Trapdoors for Lattices: Signatures, ID-Based Encryption, and Beyond

Chris Peikert
Georgia Institute of Technology

Symbolic Computations and Post-Quantum Cryptography

The Internet, 2 Mar 2011

Talk Agenda

1. Lattice-based trapdoor functions and preimage sampling
2. Applications: signatures, ID-based encryption (in RO model)
3. ‘Bonsai trees:’ removing the RO & more advanced apps

Talk Agenda

1. Lattice-based trapdoor functions and preimage sampling
2. Applications: signatures, ID-based encryption (in RO model)
3. ‘Bonsai trees:’ removing the RO & more advanced apps

- C. Gentry, C. Peikert, V. Vaikuntanathan (STOC 2008)
  “Trapdoors for Hard Lattices and New Cryptographic Constructions”
- D. Cash, D. Hofheinz, E. Kiltz, C. Peikert (Eurocrypt 2010)
  “Bonsai Trees, or How to Delegate a Lattice Basis”

Main Message

Lattices admit a hierarchy of increasingly powerful 'trapdoors,' which enable many rich applications

Part 1:

Trapdoor Functions and Preimage Sampling

Digital Signatures

(Images courtesy xkcd.org)

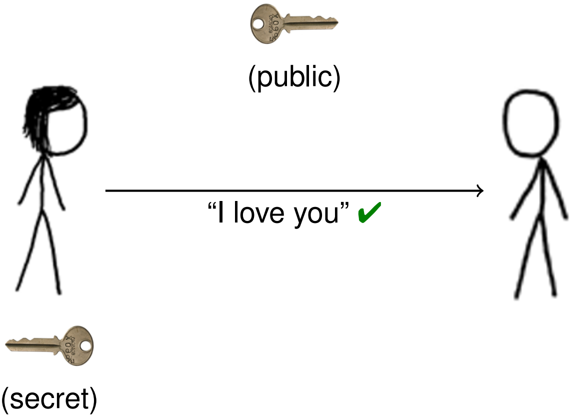
Digital Signatures

(public)

(secret)

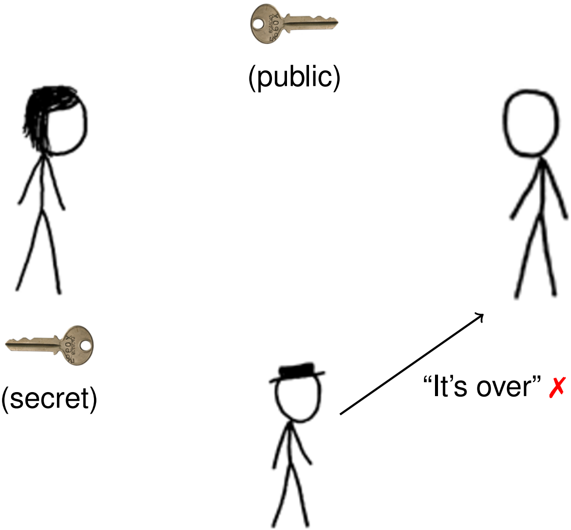
(Images courtesy xkcd.org)

Digital Signatures



(Images courtesy xkcd.org)

Digital Signatures



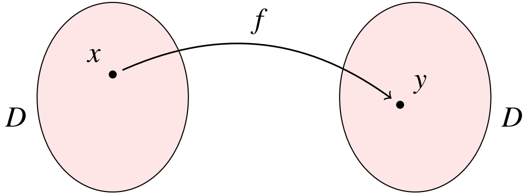
(Images courtesy xkcd.org)

Central Tool: Trapdoor Functions

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

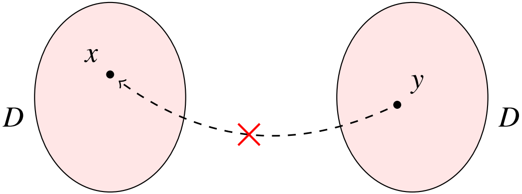
Central Tool: Trapdoor Functions

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



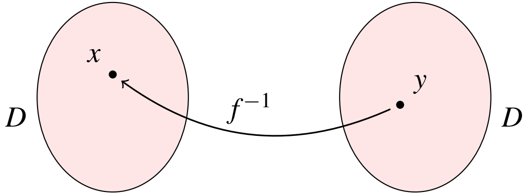
Central Tool: Trapdoor Functions

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



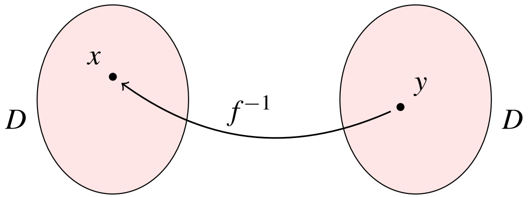
Central Tool: Trapdoor Functions

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



Central Tool: Trapdoor Functions

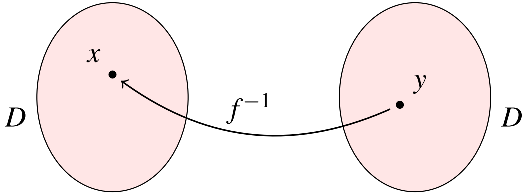
- Public function f with secret 'trapdoor' f^{-1}
- Trapdoor permutation [DH'76,RSA'77,...]



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

Central Tool: Trapdoor Functions

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



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

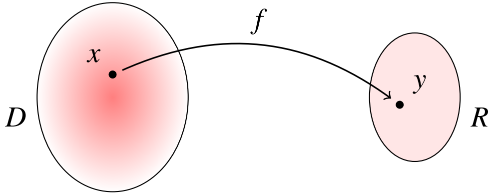
  All rely on hardness of factoring:

  ✗ Complex: 2048-bit exponentiation

  ✗ Broken by quantum algorithms [Shor'97]

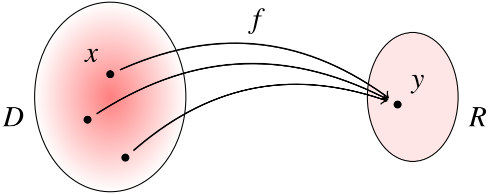
Central Tool: Trapdoor Functions

- Public function f with secret 'trapdoor' f^{-1}
- New twist: preimage sampleable trapdoor function



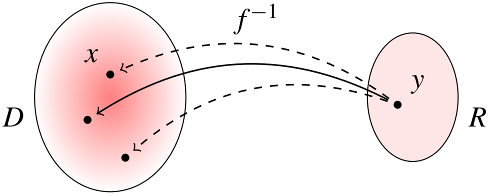
Central Tool: Trapdoor Functions

- Public function f with secret 'trapdoor' f^{-1}
- New twist: preimage sampleable trapdoor function



