Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller

$\boxed{\text{Daniele Micciancio}^1}$ Chris Peikert[2]

[1]UC San Diego

[2]Georgia Tech

April 2012

Lattice-Based Cryptography

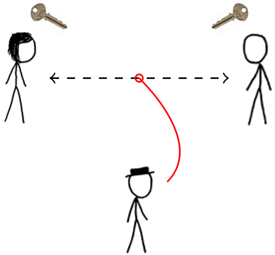
$y = g^x \bmod p$

$m^e \bmod N$

$N = p \cdot q$

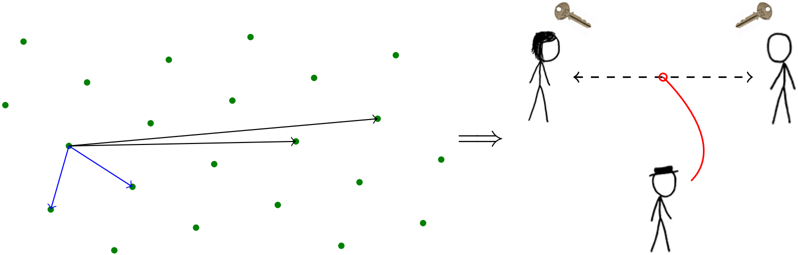
$e(g^a, g^b)$

$\Longrightarrow$



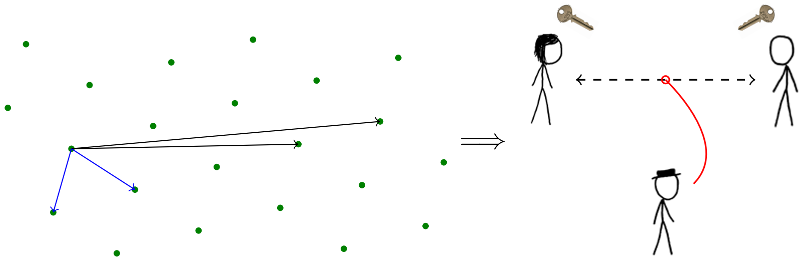
(Images courtesy xkcd.org)

Lattice-Based Cryptography



(Images courtesy xkcd.org)

Lattice-Based Cryptography

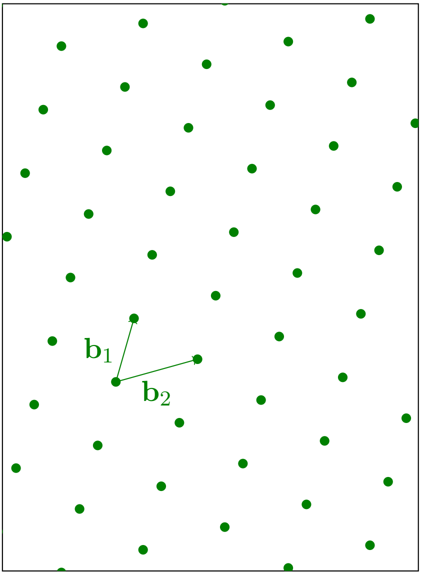


Why?

- Simple & efficient: linear, highly parallel operations
- Resist quantum attacks (so far)
- Secure under worst-case hardness assumptions [Ajtai'96,. . .]
- Solve 'holy grail' problems like FHE [Gentry'09,. . .]

(Images courtesy xkcd.org)

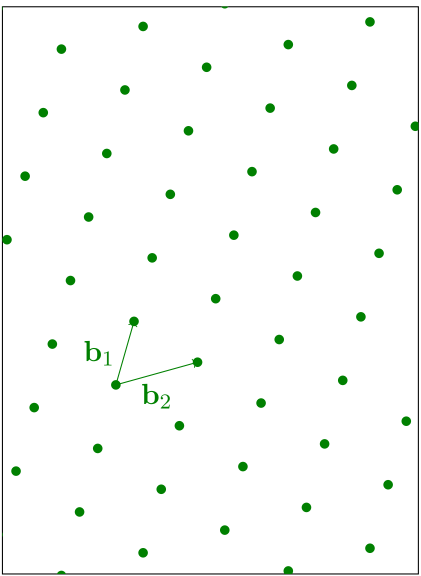
Point Lattices



A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^d$:

$$\Lambda = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z}$$

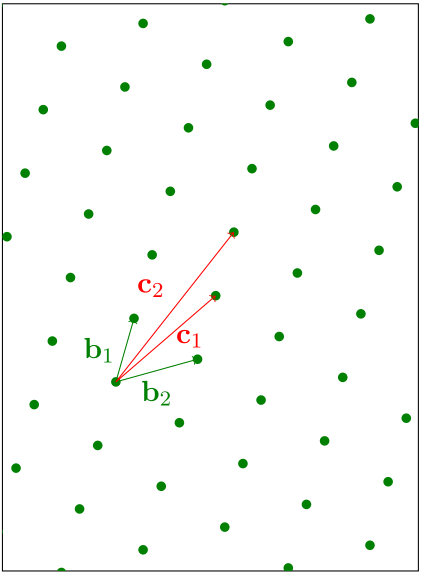
Point Lattices



A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^d$:

$$\Lambda = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{B}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$$

