Session #9: Trapdoors and Applications

Chris Peikert
Georgia Institute of Technology

Winter School on Lattice-Based Cryptography and Applications
Bar-Ilan University, Israel
19 Feb 2012 – 22 Feb 2012
Agenda

1. Lattices and short ‘trapdoor’ bases

2. Lattice-based ‘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

(public)

“I love you” ✓

(secret)

(Images courtesy xkcd.org)
Digital Signatures

(secret)

(public)

“It’s over” X

(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, . . .] (PSF)
Central Tool: Trapdoor Functions

- Public function \( f \) generated with secret ‘trapdoor’ \( f^{-1} \)
- Trapdoor permutation [DH’76, RSA’77, …] (PSF)
Central Tool: Trapdoor Functions

- Public function $f$ generated with secret ‘trapdoor’ $f^{-1}$
- Trapdoor permutation [DH’76, RSA’77, …] (PSF)
Central Tool: Trapdoor Functions

- Public function $f$ generated with secret ‘trapdoor’ $f^{-1}$
- Trapdoor permutation [DH’76, RSA’77, …] (PSF)

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

\[ \begin{align*}
D & \rightarrow f^{-1} \rightarrow D \\
\bullet & \rightarrow & \bullet
\end{align*} \]
Central Tool: Trapdoor Functions

- Public function $f$ generated with secret ‘trapdoor’ $f^{-1}$
- Trapdoor permutation [DH’76,RSA’77,…] (PSF)

\[ D \xrightarrow{x} f^{-1} \xrightarrow{y} D \]

- ‘Hash and sign:’ $pk = f$, $sk = f^{-1}$. $\text{Sign}(msg) = f^{-1}(H(msg))$.
- Candidate TDPs: [RSA’78,Rabin’79,Paillier’99] (‘general assumption’) All rely on hardness of factoring:
  - $\times$ Complex: 2048-bit exponentiation
  - $\times$ 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)

\[ D \xrightarrow{f} R \]

\[ f^{-1}(x) = y \]

'Hash and sign:'

\[ \text{pk} = f, \quad \text{sk} = f^{-1}. \text{Sign}(\text{msg}) = f^{-1}(H(\text{msg})) \]
Central Tool: Trapdoor Functions

- Public function $f$ generated with secret ‘trapdoor’ $f^{-1}$
- New twist [GPV’08]: preimage sampleable trapdoor function (PSF)
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:’ $pk = f$, $sk = f^{-1}$. $\text{Sign}(msg) = f^{-1}(H(msg))$. 
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) = \( f^{-1}(H(msg)) \).

Still secure! Can generate \((x, y)\) in two equivalent ways:

<table>
<thead>
<tr>
<th>REALITY</th>
<th>PROOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \leftarrow f^{-1} )</td>
<td>( x \leftarrow D \rightarrow f )</td>
</tr>
<tr>
<td>( y \leftarrow R )</td>
<td>( y \leftarrow y )</td>
</tr>
</tbody>
</table>
Part 1:

Constructing Preimage Sampleable Trapdoor Functions (PSFs)
Heuristic TDF & Signature Scheme \([\text{GGH}'96]\)

- **Key idea:** \(pk = \text{‘bad’ basis } B\) for \(L\), \(sk = \text{‘short’ trapdoor basis } S\)

Technical Issues

1. Generating ‘hard’ lattice together with short basis (later)
2. Signing algorithm leaks secret basis! ⋆
   - Total break after several signatures \([\text{NguyenRegev}'06]\)
Heuristic TDF & Signature Scheme \[GGH'96\]

- Key idea: \( pk = \) ‘bad’ basis \( \mathcal{B} \) for \( \mathcal{L} \), \( sk = \) ‘short’ trapdoor basis \( \mathcal{S} \)

- Sign \( H(\text{msg}) \in \mathbb{R}^n \) with “nearest-plane” algorithm \[Babai'86\]

\[\text{Diagram:}\]
Heuristic TDF & Signature Scheme [GGH’96]