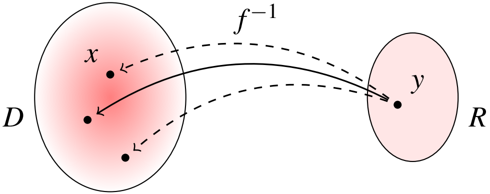
Central Tool: Trapdoor Functions

- Public function f with secret 'trapdoor' f^{-1}
- New twist: preimage sampleable trapdoor function



Central Tool: Trapdoor Functions

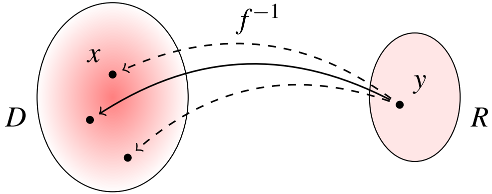
- Public function f with secret 'trapdoor' f^{-1}
- New twist: preimage sampleable trapdoor function



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

Central Tool: Trapdoor Functions

- Public function f with secret 'trapdoor' f^{-1}
- New twist: preimage sampleable trapdoor function

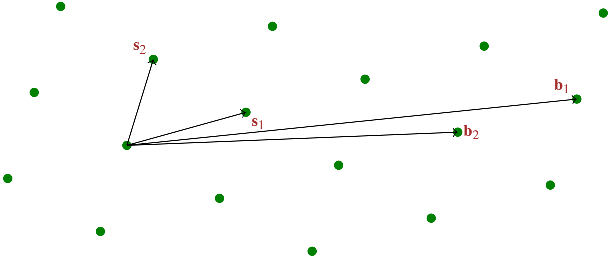


- 'Hash and sign:' $pk = f$, $sk = f^{-1}$. Sign(msg) $= f^{-1}(H(\text{msg}))$.
- Still secure! Can generate (x, y) in two equivalent ways:

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

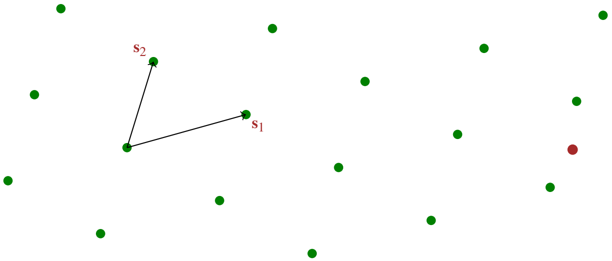
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



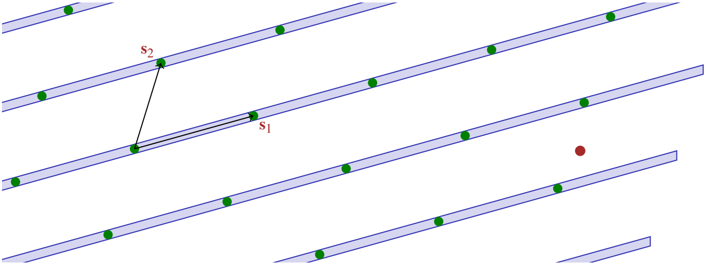
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



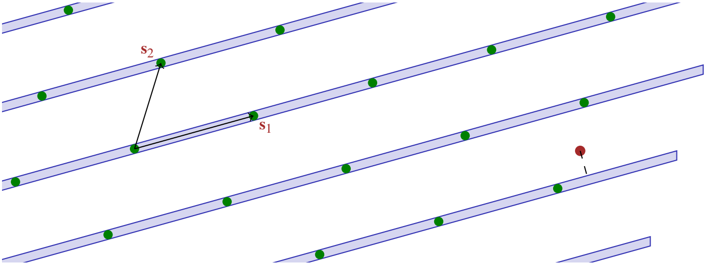
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



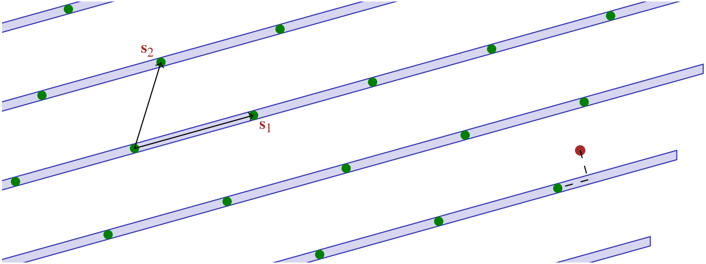
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



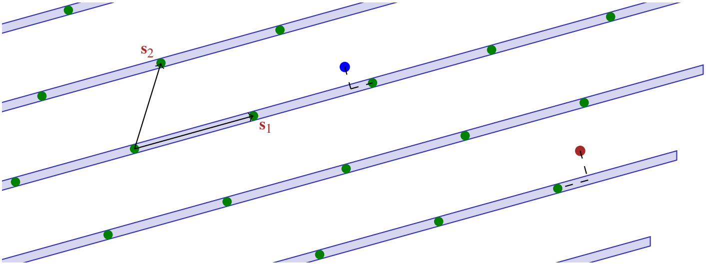
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



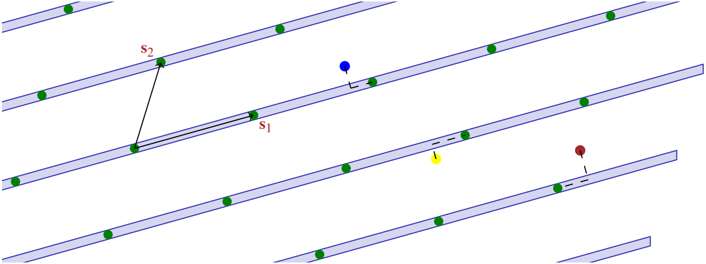
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



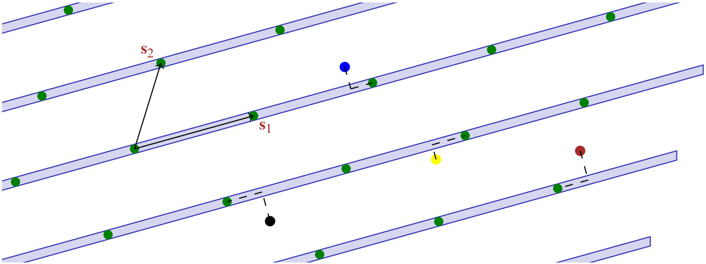
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



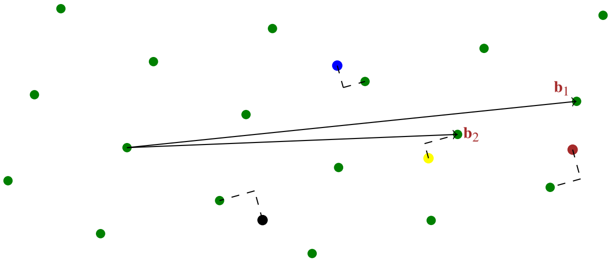
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