Point Lattices



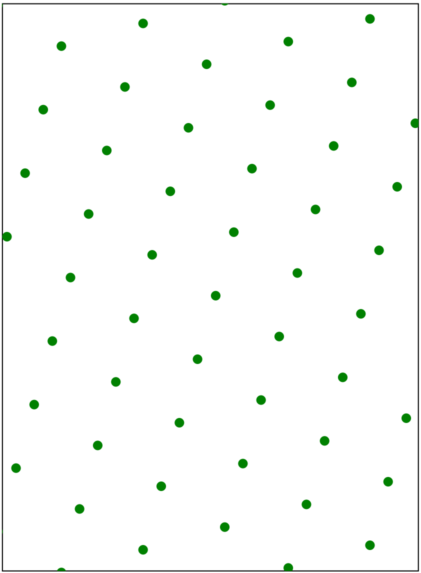
A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^d$:

$$\Lambda = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{B}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$$

The same lattice has many bases

$$\Lambda = \{\mathbf{C}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$$

Point Lattices



A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^d$:

$$\Lambda = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{B}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$$

The same lattice has many bases

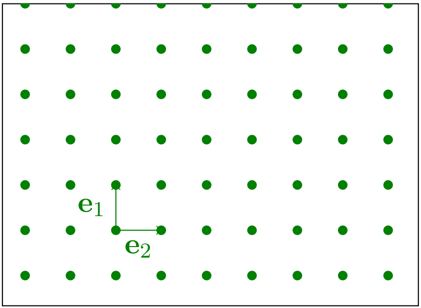
$$\Lambda = \{\mathbf{C}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$$

Definition (Lattice)

Discrete additive subgroup of $\mathbb{R}^d$

E.g. $\Lambda = \{\mathbf{x} \in \mathbb{Z}^d \colon \mathbf{A}\mathbf{x} = \mathbf{0}\}$

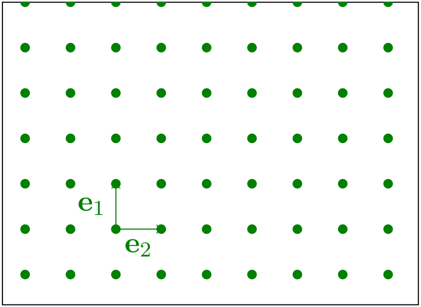
Point Lattices: Examples



The simplest lattice in n-dimensional space is the integer lattice

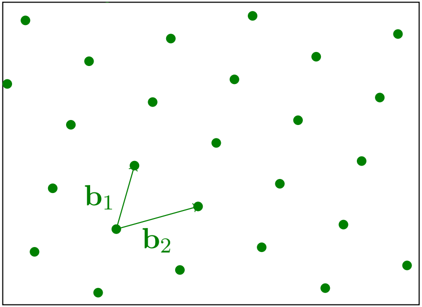
$$\Lambda = \mathbb{Z}^n$$

Point Lattices: Examples



The simplest lattice in n-dimensional space is the integer lattice

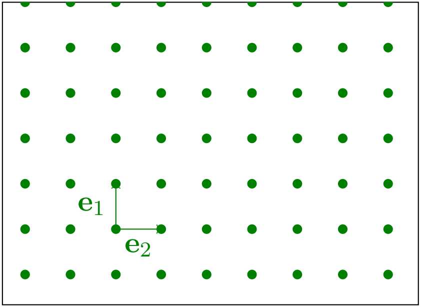
$$\Lambda = \mathbb{Z}^n$$



Other lattices are obtained by applying a linear transformation

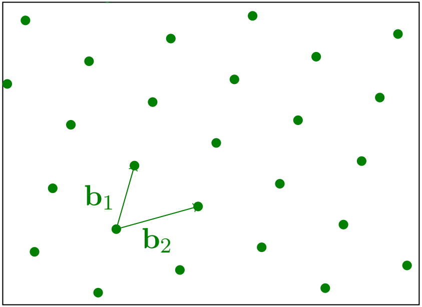
$$\Lambda = \mathbf{B}\mathbb{Z}^n \qquad (\mathbf{B} \in \mathbb{R}^{d \times n})$$

Point Lattices: Examples



The simplest lattice in n-dimensional space is the integer lattice

$$\Lambda = \mathbb{Z}^n$$



Other lattices are obtained by applying a linear transformation

$$\Lambda = \mathbf{B}\mathbb{Z}^n \qquad (\mathbf{B} \in \mathbb{R}^{d \times n})$$

Remark

All lattices have the same group structure, but different geometry

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 $q = \mathrm{poly}(n)$, $m = \Omega(n \log q)$.

