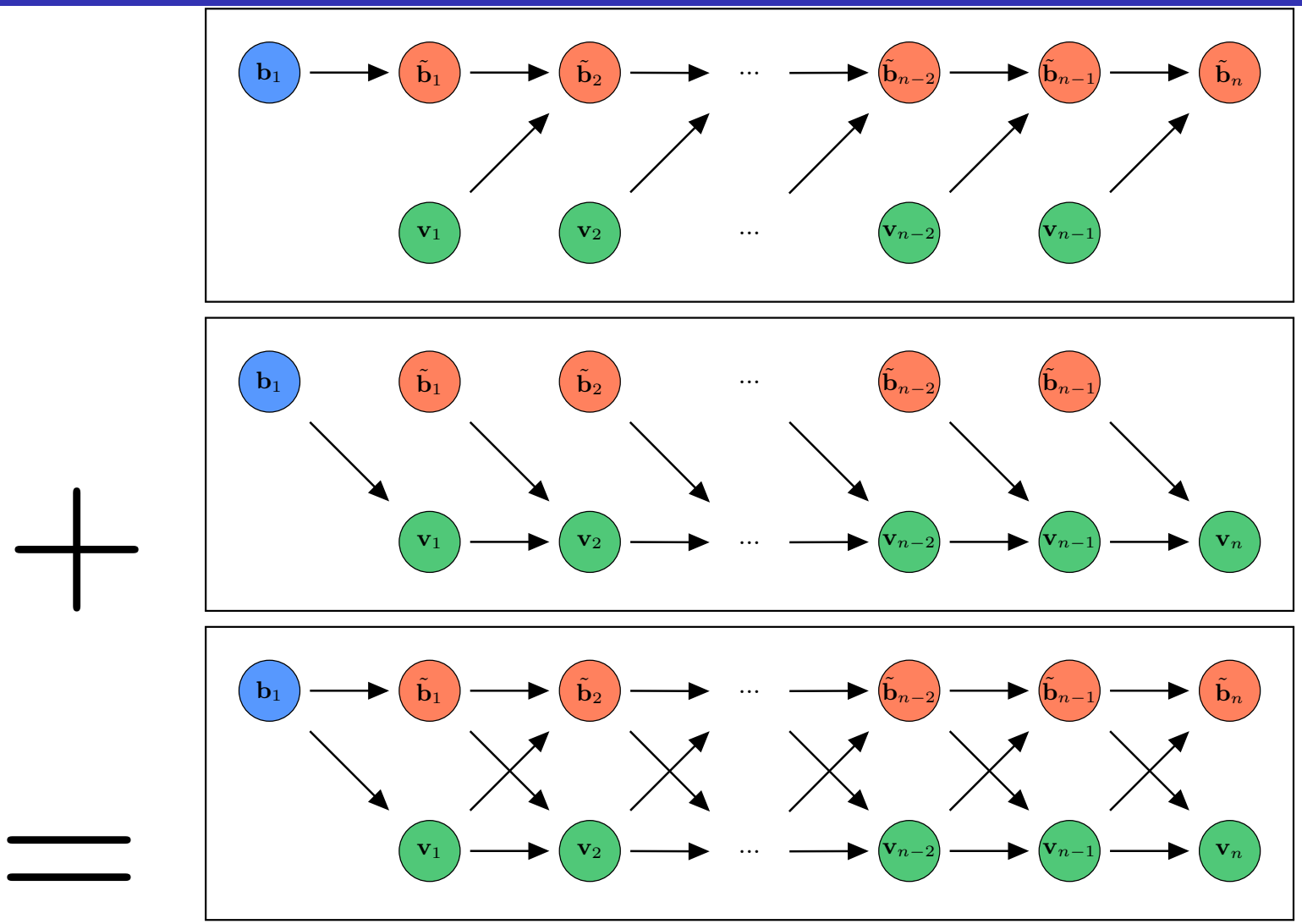
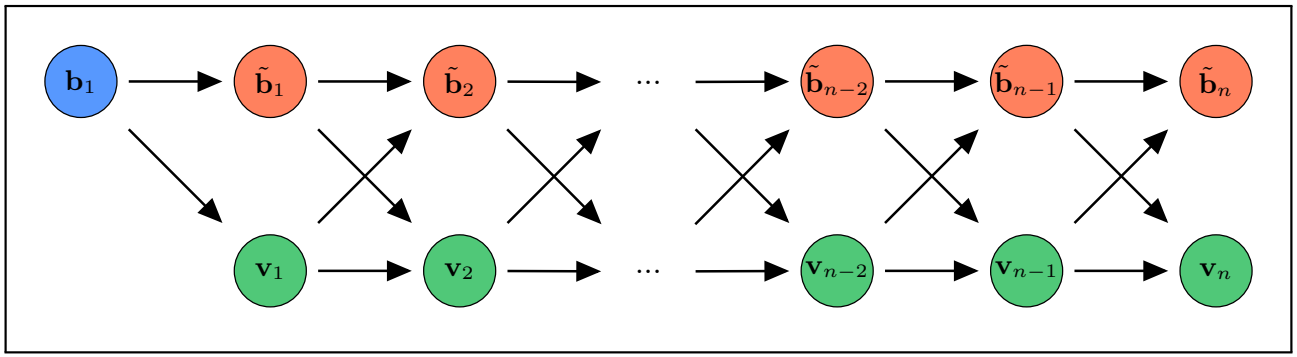


