Lattice-Based Cryptography: Trapdoors, Discrete Gaussians, and Applications

Chris Peikert
Georgia Institute of Technology

crypt@b-it 2013

Agenda

1. "Strong trapdoors" for lattices

2. Discrete Gaussians, sampling, and "preimage sampleable" functions

3. Applications: signatures, ID-based encryption (in RO model)

Digital Signatures

(Images courtesy xkcd.org)

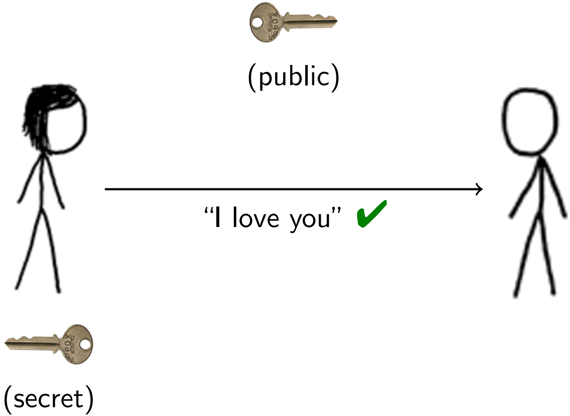
Digital Signatures

(public)

(secret)

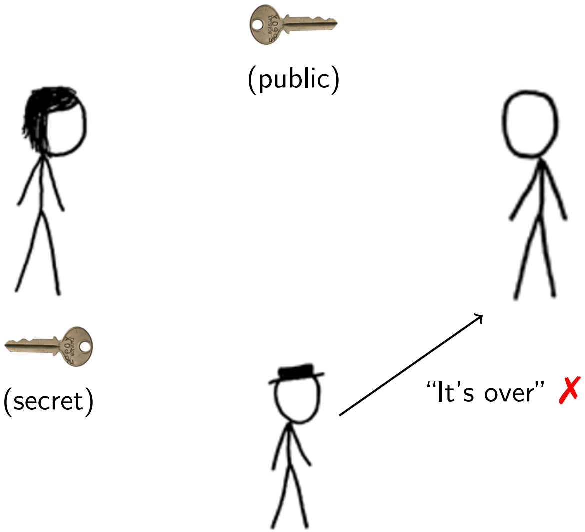
(Images courtesy xkcd.org)

Digital Signatures



(Images courtesy xkcd.org)

Digital Signatures



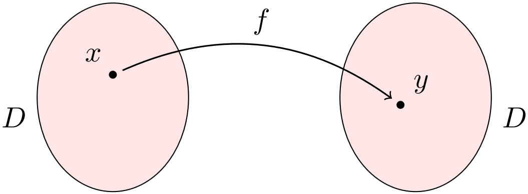
(Images courtesy xkcd.org)

Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}

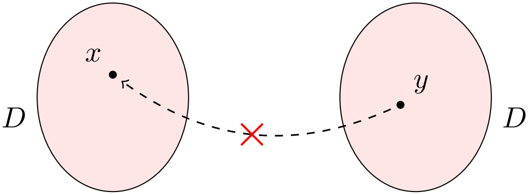
Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- Trapdoor permutation [DH'76,RSA'77,...] (TDP)



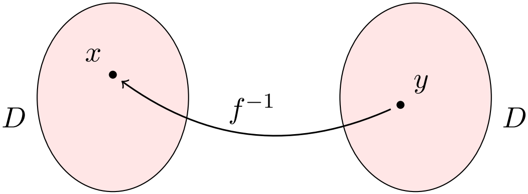
Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- Trapdoor permutation [DH'76,RSA'77,...] (TDP)



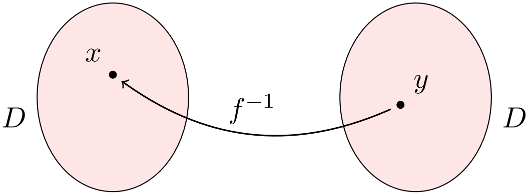
Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- Trapdoor permutation [DH'76,RSA'77,...] (TDP)



Central Tool: Trapdoor Functions

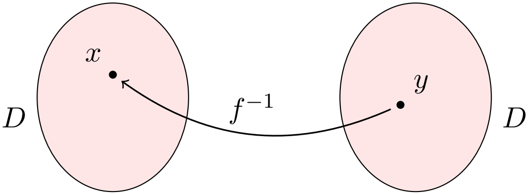
- Public function f generated with secret 'trapdoor' f^{-1}
- Trapdoor permutation [DH'76,RSA'77,...] (TDP)



- 'Hash and sign:' $pk = f$, $sk = f^{-1}$. Sign(msg) $= f^{-1}(H(\mathsf{msg}))$.

Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- Trapdoor permutation [DH'76,RSA'77,...] (TDP)



- 'Hash and sign:' $pk = f$, $sk = f^{-1}$. Sign(msg) $= f^{-1}(H(\text{msg}))$.
- Candidate TDPs: [RSA'78,Rabin'79,Paillier'99] ('general assumption')

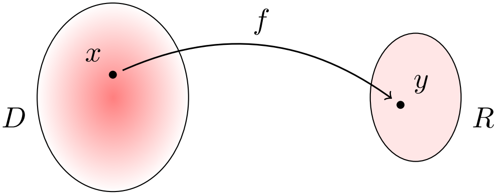
  All rely on hardness of factoring:

  ✗ Complex: 2048-bit exponentiation

  ✗ Broken by quantum algorithms [Shor'97]

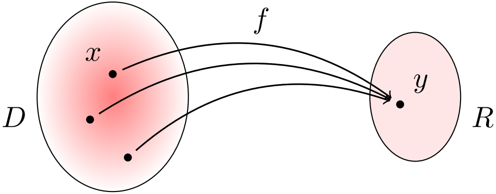
Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- New twist [GPV'08]: preimage sampleable trapdoor function (PSF)



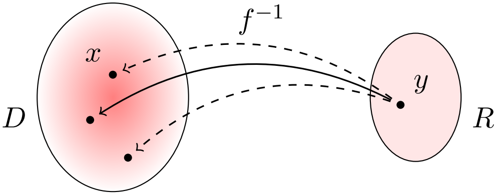
Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- New twist [GPV'08]: preimage sampleable trapdoor function (PSF)



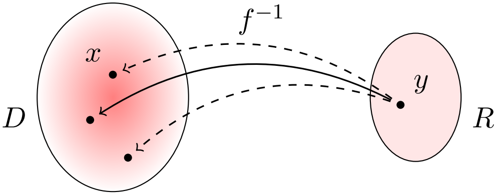
Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- New twist [GPV'08]: preimage sampleable trapdoor function (PSF)



Central Tool: Trapdoor Functions

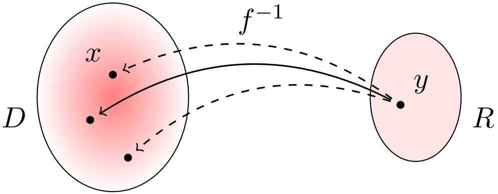
- Public function f generated with secret 'trapdoor' f^{-1}
- New twist [GPV'08]: preimage sampleable trapdoor function (PSF)



- 'Hash and sign:' $pk = f$, $sk = f^{-1}$. Sign(msg) $= f^{-1}(H(\text{msg}))$.