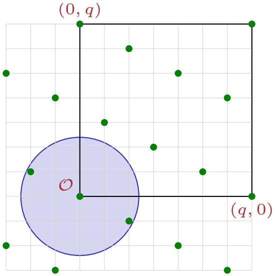
$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("short" $\mathbf{x}$, surjective)	("very short" $\mathbf{e}$, injective)
CRHF if SIS hard [Ajtai'96,...]	OWF if LWE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

Lattice-Based One-Way Functions

- Public key $\left[\cdots \ \mathbf{A} \ \cdots\right] \in \mathbb{Z}_q^{n\times m}$ for $q = \text{poly}(n)$, $m = \Omega(n \log q)$.

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("short" $\mathbf{x}$, surjective)	("very 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 $\left[\cdots \ \mathbf{A} \ \cdots\right] \in \mathbb{Z}_q^{n \times m}$ for $q = \mathrm{poly}(n)$, $m = \Omega(n \log q)$.

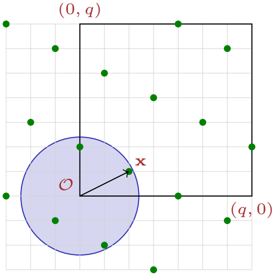
$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$
("short" $\mathbf{x}$, surjective)

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

$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("very short" $\mathbf{e}$, injective)

OWF if LWE hard [Regev'05,P'09]

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

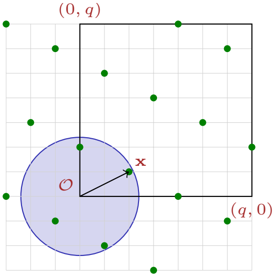


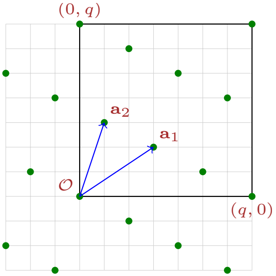
Lattice-Based One-Way Functions

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

$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$	$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$
("short" $\mathbf{x}$, surjective)	("very 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 $q = \mathrm{poly}(n)$, $m = \Omega(n \log q)$.

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

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

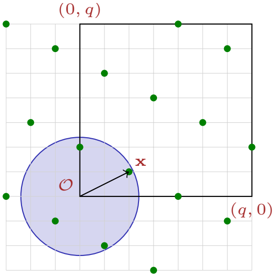
CRHF if SIS hard [Ajtai'96,...]

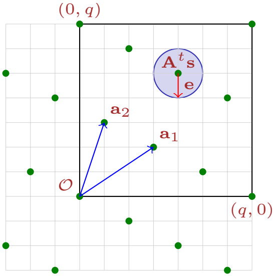
$$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q \in \mathbb{Z}_q^m$$

("very short" $\mathbf{e}$, injective)

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 $q = \text{poly}(n)$, $m = \Omega(n \log q)$.

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

- Remark:
  - ⋆ $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially equivalent functions
  - ⋆ See e.g. "Duality in lattice cryptography" [M'10]
  - ⋆ Main difference: $\mathbf{e}$ is even shorter than $\mathbf{x}$
  - ⋆ Notational convention:

Function	$\mathbf{x}/\mathbf{e}$	Injective	Surjective
$f_{\mathbf{A}}$	short	✗	✔
$g_{\mathbf{A}}$	very short	✔	✗

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)	("very short" $\mathbf{e}$, injective)
CRHF if SIS hard [Ajtai'96,...]	OWF if LWE hard [Regev'05,P'09]

- Remark:
  - $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially equivalent functions
  - See e.g. "Duality in lattice cryptography" [M'10]
  - Main difference: $\mathbf{e}$ is even shorter than $\mathbf{x}$
  - Notational convention:

Function	$\mathbf{x}/\mathbf{e}$	Injective	Surjective
$f_{\mathbf{A}}$	short	✗	✔
$g_{\mathbf{A}}$	very short	✔	✗

- $f_{\mathbf{A}}$, $g_{\mathbf{A}}$ in forward direction yield CRHFs, CPA-secure 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
$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t \bmod q$:

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 $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}') = \mathbf{A}\mathbf{x}' \bmod q$:

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

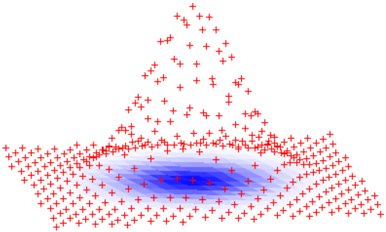
with prob $\propto \exp(-\|\mathbf{x}\|^2/\sigma^2)$.

Invert

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

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 $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}') = \mathbf{A}\mathbf{x}' \bmod q$:

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

with prob $\propto \exp(-\|\mathbf{x}\|^2/\sigma^2)$.