Figure: Computing the Gram-Schmidt Orthogonalization with a double recursion
One $\longrightarrow$ arrow takes time $O(n)$. Total time complexity $= O(n^2)$

GSO for Isometric Bases



Algorithm `Isometric_GSO`$(\mathbf{B})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$

Ensure: GSO basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

$\tilde{\mathbf{b}}_1 \leftarrow \mathbf{b}_1$

$\mathbf{v}_1 \leftarrow \mathbf{b}_1$

for $k = 1, ..., n-1$ **do**

$$\tilde{\mathbf{b}}_{k+1} \leftarrow r(\tilde{\mathbf{b}}_k) - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k) \rangle}{\|\mathbf{v}_k\|^2} \mathbf{v}_k$$

$$\mathbf{v}_{k+1} \leftarrow \mathbf{v}_k - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k) \rangle}{\|\tilde{\mathbf{b}}_k\|^2} r(\tilde{\mathbf{b}}_k)$$

end for

return $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

Algorithm `Classical_GSO`$(\mathbf{B})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$

Ensure: GSO basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

$\tilde{\mathbf{b}}_1 \leftarrow \mathbf{b}_1$

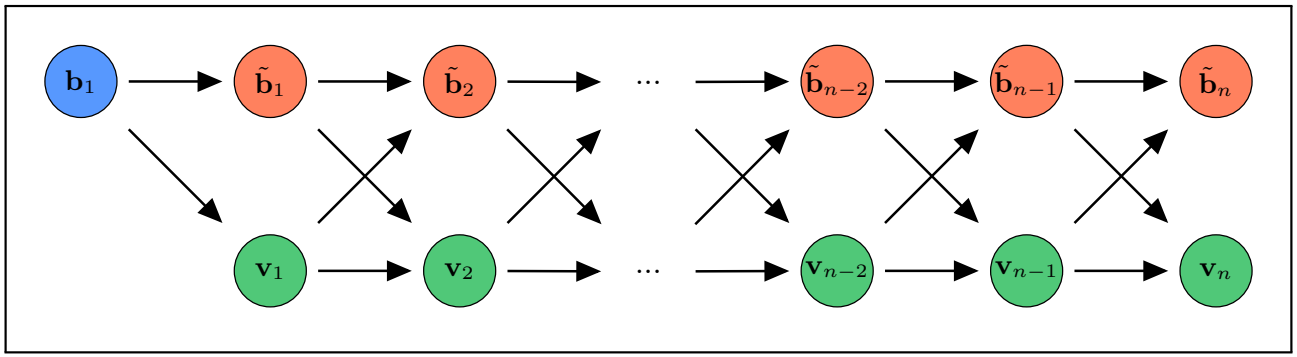
for $k = 1, ..., n-1$ **do**

$$\tilde{\mathbf{b}}_{k+1} \leftarrow r(\mathbf{b}_k) - \sum_{j<k} \frac{\langle \tilde{\mathbf{b}}_j, r(\mathbf{b}_k) \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j$$

end for

return $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

GSO for Isometric Bases



Algorithm `Isometric_GSO`$(\mathbf{B})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$

Ensure: GSO basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

$\tilde{\mathbf{b}}_1 \leftarrow \mathbf{b}_1$

$\mathbf{v}_1 \leftarrow \mathbf{b}_1$

for $k = 1, ..., n-1$ **do**

$\tilde{\mathbf{b}}_{k+1} \leftarrow r(\tilde{\mathbf{b}}_k) - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k) \rangle}{\|\mathbf{v}_k\|^2} \mathbf{v}_k$

$\mathbf{v}_{k+1} \leftarrow \mathbf{v}_k - \frac{\langle \mathbf{v}_k, r(\tilde{\mathbf{b}}_k) \rangle}{\|\tilde{\mathbf{b}}_k\|^2} r(\tilde{\mathbf{b}}_k)$

end for

return $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

Algorithm `Classical_GSO`$(\mathbf{B})$

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$

Ensure: GSO basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

$\tilde{\mathbf{b}}_1 \leftarrow \mathbf{b}_1$

for $k = 1, ..., n-1$ **do**

$\tilde{\mathbf{b}}_{k+1} \leftarrow r(\mathbf{b}_k) - \sum_{j<k} \frac{\langle \tilde{\mathbf{b}}_j, r(\mathbf{b}_k) \rangle}{\|\tilde{\mathbf{b}}_j\|^2} \tilde{\mathbf{b}}_j$

end for

return $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, ..., \tilde{\mathbf{b}}_n\}$

Further optimizations: We can avoid 2 out 3 scalar products $\Rightarrow$ up to 67% faster

Gram-Schmidt Decomposition (GSD)

Definition (Gram-Schmidt Decomposition (GSD))

Write $\mathbf{B} = \mu \times \tilde{\mathbf{B}}$, where $\tilde{\mathbf{B}}$ is the GSO of $\mathbf{B}$

Applications:

- In cryptology: lattice reduction (LLL, BKZ...), Gaussian Sampling
- Outside cryptology: solving least square problems, linear systems, computing eigenvalues...