Central Tool: Trapdoor Functions

- Public function f generated with secret 'trapdoor' f^{-1}
- New twist [GPV'08]: preimage sampleable trapdoor function (PSF)

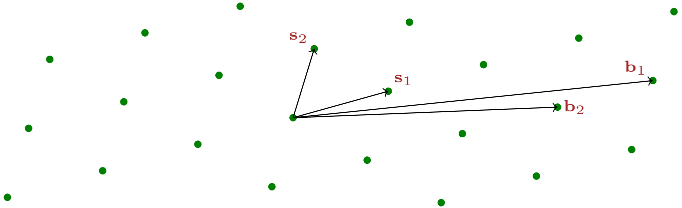


- 'Hash and sign:' $pk = f$, $sk = f^{-1}$. Sign(msg) $= f^{-1}(H(\text{msg}))$.
- Still secure! Can generate (x, y) in two equivalent ways:

REALITY	PROOF
$x \xleftarrow{f^{-1}} y \longleftarrow R$	$D \longrightarrow x \xrightarrow{f} y$

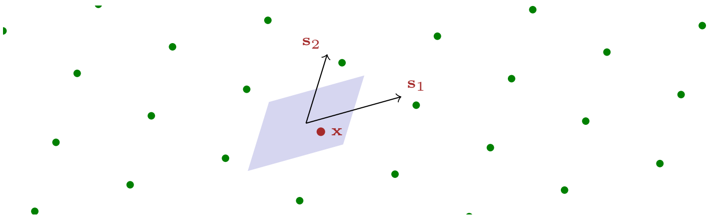
Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$



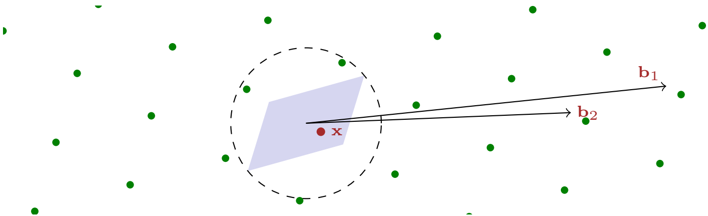
Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]



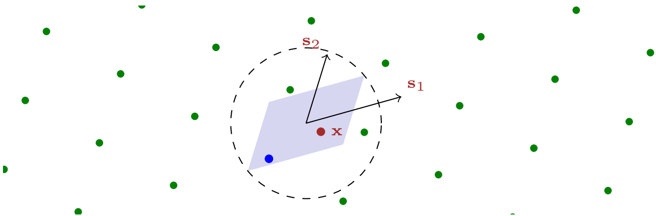
Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\mathsf{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\mathsf{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\mathsf{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough



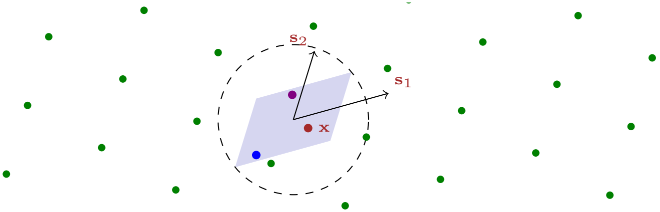
Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough



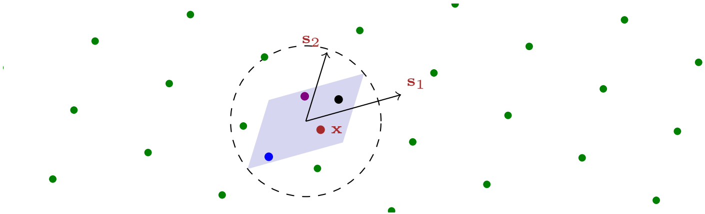
Candidate Signature Scheme [GGH'96]

- Key idea: pk = "bad" basis $\mathbf{B}$ for $\mathcal{L}$, sk = "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\mathsf{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\mathsf{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\mathsf{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough



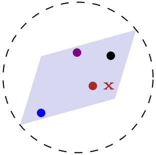
Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough



Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

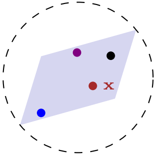


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)

Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

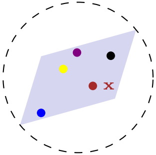


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)
2. Signing algorithm leaks secret basis!
   - ⋆ Total break after 100s-1000s of signatures [NguyenRegev'06]

Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

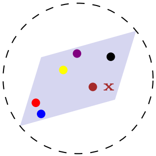


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)
2. Signing algorithm leaks secret basis!
   - ⋆ Total break after 100s-1000s of signatures [NguyenRegev'06]

Candidate Signature Scheme [GGH'96]

- Key idea: pk = "bad" basis $\mathbf{B}$ for $\mathcal{L}$, sk = "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

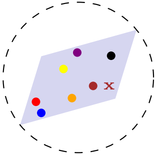


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)
2. Signing algorithm leaks secret basis!
   - ⋆ Total break after 100s-1000s of signatures [NguyenRegev'06]

Candidate Signature Scheme [GGH'96]

- Key idea: pk = "bad" basis $\mathbf{B}$ for $\mathcal{L}$, sk = "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

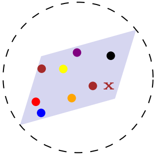


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)
2. Signing algorithm leaks secret basis!
   - ★ Total break after 100s-1000s of signatures [NguyenRegev'06]

Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

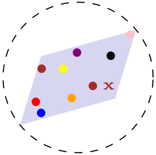


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)
2. Signing algorithm leaks secret basis!
   - ⋆ Total break after 100s-1000s of signatures [NguyenRegev'06]

Candidate Signature Scheme [GGH'96]

- Key idea: $pk =$ "bad" basis $\mathbf{B}$ for $\mathcal{L}$, $sk =$ "short" trapdoor basis $\mathbf{S}$
- Sign: $H(\text{msg}) = \mathbf{c} + \mathcal{L}$; get short $\mathbf{x} \in \mathbf{c} + \mathcal{L}$ via round-off [Babai'86]
- Verify$(\text{msg}, \mathbf{x})$ check $\mathbf{x} \in H(\text{msg}) = \mathbf{c} + \mathcal{L}$, and $\mathbf{x}$ short enough

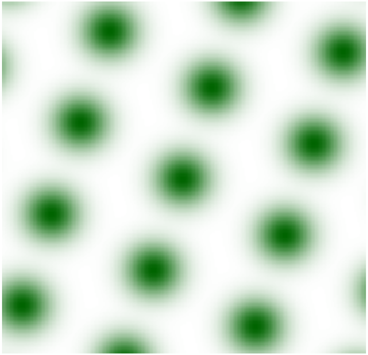


Technical Issues

1. Generating "hard" lattice together with short basis (tomorrow)
2. Signing algorithm leaks secret basis!
   - ⋆ Total break after 100s-1000s of signatures [NguyenRegev'06]

Key Concept: Blurring a Lattice [Regev'03,MR'04]

Key Concept: Blurring a Lattice [Regev'03,MR'04]



Key Concept: Blurring a Lattice [Regev'03,MR'04]

Key Concept: Blurring a Lattice [Regev'03,MR'04]

Question: How much blur makes it uniform?

Gaussians

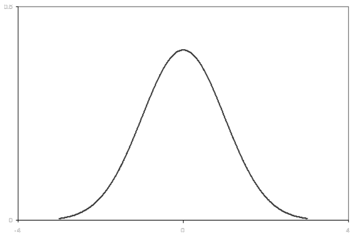
Gaußians

Gaußians

- The 1-dim Gaussian function: (pdf of normal dist w/ std dev $1/\sqrt{2\pi}$)

$$\rho(x) \stackrel{\Delta}{=} \exp(-\pi \cdot x^2).$$

Also define $\rho_s(x) \stackrel{\Delta}{=} \rho(x/s) = \exp(-\pi \cdot (x/s)^2)$.



Gaußians

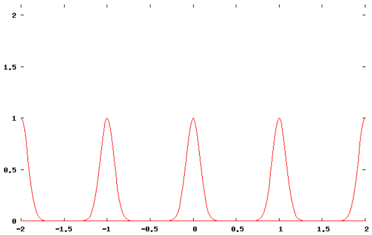
- The 1-dim Gaussian function: (pdf of normal dist w/ std dev $1/\sqrt{2\pi}$)

$$\rho(x) \stackrel{\Delta}{=} \exp(-\pi \cdot x^2).$$

  Also define $\rho_s(x) \stackrel{\Delta}{=} \rho(x/s) = \exp(-\pi \cdot (x/s)^2)$.

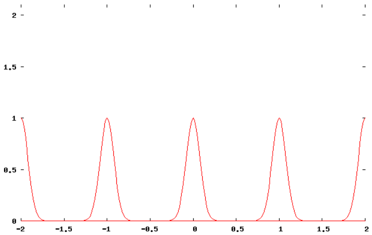
- Sum of Gaussians centered at lattice points:

$$f_s(c) = \sum_{z \in \mathbb{Z}} \rho_s(c - z) = \rho_s(c + \mathbb{Z}).$$



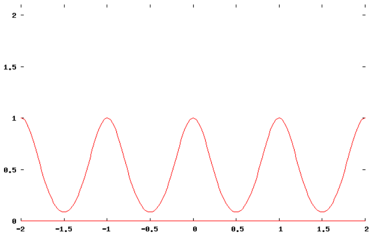
Gaußians

- The 1-dim Gaussian function: (pdf of normal dist w/ std dev $1/\sqrt{2\pi}$)
$$\rho(x) \stackrel{\Delta}{=} \exp(-\pi \cdot x^2).$$
Also define $\rho_s(x) \stackrel{\Delta}{=} \rho(x/s) = \exp(-\pi \cdot (x/s)^2)$.
- Sum of Gaussians centered at lattice points:
$$f_s(c) = \sum_{z \in \mathbb{Z}} \rho_s(c - z) = \rho_s(c + \mathbb{Z}).$$
- Fact: $\rho_s(c + \mathbb{Z}) \in [1 \pm \frac{\varepsilon}{1-\varepsilon}] \cdot s$ for *all* $c \in \mathbb{R}$, where $\varepsilon \leq 2\exp(-\pi s^2)$.