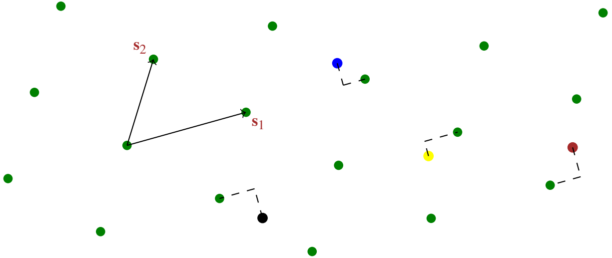
GGH Signatures [GoldreichGoldwasserHalevi'96]

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



GGH Signatures [GoldreichGoldwasserHalevi'96]

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

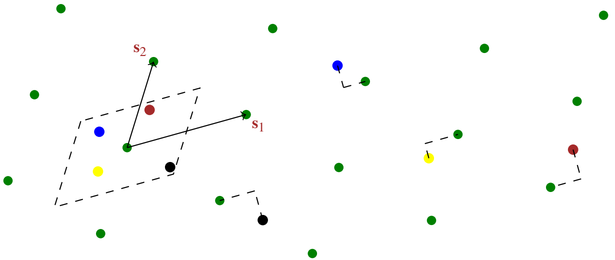


Technical Issues

1. Generating 'hard' lattice together with short basis

GGH Signatures [GoldreichGoldwasserHalevi'96]

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

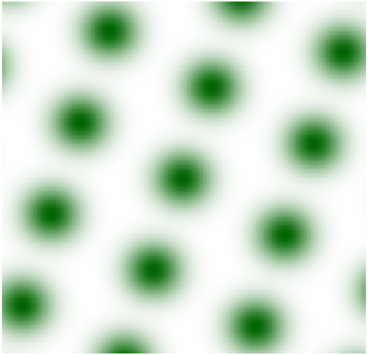


Technical Issues

1. Generating 'hard' lattice together with short basis
2. Signing algorithm leaks secret basis!
   - ⋆ Total break after several signatures [NguyenRegev'06]

Blurring a Lattice

Blurring a Lattice



Blurring a Lattice

Blurring a Lattice

'Uniform' in $\mathbb{R}^n$ when Gaussian std dev $\geq$ minimum basis length

Blurring a Lattice

'Uniform' in $\mathbb{R}^n$ when Gaussian std dev $\geq$ minimum basis length

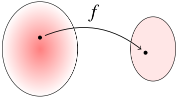
- First used in worst/average-case reductions [Regev'03,MiccReg'04,...]

Blurring a Lattice

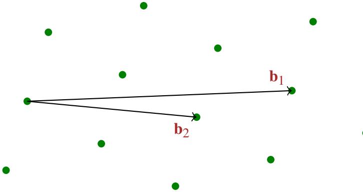
‘Uniform’ in $\mathbb{R}^n$ when Gaussian std dev $\geq$ minimum basis length

- First used in worst/average-case reductions [Regev’03,MiccReg’04,...]
- Now an essential ingredient in many crypto protocols
  [GPV’08,PV’08,ACPS’09,CHKP’10,OPW’11,...]

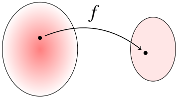
Trapdoor Function: Evaluation



- ‘Bad’ basis for $\mathcal{L}$ specifies f

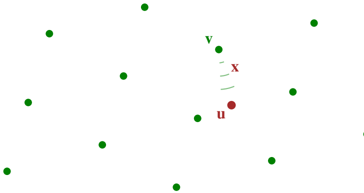


Trapdoor Function: Evaluation

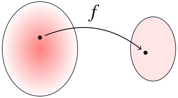


- ‘Bad’ basis for $\mathcal{L}$ specifies f
- $f(\mathbf{v}, \mathbf{x}) = \mathbf{v} + \mathbf{x}$ for $\mathbf{v} \in \mathcal{L}$, Gaussian $\mathbf{x}$.

  $\Rightarrow$ Output $\mathbf{u}$ is uniform over $\mathbb{R}^n$.

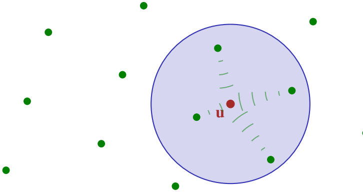


Trapdoor Function: Evaluation

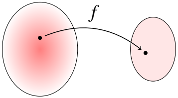


- 'Bad' basis for $\mathcal{L}$ specifies f
- $f(\mathbf{v}, \mathbf{x}) = \mathbf{v} + \mathbf{x}$ for $\mathbf{v} \in \mathcal{L}$, Gaussian $\mathbf{x}$.

  $\Rightarrow$ Output $\mathbf{u}$ is uniform over $\mathbb{R}^n$.
- Inverting $\Leftrightarrow$ decoding $\mathbf{u}$ (hard?)

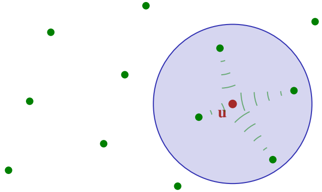


Trapdoor Function: Evaluation

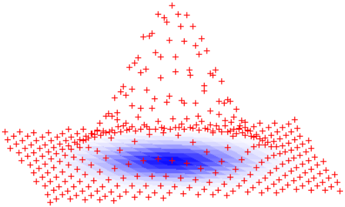


- 'Bad' basis for $\mathcal{L}$ specifies f
- $f(\mathbf{v}, \mathbf{x}) = \mathbf{v} + \mathbf{x}$ for $\mathbf{v} \in \mathcal{L}$, Gaussian $\mathbf{x}$.

  $\Rightarrow$ Output $\mathbf{u}$ is uniform over $\mathbb{R}^n$.
- Inverting $\Leftrightarrow$ decoding $\mathbf{u}$ (hard?)

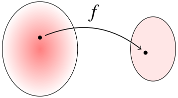


- Distribution of preimage offsets $\mathbf{x}$ is a discrete Gaussian $D_{\mathcal{L},\mathbf{u}}$



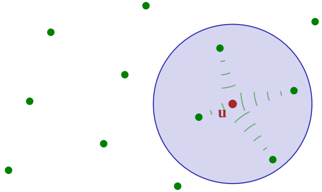
Analyzed in
[Ban'93,B'95,R'03,AR'04,MR'04,P'07...]

Trapdoor Function: Evaluation

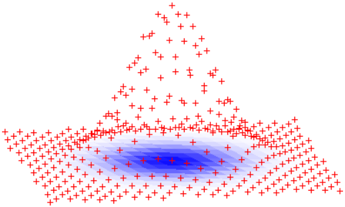


- 'Bad' basis for $\mathcal{L}$ specifies f
- $f(\mathbf{v}, \mathbf{x}) = \mathbf{v} + \mathbf{x}$ for $\mathbf{v} \in \mathcal{L}$, Gaussian $\mathbf{x}$.

  $\Rightarrow$ Output $\mathbf{u}$ is uniform over $\mathbb{R}^n$.
- Inverting $\Leftrightarrow$ decoding $\mathbf{u}$ (hard?)



