Lattice-Based Cryptography: Constructing Trapdoors and More Applications

Chris Peikert
Georgia Institute of Technology

crypt@b-it 2013

Lattice-Based One-Way Functions

- Public key $\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix} \in \mathbb{Z}_q^{n\times m}$ for $q = \mathsf{poly}(n)$, $m = \Omega(n \log q)$.

Lattice-Based One-Way Functions

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

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

("short" $\mathbf{x}$, surjective)

CRHF if SIS hard [Ajtai'96,...]

Lattice-Based One-Way Functions

- Public key $\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$ for $q = \mathsf{poly}(n)$, $m = \Omega(n \log q)$.

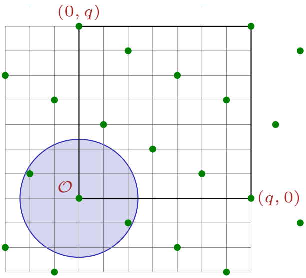
$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("short" $\mathbf{x}$, surjective)	("short" $\mathbf{e}$, injective)
CRHF if SIS hard [Ajtai'96,...]	OWF if LWE hard [Regev'05,P'09]

Lattice-Based One-Way Functions

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

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("short" $\mathbf{x}$, surjective)	("short" $\mathbf{e}$, injective)
CRHF if SIS hard [Ajtai'96,...]	OWF if LWE hard [Regev'05,P'09]

- Lattice interpretation: $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\}$

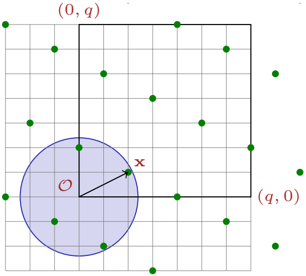


Lattice-Based One-Way Functions

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

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("short" $\mathbf{x}$, surjective)	("short" $\mathbf{e}$, injective)
CRHF if SIS hard [Ajtai'96,...]	OWF if LWE hard [Regev'05,P'09]

- Lattice interpretation: $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\}$



Lattice-Based One-Way Functions

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

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$ ("short" $\mathbf{x}$, surjective)	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$ ("short" $\mathbf{e}$, injective)
CRHF if SIS hard [Ajtai'96,...]	OWF if LWE hard [Regev'05,P'09]

- $f_{\mathbf{A}}$, $g_{\mathbf{A}}$ in forward direction yield CRHFs, CPA security (w/FHE!)

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

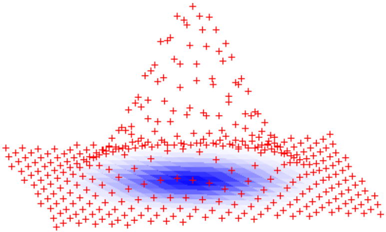
find the unique preimage $\mathbf{s}$

(equivalently, $\mathbf{e}$)

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 $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample random $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$ with prob $\propto \exp(-\|\mathbf{x}\|^2/s^2)$.	find the unique preimage $\mathbf{s}$ (equivalently, $\mathbf{e}$)



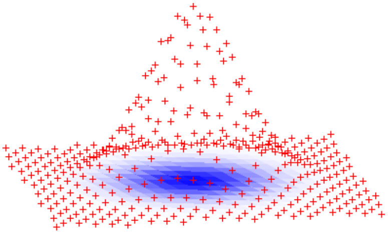
Trapdoor Inversion

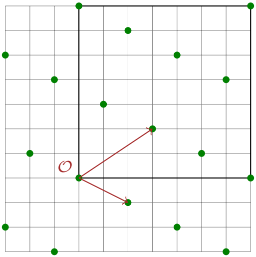
- Many cryptographic applications need to invert $f_{\mathbf{A}}$ and/or $g_{\mathbf{A}}$.

Invert $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}') = \mathbf{A}\mathbf{x}'$:	Invert $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$:
sample random $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$	find the unique preimage $\mathbf{s}$
with prob $\propto \exp(-\lVert\mathbf{x}\rVert^2/s^2)$.	(equivalently, $\mathbf{e}$)

- How? Use a "strong trapdoor" for $\mathbf{A}$: a short basis of $\Lambda^{\perp}(\mathbf{A})$

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





Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.
- Verify($msg, \mathbf{x}$): check $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = H(msg)$ and $\mathbf{x}$ short enough.

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.
- Verify$(msg, \mathbf{x})$: check $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = H(msg)$ and $\mathbf{x}$ short enough.
- Security: finding short enough preimages in $f_{\mathbf{A}}$ must be hard.

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.
- Verify($msg, \mathbf{x}$): check $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = H(msg)$ and $\mathbf{x}$ short enough.
- Security: finding short enough preimages in $f_{\mathbf{A}}$ must be hard.

Other "Black-Box" Applications of f^{-1}, g^{-1}

- Standard Model (no RO) signatures [CHKP'10,R'10,B'10]
- SM CCA-secure encryption [PW'08,P'09]
- SM (Hierarchical) IBE [GPV'08,CHKP'10,ABB'10a,ABB'10b]
- Many more: OT, NISZK, homom enc/sigs, deniable enc, func enc, ...
  [PVW'08,PV'08,GHV'10,GKV'10,BF'10a,BF'10b,OPW'11,AFV'11,ABVVW'11,...]

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.
- Verify($msg, \mathbf{x}$): check $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = H(msg)$ and $\mathbf{x}$ short enough.
- Security: finding short enough preimages in $f_{\mathbf{A}}$ must be hard.

Some Drawbacks...

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

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.
- Verify($msg, \mathbf{x}$): check $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = H(msg)$ and $\mathbf{x}$ short enough.
- Security: finding short enough preimages in $f_{\mathbf{A}}$ must be hard.

Some Drawbacks. . .