Gaußians

- The 1-dim Gaussian function: (pdf of normal dist w/ std dev $1/\sqrt{2\pi}$)
$$\rho(x) \stackrel{\Delta}{=} \exp(-\pi \cdot x^2).$$
Also define $\rho_s(x) \stackrel{\Delta}{=} \rho(x/s) = \exp(-\pi \cdot (x/s)^2)$.
- Sum of Gaussians centered at lattice points:
$$f_s(c) = \sum_{z \in \mathbb{Z}} \rho_s(c - z) = \rho_s(c + \mathbb{Z}).$$
- Fact: $\rho_s(c + \mathbb{Z}) \in [1 \pm \frac{\varepsilon}{1-\varepsilon}] \cdot s$ for *all* $c \in \mathbb{R}$, where $\varepsilon \leq 2\exp(-\pi s^2)$.



Gaußians

- The 1-dim Gaussian function: (pdf of normal dist w/ std dev $1/\sqrt{2\pi}$)

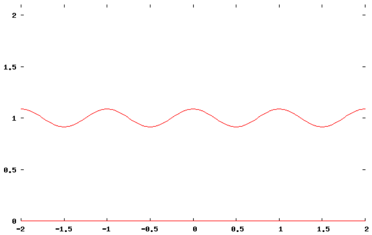
$$\rho(x) \stackrel{\Delta}{=} \exp(-\pi \cdot x^2).$$

  Also define $\rho_s(x) \stackrel{\Delta}{=} \rho(x/s) = \exp(-\pi \cdot (x/s)^2)$.

- Sum of Gaussians centered at lattice points:

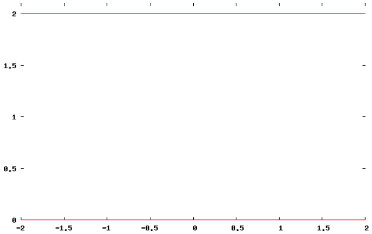
$$f_s(c) = \sum_{z \in \mathbb{Z}} \rho_s(c - z) = \rho_s(c + \mathbb{Z}).$$

- Fact: $\rho_s(c + \mathbb{Z}) \in [1 \pm \frac{\varepsilon}{1-\varepsilon}] \cdot s$ for *all* $c \in \mathbb{R}$, where $\varepsilon \leq 2\exp(-\pi s^2)$.



Gaußians

- The 1-dim Gaussian function: (pdf of normal dist w/ std dev $1/\sqrt{2\pi}$)
$$\rho(x) \stackrel{\Delta}{=} \exp(-\pi \cdot x^2).$$
Also define $\rho_s(x) \stackrel{\Delta}{=} \rho(x/s) = \exp(-\pi \cdot (x/s)^2)$.
- Sum of Gaussians centered at lattice points:
$$f_s(c) = \sum_{z \in \mathbb{Z}} \rho_s(c - z) = \rho_s(c + \mathbb{Z}).$$
- Fact: $\rho_s(c + \mathbb{Z}) \in [1 \pm \frac{\varepsilon}{1-\varepsilon}] \cdot s$ for *all* $c \in \mathbb{R}$, where $\varepsilon \leq 2\exp(-\pi s^2)$.



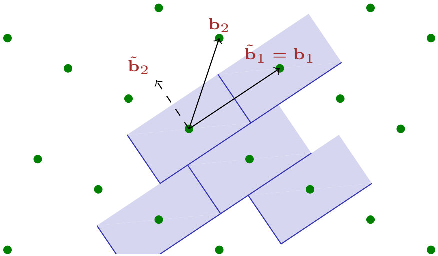
n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

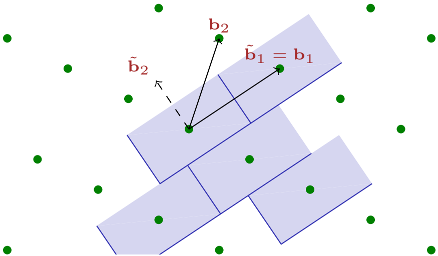
- <u>Fact</u>: Suppose $\mathcal{L}$ has a basis $\mathbf{B}$ with $M = \max_i \|\tilde{\mathbf{b}}_i\|$. Then
  $$\rho_s(\mathbf{c} + \mathcal{L}) \in [1 \pm \varepsilon] \cdot s^n$$
  for *all* $\mathbf{c} \in \mathbb{R}^n$, where $\varepsilon \leq 2n \cdot \exp(-\pi(s/M)^2)$.



n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

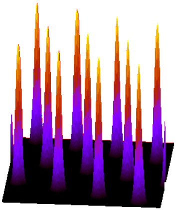
- <u>Fact</u>: Suppose $\mathcal{L}$ has a basis $\mathbf{B}$ with $M = \max_i \|\tilde{\mathbf{b}}_i\|$. Then
  $$\rho_s(\mathbf{c} + \mathcal{L}) \in [1 \pm \varepsilon] \cdot s^n$$
  for *all* $\mathbf{c} \in \mathbb{R}^n$, where $\varepsilon \leq 2n \cdot \exp(-\pi(s/M)^2)$.
  So $s \approx M\sqrt{\log n}$ suffices for near-uniformity.



n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

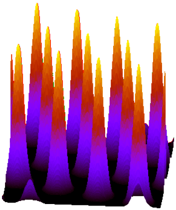
- <u>Fact</u>: Suppose $\mathcal{L}$ has a basis $\mathbf{B}$ with $M = \max_i \|\tilde{\mathbf{b}}_i\|$. Then
  $$\rho_s(\mathbf{c} + \mathcal{L}) \in [1 \pm \varepsilon] \cdot s^n$$
  for *all* $\mathbf{c} \in \mathbb{R}^n$, where $\varepsilon \leq 2n \cdot \exp(-\pi(s/M)^2)$.
  So $s \approx M\sqrt{\log n}$ suffices for near-uniformity.



n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

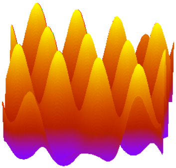
- Fact: Suppose $\mathcal{L}$ has a basis $\mathbf{B}$ with $M = \max_i \|\tilde{\mathbf{b}}_i\|$. Then
  $$\rho_s(\mathbf{c} + \mathcal{L}) \in [1 \pm \varepsilon] \cdot s^n$$
  for *all* $\mathbf{c} \in \mathbb{R}^n$, where $\varepsilon \leq 2n \cdot \exp(-\pi(s/M)^2)$.
  So $s \approx M\sqrt{\log n}$ suffices for near-uniformity.



n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

- Fact: Suppose $\mathcal{L}$ has a basis $\mathbf{B}$ with $M = \max_i \|\tilde{\mathbf{b}}_i\|$. Then
  $$\rho_s(\mathbf{c} + \mathcal{L}) \in [1 \pm \varepsilon] \cdot s^n$$
  for *all* $\mathbf{c} \in \mathbb{R}^n$, where $\varepsilon \leq 2n \cdot \exp(-\pi(s/M)^2)$.
  So $s \approx M\sqrt{\log n}$ suffices for near-uniformity.



n-dimensional Gaussians

- The n-dim Gaussian: $\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \rho(x_1) \cdots \rho(x_n)$.
  Clearly, it is rotationally invariant.

- Fact: Suppose $\mathcal{L}$ has a basis $\mathbf{B}$ with $M = \max_i \|\tilde{\mathbf{b}}_i\|$. Then
  $$\rho_s(\mathbf{c} + \mathcal{L}) \in [1 \pm \varepsilon] \cdot s^n$$
  for *all* $\mathbf{c} \in \mathbb{R}^n$, where $\varepsilon \leq 2n \cdot \exp(-\pi(s/M)^2)$.
  So $s \approx M\sqrt{\log n}$ suffices for near-uniformity.