- Key idea: \( pk = \text{‘bad’ basis } B \text{ for } L, \ sk = \text{‘short’ trapdoor basis } S \)
- Sign \( H(\text{msg}) \in \mathbb{R}^n \) with “nearest-plane” algorithm [Babai’86]
Heuristic TDF & Signature Scheme \cite{GGH96}

- Key idea: $pk = \text{‘bad’ basis } B \text{ for } L$, $sk = \text{‘short’ trapdoor basis } S$

- Sign $H(msg) \in \mathbb{R}^n$ with “nearest-plane” algorithm \cite{Babai86}

\[\text{s}_1\text{ and }\text{s}_2\]
Heuristic TDF & Signature Scheme [GGH’96]

- Key idea: $pk = \text{‘bad’ basis } B \text{ for } L, \ sk = \text{‘short’ trapdoor basis } S$
- Sign $H(\text{msg}) \in \mathbb{R}^n$ with “nearest-plane” algorithm [Babai’86]
Heuristic TDF & Signature Scheme [GGH'96]

- Key idea: $pk = \text{'bad' basis } B \text{ for } \mathcal{L}, sk = \text{'short' trapdoor basis } S$
- Sign $H(\text{msg}) \in \mathbb{R}^n$ with “nearest-plane” algorithm [Babai'86]
Heuristic TDF & Signature Scheme \[ \text{[GGH'96]} \]

- Key idea: \( pk = \) ‘bad’ basis \( B \) for \( L \), \( sk = \) ‘short’ trapdoor basis \( S \)
- Sign \( H(\text{msg}) \in \mathbb{R}^n \) with “nearest-plane” algorithm [Babai’86]
Heuristic TDF & Signature Scheme [GGH’96]

- Key idea: \( pk = \text{‘bad’ basis } \mathbb{B} \text{ for } \mathcal{L}, \ sk = \text{‘short’ trapdoor basis } \mathbb{S} \)
- Sign \( H(\text{msg}) \in \mathbb{R}^n \) with “nearest-plane” algorithm [Babai’86]
Heuristic TDF & Signature Scheme [GGH’96]

- Key idea: $pk = \text{‘bad’ basis } \mathcal{B} \text{ for } \mathcal{L}, \ sk = \text{‘short’ trapdoor basis } \mathcal{S}$
- Sign $H(\text{msg}) \in \mathbb{R}^n$ with “nearest-plane” algorithm [Babai’86]
Heuristic TDF & Signature Scheme [GGH’96]

- Key idea: $pk = \text{‘bad’ basis } B \text{ for } L, \ sk = \text{‘short’ trapdoor basis } S$
- Sign $H(\text{msg}) \in \mathbb{R}^n$ with “nearest-plane” algorithm [Babai’86]

Technical Issues

1. Generating ‘hard’ lattice together with short basis (later)
Heuristic TDF & Signature Scheme [GGH’96]

- Key idea: \( pk = \text{‘bad’ basis } B \text{ for } \mathcal{L}, sk = \text{‘short’ trapdoor basis } S \)
- Sign \( H(\text{msg}) \in \mathbb{R}^n \) with “nearest-plane” algorithm [Babai’86]

Technical Issues

1. Generating ‘hard’ lattice together with short basis (later)
2. Signing algorithm leaks secret basis!
   - ★ Total break after several signatures [NguyenRegev’06]
Blurring a Lattice

'Uniform' in $\mathbb{R}^n$ when standard deviation $\geq$ max length of some basis

▶ First used in worst/average-case reductions [Regev'03, MR'04, ...]

▶ Now an essential ingredient in many crypto schemes [GPV'08, ...]
Blurring a Lattice

Uniform' in $\mathbb{R}^n$ when std dev $\geq$ max length of some basis

▶ First used in worst/average-case reductions [Regev'03, MR'04, ...]