Speeding up GSD for Isometric Bases

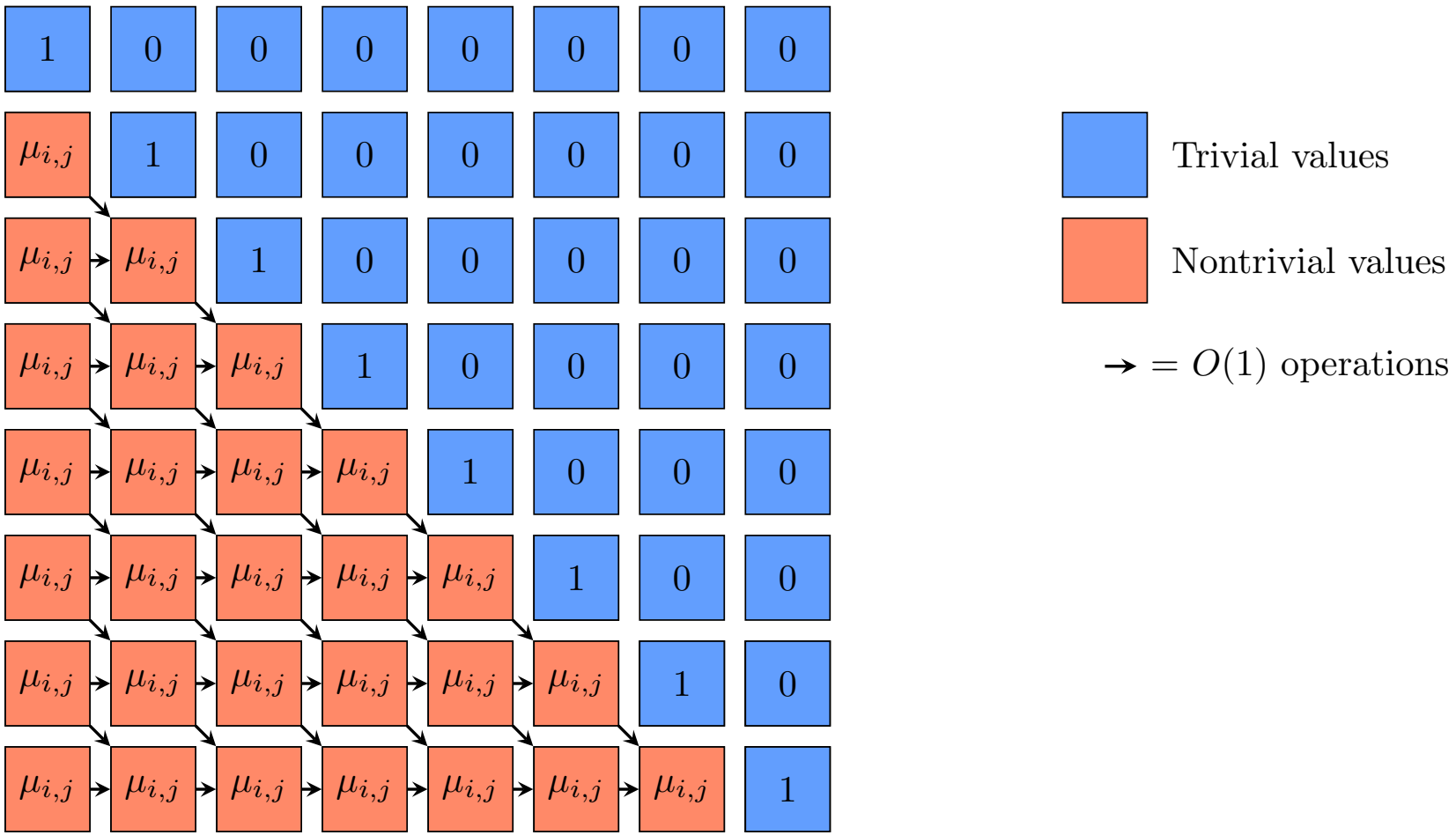
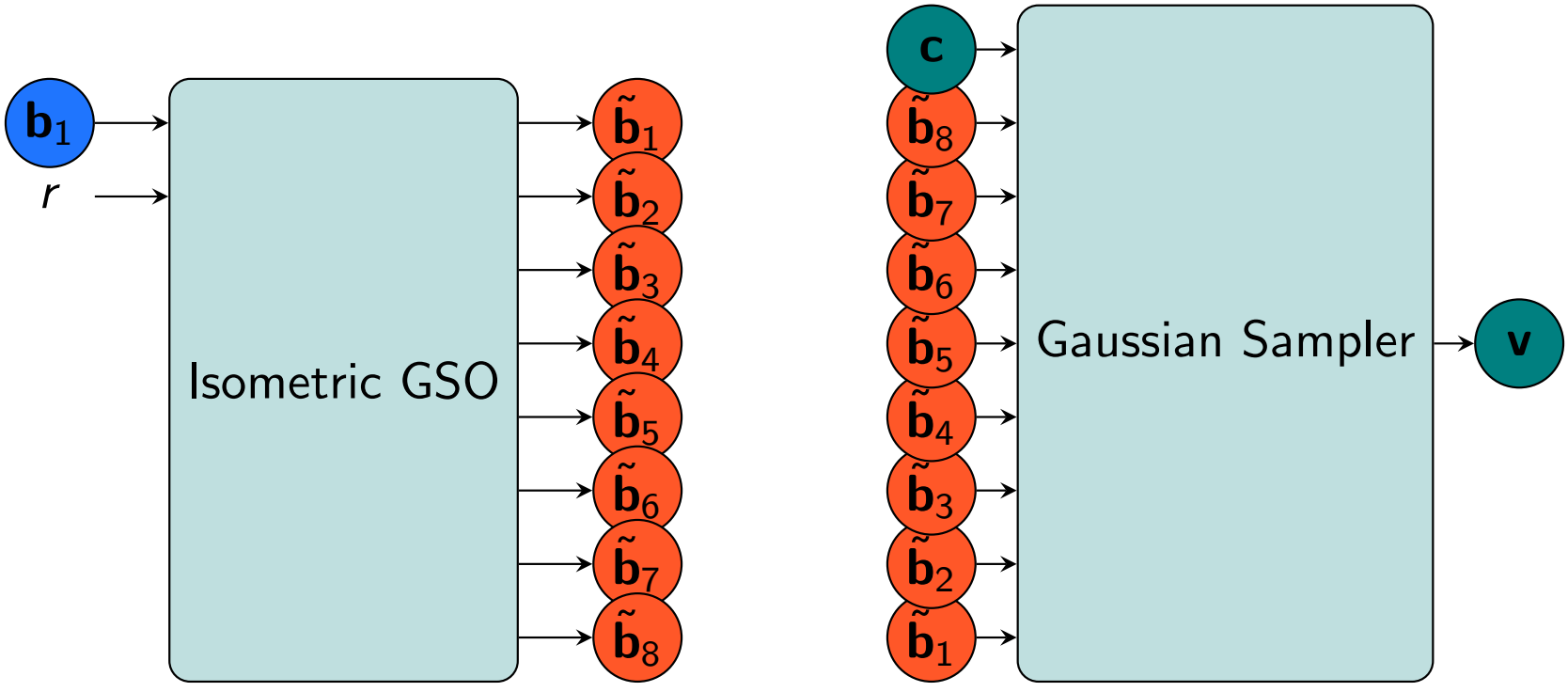