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

✗ Known inversion algorithms trade quality for efficiency

Applications of Strong Trapdoors

Canonical App: [GPV'08] Signatures

- $pk = \mathbf{A}$, $sk =$ short basis for $\mathbf{A}$, random oracle $H \colon \{0,1\}^* \to \mathbb{Z}_q^n$.
- Sign(msg): let $\mathbf{u} = H(msg)$ and output Gaussian $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$.
- Verify($msg, \mathbf{x}$): check $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} = H(msg)$ and $\mathbf{x}$ short enough.
- Security: finding short enough preimages in $f_{\mathbf{A}}$ must be hard.

Some Drawbacks. . .

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

✗ Known inversion algorithms trade quality for efficiency

	tight, iterative, fp	looser, parallel, offline
$g_{\mathbf{A}}^{-1}$	[Babai'86]	[Babai'86]
$f_{\mathbf{A}}^{-1}$	[Klein'01,GPV'08]	[P'10]

Today

"Strong" trapdoor generation and inversion algorithms:

Today

"Strong" trapdoor generation and inversion algorithms:

✔ Very simple & fast

- ⋆ Generation: one matrix mult. No HNF or inversion (cf. [A'99,AP'09])
- ⋆ Inversion of $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: practical, parallel, & mostly offline
- ⋆ No more efficiency-vs-quality tradeoff

Today

"Strong" trapdoor generation and inversion algorithms:

- ✔ Very simple & fast
  - ⋆ Generation: one matrix mult. No HNF or inversion (cf. [A'99,AP'09])
  - ⋆ Inversion of $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: practical, parallel, & mostly offline
  - ⋆ No more efficiency-vs-quality tradeoff
- ✔ Tighter parameters m and s
  - ⋆ Asymptotically optimal with small constant factors

Today

"Strong" trapdoor generation and inversion algorithms:

- ✔ Very simple & fast
  - ⋆ Generation: one matrix mult. No HNF or inversion (cf. [A'99,AP'09])
  - ⋆ Inversion of $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: practical, parallel, & mostly offline
  - ⋆ No more efficiency-vs-quality tradeoff
- ✔ Tighter parameters m and s
  - ⋆ Asymptotically optimal with small constant factors
- ✔ New kind of trapdoor — not a basis! (But just as powerful.)

Today

"Strong" trapdoor generation and inversion algorithms:

- ✔ Very simple & fast
  - ⋆ Generation: one matrix mult. No HNF or inversion (cf. [A'99,AP'09])
  - ⋆ Inversion of $f_{\mathbf{A}}$, $g_{\mathbf{A}}$: practical, parallel, & mostly offline
  - ⋆ No more efficiency-vs-quality tradeoff
- ✔ Tighter parameters m and s
  - ⋆ Asymptotically optimal with small constant factors
- ✔ New kind of trapdoor — not a basis! (But just as powerful.)
- ✔ More efficient applications: CCA, (H)IBE in standard model

Overview of Methods

1. Design a fixed, public lattice defined by "gadget" matrix $\mathbf{G}$.

   Design fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.

Overview of Methods

1. Design a fixed, public lattice defined by "gadget" matrix $\mathbf{G}$.

   Design fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.

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

   (The transformation is the trapdoor!)

Overview of Methods

1. Design a fixed, public lattice defined by "gadget" matrix $\mathbf{G}$.

   Design fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.

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

   (The transformation is the trapdoor!)

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

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}.$$

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}.$$

- To invert LWE function $g_{\mathbf{g}} \colon \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$:

$$s \cdot \mathbf{g} + \mathbf{e} = \begin{bmatrix} s + e_0 & 2s + e_1 & \cdots & 2^{k-1}s + e_{k-1} \end{bmatrix} \bmod q.$$

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1\times k}.$$

- To invert LWE function $g_{\mathbf{g}} \colon \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$:

$$s \cdot \mathbf{g} + \mathbf{e} = \begin{bmatrix} s + e_0 & 2s + e_1 & \cdots & 2^{k-1}s + e_{k-1} \end{bmatrix} \bmod q.$$

  - ⋆ Get $\mathrm{lsb}(s)$ from $2^{k-1}s + e_{k-1}$. Then get next bit of s, etc.
    Works exactly when every $e_i \in [-\frac{q}{4}, \frac{q}{4})$.

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1\times k}.$$

- To invert LWE function $g_{\mathbf{g}} \colon \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$:

$$s \cdot \mathbf{g} + \mathbf{e} = \begin{bmatrix} s + e_0 & 2s + e_1 & \cdots & 2^{k-1}s + e_{k-1} \end{bmatrix} \bmod q.$$

  - ⋆ Get lsb(s) from $2^{k-1}s + e_{k-1}$. Then get next bit of s, etc. Works exactly when every $e_i \in [-\frac{q}{4}, \frac{q}{4})$.
  - ⋆ OR round entries and look up in table.

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}.$$

- To invert LWE function $g_{\mathbf{g}} \colon \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$:

$$s \cdot \mathbf{g} + \mathbf{e} = \begin{bmatrix} s + e_0 & 2s + e_1 & \cdots & 2^{k-1}s + e_{k-1} \end{bmatrix} \bmod q.$$

  - ⋆ Get lsb(s) from $2^{k-1}s + e_{k-1}$. Then get next bit of s, etc. Works exactly when every $e_i \in [-\frac{q}{4}, \frac{q}{4})$.
  - ⋆ OR round entries and look up in table.

- To sample Gaussian preimage for $u = f_{\mathbf{g}}(\mathbf{x}) := \langle \mathbf{g}, \mathbf{x} \rangle$:

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}.$$

- To invert LWE function $g_{\mathbf{g}} \colon \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$:

$$s \cdot \mathbf{g} + \mathbf{e} = \begin{bmatrix} s + e_0 & 2s + e_1 & \cdots & 2^{k-1}s + e_{k-1} \end{bmatrix} \bmod q.$$

  - ⋆ Get lsb(s) from $2^{k-1}s + e_{k-1}$. Then get next bit of s, etc. Works exactly when every $e_i \in [-\frac{q}{4}, \frac{q}{4})$.
  - ⋆ <u>OR</u> round entries and look up in table.