Invert

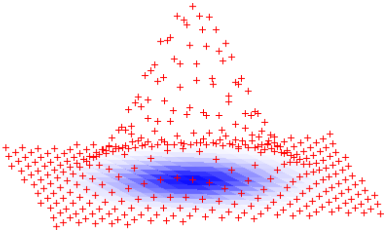
$g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$:

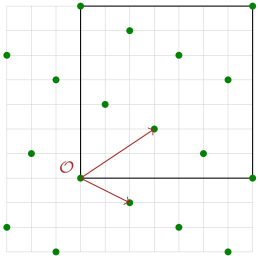
find the unique preimage $\mathbf{s}$

(equivalently, $\mathbf{e}$)

- <u>How?</u> 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

Applications of f^{-1}, g^{-1}

- "Hash and Sign" signatures in Random oracle (RO) model [GPV'08]
- 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

Applications of f^{-1}, g^{-1}

- "Hash and Sign" signatures in Random oracle (RO) model [GPV'08]
- 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,. . .]

Some Drawbacks. . .

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

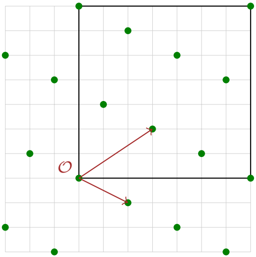
✗ Known inversion algorithms trade quality for efficiency

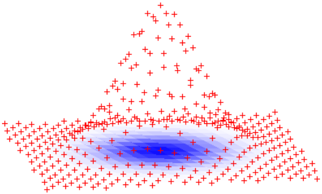
	tight, iterative, fp	looser, parallel, offline
$g_{\mathbf{A}}^{-1}$	[Babai'86]	[Babai'86]
$f_{\mathbf{A}}^{-1}$	[Klein'01,GPV'08]	[P'10]

Taming the Parameters

$$n\left\{\underbrace{\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix}}_{m}\right.$$

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

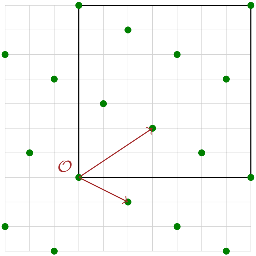


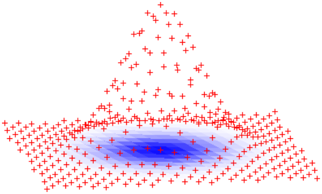


Taming the Parameters

$$n\left\{\underbrace{\begin{bmatrix}\cdots & \mathbf{A} & \cdots\end{bmatrix}}_{m}\right.$$

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



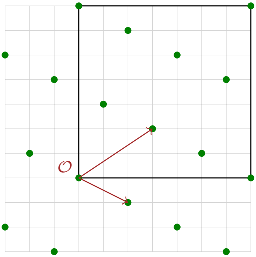


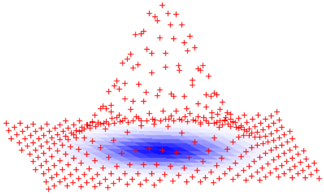
1. Trapdoor construction yields some lattice dim $m = \Omega(n \log q)$.

Taming the Parameters

$$n\left\{\underbrace{\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix}}_{m}\right.$$

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



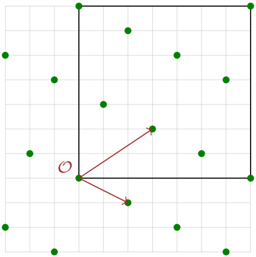


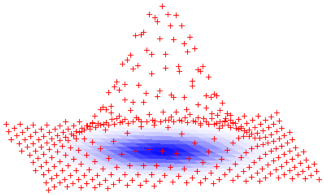
1. Trapdoor construction yields some lattice dim $m = \Omega(n \log q)$.
2. Basis "quality" $\approx$ lengths of basis vectors $\approx$ Gaussian std dev σ.

Taming the Parameters

$$n\left\{\underbrace{\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix}}_{m}\right.$$

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



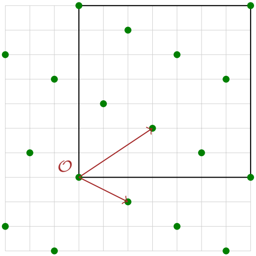


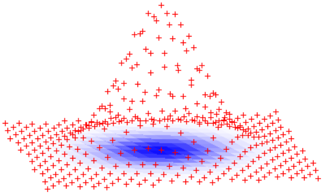
1. Trapdoor construction yields some lattice dim $m = \Omega(n \log q)$.
2. Basis "quality" $\approx$ lengths of basis vectors $\approx$ Gaussian std dev σ.
3. Dimension m, std dev $\sigma \implies$ preimage length $\beta = \|\mathbf{x}\| \approx \sigma\sqrt{m}$.