Figure: Fast computation of the matrix μ for an isometric basis.
Overall time and space complexity: $O(n^2)$

GSO and Gaussian Sampler

Now both GSO and the Gaussian Sampler run in time $O(n^2)$.

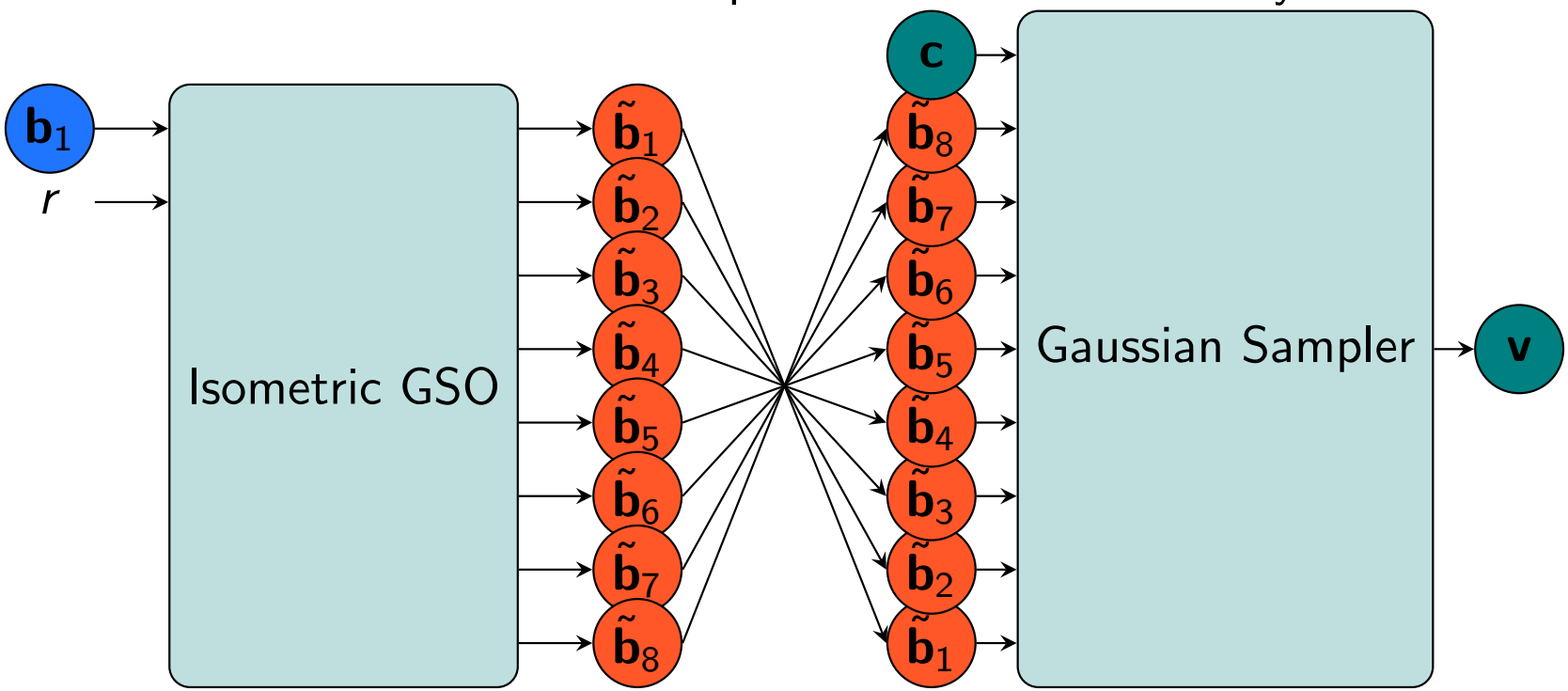
Can we directly plug together the Isometric GSO and the Gaussian Sampler to both run on-the-fly?