- To sample Gaussian preimage for $u = f_{\mathbf{g}}(\mathbf{x}) := \langle \mathbf{g}, \mathbf{x} \rangle$:
  - ⋆ For $i \leftarrow 0, \ldots, k-1$: choose $x_i \leftarrow (2\mathbb{Z} + u)$, let $u \leftarrow (u - x_i)/2 \in \mathbb{Z}$.

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Let $q = 2^k$. Define 1-by-k "parity check" vector

$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}.$$

- To invert LWE function $g_{\mathbf{g}} \colon \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$:

$$s \cdot \mathbf{g} + \mathbf{e} = \begin{bmatrix} s + e_0 & 2s + e_1 & \cdots & 2^{k-1}s + e_{k-1} \end{bmatrix} \bmod q.$$

  - ⋆ Get lsb(s) from $2^{k-1}s + e_{k-1}$. Then get next bit of s, etc. Works exactly when every $e_i \in [-\frac{q}{4}, \frac{q}{4})$.
  - ⋆ OR round entries and look up in table.

- To sample Gaussian preimage for $u = f_{\mathbf{g}}(\mathbf{x}) := \langle \mathbf{g}, \mathbf{x} \rangle$:
  - ⋆ For $i \leftarrow 0, \ldots, k-1$: choose $x_i \leftarrow (2\mathbb{Z} + u)$, let $u \leftarrow (u - x_i)/2 \in \mathbb{Z}$.
  - ⋆ OR presample many $\mathbf{x} \leftarrow \mathbb{Z}^k$ and store in q 'buckets' $f_{\mathbf{g}}(\mathbf{x})$ for later.

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Another view: for $\mathbf{g} = \begin{bmatrix} 1 & 2 & \cdots & 2^{k-1} \end{bmatrix}$ the lattice $\Lambda^{\perp}(\mathbf{g})$ has basis

$$\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}, \quad \text{with } \tilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k.$$

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Another view: for $\mathbf{g} = \begin{bmatrix} 1 & 2 & \cdots & 2^{k-1} \end{bmatrix}$ the lattice $\Lambda^{\perp}(\mathbf{g})$ has basis

$$\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}, \quad \text{with } \tilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k.$$

The iterative inversion algorithms for $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are special cases of the (randomized) "nearest-plane" algorithm [Babai'86,Klein'01,GPV'08].

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Another view: for $\mathbf{g} = \begin{bmatrix} 1 & 2 & \cdots & 2^{k-1} \end{bmatrix}$ the lattice $\Lambda^\perp(\mathbf{g})$ has basis

$$\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}, \quad \text{with } \tilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k.$$

The iterative inversion algorithms for $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are special cases of the (randomized) "nearest-plane" algorithm [Babai'86,Klein'01,GPV'08].

- Define $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} = \begin{bmatrix} \cdots \mathbf{g} \cdots & & & \\ & \cdots \mathbf{g} \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g} \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$.

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Another view: for $\mathbf{g} = \begin{bmatrix} 1 & 2 & \cdots & 2^{k-1} \end{bmatrix}$ the lattice $\Lambda^{\perp}(\mathbf{g})$ has basis

$$\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}, \quad \text{with } \tilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k.$$

  The iterative inversion algorithms for $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are special cases of the (randomized) "nearest-plane" algorithm [Babai'86,Klein'01,GPV'08].

- Define $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} = \begin{bmatrix} \cdots \mathbf{g} \cdots & & & \\ & \cdots \mathbf{g} \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g} \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$.

  Now $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$ reduce to n parallel (and offline) calls to $f_{\mathbf{g}}^{-1}$, $g_{\mathbf{g}}^{-1}$.

Step 1: Gadget $\mathbf{G}$ and Inversion Algorithms

- Another view: for $\mathbf{g} = \begin{bmatrix} 1 & 2 & \cdots & 2^{k-1} \end{bmatrix}$ the lattice $\Lambda^{\perp}(\mathbf{g})$ has basis

$$\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}, \quad \text{with } \tilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k.$$

  The iterative inversion algorithms for $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are special cases of the (randomized) "nearest-plane" algorithm [Babai'86,Klein'01,GPV'08].

- Define $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g} = \begin{bmatrix} \cdots \mathbf{g} \cdots & & & \\ & \cdots \mathbf{g} \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g} \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$.

  Now $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$ reduce to n parallel (and offline) calls to $f_{\mathbf{g}}^{-1}$, $g_{\mathbf{g}}^{-1}$.

  Also applies to $\mathbf{H} \cdot \mathbf{G}$ for any invertible $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$.

Step 2: Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$

1. Define semi-random $[\bar{\mathbf{A}} \mid \mathbf{G}]$ for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$.

   Note: $f^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$, $g^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$ easily reduce to $f^{-1}_{\mathbf{G}}$, $g^{-1}_{\mathbf{G}}$ [CHKP'10].

Step 2: Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$

1. Define semi-random $[\bar{\mathbf{A}} \mid \mathbf{G}]$ for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$.

   Note: $f^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$, $g^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$ easily reduce to $f^{-1}_{\mathbf{G}}$, $g^{-1}_{\mathbf{G}}$ [CHKP'10].

2. Choose "short" (Gaussian) $\mathbf{R} \leftarrow \mathbb{Z}^{\bar{m} \times n \log q}$ and let

$$\mathbf{A} := [\bar{\mathbf{A}} \mid \mathbf{G}] \underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix}}_{\text{unimodular}} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

Step 2: Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$

1. Define semi-random $[\bar{\mathbf{A}} \mid \mathbf{G}]$ for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$.

   Note: $f^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$, $g^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$ easily reduce to $f^{-1}_{\mathbf{G}}$, $g^{-1}_{\mathbf{G}}$ [CHKP'10].

2. Choose "short" (Gaussian) $\mathbf{R} \leftarrow \mathbb{Z}^{\bar{m} \times n \log q}$ and let

$$\mathbf{A} := [\bar{\mathbf{A}} \mid \mathbf{G}] \underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix}}_{\text{unimodular}} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

   ⋆ $\mathbf{A}$ is uniform if $[\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is: leftover hash lemma for $\bar{m} \approx n \log q$.