▶ Now an essential ingredient in many crypto schemes [GPV'08, ...]
Blurring a Lattice
Blurring a Lattice

‘Uniform’ in $\mathbb{R}^n$ when \( \text{std dev} \geq \text{max length of some basis} \)
Blurring a Lattice

Gaussian mod $\mathcal{L}$ is uniform when $\text{std dev} \geq \text{max length of some basis}$
Blurring a Lattice

Gaussian mod $\mathcal{L}$ is uniform when $\text{std dev} \geq \text{max length of some basis}$

- First used in worst/average-case reductions [Regev’03, MR’04, …]
Blurring a Lattice

Gaussian mod $\mathcal{L}$ is uniform when std dev $\geq \text{max length of some basis}$

- First used in worst/average-case reductions [Regev'03, MR'04, ...]
- Now an essential ingredient in many crypto schemes [GPV'08, ...]
‘Hard’ description of $\mathcal{L}$ specifies $f$.
Concretely: SIS matrix $A$ defines $f_A$.
Preimage Sampleable TDF: Evaluation

- ‘Hard’ description of \( \mathcal{L} \) specifies \( f \).
  Concretely: SIS matrix \( A \) defines \( f_A \).

- \( f(x) = x \mod \mathcal{L} \) for Gaussian \( x \).
  Concretely: \( f_A(x) = Ax = u \in \mathbb{Z}_q^n \).

Inverting \( \iff \) decoding syndrome \( u \iff \) solving SIS.
Preimage Sampleable TDF: Evaluation

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

- $f(x) = x \mod \mathcal{L}$ for Gaussian $x$.
  Concretely: $f_A(x) = Ax = u \in \mathbb{Z}_q^n$.

- Inverting $\iff$ decoding syndrome $u$ $\iff$ solving SIS.
‘Hard’ description of $\mathcal{L}$ specifies $f$.
Concretely: SIS matrix $A$ defines $f_A$.

$f(x) = x \mod \mathcal{L}$ for Gaussian $x$.
Concretely: $f_A(x) = Ax = u \in \mathbb{Z}_q^n$.

Inverting $\iff$ decoding syndrome $u$ $\iff$ solving SIS.

Given $u$, conditional distrib. of $x$ is the discrete Gaussian $D_{\mathcal{L}u}$.
Preimage Sampling: Method \#1

- Sample $D_{L_u}$ given any ‘short enough’ basis $S$: $\max \|s_i\| \leq \text{std dev}$

  - Unlike [GGH’96], output distribution leaks no information about $S$!
Preimage Sampling: Method #1

- Sample $D_{L_u}$ given any ‘short enough’ basis $S$: $\max \|s_i\| \leq \text{std dev}$
  - Unlike [GGH’96], output distribution leaks no information about $S$

- “Nearest-plane” algorithm with randomized rounding [Klein’00,GPV’08]
Preimage Sampling: Method #1

- Sample $D_{L_u}$ given any ‘short enough’ basis $S$: $\max ||s_i|| \leq \text{std dev}$
  - Unlike [GGH’96], output distribution leaks no information about $S$!

- “Nearest-plane” algorithm with randomized rounding [Klein’00,GPV’08]
Preimage Sampling: Method #1

- Sample $D_{\mathcal{L}_u}$ given any ‘short enough’ basis $S$: $\max \|s_i\| \leq \text{std dev}$
  - Unlike [GGH’96], output distribution leaks no information about $S$!

- “Nearest-plane” algorithm with randomized rounding [Klein’00,GPV’08]
Preimage Sampling: Method #1

- Sample $D_{L_u}$ given any ‘short enough’ basis $S$: $\max ||s_i|| \leq \text{std dev}$
  - Unlike [GGH’96], output distribution leaks no information about $S$!

- “Nearest-plane” algorithm with randomized rounding [Klein’00,GPV’08]
Preimage Sampling: Method #1

Sample $D_{L_u}$ given any 'short enough' basis $S$: $\max \|s_i\| \leq \text{std dev}$

Unlike [GGH'96], output distribution leaks no information about $S$!

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

Proof idea: $D_{L_u}(\text{plane})$ depends only on $\text{dist}(0, \text{plane})$; not affected by shift within plane
Performance of Nearest-Plane Method?

Good News, and Bad News...

✓ **Tight**: std dev $\approx \max \|\tilde{s}_i\| = \max$ dist between adjacent planes
Performance of Nearest-Plane Method?

**Good News, and Bad News. . .**

- **✓** Tight: std dev $\approx \max \|\tilde{s}_i\| = \max$ dist between adjacent planes
- **✗** Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic
Performance of Nearest-Plane Method?