Taming the Parameters

$$n\left\{\underbrace{\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix}}_{m}\right.$$

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



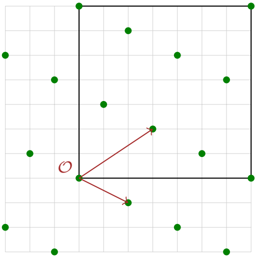


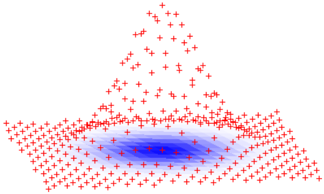
1. Trapdoor construction yields some lattice dim $m = \Omega(n \log q)$.
2. Basis “quality” $\approx$ lengths of basis vectors $\approx$ Gaussian std dev σ.
3. Dimension m, std dev $\sigma \implies$ preimage length $\beta = \|\mathbf{x}\| \approx \sigma\sqrt{m}$.
4. Choose n, q so that finding β-bounded preimages is hard.

Taming the Parameters

$$n\left\{\underbrace{\begin{bmatrix} \cdots & \mathbf{A} & \cdots \end{bmatrix}}_{m}\right.$$

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





1. Trapdoor construction yields some lattice dim $m = \Omega(n \log q)$.
2. Basis "quality" $\approx$ lengths of basis vectors $\approx$ Gaussian std dev σ.
3. Dimension m, std dev $\sigma \implies$ preimage length $\beta = \|\mathbf{x}\| \approx \sigma\sqrt{m}$.
4. Choose n, q so that finding β-bounded preimages is hard.

✔ Better dimension m & quality σ

$\implies$ "win-win-win" in security-keysize-runtime

Our Contributions

New “strong” trapdoor generation and inversion algorithms:

Our Contributions

New "strong" trapdoor generation and inversion algorithms:

✔ Very simple & fast

- ⋆ Generation: one matrix mult. No HNF or inverses (cf. [A'99,AP'09])
- ⋆ Inversion: practical, parallel, & mostly offline
- ⋆ No more efficiency-vs-quality tradeoff

Our Contributions

New "strong" trapdoor generation and inversion algorithms:

- ✔ Very simple & fast
  - ⋆ Generation: one matrix mult. No HNF or inverses (cf. [A'99,AP'09])
  - ⋆ Inversion: practical, parallel, & mostly offline
  - ⋆ No more efficiency-vs-quality tradeoff
- ✔ Tighter parameters m and σ
  - ⋆ Asymptotically optimal with small constant factors
  - ⋆ Ex improvement: 32x in dim m, 25x in quality $\sigma \Rightarrow$ 67x in keysize

Our Contributions

New "strong" trapdoor generation and inversion algorithms:

- ✔ Very simple & fast
  - ⋆ Generation: one matrix mult. No HNF or inverses (cf. [A'99,AP'09])
  - ⋆ Inversion: practical, parallel, & mostly offline
  - ⋆ No more efficiency-vs-quality tradeoff
- ✔ Tighter parameters m and σ
  - ⋆ Asymptotically optimal with small constant factors
  - ⋆ Ex improvement: 32x in dim m, 25x in quality $\sigma \Rightarrow$ 67x in keysize
- ✔ New kind of trapdoor — not a basis! (But just as powerful.)
  - ⋆ Half the dimension of a basis $\Rightarrow$ 4x size improvement
  - ⋆ Delegation: size grows as $O(\text{dim})$, versus $O(\text{dim}^2)$ [CHKP'10]

Our Contributions

New "strong" trapdoor generation and inversion algorithms:

- ✔ Very simple & fast
  - ⋆ Generation: one matrix mult. No HNF or inverses (cf. [A'99,AP'09])
  - ⋆ Inversion: practical, parallel, & mostly offline
  - ⋆ No more efficiency-vs-quality tradeoff
- ✔ Tighter parameters m and σ
  - ⋆ Asymptotically optimal with small constant factors
  - ⋆ Ex improvement: 32x in dim m, 25x in quality $\sigma \Rightarrow$ 67x in keysize
- ✔ New kind of trapdoor — not a basis! (But just as powerful.)
  - ⋆ Half the dimension of a basis $\Rightarrow$ 4x size improvement
  - ⋆ Delegation: size grows as $O(\text{dim})$, versus $O(\text{dim}^2)$ [CHKP'10]
- ✔ More efficient applications (beyond "black-box" improvements)

Concrete Parameter Improvements

	Before [AP'09]	**Now** (fast f^{-1})	**Improvement**
Dim m	slow f^{-1}: $> 5n \log q$ fast f^{-1}: $> n \log^2 q$	$2n \log q$ ($\overset{s}{\approx}$) $n(1 + \log q)$ ($\overset{c}{\approx}$)	$2.5 - \log q$

Concrete Parameter Improvements

	Before [AP'09]	**Now** (fast f^{-1})	**Improvement**
Dim m	slow f^{-1}: $> 5n \log q$ fast f^{-1}: $> n \log^2 q$	$2n \log q$ ($\stackrel{s}{\approx}$) $n(1 + \log q)$ ($\stackrel{c}{\approx}$)	$2.5 - \log q$
Quality σ	slow f^{-1}: $20\sqrt{n \log q}$ fast f^{-1}: $16\sqrt{n \log^2 q}$	$1.6\sqrt{n \log q}$	$12.5 - 10\sqrt{\log q}$