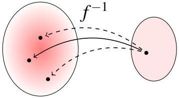
- Distribution of preimage offsets $\mathbf{x}$ is a discrete Gaussian $D_{\mathcal{L},\mathbf{u}}$



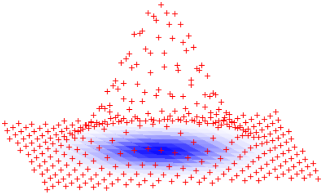
Analyzed in
[Ban'93,B'95,R'03,AR'04,MR'04,P'07...]

Typical fact: $\|D_{\mathcal{L},\mathbf{u}}\| \le \sqrt{n} \cdot$ std dev

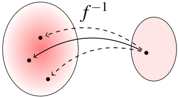
Preimage Sampling



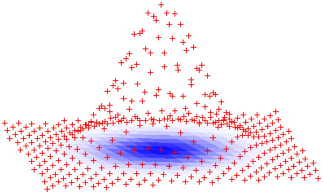
- Sample $D_{\mathcal{L},\mathbf{u}}$ given any 'short enough' basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!

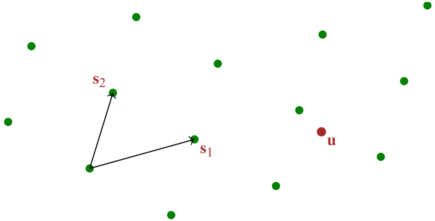


Preimage Sampling

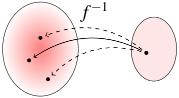


- Sample $D_{\mathcal{L},\mathbf{u}}$ given any 'short enough' basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!
- Randomized "nearest-plane" algorithm [Babai'86,Klein'00,GPV'08]



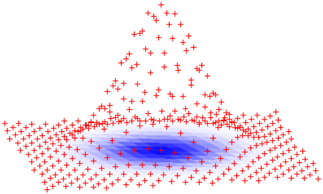


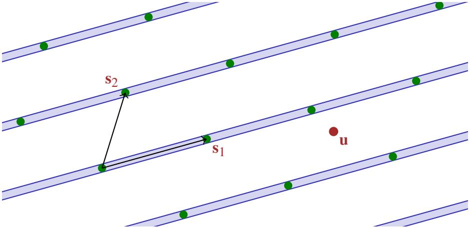
Preimage Sampling



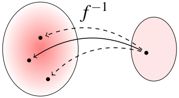
- Sample $D_{\mathcal{L},\mathbf{u}}$ given any 'short enough' basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!

- Randomized "nearest-plane" algorithm [Babai'86,Klein'00,GPV'08]



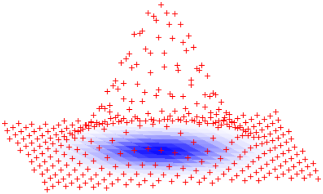


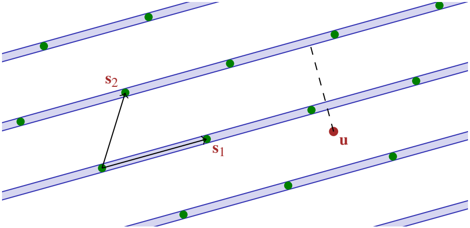
Preimage Sampling



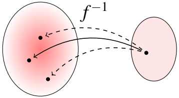
- Sample $D_{\mathcal{L},\mathbf{u}}$ given any 'short enough' basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!

- Randomized "nearest-plane" algorithm [Babai'86,Klein'00,GPV'08]

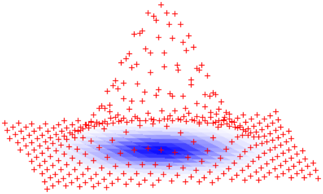


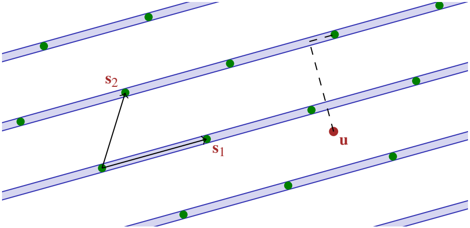


Preimage Sampling

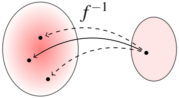


- Sample $D_{\mathcal{L},\mathbf{u}}$ given any 'short enough' basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!
- Randomized "nearest-plane" algorithm [Babai'86,Klein'00,GPV'08]



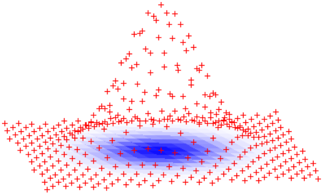


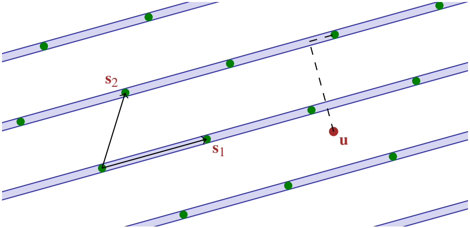
Preimage Sampling



- Sample $D_{\mathcal{L},\mathbf{u}}$ given any ‘short enough’ basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!

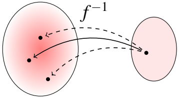
- Randomized “nearest-plane” algorithm [Babai’86,Klein’00,GPV’08]





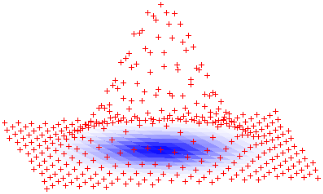
- Proof idea: $D_{\mathcal{L},\mathbf{u}}(\text{plane})$ depends only on $\mathrm{dist}(\mathbf{u}, \text{plane})$

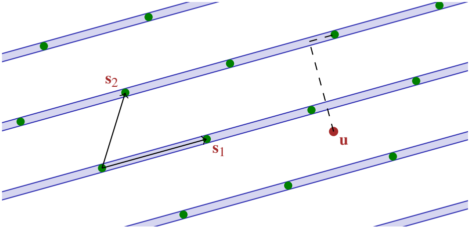
Preimage Sampling



- Sample $D_{\mathcal{L},\mathbf{u}}$ given any 'short enough' basis $\mathbf{S}$: $\max\|\tilde{\mathbf{s}}_i\| \leq$ std dev
  - ⋆ Output distribution leaks no information about $\mathbf{S}$!

- Randomized "nearest-plane" algorithm [Babai'86,Klein'00,GPV'08]





- Proof idea: $D_{\mathcal{L},\mathbf{u}}(\text{plane})$ depends only on $\text{dist}(\mathbf{u}, \text{plane})$

- [P'10]: Efficient & parallel algorithm for std dev $\geq s_1(\mathbf{S}) \approx \max\|\tilde{\mathbf{s}}_i\|$

A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$

A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \bmod q$$

A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \bmod q$$

Easy to find a 'long' solution: e.g., $\mathbf{z} = (q, 0, \ldots, 0)$

— but very hard to find a 'short' one!