GSO and Gaussian Sampler

Now both GSO and the Gaussian Sampler run in time $O(n^2)$.

Can we directly plug together the Isometric GSO and the Gaussian Sampler to both run on-the-fly?



Not really.

Linear-Space Gaussian Sampler

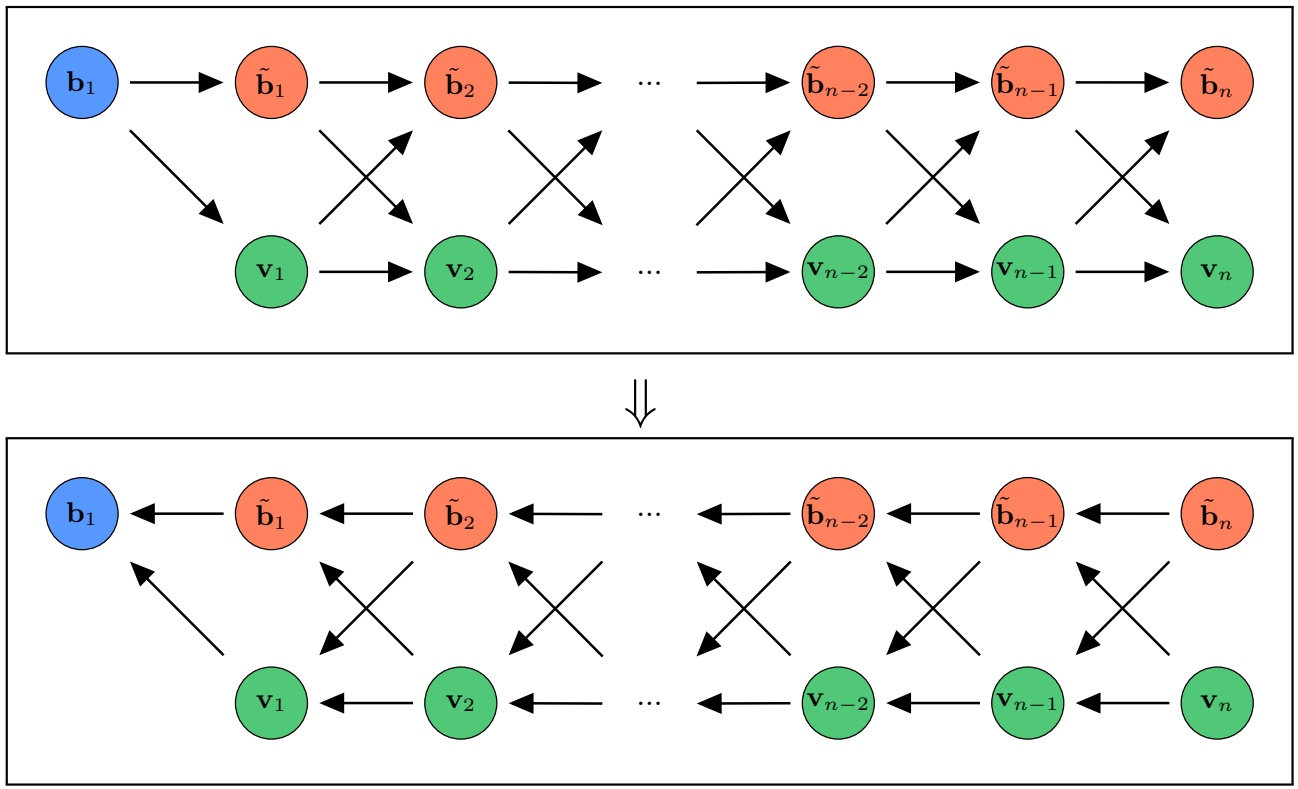
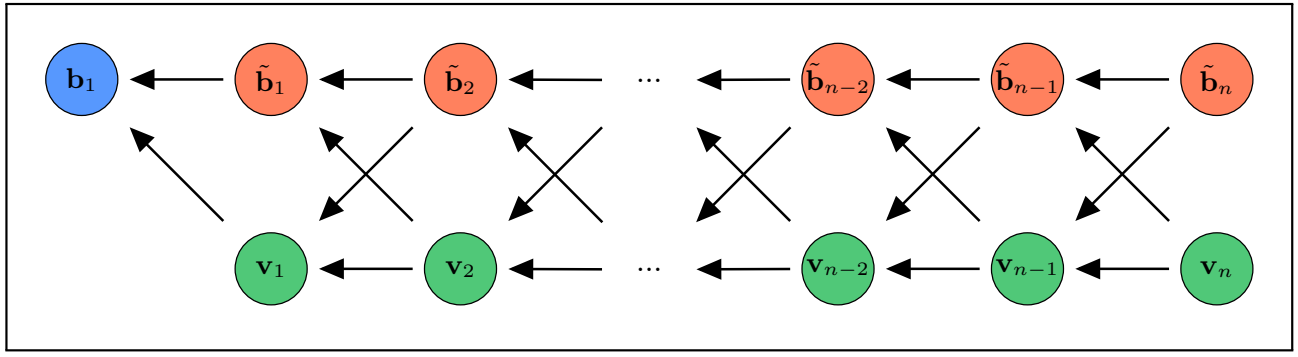