<table>
<thead>
<tr>
<th>Good News, and Bad News...</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Tight: std dev $\approx \max |\tilde{S}_i| = \max$ dist between adjacent planes</td>
</tr>
<tr>
<td>✗ Not efficient: runtime $= \Omega(n^3)$, high-precision arithmetic</td>
</tr>
<tr>
<td>✗ Inherently sequential: $n$ adaptive iterations</td>
</tr>
</tbody>
</table>
Performance of Nearest-Plane Method?

Good News, and Bad News...

- **Tight**: \( \text{std dev} \approx \max \| \tilde{s}_i \| = \max \text{dist between adjacent planes} \)
- **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 Method?

### Good News, and Bad News...

- **✓** Tight: $\text{std dev} \approx \max \|\tilde{S}_i\| = \max \text{dist between adjacent planes}$
- **✗** 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 Method?

**Good News, and Bad News...**

- **Tight:** \( \text{std dev} \approx \max \| \tilde{s}_i \| = \max \text{dist between adjacent planes} \)
- **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 Method?

<table>
<thead>
<tr>
<th>Good News, and Bad News...</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Tight: std dev ≈ max ∥\tilde{s}_i∥ = max dist between adjacent planes</td>
</tr>
<tr>
<td>✗ Not efficient: runtime = \Omega(n^3), high-precision arithmetic</td>
</tr>
<tr>
<td>✗ Inherently sequential: n adaptive iterations</td>
</tr>
<tr>
<td>✗ No efficiency improvement in the ring setting [NTRU'98,M'02,...]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Different Sampling Algorithm [P'10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Simple &amp; efficient: (n^2) online adds and mults (mod (q))</td>
</tr>
<tr>
<td>Even better: (\tilde{O}(n)) time in the ring setting</td>
</tr>
<tr>
<td>▶ Fully parallel: (n^2/P) operations on any (P \leq n^2) processors</td>
</tr>
</tbody>
</table>
Performance of Nearest-Plane Method?

Good News, and Bad News...

- **Tight:** $\text{std dev} \approx \max \|\tilde{s}_i\| = \max \text{dist between adjacent planes}$
- **Not efficient:** $\text{runtime} = \Omega(n^3)$, 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] ‘simple rounding:’ $c \mapsto S \cdot \text{frac}(S^{-1} \cdot c)$ . (Fast & parallel!)

\[\text{coset } \mathcal{L} + c\]
A First Attempt

- [Babai’86] ‘simple rounding:’ $c \mapsto S \cdot \text{frac}(S^{-1} \cdot c)$ . (Fast & parallel!)
- **Deterministic** rounding is **insecure** [NR’06] . . .
A First Attempt

- [Babai’86] ‘simple rounding:’ $c \mapsto S \cdot \text{frac}(S^{-1} \cdot c)_S$. (Fast & parallel!)
- Deterministic rounding is insecure [NR’06] …
  … but what about randomized rounding?

\[ \text{coset } \mathcal{L} + c \]
A First Attempt

- [Babai’86] ‘simple rounding:’ $c \mapsto S \cdot \frac{S^{-1} \cdot c}{\|S^{-1} \cdot c\|}$. (Fast & parallel!)
- Deterministic rounding is insecure [NR’06] . . .

... but what about randomized rounding?

Non-spherical discrete Gaussian: has covariance

\[ \Sigma := E_{x}[x \cdot x^t] \approx S \cdot S^t. \]

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

coset $\mathcal{L} + c$
A First Attempt

- [Babai’86] ‘simple rounding:’  $c \mapsto S \cdot \text{frac}(S^{-1} \cdot c)_S$.  (Fast & parallel!)
- Deterministic rounding is **insecure** [NR’06] . . .
  
  . . . but what about **randomized** rounding?

- **Non-spherical** discrete Gaussian: has **covariance**

  $$
  \Sigma := \mathbb{E}_x \left[ x \cdot x^t \right] \approx S \cdot S^t.
  $$
A First Attempt

- [Babai’86] ‘simple rounding:’ \( c \mapsto S \cdot \text{frac}(S^{-1} \cdot c) \). (Fast & parallel!)
- Deterministic rounding is insecure [NR’06] . . .
  