A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \bmod q$$

Easy to find a 'long' solution: e.g., $\mathbf{z} = (q, 0, \ldots, 0)$

— but very hard to find a 'short' one!

Theorem: Worst-Case/Average-Case [Ajtai'96,...,MR'04,GPV'08]

For uniform $\mathbf{A}$ and $q \geq \beta\sqrt{n}$, finding solution $\mathbf{z} \neq \mathbf{0}$ where $\|\mathbf{z}\| \leq \beta$

$$\Downarrow$$

Solving $\beta\sqrt{n}$-approx GapSVP & more, on any n-dim lattice!

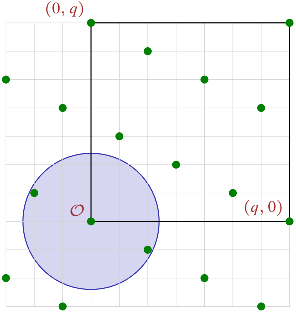
A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \bmod q$$

Putting it all together:

1. Solutions $\mathbf{z}$ form a 'hard' lattice $\mathcal{L} \subseteq \mathbb{Z}^m$



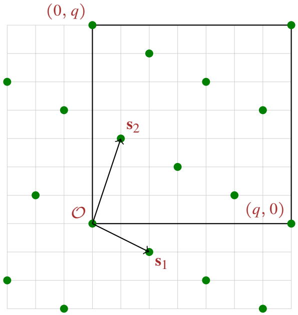
A Secure Instantiation [Ajtai96,...]

- Let n = sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \mod q$$

Putting it all together:

1. Solutions $\mathbf{z}$ form a 'hard' lattice $\mathcal{L} \subseteq \mathbb{Z}^m$
2. [Ajtai'99,AlwenP'09]: can generate uniform $\mathbf{A}$ together with a short basis $\mathbf{S}$ (i.e., $\mathbf{AS} = \mathbf{0}$).



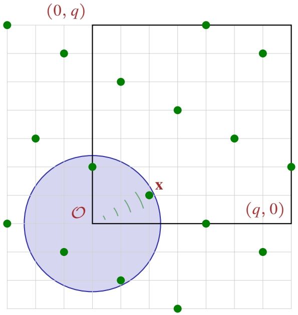
A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \bmod q$$

Putting it all together:

1. Solutions $\mathbf{z}$ form a 'hard' lattice $\mathcal{L} \subseteq \mathbb{Z}^m$
2. [Ajtai'99,AlwenP'09]: can generate uniform $\mathbf{A}$ together with a short basis $\mathbf{S}$ (i.e., $\mathbf{AS} = \mathbf{0}$).
3. Gaussian $\mathbf{x} \leftrightarrow$ syndrome $\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})$



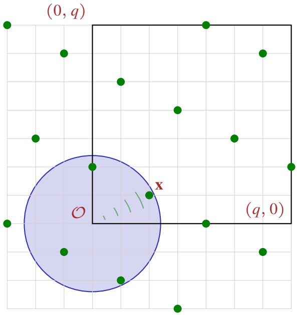
A Secure Instantiation [Ajtai96,...]

- Let $n =$ sec param, $q = \text{poly}(n) \longrightarrow$ additive group $\mathbb{Z}_q^n$
- Given $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, consider integer solutions $\mathbf{z} \in \mathbb{Z}^m$ of:

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \underbrace{\begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{pmatrix}}_{m \gg n} \begin{pmatrix} \\ \mathbf{z} \\ \\ \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{0} \\ | \end{pmatrix} \bmod q$$

Putting it all together:

1. Solutions $\mathbf{z}$ form a 'hard' lattice $\mathcal{L} \subseteq \mathbb{Z}^m$
2. [Ajtai'99,AlwenP'09]: can generate uniform $\mathbf{A}$ together with a short basis $\mathbf{S}$ (i.e., $\mathbf{AS} = \mathbf{0}$).
3. Gaussian $\mathbf{x} \leftrightarrow$ syndrome $\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})$
   - ⋆ Given $\mathbf{u}$, hard to find short $\mathbf{x} \in f_{\mathbf{A}}^{-1}(\mathbf{u})$.
   - ⋆ But given basis $\mathbf{S}$, can sample $f_{\mathbf{A}}^{-1}(\mathbf{u})$!



Part 2:

Identity-Based Encryption

Identity-Based Encryption

- Proposed by [Shamir'84]:

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - ‘Master’ keys mpk (public) and msk (held by trusted authority)

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - 'Master' keys mpk (public) and msk (held by trusted authority)
  - Given mpk, can encrypt to ID "Alice" or "Bob" or ...

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - 'Master' keys mpk (public) and msk (held by trusted authority)
  - Given mpk, can encrypt to ID "Alice" or "Bob" or ...
  - Using msk, authority can calculate sk_{Alice} or sk_{Bob} or ...

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - ‘Master’ keys mpk (public) and msk (held by trusted authority)
  - Given mpk, can encrypt to ID “Alice” or “Bob” or …
  - Using msk, authority can calculate sk_{Alice} or sk_{Bob} or …
  - Messages to Carol remain secret, even given $sk_{\text{Alice}}, sk_{\text{Bob}}, \ldots$

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - ‘Master’ keys mpk (public) and msk (held by trusted authority)
  - Given mpk, can encrypt to ID “Alice” or “Bob” or ...
  - Using msk, authority can calculate sk_{Alice} or sk_{Bob} or ...
  - Messages to Carol remain secret, even given $sk_{\text{Alice}}, sk_{\text{Bob}}, \ldots$

(Fast-forward 17 years...)

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - ★ 'Master' keys mpk (public) and msk (held by trusted authority)
  - ★ Given mpk, can encrypt to ID "Alice" or "Bob" or ...
  - ★ Using msk, authority can calculate sk_{Alice} or sk_{Bob} or ...
  - ★ Messages to Carol remain secret, even given $sk_{\text{Alice}}, sk_{\text{Bob}}, \ldots$

(Fast-forward 17 years...)

- [BonehFranklin'01,...]: construction using bilinear pairings

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - ‘Master’ keys mpk (public) and msk (held by trusted authority)
  - Given mpk, can encrypt to ID “Alice” or “Bob” or ...
  - Using msk, authority can calculate sk_{Alice} or sk_{Bob} or ...
  - Messages to Carol remain secret, even given $sk_{\text{Alice}}, sk_{\text{Bob}}, \ldots$

(Fast-forward 17 years...)