Figure: Reverting Isometric GSO in order to do linear-space Gaussian Sampling
One $\longrightarrow$ arrow takes time $O(n)$ and space $O(1)$ (besides the vectors $\tilde{\mathbf{b}}_k, \mathbf{v}_k$).

A Linear Space Gaussian Sampler



Algorithm

`Classic_Sampler`$(\mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$

Require: $\mathbf{B}$, $\tilde{\mathbf{B}}$, σ, $\mathbf{c}$

Ensure: $\mathbf{z}$ sampled in $\mathcal{D}_{\Lambda(\mathbf{B}),\sigma,\mathbf{c}}$

$\mathbf{c}_n \leftarrow \mathbf{c}$

for $k \leftarrow n, ..., 1$ **do**

$d_k \leftarrow \langle \mathbf{c}_k, \tilde{\mathbf{b}}_k \rangle / \|\tilde{\mathbf{b}}_k\|^2$

$\sigma_k \leftarrow \sigma / \|\tilde{\mathbf{b}}_k\|$

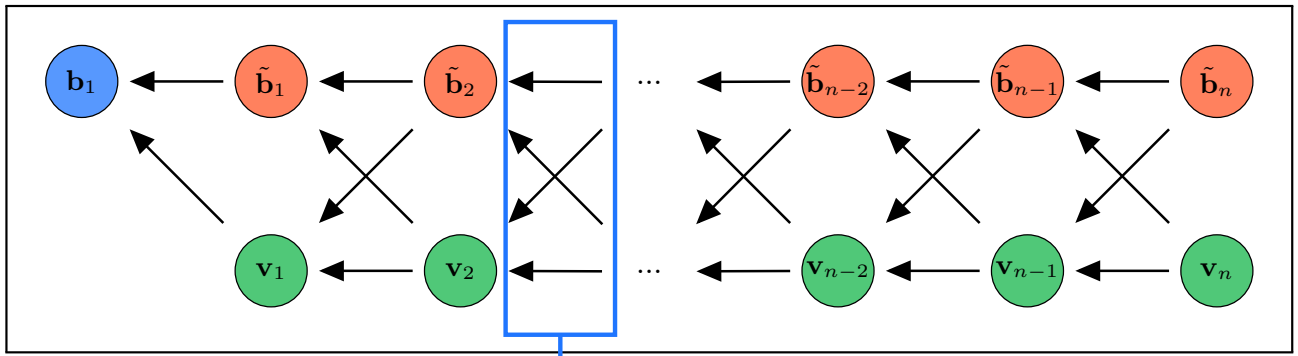
$z_k \leftarrow D_{\mathbb{Z},\sigma_k,d_k}$

$\mathbf{c}_{k-1} \leftarrow \mathbf{c}_k - z_k \mathbf{b}_k$

end for

return $\mathbf{c} - \mathbf{c}_0$

A Linear Space Gaussian Sampler



Algorithm
`Classic_Sampler`$(\mathbf{B}, \tilde{\mathbf{B}}, \sigma, \mathbf{c})$

Require: $\mathbf{B}$, $\tilde{\mathbf{B}}$, σ, $\mathbf{c}$
Ensure: $\mathbf{z}$ sampled in $\mathcal{D}_{\Lambda(\mathbf{B}),\sigma,\mathbf{c}}$