Discrete Gaussians

- Define the discrete Gaussian distribution over coset $\mathbf{c} + \mathcal{L}$ as

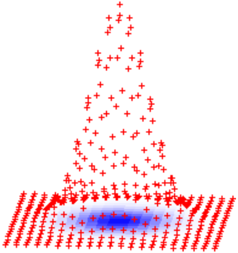
$$D_{\mathbf{c}+\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{c}+\mathcal{L})} \text{ for all } \mathbf{x} \in \mathbf{c} + \mathcal{L}.$$

Discrete Gaussians

- Define the discrete Gaussian distribution over coset $\mathbf{c} + \mathcal{L}$ as

$$D_{\mathbf{c}+\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{c}+\mathcal{L})} \text{ for all } \mathbf{x} \in \mathbf{c} + \mathcal{L}.$$

- Consider the following experiment:
  1. Choose $\mathbf{x} \in \mathbb{Z}^n$ from $D_{\mathbb{Z}^n,s}$.

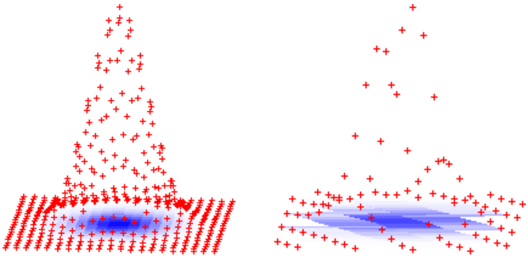


Discrete Gaussians

- Define the discrete Gaussian distribution over coset $\mathbf{c} + \mathcal{L}$ as

$$D_{\mathbf{c}+\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{c}+\mathcal{L})} \text{ for all } \mathbf{x} \in \mathbf{c} + \mathcal{L}.$$

- Consider the following experiment:
  1. Choose $\mathbf{x} \in \mathbb{Z}^n$ from $D_{\mathbb{Z}^n,s}$.
  2. Reveal coset $\mathbf{x} + \mathcal{L}$. (e.g., as $\bar{\mathbf{x}} = \mathbf{x} \bmod \mathbf{B}$ for some basis $\mathbf{B}$)



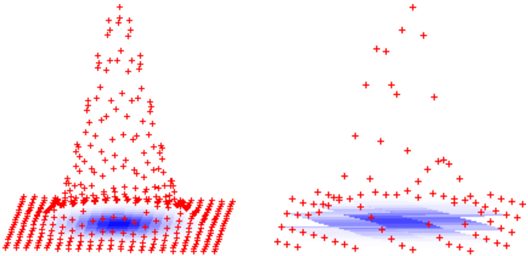
Discrete Gaussians

- Define the discrete Gaussian distribution over coset $\mathbf{c} + \mathcal{L}$ as

$$D_{\mathbf{c}+\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{c}+\mathcal{L})} \text{ for all } \mathbf{x} \in \mathbf{c}+\mathcal{L}.$$

- Consider the following experiment:
  1. Choose $\mathbf{x} \in \mathbb{Z}^n$ from $D_{\mathbb{Z}^n,s}$.
  2. Reveal coset $\mathbf{x} + \mathcal{L}$. (e.g., as $\bar{\mathbf{x}} = \mathbf{x} \bmod \mathbf{B}$ for some basis $\mathbf{B}$)

  Immediate facts:



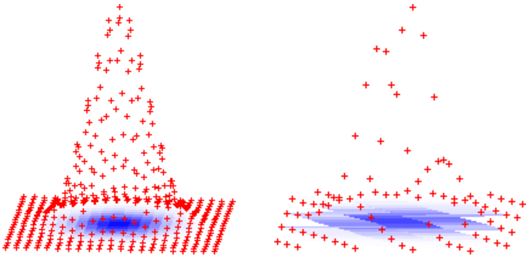
Discrete Gaussians

- Define the discrete Gaussian distribution over coset $\mathbf{c} + \mathcal{L}$ as

$$D_{\mathbf{c}+\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{c}+\mathcal{L})} \text{ for all } \mathbf{x} \in \mathbf{c} + \mathcal{L}.$$

- Consider the following experiment:
  1. Choose $\mathbf{x} \in \mathbb{Z}^n$ from $D_{\mathbb{Z}^n,s}$.
  2. Reveal coset $\mathbf{x} + \mathcal{L}$. (e.g., as $\bar{\mathbf{x}} = \mathbf{x} \bmod \mathbf{B}$ for some basis $\mathbf{B}$)

  Immediate facts:
  1. Every coset $\mathbf{c} + \mathcal{L}$ is equally* likely: we get uniform dist over $\mathbb{Z}^n/\mathcal{L}$.



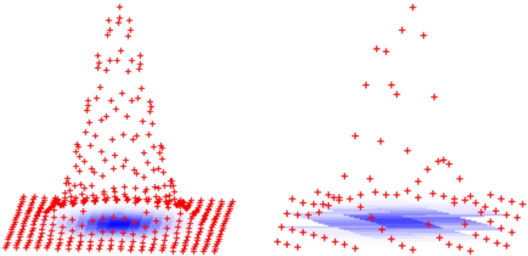
Discrete Gaussians

- Define the discrete Gaussian distribution over coset $\mathbf{c} + \mathcal{L}$ as

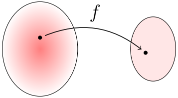
$$D_{\mathbf{c}+\mathcal{L},s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbf{c}+\mathcal{L})} \text{ for all } \mathbf{x} \in \mathbf{c} + \mathcal{L}.$$

- Consider the following experiment:
  1. Choose $\mathbf{x} \in \mathbb{Z}^n$ from $D_{\mathbb{Z}^n,s}$.
  2. Reveal coset $\mathbf{x} + \mathcal{L}$. (e.g., as $\bar{\mathbf{x}} = \mathbf{x} \bmod \mathbf{B}$ for some basis $\mathbf{B}$)

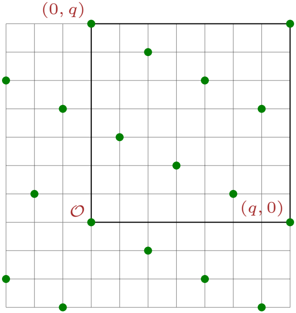
  Immediate facts:
  1. Every coset $\mathbf{c} + \mathcal{L}$ is equally* likely: we get uniform dist over $\mathbb{Z}^n/\mathcal{L}$.
  2. Given that $\mathbf{x} \in \mathbf{c} + \mathcal{L}$, it has conditional distribution $D_{\mathbf{c}+\mathcal{L},s}$.



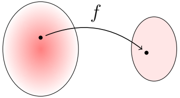
Preimage Sampleable TDF: Evaluation



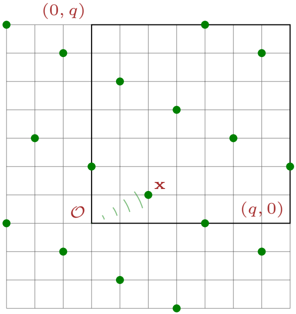
- ‘Hard’ description of $\mathcal{L}$ specifies f.
  Concretely: SIS matrix $\mathbf{A}$ defines $f_{\mathbf{A}}$.



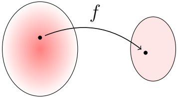
Preimage Sampleable TDF: Evaluation



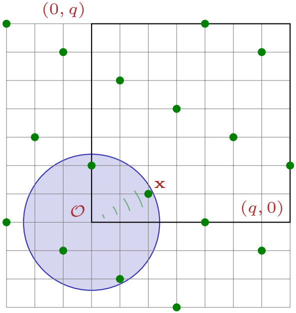
- 'Hard' description of $\mathcal{L}$ specifies f.
  Concretely: SIS matrix $\mathbf{A}$ defines $f_{\mathbf{A}}$.
- $f(\mathbf{x}) = \mathbf{x} \bmod \mathcal{L}$ for Gaussian $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$.
  Concretely: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = \mathbf{u} \in \mathbb{Z}_q^n$.



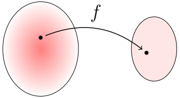
Preimage Sampleable TDF: Evaluation



- 'Hard' description of $\mathcal{L}$ specifies f.
  Concretely: SIS matrix $\mathbf{A}$ defines $f_{\mathbf{A}}$.
- $f(\mathbf{x}) = \mathbf{x} \bmod \mathcal{L}$ for Gaussian $\mathbf{x} \leftarrow D_{\mathbb{Z}^m, s}$.
  Concretely: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = \mathbf{u} \in \mathbb{Z}_q^n$.
- Inverting $f_{\mathbf{A}}$ $\Leftrightarrow$ decoding unif syndrome $\mathbf{u}$
  $\Leftrightarrow$ solving SIS.