  . . . but what about randomized rounding?

- Non-spherical discrete Gaussian: has covariance

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

Covariance can be measured — and it leaks \( S \)! (up to rotation)
Inspiration: Some Facts About Gaussians

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

(pos def means: $u^T \Sigma u > 0$ for all unit $u$.)

Lattice-Based Crypto & Applications, Bar-Ilan University, Israel 2012 12/19
Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix $\Sigma$.
   (pos def means: $u^t \Sigma u > 0$ for all unit $u$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2 I$. 

Lattice-Based Crypto & Applications, Bar-Ilan University, Israel 2012 12/19
Inspiration: Some Facts About Gaussians

1 Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix $\Sigma$.
   (pos def means: $u^t \Sigma u > 0$ for all unit $u$.)

Spherical Gaussian $\leftrightarrow$ covariance $s^2 I$.

2 Convolution of Gaussians:

$$\Sigma_1 + \Sigma_2 = \Sigma = s^2 I$$
Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix $\Sigma$.
   (pos def means: $u^t \Sigma u > 0$ for all unit $u$.)

   Spherical Gaussian $\leftrightarrow$ covariance $s^2 I$.

2. Convolution of Gaussians:

   $$\Sigma_1 + \Sigma_2 = \Sigma = s^2 I$$

3. Given $\Sigma_1$, how small can $s$ be? For $\Sigma_2 := s^2 I - \Sigma_1$,
Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix $\Sigma$.
   
   (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:

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

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

   $\mathbf{u}^T \Sigma_2 \mathbf{u} = s^2 - \mathbf{u}^T \Sigma_1 \mathbf{u} > 0 \iff s^2 > \max \lambda_i(\Sigma_1)$
Inspiration: Some Facts About Gaussians

1. Continuous Gaussian $\leftrightarrow$ positive definite covariance matrix $\Sigma$.
   
   (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:

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

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

   $\mathbf{u}^T \Sigma_2 \mathbf{u} = s^2 - \mathbf{u}^T \Sigma_1 \mathbf{u} > 0$ $\iff$ $s^2 > \max \lambda_i(\Sigma_1)$

   For $\Sigma_1 = \mathbf{S} \mathbf{S}^t$, can use any $s > s_1(\mathbf{S}) := \max$ singular val of $\mathbf{S}$. 
‘Convolution’ Sampling Algorithm [P’10]

- Given basis $S$, coset $L + c$, and std dev $s > s_1(S)$,

$$\Sigma_1 = S S^t$$
‘Convolution’ Sampling Algorithm [P’10]

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

\[
\Sigma_1 = \mathbf{S} \mathbf{S}^t \quad \Sigma_2
\]
‘Convolution’ Sampling Algorithm [P’10]

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

1. Generate perturbation $p$ with covariance $\Sigma_2 := s^2 I - \Sigma_1 > 0$
2. Randomly simple-round $p$ to $\mathcal{L} + c$

\[
\Sigma_1 = SS^t \quad \Sigma_2
\]
‘Convolution’ Sampling Algorithm [P’10]

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

1. Generate perturbation $p$ with covariance $\Sigma_2 := s^2 I - \Sigma_1 > 0$
2. Randomly simple-round $p$ to $\mathcal{L} + c$

$\Sigma_1 = SS^t$  \hspace{1cm} $\Sigma_2$

**Convolution* Theorem**

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

---

Lattice-Based Crypto & Applications, Bar-Ilan University, Israel 2012 13/19
‘Convolution’ Sampling Algorithm [P’10]

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

$$\Sigma_1 = S S^t \quad \Sigma_2$$

**Convolution* Theorem**

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

(*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 simple-round $\mathbf{p}$ to $\mathcal{L} + \mathbf{c}$

\[
\Sigma_1 = \mathbf{S} \mathbf{S}^t \quad \Sigma_2
\]

Optimizations

1. Precompute perturbations offline
‘Convolution’ Sampling Algorithm [P’10]

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

\[ \Sigma_1 = SS^t \quad \Sigma_2 \]