$\mathbf{c}_n \leftarrow \mathbf{c}$
for $k \leftarrow n, ..., 1$ **do**
  $d_k \leftarrow \langle \mathbf{c}_k, \tilde{\mathbf{b}}_k \rangle / \|\tilde{\mathbf{b}}_k\|^2$
  $\sigma_k \leftarrow \sigma / \|\tilde{\mathbf{b}}_k\|$
  $z_k \leftarrow D_{\mathbb{Z},\sigma_k,d_k}$
  $\mathbf{c}_{k-1} \leftarrow \mathbf{c}_k - z_k \mathbf{b}_k$
end for
return $\mathbf{c} - \mathbf{c}_0$

Algorithm
`Compact_Sampler`$(\mathbf{B}, \tilde{\mathbf{b}}_n, \mathbf{v}_n, \sigma, \mathbf{c}, (H_k, I_k)_k)$

Require: $\mathbf{B}$, $\tilde{\mathbf{b}}_n$, $\mathbf{v}_n$, σ, $\mathbf{c}$
Ensure: $\mathbf{z}$ sampled in $\mathcal{D}_{\Lambda(\mathbf{B}),\sigma,\mathbf{c}}$

$\mathbf{c}_n \leftarrow \mathbf{c}$
for $k \leftarrow n, ..., 1$ **do**
  $d_k \leftarrow \langle \mathbf{c}_k, \tilde{\mathbf{b}}_k \rangle / \|\tilde{\mathbf{b}}_k\|^2$
  $\sigma_k \leftarrow \sigma / \|\tilde{\mathbf{b}}_k\|$
  $z_k \leftarrow D_{\mathbb{Z},\sigma_k,d_k}$
  $\mathbf{c}_{k-1} \leftarrow \mathbf{c}_k - z_k \mathbf{b}_k$
  $\tilde{\mathbf{b}}_{k-1} = r^{-1}(H_k \tilde{\mathbf{b}}_k + I_k \mathbf{v}_k)$
  $\mathbf{v}_{k-1} = I_k \tilde{\mathbf{b}}_k + H_i \mathbf{v}_k$
end for
return $\mathbf{c} - \mathbf{c}_0$

Timings and Space Requirements

Table: Timings and space requirements of the classic and compact Gaussian Samplers (Classic GS and Compact GS)[1].

Statistical distance from ideal		2^{-128}
Precision needed	Classic GS	163 bits
	Compact GS	193 bits
Running time	Classic GS	170 ms
	Compact GS	521 ms
Space requirement	Classic GS	163 Mb
	Compact GS	0.47 Mb

- Running time: $\times$**3.06**
- Space requirement: /**340**

[1] The implementation was done in `C++` using `GMP`. Timings were performed on an Intel Core i5-3210M laptop with a 2.5GHz CPU and 6GB RAM.

A few open questions:

- Precision analysis of Isometric GSO?
- Better precision analysis of Gaussian Sampler?
- Combine with ideas of [DN12]?
- How does this link to Arnoldi iteration and Lanczos Algorithm?
- Use same principles to improve other algorithms?