Step 2: Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$

1. Define semi-random $[\bar{\mathbf{A}} \mid \mathbf{G}]$ for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$.

   Note: $f^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$, $g^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$ easily reduce to $f^{-1}_{\mathbf{G}}$, $g^{-1}_{\mathbf{G}}$ [CHKP'10].

2. Choose "short" (Gaussian) $\mathbf{R} \leftarrow \mathbb{Z}^{\bar{m} \times n \log q}$ and let

$$\mathbf{A} := [\bar{\mathbf{A}} \mid \mathbf{G}] \underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix}}_{\text{unimodular}} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

   ⋆ $\mathbf{A}$ is uniform if $[\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is: leftover hash lemma for $\bar{m} \approx n \log q$.

   (With $\mathbf{G} = \mathbf{0}$, we get the "key trick" constructing $\mathbf{A}$ with a "weak" trapdoor of ≥ 1 short vector, but not a full basis.)

Step 2: Randomize $\mathbf{G} \leftrightarrow \mathbf{A}$

1. Define semi-random $[\bar{\mathbf{A}} \mid \mathbf{G}]$ for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$.

   Note: $f^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$, $g^{-1}_{[\bar{\mathbf{A}}|\mathbf{G}]}$ easily reduce to $f^{-1}_{\mathbf{G}}$, $g^{-1}_{\mathbf{G}}$ [CHKP'10].

2. Choose "short" (Gaussian) $\mathbf{R} \leftarrow \mathbb{Z}^{\bar{m} \times n \log q}$ and let

$$\mathbf{A} := [\bar{\mathbf{A}} \mid \mathbf{G}] \underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix}}_{\text{unimodular}} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

   - ⋆ $\mathbf{A}$ is uniform if $[\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is: leftover hash lemma for $\bar{m} \approx n \log q$.

     (With $\mathbf{G} = \mathbf{0}$, we get the "key trick" constructing $\mathbf{A}$ with a "weak" trapdoor of ≥ 1 short vector, but not a full basis.)

   - ⋆ $[\mathbf{I} \mid \bar{\mathbf{A}} \mid -(\bar{\mathbf{A}}\mathbf{R}_1 + \mathbf{R}_2)]$ is pseudorandom (under LWE) for $\bar{m} = n$.

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

Definition

- $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ ($\mathbf{H}$ invertible) if

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{G}.$$

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

Definition

- $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ ($\mathbf{H}$ invertible) if

$$\mathbf{A} \cdot \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right] = \mathbf{H} \cdot \mathbf{G}.$$

- The quality of $\mathbf{R}$ is $s_1(\mathbf{R}) := \max_{\|\mathbf{u}\|=1} \|\mathbf{R}\mathbf{u}\|$. (smaller is better.)

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

Definition

- $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ ($\mathbf{H}$ invertible) if

$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{G}.$$

- The quality of $\mathbf{R}$ is $s_1(\mathbf{R}) := \max_{\|\mathbf{u}\|=1} \|\mathbf{R}\mathbf{u}\|$. (smaller is better.)
- Fact: $s_1(\mathbf{R}) \approx (\sqrt{\text{rows}} + \sqrt{\text{cols}}) \cdot r$ for Gaussian entries w/ std dev r.

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

Definition

- $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ ($\mathbf{H}$ invertible) if

$$\mathbf{A} \cdot \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right] = \mathbf{H} \cdot \mathbf{G}.$$

- The quality of $\mathbf{R}$ is $s_1(\mathbf{R}) := \max_{\|\mathbf{u}\|=1} \|\mathbf{R}\mathbf{u}\|$. (smaller is better.)
- Fact: $s_1(\mathbf{R}) \approx (\sqrt{\text{rows}} + \sqrt{\text{cols}}) \cdot r$ for Gaussian entries w/ std dev r.
- Note: $\mathbf{R}$ is a trapdoor for $\mathbf{A} - [\mathbf{0} \mid \mathbf{H}' \cdot \mathbf{G}]$ w/tag $(\mathbf{H} - \mathbf{H}')$ [ABB'10].

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

Definition

- $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ ($\mathbf{H}$ invertible) if
$$\mathbf{A} \cdot \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{H} \cdot \mathbf{G}.$$
- The quality of $\mathbf{R}$ is $s_1(\mathbf{R}) := \max_{\|\mathbf{u}\|=1} \|\mathbf{R}\mathbf{u}\|$. (smaller is better.)
- Fact: $s_1(\mathbf{R}) \approx (\sqrt{\text{rows}} + \sqrt{\text{cols}}) \cdot r$ for Gaussian entries w/ std dev r.
- Note: $\mathbf{R}$ is a trapdoor for $\mathbf{A} - [\mathbf{0} \mid \mathbf{H}' \cdot \mathbf{G}]$ w/tag $(\mathbf{H} - \mathbf{H}')$ [ABB'10].

Relating New and Old Trapdoors

Given a basis $\mathbf{S}$ for $\Lambda^\perp(\mathbf{G})$ and a trapdoor $\mathbf{R}$ for $\mathbf{A}$,
we can efficiently construct a basis $\mathbf{S}_\mathbf{A}$ for $\Lambda^\perp(\mathbf{A})$
where $\|\tilde{\mathbf{S}}_\mathbf{A}\| \le (s_1(\mathbf{R}) + 1) \cdot \|\tilde{\mathbf{S}}\|$.

A New Trapdoor Notion

- We constructed $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

Definition

- $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ ($\mathbf{H}$ invertible) if

$$\mathbf{A} \cdot \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right] = \mathbf{H} \cdot \mathbf{G}.$$

- The quality of $\mathbf{R}$ is $s_1(\mathbf{R}) := \max_{\|\mathbf{u}\|=1} \|\mathbf{R}\mathbf{u}\|$. (smaller is better.)
- Fact: $s_1(\mathbf{R}) \approx (\sqrt{\text{rows}} + \sqrt{\text{cols}}) \cdot r$ for Gaussian entries w/ std dev r.
- Note: $\mathbf{R}$ is a trapdoor for $\mathbf{A} - [\mathbf{0} \mid \mathbf{H}' \cdot \mathbf{G}]$ w/tag $(\mathbf{H} - \mathbf{H}')$ [ABB'10].