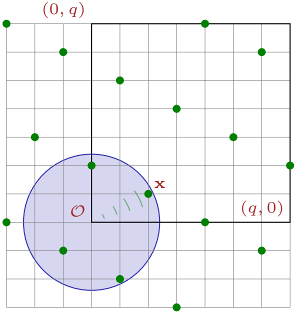
Preimage Sampleable TDF: Evaluation



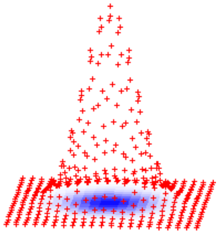
- 'Hard' description of $\mathcal{L}$ specifies f.
  Concretely: SIS matrix $\mathbf{A}$ defines $f_{\mathbf{A}}$.

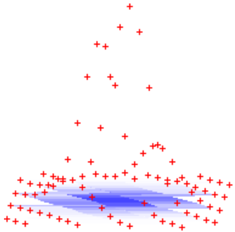
- $f(\mathbf{x}) = \mathbf{x} \bmod \mathcal{L}$ for Gaussian $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$.
  Concretely: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = \mathbf{u} \in \mathbb{Z}_q^n$.

- Inverting $f_{\mathbf{A}}$ $\Leftrightarrow$ decoding unif syndrome $\mathbf{u}$
  $\Leftrightarrow$ solving SIS.

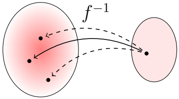


- Given $\mathbf{u}$, conditional distrib. of $\mathbf{x}$ is the discrete Gaussian $D_{\mathcal{L}_{\mathbf{u}}^{\perp}(\mathbf{A}),s}$.

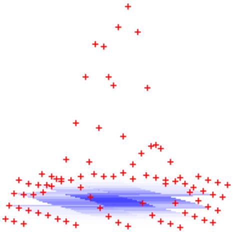




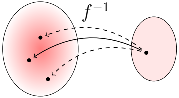
Preimage Sampling: Method #1



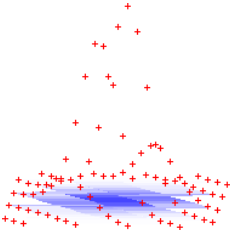
- Sample $D_{\mathcal{L}^\perp_{\mathbf{u}}(\mathbf{A}),s}$ given any short enough basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \le s$.
  - ⋆ Unlike [GGH'96], output leaks nothing about $\mathbf{S}$! (the bound s is public)

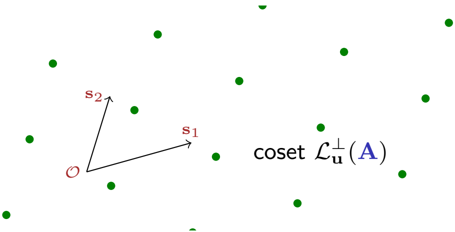


Preimage Sampling: Method #1

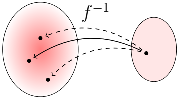


- Sample $D_{\mathcal{L}^{\perp}_{\mathbf{u}}(\mathbf{A}),s}$ given any short enough basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq s$.
  - ★ Unlike [GGH'96], output leaks nothing about $\mathbf{S}$! (the bound s is public)
- "Nearest-plane" algorithm with randomized rounding [Klein'00,GPV'08]



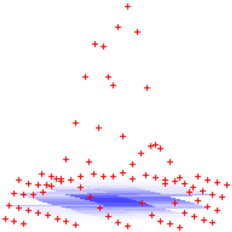


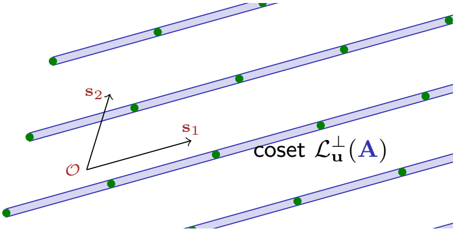
Preimage Sampling: Method #1



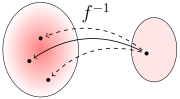
- Sample $D_{\mathcal{L}^{\perp}_{\mathbf{u}}(\mathbf{A}),s}$ given any short enough basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq s$.
  - ★ Unlike [GGH'96], output leaks nothing about $\mathbf{S}$! (the bound s is public)

- "Nearest-plane" algorithm with randomized rounding [Klein'00,GPV'08]



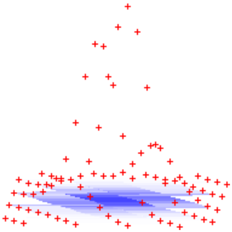


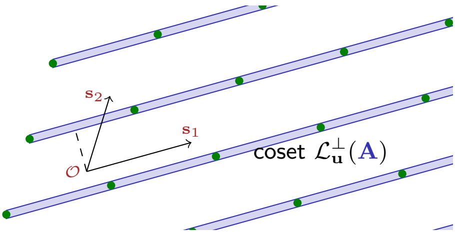
Preimage Sampling: Method #1



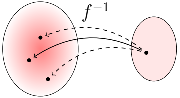
- Sample $D_{\mathcal{L}^{\perp}_{\mathbf{u}}(\mathbf{A}),s}$ given any short enough basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \le s$.
  - ⋆ Unlike [GGH'96], output leaks nothing about $\mathbf{S}$! (the bound s is public)

- "Nearest-plane" algorithm with randomized rounding [Klein'00,GPV'08]



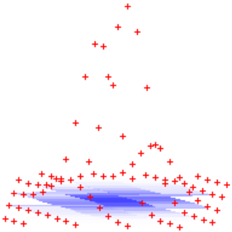


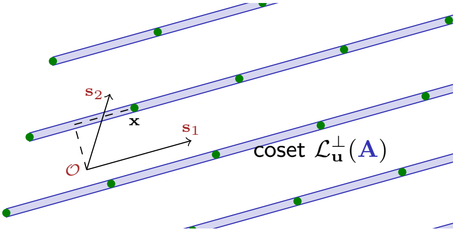
Preimage Sampling: Method #1



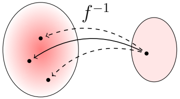
- Sample $D_{\mathcal{L}^{\perp}_{\mathbf{u}}(\mathbf{A}),s}$ given any short enough basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq s$.
  - ⋆ Unlike [GGH'96], output leaks nothing about $\mathbf{S}$! (the bound s is public)

- "Nearest-plane" algorithm with randomized rounding [Klein'00,GPV'08]



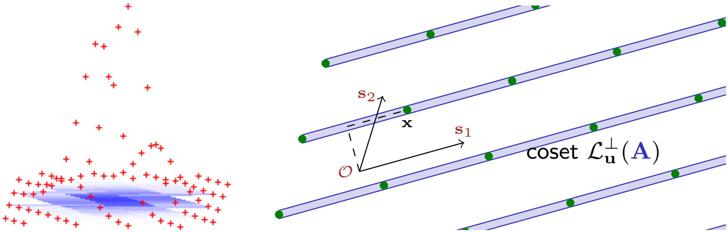


Preimage Sampling: Method #1



- Sample $D_{\mathcal{L}^{\perp}_{\mathbf{u}}(\mathbf{A}),s}$ given any short enough basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq s$.
  - ⋆ Unlike [GGH'96], output leaks nothing about $\mathbf{S}$! (the bound s is public)

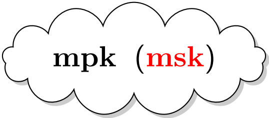
- "Nearest-plane" algorithm with randomized rounding [Klein'00,GPV'08]



- Proof idea: $\rho_s((\mathbf{c} + \mathcal{L}) \cap \text{plane})$ depends only on $\text{dist}(\mathbf{0}, \text{plane})$; essentially no dependence on shift within plane

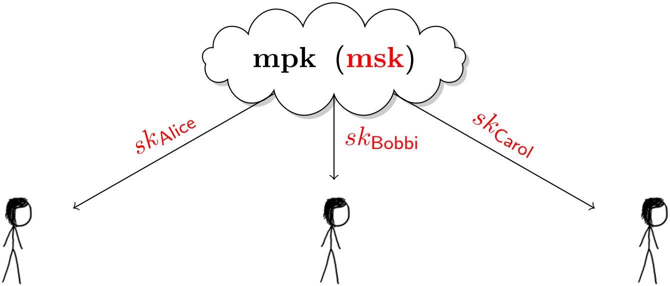
Identity-Based Encryption

- Proposed by [Shamir'84]: could this exist?