- [BonehFranklin'01,...]: construction using bilinear pairings
- [Cocks'01,BGH'07]: quadratic residuosity (mod $N = pq$)

Identity-Based Encryption

- Proposed by [Shamir'84]:
  - ★ 'Master' keys mpk (public) and msk (held by trusted authority)
  - ★ Given mpk, can encrypt to ID "Alice" or "Bob" or ...
  - ★ Using msk, authority can calculate sk_{Alice} or sk_{Bob} or ...
  - ★ Messages to Carol remain secret, even given $sk_{\text{Alice}}, sk_{\text{Bob}}, \ldots$

(Fast-forward 17 years...)

- [BonehFranklin'01,...]: construction using bilinear pairings
- [Cocks'01,BGH'07]: quadratic residuosity (mod $N = pq$)
- [GPV'08]: lattices!

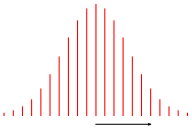
‘Learning With Errors’ (LWE) Problem [Regev’05]

- Secret $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a}_i \in \mathbb{Z}_q^n$ (here q is prime)

‘Learning With Errors’ (LWE) Problem [Regev’05]

- Secret $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a}_i \in \mathbb{Z}_q^n$ (here q is prime)
- **Goal:** distinguish $(\mathbf{a}_i \ , \ b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from uniform $(\mathbf{a}_i \ , \ b_i)$

$$
\begin{array}{ccc}
\mathbf{a}_1 & , & b_1 = \langle \mathbf{a}_1 \ , \ \mathbf{s} \rangle + e_1 \\
\mathbf{a}_2 & , & b_2 = \langle \mathbf{a}_2 \ , \ \mathbf{s} \rangle + e_2 \\
 & \vdots &
\end{array}
$$

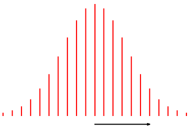


$\sqrt{n} \leq$ error $\ll q$

‘Learning With Errors’ (LWE) Problem [Regev’05]

- Secret $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a}_i \in \mathbb{Z}_q^n$ (here q is prime)
- **Goal:** distinguish $(\mathbf{A}\ ,\ \mathbf{b} = \mathbf{A}^t\mathbf{s} + \mathbf{e})$ from uniform $(\mathbf{A}\ ,\ \mathbf{b})$

$$m\left\{\begin{pmatrix} & \vdots & \\ & \mathbf{A}^t & \\ & \vdots & \end{pmatrix}\right. \quad , \quad \begin{pmatrix} \vdots \\ \mathbf{b} \\ \vdots \end{pmatrix} = \mathbf{A}^t\mathbf{s} + \mathbf{e}$$

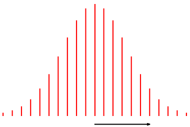


$\sqrt{n} \leq$ error $\ll q$

‘Learning With Errors’ (LWE) Problem [Regev’05]

- Secret $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a}_i \in \mathbb{Z}_q^n$ (here q is prime)
- **Goal:** distinguish $(\mathbf{A}\ ,\ \mathbf{b} = \mathbf{A}^t\mathbf{s} + \mathbf{e})$ from uniform $(\mathbf{A}\ ,\ \mathbf{b})$

$$m\left\{\begin{pmatrix} \vdots \\ \mathbf{A}^t \\ \vdots \end{pmatrix}\right. \quad , \quad \begin{pmatrix} \vdots \\ \mathbf{b} \\ \vdots \end{pmatrix} = \mathbf{A}^t\mathbf{s} + \mathbf{e}$$



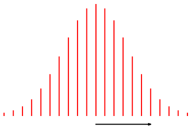
$\sqrt{n} \leq$ error $\ll q$

- Recall: as hard as worst-case lattice problems [Regev’05,P’09]

'Learning With Errors' (LWE) Problem [Regev'05]

- Secret $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a}_i \in \mathbb{Z}_q^n$ (here q is prime)
- **Goal:** distinguish $(\mathbf{A}\ ,\ \mathbf{b} = \mathbf{A}^t\mathbf{s} + \mathbf{e})$ from uniform $(\mathbf{A}\ ,\ \mathbf{b})$

$$m\left\{\begin{pmatrix} \vdots \\ \mathbf{A}^t \\ \vdots \end{pmatrix}\right. \quad , \quad \begin{pmatrix} \vdots \\ \mathbf{b} \\ \vdots \end{pmatrix} = \mathbf{A}^t\mathbf{s} + \mathbf{e}$$



$\sqrt{n} \leq$ error $\ll q$

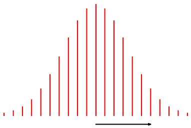
- Recall: as hard as worst-case lattice problems [Regev'05,P'09]
- Observe: given short nonzero $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$,

$$\begin{aligned} \langle \mathbf{z}, \mathbf{b} \rangle &= \langle \mathbf{A}\mathbf{z}, \mathbf{s} \rangle + \langle \mathbf{z}, \mathbf{e} \rangle \approx 0 \bmod q \\ \langle \mathbf{z}, \mathbf{b} \rangle &= \text{uniform} \bmod q \end{aligned}$$

‘Learning With Errors’ (LWE) Problem [Regev’05]

- Secret $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a}_i \in \mathbb{Z}_q^n$ (here q is prime)
- **Goal:** distinguish $(\mathbf{A}\ ,\ \mathbf{b} = \mathbf{A}^t\mathbf{s} + \mathbf{e})$ from uniform $(\mathbf{A}\ ,\ \mathbf{b})$

$$m \left\{ \begin{pmatrix} \vdots \\ \mathbf{A}^t \\ \vdots \end{pmatrix} \right. \quad , \quad \begin{pmatrix} \vdots \\ \mathbf{b} \\ \vdots \end{pmatrix} = \mathbf{A}^t\mathbf{s} + \mathbf{e}$$



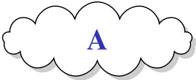
$\sqrt{n} \leq$ error $\ll q$

- Recall: as hard as worst-case lattice problems [Regev’05,P’09]
- Observe: given short nonzero $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Az} = \mathbf{0} \bmod q$,

$$\begin{aligned} \langle \mathbf{z}, \mathbf{b} \rangle &= \langle \mathbf{Az}, \mathbf{s} \rangle + \langle \mathbf{z}, \mathbf{e} \rangle \approx 0 \bmod q \\ \langle \mathbf{z}, \mathbf{b} \rangle &= \text{uniform} \bmod q \end{aligned}$$

$\Longrightarrow$ $\mathbf{z}$ is a ‘weak’ trapdoor, for distinguishing LWE from uniform

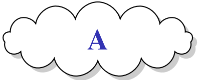
Warm-Up: Public-Key Encryption



$\mathbf{x} \leftarrow$ Gauss

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

Warm-Up: Public-Key Encryption

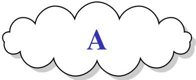


 $\mathbf{x} \leftarrow \text{Gauss}$

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

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

Warm-Up: Public-Key Encryption



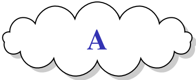
$\mathbf{x} \leftarrow \text{Gauss}$

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

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

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

Warm-Up: Public-Key Encryption



$\mathbf{x} \leftarrow$ Gauss

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

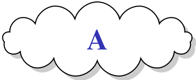
$$\xrightarrow[\text{(public key)}]{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