Relating New and Old Trapdoors

Given a basis $\mathbf{S}$ for $\Lambda^\perp(\mathbf{G})$ and a trapdoor $\mathbf{R}$ for $\mathbf{A}$,
we can efficiently construct a basis $\mathbf{S}_\mathbf{A}$ for $\Lambda^\perp(\mathbf{A})$
where $\|\tilde{\mathbf{S}}_\mathbf{A}\| \le (s_1(\mathbf{R}) + 1) \cdot \|\tilde{\mathbf{S}}\|$.

(But we'll never need to.)

Step 3: Reduce $f_{\mathbf{A}}^{-1}$, $g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ (w/tag $\mathbf{H} = \mathbf{I}$): $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$.

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ (w/tag $\mathbf{H} = \mathbf{I}$): $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$.

Inverting LWE Function

Given $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, recover $\mathbf{s}$ from

$$\mathbf{b}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right].$$

Works if each entry of $\mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$ in $[-\frac{q}{4}, \frac{q}{4}) \Leftarrow \|\mathbf{e}\| < q/(4s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]))$.

Step 3: Reduce $f_{\mathbf{A}}^{-1}$, $g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ (w/tag $\mathbf{H} = \mathbf{I}$): $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$.

Inverting LWE Function

Given $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, recover $\mathbf{s}$ from

$$\mathbf{b}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right].$$

Works if each entry of $\mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$ in $[-\frac{q}{4}, \frac{q}{4}) \Leftarrow \|\mathbf{e}\| < q/(4s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]))$.

Sampling Gaussian Preimages

Given $\mathbf{u}$, sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$ and output $\mathbf{x} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$?

- We have $\mathbf{A}\mathbf{x} = \mathbf{G}\mathbf{z} = \mathbf{u}$ as desired.

Step 3: Reduce $f_{\mathbf{A}}^{-1}, g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}, g_{\mathbf{G}}^{-1}$

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ (w/tag $\mathbf{H} = \mathbf{I}$): $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$.

Inverting LWE Function

Given $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, recover $\mathbf{s}$ from

$$\mathbf{b}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right].$$

Works if each entry of $\mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$ in $[-\frac{q}{4}, \frac{q}{4}) \Leftarrow \|\mathbf{e}\| < q/(4s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]))$.

Sampling Gaussian Preimages

Given $\mathbf{u}$, sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$ and output $\mathbf{x} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$?

- We have $\mathbf{A}\mathbf{x} = \mathbf{G}\mathbf{z} = \mathbf{u}$ as desired.
- Problem: $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ is non-spherical Gaussian, leaks $\mathbf{R}$!

Step 3: Reduce $f_{\mathbf{A}}^{-1}$, $g_{\mathbf{A}}^{-1}$ to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ (w/tag $\mathbf{H} = \mathbf{I}$): $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$.

Inverting LWE Function

Given $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, recover $\mathbf{s}$ from

$$\mathbf{b}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right].$$

Works if each entry of $\mathbf{e}^t\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]$ in $[-\frac{q}{4}, \frac{q}{4}) \Leftarrow \|\mathbf{e}\| < q/(4s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]))$.

Sampling Gaussian Preimages

Given $\mathbf{u}$, sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$ and output $\mathbf{x} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$?

- We have $\mathbf{A}\mathbf{x} = \mathbf{G}\mathbf{z} = \mathbf{u}$ as desired.
- Problem: $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ is non-spherical Gaussian, leaks $\mathbf{R}$!
- Solution: use offline 'perturbation' [P'10] to get spherical Gaussian w/ std dev $\approx s_1(\mathbf{R})$: output $\mathbf{x} = \mathbf{p} + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$.

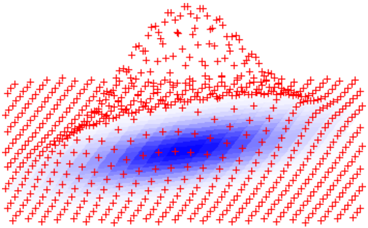
A First Attempt

- Given $\mathbf{u}$, sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$ and output $\mathbf{x} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mathbf{z} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$?

A First Attempt

- Given $\mathbf{u}$, sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$ and output $\mathbf{x} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$?
- $\mathbf{x}_1 = \mathbf{R}\mathbf{z}$ has a non-spherical Gaussian distribution of covariance

$$\Sigma := \mathbb{E}_{\mathbf{x}}\left[\mathbf{x} \cdot \mathbf{x}^t\right] = \mathbb{E}_{\mathbf{z}}\left[\mathbf{R} \cdot \mathbf{z}\mathbf{z}^t \cdot \mathbf{R}^t\right] \approx s^2 \cdot \mathbf{R}\mathbf{R}^t.$$

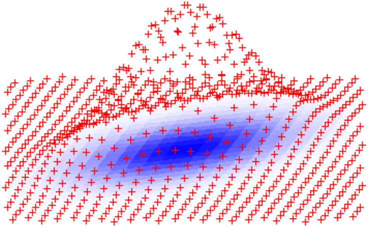


A First Attempt

- Given $\mathbf{u}$, sample $\mathbf{z} \leftarrow f_{\mathbf{G}}^{-1}(\mathbf{u})$ and output $\mathbf{x} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \mathbf{z} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$?
- $\mathbf{x}_1 = \mathbf{R}\mathbf{z}$ has a non-spherical Gaussian distribution of covariance

$$\Sigma := \mathbb{E}_{\mathbf{x}}\left[\mathbf{x} \cdot \mathbf{x}^t\right] = \mathbb{E}_{\mathbf{z}}\left[\mathbf{R} \cdot \mathbf{z}\mathbf{z}^t \cdot \mathbf{R}^t\right] \approx s^2 \cdot \mathbf{R}\mathbf{R}^t.$$