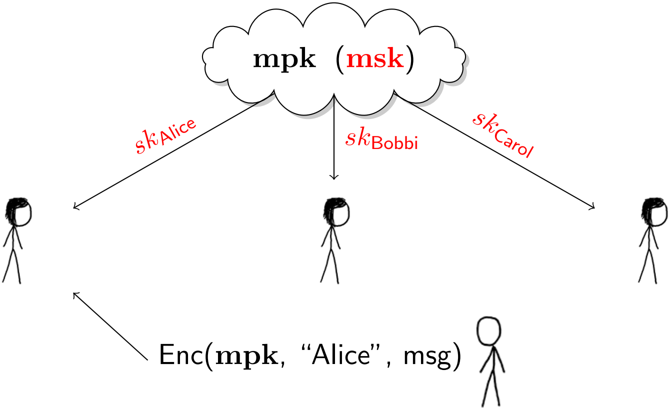
Identity-Based Encryption

- Proposed by [Shamir'84]: could this exist?



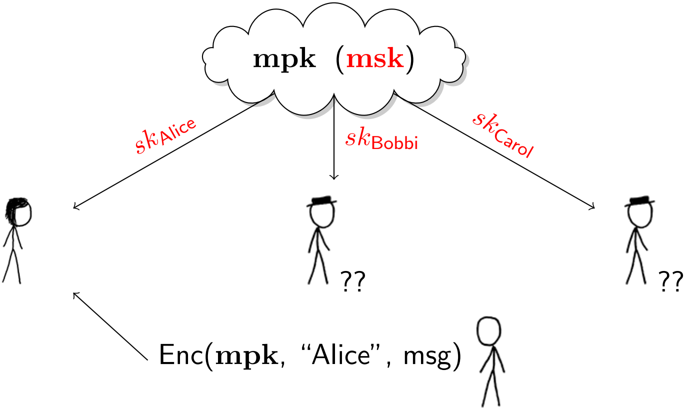
Identity-Based Encryption

- Proposed by [Shamir'84]: could this exist?



Identity-Based Encryption

- Proposed by [Shamir'84]: could this exist?



Fast-Forward 17 Years...

1. [BonehFranklin'01,...]: first IBE construction, using "new math"
   (elliptic curves w/ bilinear pairings)

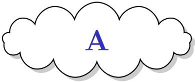
Fast-Forward 17 Years...

1. [BonehFranklin'01,...]: first IBE construction, using "new math"
   (elliptic curves w/ bilinear pairings)
2. [Cocks'01,BGH'07]: quadratic residuosity mod $N = pq$ [GM'82]

Fast-Forward 17 Years…

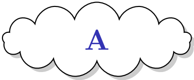
1. [BonehFranklin'01,…]: first IBE construction, using "new math" (elliptic curves w/ bilinear pairings)
2. [Cocks'01,BGH'07]: quadratic residuosity mod $N = pq$ [GM'82]
3. [GPV'08]: lattices!

Recall: 'Dual' LWE Cryptosystem



$\mathbf{x} \leftarrow$ Gauss

Recall: ‘Dual’ LWE Cryptosystem



$\mathbf{x} \leftarrow$ Gauss

$$\xrightarrow[\text{(public key)}]{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

Recall: ‘Dual’ LWE Cryptosystem

$\mathbf{A}$

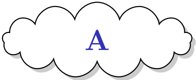
$\mathbf{x} \leftarrow$ Gauss

$\mathbf{s}, \mathbf{e}$

$$\xrightarrow[\text{(public key)}]{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

$$\xleftarrow[\text{(ciphertext ‘preamble’)}]{\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t}$$

Recall: ‘Dual’ LWE Cryptosystem



$\mathbf{x} \leftarrow$ Gauss

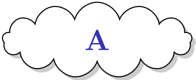
$\mathbf{s}, \mathbf{e}$

$$\xrightarrow[\text{(public key)}]{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

$$\xleftarrow[\text{(ciphertext `preamble')}]{\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t}$$

$$\xleftarrow[\text{(`payload')}]{b' = \mathbf{s}^t\,\mathbf{u} + e' + \text{bit} \cdot \frac{q}{2}}$$

Recall: ‘Dual’ LWE Cryptosystem



$\mathbf{A}$

$\mathbf{x} \leftarrow$ Gauss

$\mathbf{s}, \mathbf{e}$

$$\xrightarrow{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

(public key)

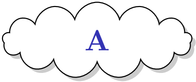
$$\xleftarrow{\mathbf{b}^t = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t}$$

(ciphertext ‘preamble’)

$$b' - \mathbf{b}^t \mathbf{x} \approx \text{bit} \cdot \frac{q}{2} \qquad \xleftarrow{b' = \mathbf{s}^t \mathbf{u} + e' + \text{bit} \cdot \frac{q}{2}}$$

(‘payload’)

Recall: 'Dual' LWE Cryptosystem



$\mathbf{x} \leftarrow$ Gauss

$\mathbf{s}, \mathbf{e}$

$$\xrightarrow[\text{(public key)}]{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

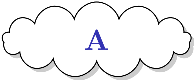
$$\xleftarrow[\text{(ciphertext `preamble')}]{\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t}$$

$b' - \mathbf{b}^t\mathbf{x} \approx \mathsf{bit} \cdot \frac{q}{2}$

$$\xleftarrow[\text{(`payload')}]{b' = \mathbf{s}^t\mathbf{u} + e' + \mathsf{bit} \cdot \frac{q}{2}}$$

? $(\mathbf{A}, \mathbf{u}, \mathbf{b}, b')$

Recall: ‘Dual’ LWE Cryptosystem



$\mathbf{x} \leftarrow$ Gauss

$\mathbf{s}, \mathbf{e}$

$$\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})$$
$$\xrightarrow{\hspace{6em}}$$
(public key)

$$\mathbf{b}^t = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t$$
$$\xleftarrow{\hspace{6em}}$$
(ciphertext ‘preamble’)

$$b' - \mathbf{b}^t \mathbf{x} \approx \text{bit} \cdot \frac{q}{2} \qquad b' = \mathbf{s}^t \mathbf{u} + e' + \text{bit} \cdot \frac{q}{2}$$
$$\xleftarrow{\hspace{6em}}$$
(‘payload’)

? $(\mathbf{A}, \mathbf{u}, \mathbf{b}, b')$

ID-Based Encryption

$\boxed{\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})}$

$mpk = \mathbf{A}$

$\mathbf{s}, \mathbf{e}$

$\boxed{\mathbf{u} = H(\text{“Alice”})}$

(‘identity’ public key)

$$\xleftarrow{\mathbf{b} = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t}$$

(ciphertext preamble)

$$b' - \mathbf{b}^t \mathbf{x} \approx \text{bit} \cdot \frac{q}{2} \qquad \xleftarrow{b' = \mathbf{s}^t \mathbf{u} + e' + \text{bit} \cdot \frac{q}{2}}$$

(‘payload’)

Tomorrow. . .

- Generating trapdoors ($\mathbf{A}$ with short basis or equivalent)

Tomorrow...

- Generating trapdoors ($\mathbf{A}$ with short basis or equivalent)
- Removing the random oracle from signatures & IBE

Tomorrow...

- Generating trapdoors ($\mathbf{A}$ with short basis or equivalent)
- Removing the random oracle from signatures & IBE
- More surprising applications

Tomorrow. . .

- Generating trapdoors ($\mathbf{A}$ with short basis or equivalent)
- Removing the random oracle from signatures & IBE
- More surprising applications

Selected bibliography for this talk:

MR'04 D. Micciancio and O. Regev, "Worst-Case to Average-Case Reductions Based on Gaussian Measures," FOCS'04 / SICOMP'07.

GPV'08 C. Gentry, C. Peikert, V. Vaikuntanathan, "Trapdoors for Hard Lattices and New Cryptographic Constructions," STOC'08.

P'10 C. Peikert, "An Efficient and Parallel Gaussian Sampler for Lattices," Crypto'10.