Conclusion

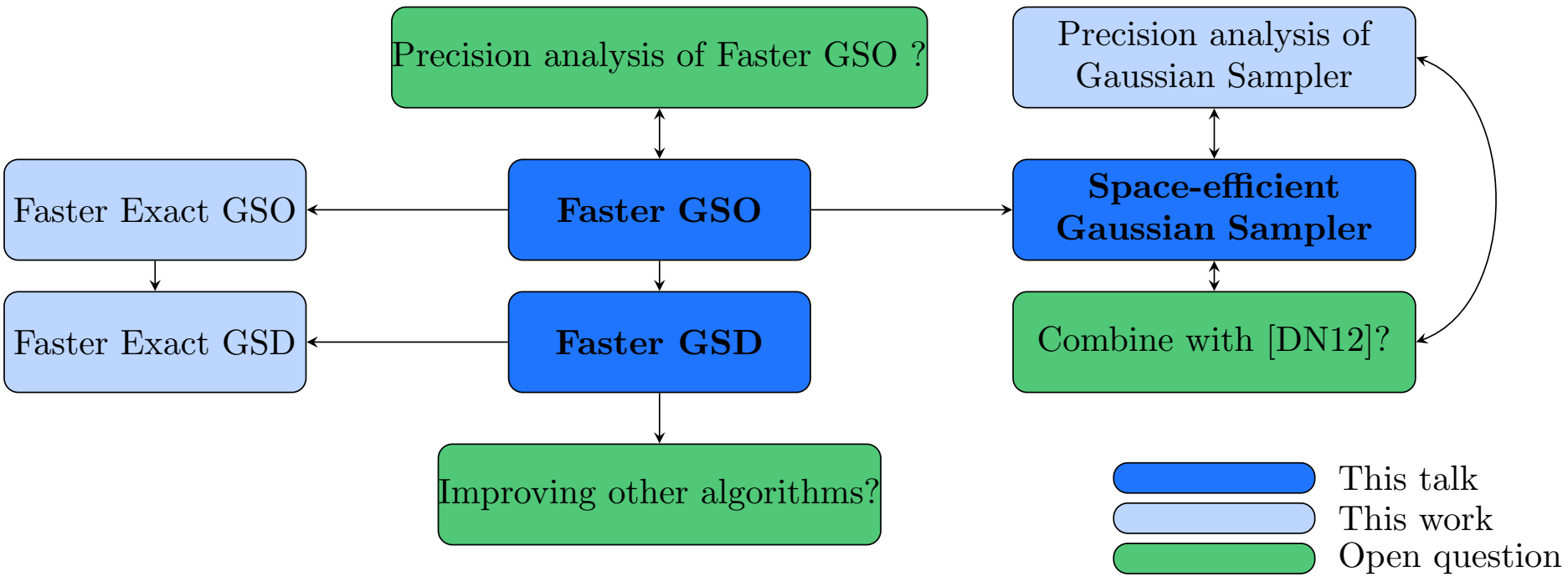


Figure: Present and future work

Conclusion

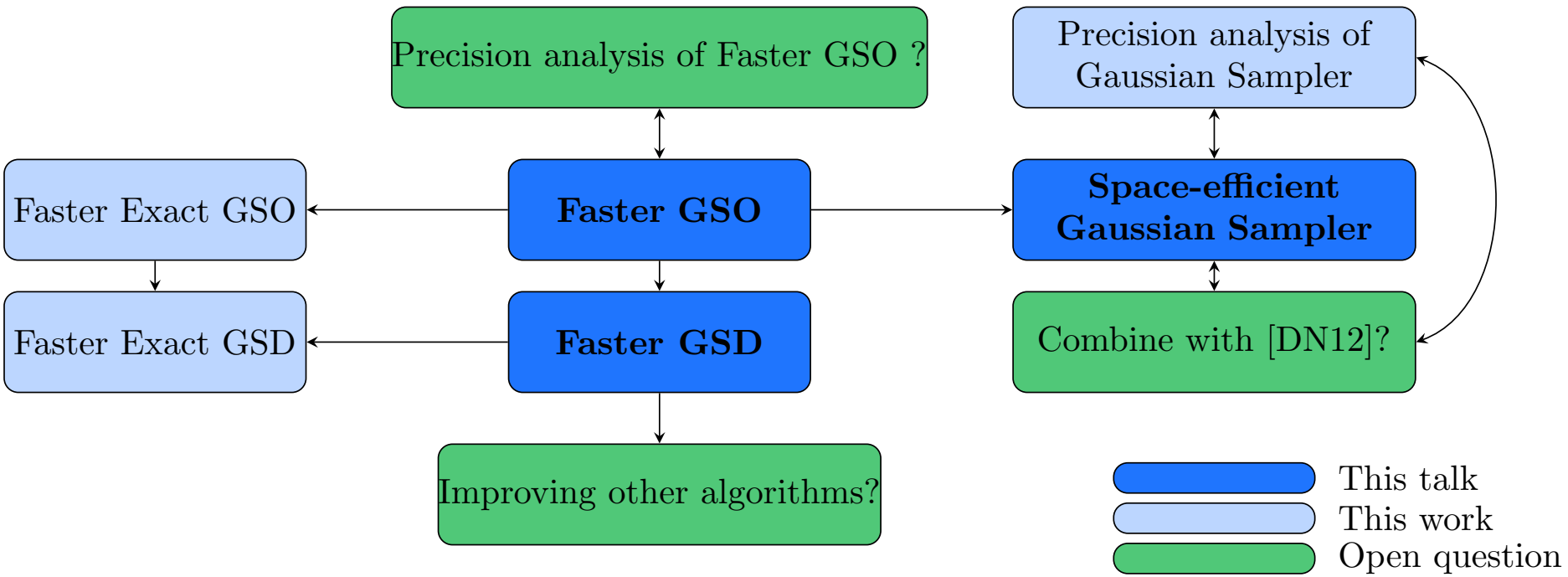


Figure: Present and future work

Thank you!

☞ Shweta Agrawal, Dan Boneh, and Xavier Boyen.
Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE.
In *CRYPTO'10*, pages 98–115, 2010.

☞ Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy.
Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits.
In *EUROCRYPT'14*, pages 533–556, 2014.

☞ Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé.
Classical hardness of learning with errors.
In *STOC*, pages 575–584, 2013.

☞ David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert.
Bonsai trees, or how to delegate a lattice basis.
In *EUROCRYPT*, pages 523–552, 2010.

☞ Léo Ducas and Phong Q. Nguyen.
Faster gaussian lattice sampling using lazy floating-point arithmetic.
In *ASIACRYPT'2012*, pages 415–432, 2012.

☞ Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan.
Trapdoors for hard lattices and new cryptographic constructions.
In *STOC*, pages 197–206, 2008.

☞ Philip N. Klein.
Finding the closest lattice vector when it's unusually close.
In *SODA'00*, pages 937–941, 2000.

☞ Vadim Lyubashevsky and Daniele Micciancio.
Generalized compact knapsacks are collision resistant.
In *ICALP (2)*, pages 144–155, 2006.