Concrete Parameter Improvements

	Before [AP'09]	**Now** (fast f^{-1})	**Improvement**
Dim m	slow f^{-1}: $> 5n \log q$ fast f^{-1}: $> n \log^2 q$	$2n \log q$ ($\overset{s}{\approx}$) $n(1 + \log q)$ ($\overset{c}{\approx}$)	$2.5 - \log q$
Quality σ	slow f^{-1}: $20\sqrt{n \log q}$ fast f^{-1}: $16\sqrt{n \log^2 q}$	$1.6\sqrt{n \log q}$	$12.5 - 10\sqrt{\log q}$

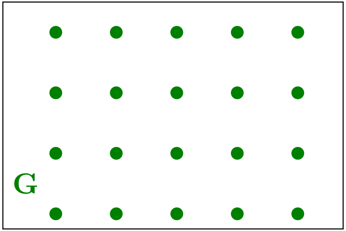
Example parameters for (ring-based) GPV signatures:

	n	q	δ to break	pk size (bits)
Before (fast f^{-1})	436	2^{32}	1.007	$\approx 17 \times 10^{6}$
Now	284	2^{24}	1.007	$\approx 36 \times 10^{4}$

Bottom line: $\approx$ 45-fold improvement in key size.

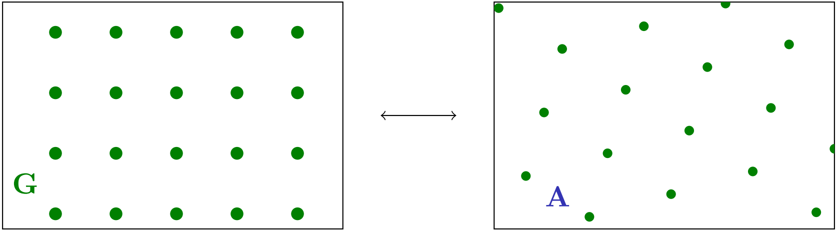
Overview of Methods

1. Design a fixed, public lattice defined by "gadget" $\mathbf{G}$.
   Give fast, parallel, offline algorithms for $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.