Bonus Material:

A Better
Discrete Gaussian Sampling
Algorithm

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime = $\Omega(n^3)$, high-precision arithmetic
- ✗ Inherently sequential: n adaptive iterations

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic
- ✗ Inherently sequential: n adaptive iterations
- ✗ No efficiency improvement in the ring setting [NTRU'98,M'02,...]

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic
- ✗ Inherently sequential: n adaptive iterations
- ✗ No efficiency improvement in the ring setting [NTRU'98,M'02,...]

A Different Sampling Algorithm [P'10]

- Simple & efficient: n^2 online adds and mults (mod q)

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic
- ✗ Inherently sequential: n adaptive iterations
- ✗ No efficiency improvement in the ring setting [NTRU'98,M'02,...]

A Different Sampling Algorithm [P'10]

- Simple & efficient: n^2 online adds and mults (mod q)

  Even better: $\tilde{O}(n)$ time in the ring setting

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic
- ✗ Inherently sequential: n adaptive iterations
- ✗ No efficiency improvement in the ring setting [NTRU'98,M'02,...]

A Different Sampling Algorithm [P'10]

- Simple & efficient: n^2 online adds and mults (mod q)
  Even better: $\tilde{O}(n)$ time in the ring setting
- Fully parallel: n^2/P operations on any $P \leq n^2$ processors

Performance of Nearest-Plane Sampling Algorithm?

Good News, and Bad News...

- ✔ Tight: std dev $s \approx \max\|\tilde{\mathbf{s}}_i\|$ = max dist between adjacent planes
- ✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic
- ✗ Inherently sequential: n adaptive iterations
- ✗ No efficiency improvement in the ring setting [NTRU'98,M'02,...]

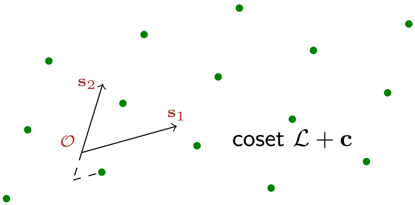
A Different Sampling Algorithm [P'10]

- Simple & efficient: n^2 online adds and mults (mod q)
  Even better: $\tilde{O}(n)$ time in the ring setting
- Fully parallel: n^2/P operations on any $P \leq n^2$ processors
- High quality: same* Gaussian std dev as nearest-plane alg

*in cryptographic applications

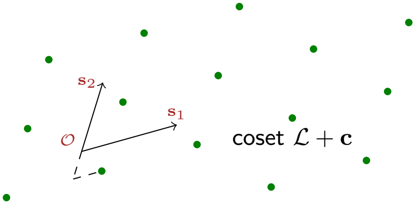
A First Attempt

- [Babai'86] "round-off:" $\mathbf{c} \mapsto \mathbf{S} \cdot \mathsf{frac}(\mathbf{S}^{-1} \cdot \mathbf{c})$. (Fast & parallel!)



A First Attempt

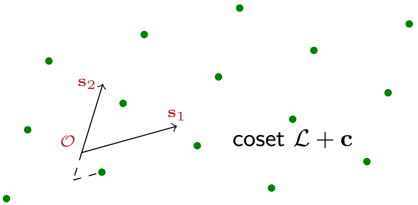
- [Babai'86] "round-off:" $\mathbf{c} \mapsto \mathbf{S} \cdot \mathsf{frac}(\mathbf{S}^{-1} \cdot \mathbf{c})$. (Fast & parallel!)
- Deterministic round-off is insecure [NR'06] ...



A First Attempt

- [Babai'86] "round-off:" $\mathbf{c} \mapsto \mathbf{S} \cdot \mathsf{frac}(\mathbf{S}^{-1} \cdot \mathbf{c})_{\$}$. (Fast & parallel!)
- Deterministic round-off is insecure [NR'06] ...

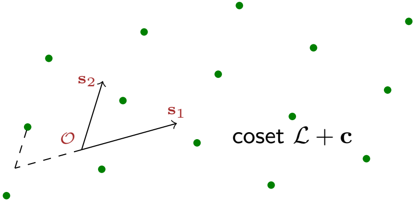
  ... but what about randomized rounding?

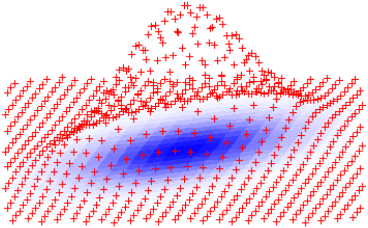


A First Attempt

- [Babai'86] "round-off:" $\mathbf{c} \mapsto \mathbf{S} \cdot \mathsf{frac}(\mathbf{S}^{-1} \cdot \mathbf{c})_{\$}$. (Fast & parallel!)
- Deterministic round-off is insecure [NR'06] ...

...but what about randomized rounding?

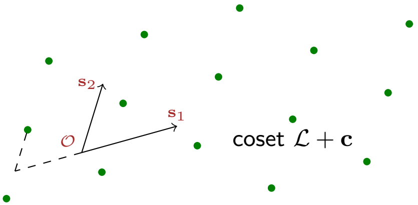


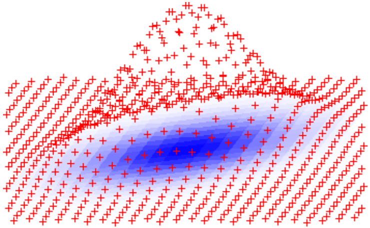


A First Attempt

- [Babai'86] "round-off:" $\mathbf{c} \mapsto \mathbf{S} \cdot \mathrm{frac}(\mathbf{S}^{-1} \cdot \mathbf{c})_{\$}$. (Fast & parallel!)
- Deterministic round-off is insecure [NR'06] ...

  ... but what about randomized rounding?





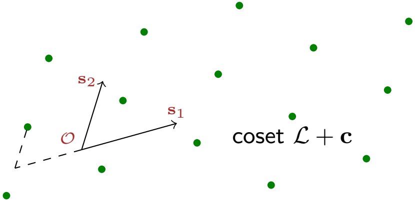
- Non-spherical discrete Gaussian: has covariance

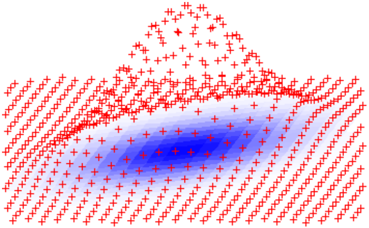
$$\Sigma := \mathbb{E}_{\mathbf{x}}\left[\mathbf{x} \cdot \mathbf{x}^t\right] \approx \mathbf{S} \cdot \mathbf{S}^t.$$

A First Attempt

- [Babai'86] "round-off:" $\mathbf{c} \mapsto \mathbf{S} \cdot \text{frac}(\mathbf{S}^{-1} \cdot \mathbf{c})_{\$}$. (Fast & parallel!)
- Deterministic round-off is insecure [NR'06] ...

  ... but what about randomized rounding?





- Non-spherical discrete Gaussian: has covariance

$$\Sigma := \mathbb{E}_{\mathbf{x}}\left[\mathbf{x} \cdot \mathbf{x}^t\right] \approx \mathbf{S} \cdot \mathbf{S}^t.$$

  Covariance can be measured — and it leaks $\mathbf{S}$! (up to rotation)

Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix Σ.

   (pos def means: $\mathbf{u}^t \, \Sigma \, \mathbf{u} > 0$ for all unit $\mathbf{u}$.)

Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix Σ.

   (pos def means: $\mathbf{u}^t \, \Sigma \, \mathbf{u} > 0$ for all unit $\mathbf{u}$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2 \, \mathbf{I}$.

Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix Σ.

   (pos def means: $\mathbf{u}^t\,\Sigma\,\mathbf{u} > 0$ for all unit $\mathbf{u}$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2\,\mathbf{I}$.

2. Convolution of Gaussians:

$$\Sigma_1 \quad + \quad \Sigma_2 \quad = \quad \Sigma = s^2\,\mathbf{I}$$

Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix Σ.

   (pos def means: $\mathbf{u}^t \, \Sigma \, \mathbf{u} > 0$ for all unit $\mathbf{u}$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2 \, \mathbf{I}$.

2. Convolution of Gaussians:

$$\Sigma_1 \quad + \quad \Sigma_2 \quad = \quad \Sigma = s^2 \, \mathbf{I}$$

3. Given Σ_1, how small can s be? For $\Sigma_2 := s^2 \, \mathbf{I} - \Sigma_1$,

Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix Σ.