Covariance can be measured — and it leaks $\mathbf{R}$! (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 $\Sigma_2 := s^2\,\mathbf{I} - \Sigma_1$,

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

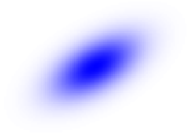
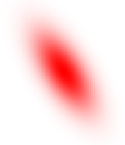
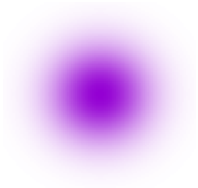
Inspiration: Some Facts About Gaussians

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 $\Sigma_2 := s^2 \, \mathbf{I} - \Sigma_1$,

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

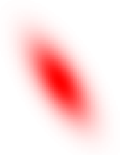
   For $\Sigma_1 = \mathbf{R} \, \mathbf{R}^t$, can use any $\boxed{s > s_1(\mathbf{R}) := \text{max singular val of } \mathbf{R}.}$

‘Convolution’ Sampling Algorithm [P’10]

- Given trapdoor $\mathbf{R}$ of $\mathbf{A}$, syndrome $\mathbf{u}$, and std dev $s > s_1(\mathbf{R})$,

'Convolution' Sampling Algorithm [P'10]

- Given trapdoor $\mathbf{R}$ of $\mathbf{A}$, syndrome $\mathbf{u}$, and std dev $s > s_1(\mathbf{R})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \mathbf{R}\mathbf{R}^t > 0$.



$(s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t)$

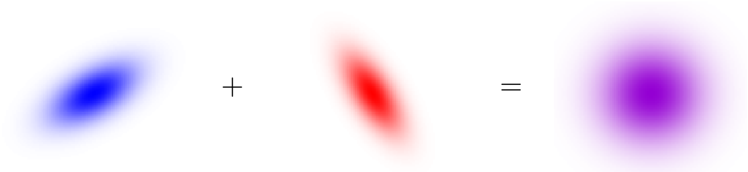
'Convolution' Sampling Algorithm [P'10]

- Given trapdoor $\mathbf{R}$ of $\mathbf{A}$, syndrome $\mathbf{u}$, and std dev $s > s_1(\mathbf{R})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \mathbf{R}\mathbf{R}^t > 0$.
  2. Sample spherical $\mathbf{z}$ s.t. $\mathbf{G}\mathbf{z} = \mathbf{u} - \mathbf{A}\mathbf{p}$.

$$\mathbf{R}\mathbf{R}^t \quad + \quad (s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t)$$

'Convolution' Sampling Algorithm [P'10]

- Given trapdoor $\mathbf{R}$ of $\mathbf{A}$, syndrome $\mathbf{u}$, and std dev $s > s_1(\mathbf{R})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \mathbf{R}\mathbf{R}^t > 0$.
  2. Sample spherical $\mathbf{z}$ s.t. $\mathbf{G}\mathbf{z} = \mathbf{u} - \mathbf{A}\mathbf{p}$.
  3. Output $\mathbf{x} = \mathbf{p} + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$. (Note: $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{p} + \mathbf{G}\mathbf{z} = \mathbf{u}$.)



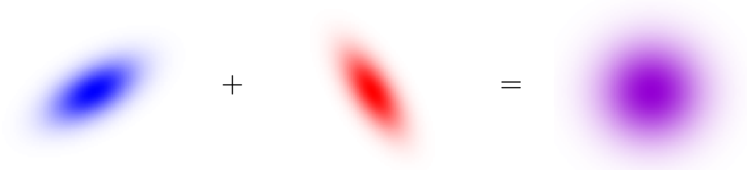
$$\mathbf{R}\mathbf{R}^t \quad + \quad (s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t) \quad = \quad s^2\,\mathbf{I}$$

Convolution* Theorem

Algorithm generates a spherical discrete Gaussian over $\mathcal{L}^{\perp}_{\mathbf{u}}(\mathbf{A})$.

‘Convolution’ Sampling Algorithm [P’10]

- Given trapdoor $\mathbf{R}$ of $\mathbf{A}$, syndrome $\mathbf{u}$, and std dev $s > s_1(\mathbf{R})$,
  1. Generate perturbation $\mathbf{p}$ with covariance $\Sigma_2 := s^2\,\mathbf{I} - \mathbf{R}\mathbf{R}^t > 0$.
  2. Sample spherical $\mathbf{z}$ s.t. $\mathbf{G}\mathbf{z} = \mathbf{u} - \mathbf{A}\mathbf{p}$.
  3. Output $\mathbf{x} = \mathbf{p} + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$. (Note: $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{p} + \mathbf{G}\mathbf{z} = \mathbf{u}$.)



$$\mathbf{R}\mathbf{R}^t \quad + \quad (s^2\mathbf{I} - \mathbf{R}\mathbf{R}^t) \quad = \quad s^2\,\mathbf{I}$$

Convolution* Theorem

Algorithm generates a spherical discrete Gaussian over $\mathcal{L}^\perp_{\mathbf{u}}(\mathbf{A})$.

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

Application: Efficient IBE *a la* [ABB'10]

- Setup: choose $\mathbf{A} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$. Let $mpk = (\mathbf{A}, \mathbf{u})$, $msk = \mathbf{R}$.
  ($\mathbf{A}$ has trapdoor $\mathbf{R}$ with tag $\mathbf{0}$.)

Application: Efficient IBE *a la* [ABB'10]

- Setup: choose $\mathbf{A} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$. Let $mpk = (\mathbf{A}, \mathbf{u})$, $msk = \mathbf{R}$.
  ($\mathbf{A}$ has trapdoor $\mathbf{R}$ with tag $\mathbf{0}$.)

