Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller

Daniele Micciancio[1] Chris Peikert[2]

[1]UC San Diego

[2]Georgia Tech

CRYPTO Rump Session
16 Aug 2011

Lattice-Based One-Way Functions

- Public key $\begin{bmatrix} \cdots \mathbf{A} \cdots \end{bmatrix} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $q = \mathrm{poly}(n)$, $m = O(n \log q)$

Lattice-Based One-Way Functions

- Public key $\begin{bmatrix} \cdots \mathbf{A} \cdots \end{bmatrix} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $q = \text{poly}(n)$, $m = O(n \log q)$

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$
(surjective)	(injective)
OWF if SIS hard [Ajtai'96]	OWF if LWE hard [Regev'05]

Lattice-Based One-Way Functions

- Public key $\begin{bmatrix} \cdots \mathbf{A} \cdots \end{bmatrix} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ for $q = \text{poly}(n)$, $m = O(n \log q)$

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$
(surjective)	(injective)
OWF if SIS hard [Ajtai'96]	OWF if LWE hard [Regev'05]

- $f_{\mathbf{A}}$, $g_{\mathbf{A}}$ in forward direction yields CRHFs, IND-CPA encryption

(. . . and not much else)

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:

find the unique preimage $\mathbf{s}, \mathbf{e}$

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$:	Invert $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$	find the unique preimage $\mathbf{s}, \mathbf{e}$

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$:	Invert $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$	find the unique preimage $\mathbf{s}, \mathbf{e}$

- How? Use a "strong trapdoor" for $\mathbf{A}$: a short basis

[Babai'86,GGH'97,Klein'01,GPV'08,P'10]

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$:	Invert $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$	find the unique preimage $\mathbf{s}, \mathbf{e}$

- How? Use a "strong trapdoor" for $\mathbf{A}$: a short basis

  [Babai'86,GGH'97,Klein'01,GPV'08,P'10]

- Crypto applications: [GPV'08, PW'08, PV'08, PVW'08, P'09, CHKP'10, R'10, ABB'10a, GHV'10, B'10, ABB'10b, GKV'10, BF'11a, BF'11b, OPW'11, ...]

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$:	Invert $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$	find the unique preimage $\mathbf{s}, \mathbf{e}$

- How? Use a "strong trapdoor" for $\mathbf{A}$: a short basis
  [Babai'86,GGH'97,Klein'01,GPV'08,P'10]

- Crypto applications: [GPV'08, PW'08, PV'08, PVW'08, P'09, CHKP'10, R'10, ABB'10a, GHV'10, B'10, ABB'10b, GKV'10, BF'11a, BF'11b, OPW'11, . . .]

Some Practical Drawbacks. . .

✗ Generating $\mathbf{A}$ with short basis is complicated & slow [Ajtai'99,AP'09]

Trapdoor Inversion

- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$:	Invert $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$	find the unique preimage $\mathbf{s}, \mathbf{e}$

- How? Use a "strong trapdoor" for $\mathbf{A}$: a short basis

  [Babai'86,GGH'97,Klein'01,GPV'08,P'10]

- Crypto applications: [GPV'08, PW'08, PV'08, PVW'08, P'09, CHKP'10, R'10, ABB'10a, GHV'10, B'10, ABB'10b, GKV'10, BF'11a, BF'11b, OPW'11, . . .]

Some Practical Drawbacks. . .

✗ Generating $\mathbf{A}$ with short basis is complicated & slow [Ajtai'99,AP'09]

✗ Inversion algorithms either are sequential & need bigints, or are for suboptimal dimension m and preimage "quality."

Our Contributions

New trapdoor generation and inversion algorithms:

Our Contributions

New trapdoor generation and inversion algorithms:

- ✔ Much, much <u>simpler & faster</u>
  - ⋆ To generate: one matrix mult.
  - ⋆ To invert $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: efficient, highly parallel, & mostly offline

Our Contributions

New trapdoor generation and inversion algorithms:

- ✔ Much, much simpler & faster
  - ⋆ To generate: one matrix mult.
  - ⋆ To invert $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: efficient, highly parallel, & mostly offline
- ✔ Tighter, more secure parameters
  - ⋆ Asymptotically optimal with small constant factors
  - ⋆ Improvements: 8x in dim m, 112x in "quality" $\Rightarrow$ 50x in keysize

Our Contributions

New trapdoor generation and inversion algorithms:

- ✔ Much, much simpler & faster
  - ⋆ To generate: one matrix mult.
  - ⋆ To invert $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: efficient, highly parallel, & mostly offline
- ✔ Tighter, more secure parameters
  - ⋆ Asymptotically optimal with small constant factors
  - ⋆ Improvements: 8x in dim m, 112x in “quality” $\Rightarrow$ 50x in keysize
- ✔ New trapdoor notion (not a basis!): 4x smaller, easier delegation

Our Contributions

New trapdoor generation and inversion algorithms:

- ✔ Much, much simpler & faster
  - ⋆ To generate: one matrix mult.
  - ⋆ To invert $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: efficient, highly parallel, & mostly offline
- ✔ Tighter, more secure parameters
  - ⋆ Asymptotically optimal with small constant factors
  - ⋆ Improvements: 8x in dim m, 112x in “quality” $\Rightarrow$ 50x in keysize
- ✔ New trapdoor notion (not a basis!): 4x smaller, easier delegation
- ✔ More efficient applications beyond “black box” improvements:
  - ⋆ CCA encryption with smaller keys (subsumes [PW’08,P’09,ABB’10a])
  - ⋆ Short, standard-model signatures (improves [CHKP’10,R’10,B’10])

Trapdoor Generation and Algorithms

1. Start from a special (fixed, public) lattice defined by $\mathbf{G}$.

   - Give very fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$
   - Concretely,

$$\mathbf{G} = \mathbf{I}_n \otimes [1, 2, 4, \ldots, \tfrac{q}{2}] = \begin{bmatrix} 1\ 2\ \cdots\ \frac{q}{2} & & & \\ & 1\ 2\ \cdots\ \frac{q}{2} & & \\ & & \ddots & \\ & & & 1\ 2\ \cdots\ \frac{q}{2} \end{bmatrix}$$

Trapdoor Generation and Algorithms

1. Start from a special (fixed, public) lattice defined by $\mathbf{G}$.
   - ★ Give very fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$
   - ★ Concretely,

$$\mathbf{G} = \mathbf{I}_n \otimes [1, 2, 4, \ldots, \tfrac{q}{2}] = \begin{bmatrix} 1\ 2\ \cdots\ \frac{q}{2} & & & \\ & 1\ 2\ \cdots\ \frac{q}{2} & & \\ & & \ddots & \\ & & & 1\ 2\ \cdots\ \frac{q}{2} \end{bmatrix}$$

2. Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$ with a "nice" unimodular transformation:

$$\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G}] \begin{bmatrix} \mathbf{I} & \mathbf{R} \\ & \mathbf{I} \end{bmatrix} = [\bar{\mathbf{A}} \mid \mathbf{G} + \bar{\mathbf{A}}\mathbf{R}]$$

Trapdoor Generation and Algorithms

1. Start from a special (fixed, public) lattice defined by $\mathbf{G}$.
   - Give very fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$
   - Concretely,

$$\mathbf{G} = \mathbf{I}_n \otimes [1, 2, 4, \ldots, \tfrac{q}{2}] = \begin{bmatrix} 1\ 2\ \cdots\ \frac{q}{2} & & & \\ & 1\ 2\ \cdots\ \frac{q}{2} & & \\ & & \ddots & \\ & & & 1\ 2\ \cdots\ \frac{q}{2} \end{bmatrix}$$

2. Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$ with a "nice" unimodular transformation:

$$\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G}] \begin{bmatrix} \mathbf{I} & \mathbf{R} \\ & \mathbf{I} \end{bmatrix} = [\bar{\mathbf{A}} \mid \mathbf{G} + \bar{\mathbf{A}}\mathbf{R}]$$

3. Efficiently reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$ with pre-/post-processing

Trapdoor Generation and Algorithms

1. Start from a special (fixed, public) lattice defined by $\mathbf{G}$.
   - Give very fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$
   - Concretely,

$$\mathbf{G} = \mathbf{I}_n \otimes [1, 2, 4, \ldots, \tfrac{q}{2}] = \begin{bmatrix} 1\ 2\ \cdots\ \frac{q}{2} & & & \\ & 1\ 2\ \cdots\ \frac{q}{2} & & \\ & & \ddots & \\ & & & 1\ 2\ \cdots\ \frac{q}{2} \end{bmatrix}$$

2. Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$ with a "nice" unimodular transformation:

$$\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G}] \begin{bmatrix} \mathbf{I} & \mathbf{R} \\ & \mathbf{I} \end{bmatrix} = [\bar{\mathbf{A}} \mid \mathbf{G} + \bar{\mathbf{A}}\mathbf{R}]$$

3. Efficiently reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$ with pre-/post-processing

Coming very soon to an eprint near you. . .