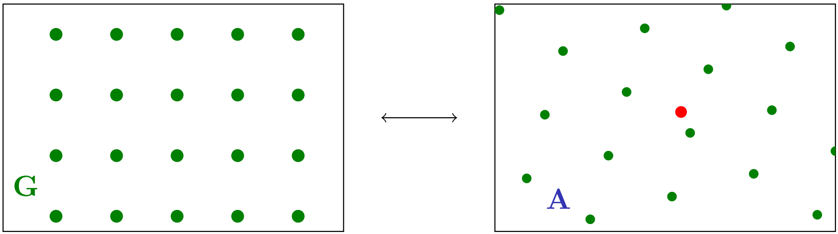
Overview of Methods

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



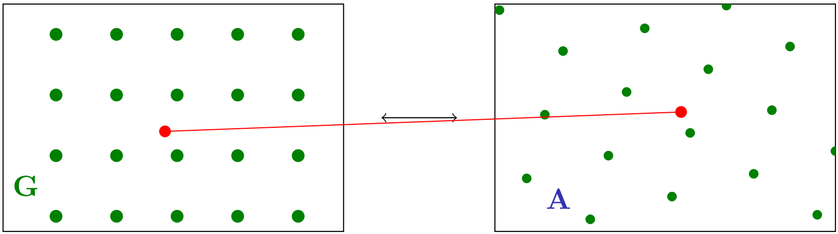
Overview of Methods

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



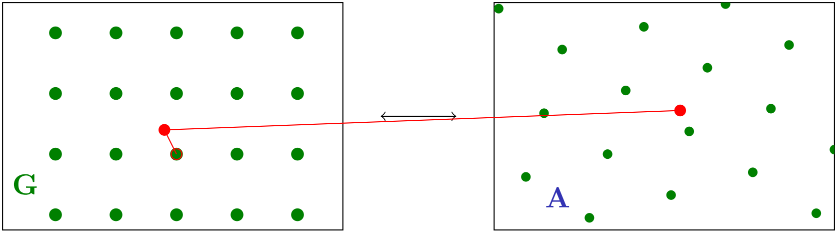
Overview of Methods

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



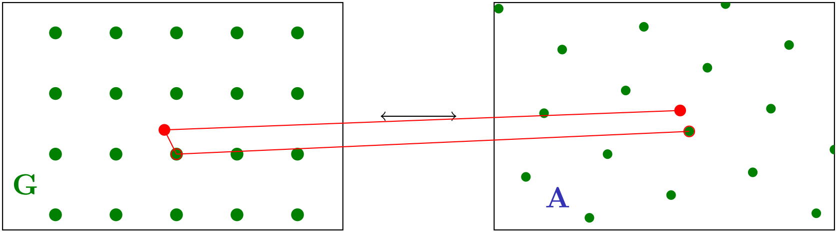
Overview of Methods

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



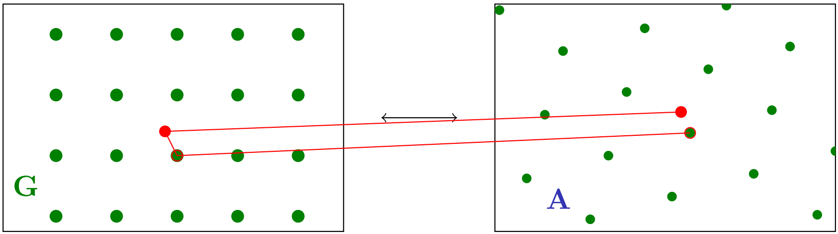
Overview of Methods

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



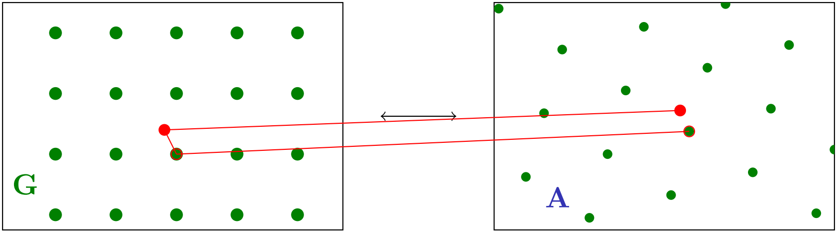
Overview of Methods

1. Design a fixed, public lattice defined by "gadget" $\mathbf{G}$.
   Give 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.
4. Problem: Transformation distorts noise.



Overview of Methods

1. Design a fixed, public lattice defined by "gadget" $\mathbf{G}$.
   Give 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.
4. Problem: Transformation distorts noise.
   Solution: add 'perturbation' during pre-/post-processing [P'10]



Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

  almost orthogonal $(\widetilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k)$,

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

  almost orthogonal ($\widetilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k$), sparse ($2k-1$ nonzero entries),

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

  almost orthogonal ($\widetilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k$), sparse ($2k-1$ nonzero entries), and low dimensional ($k = \log q = O(\log n)$)

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

  almost orthogonal ($\widetilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k$), sparse ($2k-1$ nonzero entries), and low dimensional ($k = \log q = O(\log n)$)

- $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are efficiently invertible, either by optimized versions of [Babai'86,Klein'01,GPV'08], or other specialized algorithms.

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

  almost orthogonal ($\widetilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k$), sparse ($2k-1$ nonzero entries), and low dimensional ($k = \log q = O(\log n)$)

- $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are efficiently invertible, either by optimized versions of [Babai'86,Klein'01,GPV'08], or other specialized algorithms.

Inverting f on very small inputs

Find $\mathbf{x} \in \{0,1\}^k$ such that $f_{\mathbf{g}}(\mathbf{x}) = \mathbf{g} \cdot \mathbf{x} = y \bmod q$.

Gadget $\mathbf{G}$ construction: the primitive vector $\mathbf{g}$

- Let $q = 2^k$. Define lattice $\Lambda^\perp(\mathbf{g})$ by $1 \times k$ "parity check" vector

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

- $\Lambda^\perp(\mathbf{g})$ has a short basis $\mathbf{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}$

  almost orthogonal ($\widetilde{\mathbf{S}} = 2 \cdot \mathbf{I}_k$), sparse ($2k-1$ nonzero entries), and low dimensional ($k = \log q = O(\log n)$)

- $f_{\mathbf{g}}$, $g_{\mathbf{g}}$ are efficiently invertible, either by optimized versions of [Babai'86,Klein'01,GPV'08], or other specialized algorithms.

Inverting f on very small inputs

Find $\mathbf{x} \in \{0,1\}^k$ such that $f_{\mathbf{g}}(\mathbf{x}) = \mathbf{g} \cdot \mathbf{x} = y \bmod q$.
Solution: set $\mathbf{x}$ to the binary representation of y

Gadget $\mathbf{G}$ construction: from $\mathbf{g}$ to $\mathbf{G}$

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

Gadget $\mathbf{G}$ construction: from $\mathbf{g}$ to $\mathbf{G}$

- 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 calls to $f_{\mathbf{g}}^{-1}$, $g_{\mathbf{g}}^{-1}$.

Gadget $\mathbf{G}$ construction: from $\mathbf{g}$ to $\mathbf{G}$

- 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 calls to $f_{\mathbf{g}}^{-1}$, $g_{\mathbf{g}}^{-1}$.
- Running time: almost linear in n, and trivially parallelizable up to n processors.