- Extract$(\mathbf{R}, id)$: map $id \mapsto$ invertible $\mathbf{H}_{id} \in \mathbb{Z}_q^{n \times n}$. [DF'94,...,ABB'10]
  Using $\mathbf{R}$, choose $sk_{id} = \mathbf{x} \leftarrow f_{\mathbf{A}_{id}}^{-1}(\mathbf{u})$, where

$$\mathbf{A}_{id} = \mathbf{A} + [\mathbf{0} \mid \mathbf{H}_{id} \cdot \mathbf{G}] = [\bar{\mathbf{A}} \mid \mathbf{H}_{id} \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

Application: Efficient IBE *a la* [ABB'10]

- Setup: choose $\mathbf{A} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$. Let $mpk = (\mathbf{A}, \mathbf{u})$, $msk = \mathbf{R}$.
  ($\mathbf{A}$ has trapdoor $\mathbf{R}$ with tag $\mathbf{0}$.)

- Extract$(\mathbf{R}, id)$: map $id \mapsto$ invertible $\mathbf{H}_{id} \in \mathbb{Z}_q^{n \times n}$. [DF'94,. . . ,ABB'10]
  Using $\mathbf{R}$, choose $sk_{id} = \mathbf{x} \leftarrow f_{\mathbf{A}_{id}}^{-1}(\mathbf{u})$, where

$$\mathbf{A}_{id} = \mathbf{A} + [\mathbf{0} \mid \mathbf{H}_{id} \cdot \mathbf{G}] = [\bar{\mathbf{A}} \mid \mathbf{H}_{id} \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

- Encrypt to $\mathbf{A}_{id}$, decrypt using sk_{id} as in 'dual' system [GPV'08].

Application: Efficient IBE *a la* [ABB'10]

- Setup: choose $\mathbf{A} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$. Let $mpk = (\mathbf{A}, \mathbf{u})$, $msk = \mathbf{R}$.
  ($\mathbf{A}$ has trapdoor $\mathbf{R}$ with tag $\mathbf{0}$.)

- Extract$(\mathbf{R}, id)$: map $id \mapsto$ invertible $\mathbf{H}_{id} \in \mathbb{Z}_q^{n \times n}$. [DF'94,...,ABB'10]
  Using $\mathbf{R}$, choose $sk_{id} = \mathbf{x} \leftarrow f_{\mathbf{A}_{id}}^{-1}(\mathbf{u})$, where

$$\mathbf{A}_{id} = \mathbf{A} + [\mathbf{0} \mid \mathbf{H}_{id} \cdot \mathbf{G}] = [\bar{\mathbf{A}} \mid \mathbf{H}_{id} \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

- Encrypt to $\mathbf{A}_{id}$, decrypt using sk_{id} as in 'dual' system [GPV'08].

- Security ("puncturing"): Given target id^* (selective security), set up

$$\mathbf{A} = [\bar{\mathbf{A}} \mid -\mathbf{H}_{id^*} \cdot \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \Longrightarrow \mathbf{A}_{id} = [\bar{\mathbf{A}} \mid (\mathbf{H}_{id} - \mathbf{H}_{id^*})\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

  - ★ $\mathbf{H}_{id} - \mathbf{H}_{id^*}$ is invertible for all $id \neq id^*$, so can extract sk_{id} using $\mathbf{R}$.
  - ★ $\mathbf{A}_{id^*} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$, so can embed an LWE challenge at id^*.

Trapdoor Delegation [CHKP'10]

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, i.e. $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\\mathbf{I}\end{smallmatrix}\right] = \mathbf{H}\cdot\mathbf{G}$.

Trapdoor Delegation [CHKP'10]

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, i.e. $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}\cdot\mathbf{G}$.
- To delegate a trapdoor for an extension $[\mathbf{A} \mid \mathbf{A}']$ with tag $\mathbf{H}'$, just sample Gaussian $\mathbf{R}'$ s.t.

$$[\mathbf{A} \mid \mathbf{A}']\left[\begin{smallmatrix}\mathbf{R}'\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}'\cdot\mathbf{G} \iff \mathbf{A}\mathbf{R}' = \mathbf{H}'\cdot\mathbf{G} - \mathbf{A}'.$$

Trapdoor Delegation [CHKP'10]

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, i.e. $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H} \cdot \mathbf{G}$.
- To delegate a trapdoor for an extension $[\mathbf{A} \mid \mathbf{A}']$ with tag $\mathbf{H}'$, just sample Gaussian $\mathbf{R}'$ s.t.

$$[\mathbf{A} \mid \mathbf{A}']\left[\begin{smallmatrix}\mathbf{R}'\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}' \cdot \mathbf{G} \iff \mathbf{A}\mathbf{R}' = \mathbf{H}' \cdot \mathbf{G} - \mathbf{A}'.$$

- One-way: $\mathbf{R}'$ reveals nothing about $\mathbf{R}$.

  Useful for HIBE & IB-TDFs [CHKP'10,ABB'10,BKPW'12].

Trapdoor Delegation [CHKP'10]

- Suppose $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, i.e. $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H} \cdot \mathbf{G}$.
- To delegate a trapdoor for an extension $[\mathbf{A} \mid \mathbf{A}']$ with tag $\mathbf{H}'$, just sample Gaussian $\mathbf{R}'$ s.t.

$$[\mathbf{A} \mid \mathbf{A}']\left[\begin{smallmatrix}\mathbf{R}'\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}' \cdot \mathbf{G} \iff \mathbf{A}\mathbf{R}' = \mathbf{H}' \cdot \mathbf{G} - \mathbf{A}'.$$

- One-way: $\mathbf{R}'$ reveals nothing about $\mathbf{R}$.

  Useful for HIBE & IB-TDFs [CHKP'10,ABB'10,BKPW'12].

- Note: $\mathbf{R}'$ is only width$(\mathbf{A}) \times$ width$(\mathbf{G}) = m \times n \log q$.

  So size of $\mathbf{R}'$ grows only as $O(m)$, not $\Omega(m^2)$ like a basis does.

  Also computationally efficient: $n \log q$ samples, no HNF or ToBasis.