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

$$\boxed{b' = \langle \mathbf{u}, \mathbf{s} \rangle + e'}$$

(‘pad’)

Warm-Up: Public-Key Encryption



$\mathbf{x} \leftarrow$ Gauss

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

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

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

$$\xleftarrow[\text{(‘payload’)}]{b' + \text{bit} \cdot \lfloor \frac{q}{2} \rfloor}$$

$$\boxed{b' = \langle \mathbf{u}, \mathbf{s} \rangle + e'}$$

(‘pad’)

Warm-Up: Public-Key Encryption

$\mathbf{A}$

 $\mathbf{x} \leftarrow$ Gauss

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

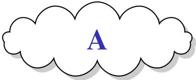
$$\xrightarrow[\text{(public key)}]{\mathbf{u} = \mathbf{A}\mathbf{x} = f_{\mathbf{A}}(\mathbf{x})}$$

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

$$\langle \mathbf{x}, \mathbf{b} \rangle \approx \langle \mathbf{u}, \mathbf{s} \rangle \qquad \xleftarrow[\text{(‘payload’)}]{b' + \text{bit} \cdot \lfloor \frac{q}{2} \rfloor} \qquad \boxed{b' = \langle \mathbf{u}, \mathbf{s} \rangle + e'}$$

(‘pad’)

Warm-Up: Public-Key Encryption



$\mathbf{x} \leftarrow$ Gauss

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

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

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

$\langle \mathbf{x}, \mathbf{b} \rangle \approx \langle \mathbf{u}, \mathbf{s} \rangle$

$$\xleftarrow[\text{(‘payload’)}]{b' + \text{bit} \cdot \lfloor \frac{q}{2} \rfloor}$$

$\boxed{b' = \langle \mathbf{u}, \mathbf{s} \rangle + e'}$

(‘pad’)

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

Warm-Up: Public-Key Encryption

$\mathbf{A}$

 $\mathbf{x} \leftarrow$ Gauss

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

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

(public key)

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

(ciphertext 'preamble')

$\langle \mathbf{x}, \mathbf{b} \rangle \approx \langle \mathbf{u}, \mathbf{s} \rangle$

$$\longleftarrow b' + \text{bit} \cdot \lfloor \tfrac{q}{2} \rceil$$

('payload')

$$\boxed{b' = \langle \mathbf{u}, \mathbf{s} \rangle + e'}$$

('pad')

- - - - - - - - - - - - - - -

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

ID-Based Encryption

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

$mpk = \mathbf{A}$

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

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

(‘identity’ key)

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

(ciphertext randomness)

$$\langle \mathbf{x}, \mathbf{b} \rangle \approx \langle \mathbf{u}, \mathbf{s} \rangle \qquad \xleftarrow{b' + \text{bit} \cdot \lfloor \frac{q}{2} \rfloor} \qquad b' = \langle \mathbf{u}, \mathbf{s} \rangle + e'$$

(‘payload’)

(‘pad’)

Part 3:

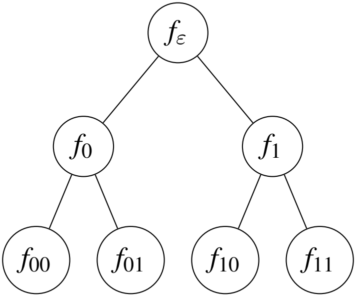
Bonsai Trees:
Removing the Random Oracle and More Advanced Applications

CONTROLLED or NATURAL ?

CONTROLLED or NATURAL ?

- Bonsai: collection of techniques for selective control of tree growth, for the creation of natural aesthetic forms

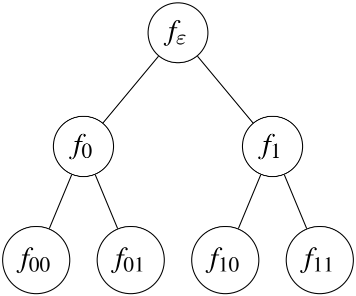
Bonsai Trees in Cryptography



1. Hierarchy of TDFs

   (Functions specified by public key, random oracle, interaction, . . .)

Bonsai Trees in Cryptography

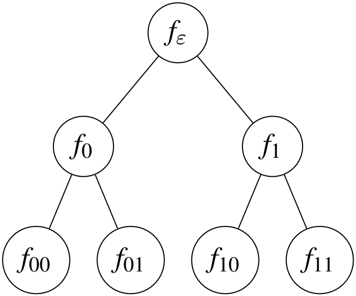


1. Hierarchy of TDFs

   (Functions specified by public key, random oracle, interaction, ...)

2. Techniques for selective 'control' of growth & delegation of control

Bonsai Trees in Cryptography



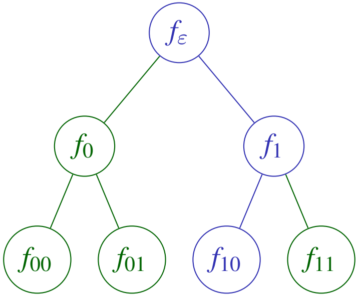
1. Hierarchy of TDFs

   (Functions specified by public key, random oracle, interaction, . . .)

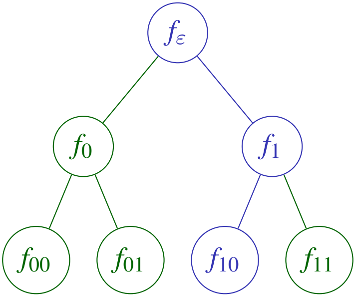
2. Techniques for selective 'control' of growth & delegation of control

3. Applications: 'hash-and-sign,' (hierarchical) IBE
   — all without random oracles!

Bonsai Trees: Abstract Properties

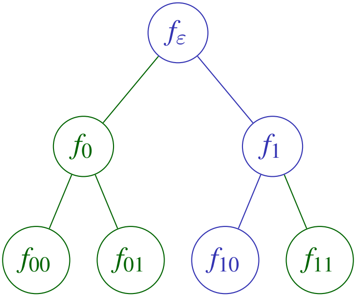


Bonsai Trees: Abstract Properties



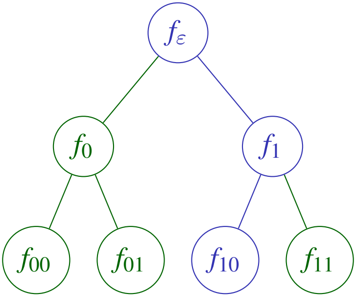
1. Controlling f_v (knowing trapdoor) $\Longrightarrow$ controlling f_{vz}, for all z.

Bonsai Trees: Abstract Properties



1. Controlling f_v (knowing trapdoor) $\Longrightarrow$ controlling f_{vz}, for all z.
2. Can grow a controlled branch off of any uncontrolled node.