Gadget $\mathbf{G}$ construction: from $\mathbf{g}$ to $\mathbf{G}$

- 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 calls to $f_{\mathbf{g}}^{-1}$, $g_{\mathbf{g}}^{-1}$.
- Running time: almost linear in n, and trivially parallelizable up to n processors.
- The lattice $\Lambda^\perp(\mathbf{G})$ has short basis

$$\mathbf{S}^{\oplus n} = \mathbf{I}_n \otimes \mathbf{S} = \begin{bmatrix} \cdots \mathbf{S} \cdots & & & \\ & \cdots \mathbf{S} \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{S} \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$$

Gadget $\mathbf{G}$ construction: from $\mathbf{g}$ to $\mathbf{G}$

- 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 calls to $f_{\mathbf{g}}^{-1}$, $g_{\mathbf{g}}^{-1}$.
- Running time: almost linear in n, and trivially parallelizable up to n processors.
- The lattice $\Lambda^{\perp}(\mathbf{G})$ has short basis

$$\mathbf{S}^{\oplus n} = \mathbf{I}_n \otimes \mathbf{S} = \begin{bmatrix} \cdots \mathbf{S} \cdots & & & \\ & \cdots \mathbf{S} \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{S} \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}$$

almost orthogonal ($\widetilde{\mathbf{S}}^{\oplus n} = 2 \cdot \mathbf{I}_{kn}$), and
sparse ($< 2kn$ nonzero entries).

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$

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

   (Computing f^{-1}, g^{-1} easily reduce to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.)

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$

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

   (Computing f^{-1}, g^{-1} easily reduce to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.)

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

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

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$

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

   (Computing f^{-1}, g^{-1} easily reduce to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.)

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 $\bar{m} \approx n \log q$.

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$

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

   (Computing f^{-1}, g^{-1} easily reduce to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.)

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 Ajtai's original method for constructing $\mathbf{A}$ with a "weak" trapdoor of ≥ 1 short vector (but not a full basis).

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$

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

   (Computing f^{-1}, g^{-1} easily reduce to $f_{\mathbf{G}}^{-1}$, $g_{\mathbf{G}}^{-1}$.)

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 Ajtai's original method for 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 $\bar{m} = n$.

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$: Efficiency

1. Linear transformation is easily computable $(\bar{m} \cdot n \log q)$

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{(\bar{m}+n\log q)\times(\bar{m}+n\log q)}$$

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$: Efficiency

1. Linear transformation is easily computable ($\bar{m} \cdot n \log q$)

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{(\bar{m}+n\log q)\times(\bar{m}+n\log q)}$$

2. Inverse transformation is just as simple

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I} & +\mathbf{R} \\ & \mathbf{I} \end{bmatrix}$$

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$: Efficiency

1. Linear transformation is easily computable ($\bar{m} \cdot n \log q$)

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{(\bar{m}+n\log q)\times(\bar{m}+n\log q)}$$

2. Inverse transformation is just as simple

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I} & +\mathbf{R} \\ & \mathbf{I} \end{bmatrix}$$

3. Batch application of $\mathbf{T}$ (or $\mathbf{T}^{-1}$) to several inputs can be computed asymptotically faster using fast matrix multiplication algorithms.

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$: Efficiency

1. Linear transformation is easily computable ($\bar{m} \cdot n \log q$)

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{(\bar{m}+n\log q)\times(\bar{m}+n\log q)}$$

2. Inverse transformation is just as simple

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I} & +\mathbf{R} \\ & \mathbf{I} \end{bmatrix}$$

3. Batch application of $\mathbf{T}$ (or $\mathbf{T}^{-1}$) to several inputs can be computed asymptotically faster using fast matrix multiplication algorithms.
4. Both $\mathbf{T}$ and $\mathbf{T}^{-1}$ introduce relatively low (in fact, optimal) distorsion because $\mathbf{R}$ has small (Gaussian) entries.

Randomized Trasformation $\mathbf{G} \leftrightarrow \mathbf{A}$: Efficiency

1. Linear transformation is easily computable ($\bar{m} \cdot n \log q$)

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{(\bar{m}+n\log q)\times(\bar{m}+n\log q)}$$

2. Inverse transformation is just as simple

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I} & +\mathbf{R} \\ & \mathbf{I} \end{bmatrix}$$

3. Batch application of $\mathbf{T}$ (or $\mathbf{T}^{-1}$) to several inputs can be computed asymptotically faster using fast matrix multiplication algorithms.
4. Both $\mathbf{T}$ and $\mathbf{T}^{-1}$ introduce relatively low (in fact, optimal) distorsion because $\mathbf{R}$ has small (Gaussian) entries.
5. A basis for $\Lambda^{\perp}(\mathbf{A})$ is easily computed using $\mathbf{T}$, but never needed: $\mathbf{R}$ serves as a new trapdoor

Conclusions

- A new, simpler, more efficient trapdoor notion and construction

Conclusions

- A new, simpler, more efficient trapdoor notion and construction
- Exposing structure of trapdoor to applications yields further efficiency improvements

Conclusions

- A new, simpler, more efficient trapdoor notion and construction
- Exposing structure of trapdoor to applications yields further efficiency improvements
- Key sizes and algorithms for "strong" trapdoors are now practical

Conclusions

- A new, simpler, more efficient trapdoor notion and construction
- Exposing structure of trapdoor to applications yields further efficiency improvements
- Key sizes and algorithms for "strong" trapdoors are now practical

Questions?