Optimizations

1. Precompute perturbations offline
2. Batch multi-sample using fast matrix multiplication
‘Convolution’ Sampling Algorithm [P’10]

Given basis \( \mathbf{S} \), coset \( \mathbf{L} + \mathbf{c} \), 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 simple-round \( \mathbf{p} \) to \( \mathbf{L} + \mathbf{c} \)

\[
\Sigma_1 = \mathbf{S} \mathbf{S}^t \quad \Sigma_2
\]

Optimizations

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

Identity-Based Encryption
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?

\[
\text{Enc}(\text{mpk}, \text{"Alice"}, \text{msg})
\]
Identity-Based Encryption

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

```
Enc(mpk, "Alice", msg)
```

```
mpk (msk)
```

```
mpk
```

```
sk_Alice
```

```
sk_Bobbi
```

```
sk_Carol
```

```
??
```

```
??
```

```
??
```
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

\[ \mathbf{x} \leftarrow \text{Gauss} \]
Recall: ‘Dual’ LWE Cryptosystem

\[ x \leftarrow \text{Gauss} \]

\[ u =Ax = f_A(x) \]

(public key)
Recall: ‘Dual’ LWE Cryptosystem

\[ x \leftarrow \text{Gauss} \]

\[ u = Ax = f_A(x) \]

(public key)

\[ b^t = s^tA + e^t \]

(ciphertext ‘preamble’)

\[ s, e \]
Recall: ‘Dual’ LWE Cryptosystem

\[ \mathbf{x} \leftarrow \text{Gauss} \]

\[ \mathbf{u} = \mathbf{A} \mathbf{x} = f_{\mathbf{A}}(\mathbf{x}) \]

(public key)

\[ \mathbf{b}^t = s^t \mathbf{A} + \mathbf{e}^t \]

(ciphertext ‘preamble’)

\[ \mathbf{b}' = s^t \mathbf{u} + \mathbf{e}' + \text{bit} \cdot \frac{q}{2} \]

('payload')

\[ \mathbf{s}, \mathbf{e} \]
Recall: ‘Dual’ LWE Cryptosystem

\[ x \leftarrow \text{Gauss} \]

\[ \mathbf{u} = A\mathbf{x} = f_A(x) \]

(public key)

\[ b^t = s^tA + e^t \]

(ciphertext ‘preamble’)

\[ b' - b^t x \approx \text{bit} \cdot \frac{q}{2} \]

\[ b' = s^t \mathbf{u} + e' + \text{bit} \cdot \frac{q}{2} \]

('payload')
Recall: ‘Dual’ LWE Cryptosystem

\[ \mathbf{x} \leftarrow \text{Gauss} \]

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

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

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

? (\(\mathbf{A}, \mathbf{u}, \mathbf{b}, \mathbf{b}'\))
Recall: ‘Dual’ LWE Cryptosystem

\[ x \leftarrow \text{Gauss} \]

\[ u = Ax = f_A(x) \]

(public key)

\[ b^t = s^tA + e^t \]

(ciphertext ‘preamble’)

\[ b' = s^t u + e' + \text{bit} \cdot \frac{q}{2} \]

('payload')

\[ b' - b^t x \approx \text{bit} \cdot \frac{q}{2} \]

? \( (A, u, b, b') \)
ID-Based Encryption

\[ mpk = A \]

\[ x \leftarrow f_A^{-1}(u) \]

\[ u = H(“Alice”) \]

\[ b = s^t A + e^t \]

\[ b' = s^t u + e' + \text{bit} \cdot \frac{q}{2} \]

\[ b' - b^t x \approx \text{bit} \cdot \frac{q}{2} \]
When We Come Back . . .

- Generating trapdoors (A with short basis)
When We Come Back... 

- Generating trapdoors ($A^\perp$ with short basis)
- Removing the random oracle from signatures & IBE
When We Come Back...

- Generating trapdoors (A with short basis)
- Removing the random oracle from signatures & IBE
- More surprising applications
When We Come Back... 

- Generating trapdoors (A with short basis)
- 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.