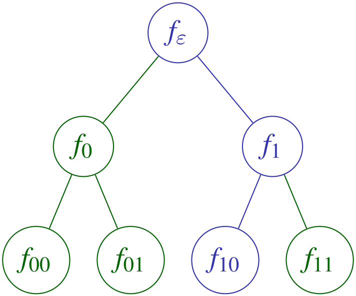
Bonsai Trees: Abstract Properties



1. Controlling f_v (knowing trapdoor) $\Longrightarrow$ controlling f_{vz}, for all z.
2. Can grow a controlled branch off of any uncontrolled node.

   (Allows simulation to embed its challenge into the tree, while still being able to answer queries.)

Bonsai Trees: Abstract Properties



1. Controlling f_v (knowing trapdoor) $\Longrightarrow$ controlling f_{vz}, for all z.
2. Can grow a controlled branch off of any uncontrolled node.

   (Allows simulation to embed its challenge into the tree, while still being able to answer queries.)
3. Can delegate control of any subtree, w/o endangering ancestors.

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

- Using $\mathbf{S}_1$, compute a short integer soln $\mathbf{X}$ to $\mathbf{A}_1\mathbf{X} = -\mathbf{A}_2 \bmod q$. Then:

$$\mathbf{A} \cdot \mathbf{S} = [\mathbf{A}_1 \mid \mathbf{A}_2] \cdot \underbrace{\begin{bmatrix} \mathbf{S}_1 & \mathbf{X} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} = \mathbf{0} \bmod q.$$

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

- Using $\mathbf{S}_1$, compute a short integer soln $\mathbf{X}$ to $\mathbf{A}_1 \mathbf{X} = -\mathbf{A}_2 \bmod q$. Then:

$$\mathbf{A} \cdot \mathbf{S} = [\mathbf{A}_1 \mid \mathbf{A}_2] \cdot \underbrace{\begin{bmatrix} \mathbf{S}_1 & \mathbf{X} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} = \mathbf{0} \bmod q.$$

(In fact, $\mathbf{X}$ need not be short — we have $\tilde{\mathbf{S}} = \left(\begin{smallmatrix} \tilde{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$, so $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_1\|$.)

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

- Using $\mathbf{S}_1$, compute a short integer soln $\mathbf{X}$ to $\mathbf{A}_1\mathbf{X} = -\mathbf{A}_2 \bmod q$.
  Then:
  $$\mathbf{A} \cdot \mathbf{S} = [\mathbf{A}_1 \mid \mathbf{A}_2] \cdot \underbrace{\begin{bmatrix} \mathbf{S}_1 & \mathbf{X} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} = \mathbf{0} \bmod q.$$

  (In fact, $\mathbf{X}$ need not be short — we have $\tilde{\mathbf{S}} = \left(\begin{smallmatrix} \tilde{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$, so $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_1\|$.)

Property 2: Grow a Controlled Branch

Given (uncontrolled) $\mathbf{A}_1$, create controlled extension $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$.

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

- Using $\mathbf{S}_1$, compute a short integer soln $\mathbf{X}$ to $\mathbf{A}_1\mathbf{X} = -\mathbf{A}_2 \bmod q$. Then:

$$\mathbf{A} \cdot \mathbf{S} = [\mathbf{A}_1 \mid \mathbf{A}_2] \cdot \underbrace{\begin{bmatrix} \mathbf{S}_1 & \mathbf{X} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} = \mathbf{0} \bmod q.$$

(In fact, $\mathbf{X}$ need not be short — we have $\tilde{\mathbf{S}} = \left(\begin{smallmatrix} \tilde{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$, so $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_1\|$.)

Property 2: Grow a Controlled Branch

Given (uncontrolled) $\mathbf{A}_1$, create controlled extension $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$.

- Just generate $\mathbf{A}_2$ with short basis $\mathbf{S}_2$.

  Then use above technique to control $\mathbf{A}$!

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

- Using $\mathbf{S}_1$, compute a short integer soln $\mathbf{X}$ to $\mathbf{A}_1 \mathbf{X} = -\mathbf{A}_2 \bmod q$. Then:

$$\mathbf{A} \cdot \mathbf{S} = [\mathbf{A}_1 \mid \mathbf{A}_2] \cdot \underbrace{\begin{bmatrix} \mathbf{S}_1 & \mathbf{X} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} = \mathbf{0} \bmod q.$$

(In fact, $\mathbf{X}$ need not be short — we have $\tilde{\mathbf{S}} = \begin{pmatrix} \tilde{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$, so $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_1\|$.)

Property 3: Securely Delegate Control ?

- Basis $\mathbf{S}$ contains $\mathbf{S}_1$, so unsafe to reveal!

Bonsai Trees: Realization

Property 1: Control $f_v \Rightarrow$ Control f_{vz}

Short basis $\mathbf{S}_1$ for $\mathbf{A}_1 \Rightarrow$ short basis $\mathbf{S}$ for $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$, for any $\mathbf{A}_2$.

- Using $\mathbf{S}_1$, compute a short integer soln $\mathbf{X}$ to $\mathbf{A}_1 \mathbf{X} = -\mathbf{A}_2 \bmod q$. Then:

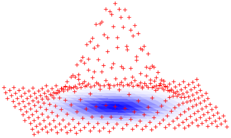
$$\mathbf{A} \cdot \mathbf{S} = [\mathbf{A}_1 \mid \mathbf{A}_2] \cdot \underbrace{\begin{bmatrix} \mathbf{S}_1 & \mathbf{X} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} = \mathbf{0} \bmod q.$$

(In fact, $\mathbf{X}$ need not be short — we have $\tilde{\mathbf{S}} = \left(\begin{smallmatrix} \tilde{\mathbf{S}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$, so $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_1\|$.)

Property 3: Securely Delegate Control ?

- Basis $\mathbf{S}$ contains $\mathbf{S}_1$, so unsafe to reveal!

  Solution: Use $\mathbf{S}$ to sample new *Gaussian* basis.



Other Applications of Today's Tools

1. Noninteractive (Statistical) Zero Knowledge [PV'08]
2. Universally Composable Oblivious Transfer [PVW'08]
3. CCA-Secure Encryption [P'09]
4. Many-add, Single-mult Homomorphic Encryption [GHV'10]
5. Bonsai trees with smaller keys [ABB'10]
6. Group signatures [GKV'10]
7. (Bi-)Deniable Encryption [OPW'11]
8. Whatever you can invent!

Closing Thoughts

- A hierarchy of trapdoors for lattices:

  Short vector (decryption)

  $<$ Short basis (sampling)

  $<$ Short basis for 'ancestor' lattice (delegation)

  $< \cdots$

Closing Thoughts

- A hierarchy of trapdoors for lattices:

  Short vector (decryption)

  $<$ Short basis (sampling)

  $<$ Short basis for 'ancestor' lattice (delegation)

  $< \cdots$

Thanks!