   (pos def means: $\mathbf{u}^t\,\Sigma\,\mathbf{u} > 0$ for all unit $\mathbf{u}$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2\,\mathbf{I}$.

2. Convolution of Gaussians:

$$\Sigma_1 \quad + \quad \Sigma_2 \quad = \quad \Sigma = s^2\,\mathbf{I}$$

3. Given Σ_1, how small can s be? For $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1$,

$$\mathbf{u}^t\,\Sigma_2\,\mathbf{u} \;=\; s^2 - \mathbf{u}^t\,\Sigma_1\,\mathbf{u} > 0 \quad \Longleftrightarrow \quad \boxed{s^2 > \max \lambda_i(\Sigma_1)}$$

Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix Σ.

   (pos def means: $\mathbf{u}^t\,\Sigma\,\mathbf{u} > 0$ for all unit $\mathbf{u}$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2\,\mathbf{I}$.

2. Convolution of Gaussians:

$$\Sigma_1 \quad + \quad \Sigma_2 \quad = \quad \Sigma = s^2\,\mathbf{I}$$

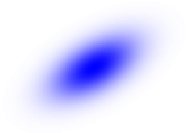
3. Given Σ_1, how small can s be? For $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1$,

$$\mathbf{u}^t\,\Sigma_2\,\mathbf{u} \;=\; s^2 - \mathbf{u}^t\,\Sigma_1\,\mathbf{u} > 0 \quad \Longleftrightarrow \quad \boxed{s^2 > \max \lambda_i(\Sigma_1)}$$

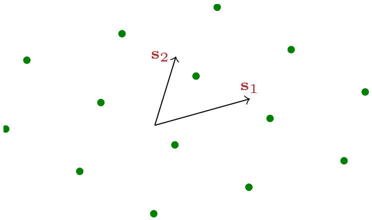
   For $\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$, can use any $\boxed{s > s_1(\mathbf{S}) := \text{max singular val of } \mathbf{S}.}$

'Convolution' Sampling Algorithm [P'10]

- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,

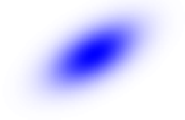


$$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$$

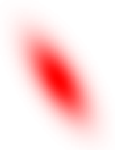


‘Convolution’ Sampling Algorithm [P’10]

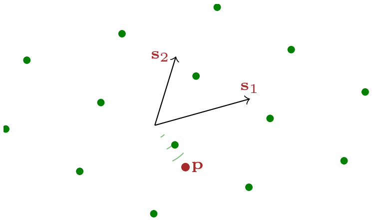
- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$

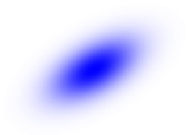


Σ_2

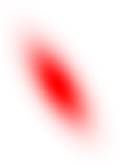


‘Convolution’ Sampling Algorithm [P’10]

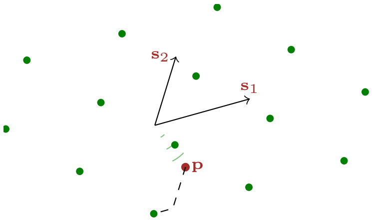
- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$
  2. Randomly round-off $\mathbf{p}$ to $\mathcal{L} + \mathbf{c}$: return $\mathbf{S} \cdot \text{frac}(\mathbf{S}^{-1} \cdot (\mathbf{c} + \mathbf{p}))_{\$}$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$

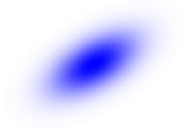


Σ_2

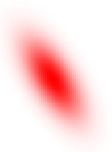


‘Convolution’ Sampling Algorithm [P’10]

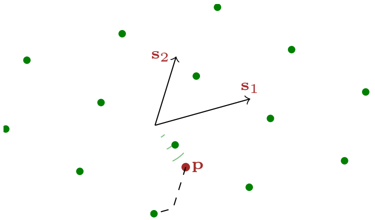
- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$
  2. Randomly round-off $\mathbf{p}$ to $\mathcal{L} + \mathbf{c}$: return $\mathbf{S} \cdot \mathrm{frac}(\mathbf{S}^{-1} \cdot (\mathbf{c} + \mathbf{p}))_{\$}$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$



Σ_2

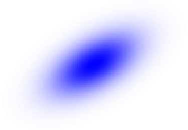


Convolution* Theorem

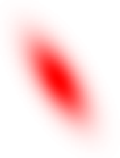
Algorithm generates a spherical discrete Gaussian over $\mathcal{L} + \mathbf{c}$.

'Convolution' Sampling Algorithm [P'10]

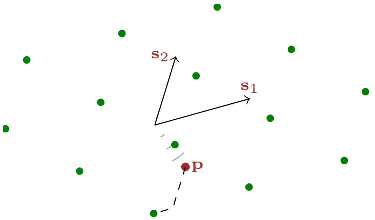
- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$
  2. Randomly round-off $\mathbf{p}$ to $\mathcal{L} + \mathbf{c}$: return $\mathbf{S} \cdot \mathsf{frac}(\mathbf{S}^{-1} \cdot (\mathbf{c} + \mathbf{p}))_{\$}$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$



Σ_2



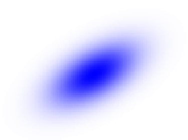
Convolution* Theorem

Algorithm generates a spherical discrete Gaussian over $\mathcal{L} + \mathbf{c}$.

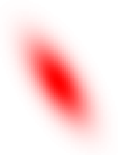
(*technically not a convolution, since step 2 depends on step 1.)

‘Convolution’ Sampling Algorithm [P’10]

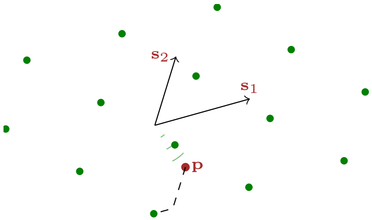
- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$
  2. Randomly round-off $\mathbf{p}$ to $\mathcal{L} + \mathbf{c}$: return $\mathbf{S} \cdot \text{frac}(\mathbf{S}^{-1} \cdot (\mathbf{c} + \mathbf{p}))_{\$}$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$



Σ_2

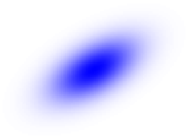


Optimizations

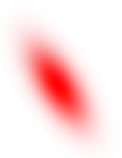
1. Precompute perturbations offline

‘Convolution’ Sampling Algorithm [P’10]

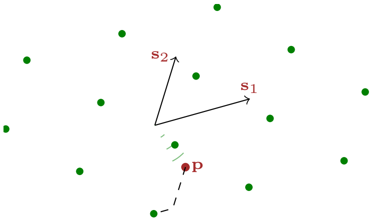
- Given basis $\mathbf{S}$, coset $\mathcal{L} + \mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$
  2. Randomly round-off $\mathbf{p}$ to $\mathcal{L} + \mathbf{c}$: return $\mathbf{S} \cdot \mathrm{frac}(\mathbf{S}^{-1} \cdot (\mathbf{c} + \mathbf{p}))_{\$}$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$



Σ_2

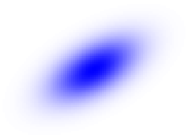


Optimizations

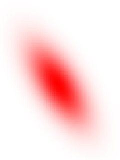
1. Precompute perturbations offline
2. Batch multi-sample using fast matrix multiplication

‘Convolution’ Sampling Algorithm [P’10]

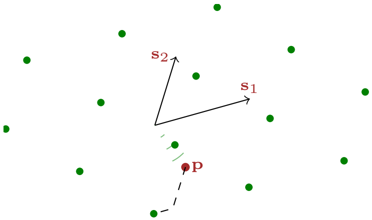
- Given basis $\mathbf{S}$, coset $\mathcal{L}+\mathbf{c}$, and std dev $s > s_1(\mathbf{S})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1 > 0$
  2. Randomly round-off $\mathbf{p}$ to $\mathcal{L}+\mathbf{c}$: return $\mathbf{S} \cdot \text{frac}(\mathbf{S}^{-1} \cdot (\mathbf{c}+\mathbf{p}))_{\$}$



$\Sigma_1 = \mathbf{S}\,\mathbf{S}^t$



Σ_2



Optimizations

1. Precompute perturbations offline
2. Batch multi-sample using fast matrix multiplication
3. More tricks & simplifications for SIS lattices (tomorrow)