Hierarchical IBE [CHKP'10,ABB'10]

- Setup(d): choose $\mathbf{A}_0, \ldots, \mathbf{A}_d$ where $\mathbf{A}_\varepsilon = [\mathbf{A}_0 \mid \mathbf{A}_1]$ has trapdoor $\mathbf{R}_\varepsilon$ for tag $\mathbf{0}$. Let $msk = sk_\varepsilon = \mathbf{R}_\varepsilon$ and $mpk = \{\mathbf{A}_i\}$.

Hierarchical IBE [CHKP'10,ABB'10]

- Setup(d): choose $\mathbf{A}_0, \ldots, \mathbf{A}_d$ where $\mathbf{A}_\varepsilon = [\mathbf{A}_0 \mid \mathbf{A}_1]$ has trapdoor $\mathbf{R}_\varepsilon$ for tag $\mathbf{0}$. Let $msk = sk_\varepsilon = \mathbf{R}_\varepsilon$ and $mpk = \{\mathbf{A}_i\}$.
- Extract(id): map $id = (id_1, \ldots, id_t) \mapsto (\mathbf{H}_{id_1}, \ldots \mathbf{H}_{id_t})$ (invertible). Let

$$\mathbf{A}_{id} = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_{id_1}\mathbf{G} \mid \cdots \mid \mathbf{A}_t + \mathbf{H}_{id_t}\mathbf{G} \mid \mathbf{A}_{t+1}].$$

Hierarchical IBE [CHKP'10,ABB'10]

- Setup(d): choose $\mathbf{A}_0, \ldots, \mathbf{A}_d$ where $\mathbf{A}_\varepsilon = [\mathbf{A}_0 \mid \mathbf{A}_1]$ has trapdoor $\mathbf{R}_\varepsilon$ for tag $\mathbf{0}$. Let $msk = sk_\varepsilon = \mathbf{R}_\varepsilon$ and $mpk = \{\mathbf{A}_i\}$.

- Extract(id): map $id = (id_1, \ldots, id_t) \mapsto (\mathbf{H}_{id_1}, \ldots \mathbf{H}_{id_t})$ (invertible). Let

$$\mathbf{A}_{id} = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_{id_1}\mathbf{G} \mid \cdots \mid \mathbf{A}_t + \mathbf{H}_{id_t}\mathbf{G} \mid \mathbf{A}_{t+1}].$$

  Delegate $sk_{id} =$ trapdoor $\mathbf{R}_{id}$ for $\mathbf{A}_{id}$ with tag $\mathbf{0}$.

  Using sk_{id}, can delegate any $sk_{id'}$ for any nontrivial extension id'.

Hierarchical IBE [CHKP'10,ABB'10]

- Setup(d): choose $\mathbf{A}_0, \ldots, \mathbf{A}_d$ where $\mathbf{A}_\varepsilon = [\mathbf{A}_0 \mid \mathbf{A}_1]$ has trapdoor $\mathbf{R}_\varepsilon$ for tag $\mathbf{0}$. Let $msk = sk_\varepsilon = \mathbf{R}_\varepsilon$ and $mpk = \{\mathbf{A}_i\}$.

- Extract(id): map $id = (id_1, \ldots, id_t) \mapsto (\mathbf{H}_{id_1}, \ldots \mathbf{H}_{id_t})$ (invertible). Let

$$\mathbf{A}_{id} = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_{id_1}\mathbf{G} \mid \cdots \mid \mathbf{A}_t + \mathbf{H}_{id_t}\mathbf{G} \mid \mathbf{A}_{t+1}].$$

  Delegate sk_{id} = trapdoor $\mathbf{R}_{id}$ for $\mathbf{A}_{id}$ with tag $\mathbf{0}$.

  Using sk_{id}, can delegate any $sk_{id'}$ for any nontrivial extension id'.

- Encrypt to $\mathbf{A}_{id}$, decrypt using $\mathbf{R}_{id}$ as in [GPV'08].

Hierarchical IBE [CHKP'10,ABB'10]

- Setup(d): choose $\mathbf{A}_0, \ldots, \mathbf{A}_d$ where $\mathbf{A}_\varepsilon = [\mathbf{A}_0 \mid \mathbf{A}_1]$ has trapdoor $\mathbf{R}_\varepsilon$ for tag $\mathbf{0}$. Let $msk = sk_\varepsilon = \mathbf{R}_\varepsilon$ and $mpk = \{\mathbf{A}_i\}$.
- Extract(id): map $id = (id_1, \ldots, id_t) \mapsto (\mathbf{H}_{id_1}, \ldots \mathbf{H}_{id_t})$ (invertible). Let

$$\mathbf{A}_{id} = [\mathbf{A}_0 \mid \mathbf{A}_1 + \mathbf{H}_{id_1}\mathbf{G} \mid \cdots \mid \mathbf{A}_t + \mathbf{H}_{id_t}\mathbf{G} \mid \mathbf{A}_{t+1}].$$

  Delegate sk_{id} = trapdoor $\mathbf{R}_{id}$ for $\mathbf{A}_{id}$ with tag $\mathbf{0}$.

  Using sk_{id}, can delegate any $sk_{id'}$ for any nontrivial extension id'.
- Encrypt to $\mathbf{A}_{id}$, decrypt using $\mathbf{R}_{id}$ as in [GPV'08].
- Security ("puncturing"): Set up mpk, trapdoor $\mathbf{R}$ with tags $= -id^*$.

Conclusions

- A simple trapdoor that's easy to generate, use, and understand.
- Key sizes and algorithms for "strong" trapdoors are now realistic, with ring techniques (tomorrow)

Selected bibliography for this talk:

CHKP'10 D. Cash, D. Hofheinz, E. Kiltz, C. Peikert, "Bonsai Trees, or How to Delegate a Lattice Basis," Eurocrypt'10 / J. Crypt'11.

ABB'10 S. Agrawal, D. Boneh, X. Boyen, "Efficient Lattice (H)IBE in the Standard Model," Eurocrypt'10.

MP'12 D. Micciancio, C. Peikert, "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller," Eurocrypt'12.