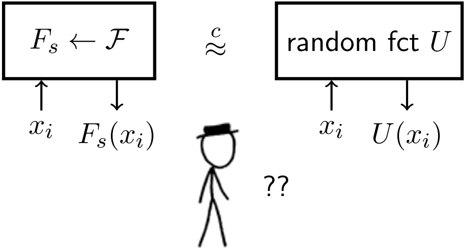
Session #6: Another Application of LWE: Pseudorandom Functions

Chris Peikert
Georgia Institute of Technology

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

Pseudorandom Functions [GGM'84]

- A family $\mathcal{F} = \{F_s : \{0,1\}^k \to D\}$ s.t. given adaptive query access,

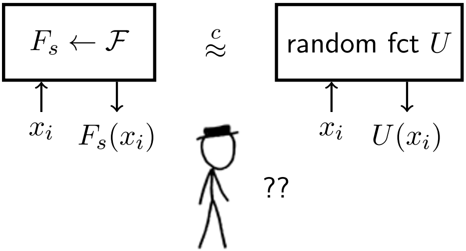


(The "seed" or "secret key" for F_s is s.)

(Images courtesy xkcd.org)

Pseudorandom Functions [GGM'84]

- A family $\mathcal{F} = \{F_s : \{0,1\}^k \to D\}$ s.t. given adaptive query access,



(The "seed" or "secret key" for F_s is s.)

- Countless applications in symmetric cryptography: (efficient) encryption, authentication, friend-or-foe . . .

(Images courtesy xkcd.org)

How to Construct PRFs

1. Heuristically: AES etc.
   - ✔ Fast!
   - ✔ Withstand known cryptanalytic techniques (linear, differential, ...)

How to Construct PRFs

1. Heuristically: AES etc.
   - ✔ Fast!
   - ✔ Withstand known cryptanalytic techniques (linear, differential, ...)
   - ✗ PRF security is subtle: want provable (reductionist) guarantees

How to Construct PRFs

1. Heuristically: AES etc.
   - ✔ Fast!
   - ✔ Withstand known cryptanalytic techniques (linear, differential, . . .)
   - ✗ PRF security is subtle: want provable (reductionist) guarantees

2. Goldreich-Goldwasser-Micali [GGM'84]
   - ✔ Based on any (doubling) PRG. $F_s(x_1 \cdots x_k) = G_{x_k}(\cdots G_{x_1}(s) \cdots)$

How to Construct PRFs

1. Heuristically: AES etc.
   - ✔ Fast!
   - ✔ Withstand known cryptanalytic techniques (linear, differential, ...)
   - ✗ PRF security is subtle: want provable (reductionist) guarantees

2. Goldreich-Goldwasser-Micali [GGM'84]
   - ✔ Based on any (doubling) PRG. $F_s(x_1 \cdots x_k) = G_{x_k}(\cdots G_{x_1}(s) \cdots)$
   - ✗ Inherently sequential: $\geq k$ iterations (circuit depth)

How to Construct PRFs

1. Heuristically: AES etc.
   - ✔ Fast!
   - ✔ Withstand known cryptanalytic techniques (linear, differential, ...)
   - ✗ PRF security is subtle: want provable (reductionist) guarantees

2. Goldreich-Goldwasser-Micali [GGM'84]
   - ✔ Based on any (doubling) PRG. $F_s(x_1 \cdots x_k) = G_{x_k}(\cdots G_{x_1}(s) \cdots)$
   - ✗ Inherently sequential: $\geq k$ iterations (circuit depth)

3. Naor-Reingold(-Rosen) [NR'95,NR'97,NRR'00]
   - ✔ Based on "synthesizers" or number theory (DDH, factoring)
   - ✔ Low-depth: NC^2, NC^1 or even TC^0 [$O(1)$ depth w/ threshold gates]

How to Construct PRFs

1. Heuristically: AES etc.
   - ✔ Fast!
   - ✔ Withstand known cryptanalytic techniques (linear, differential, ...)
   - ✗ PRF security is subtle: want provable (reductionist) guarantees

2. Goldreich-Goldwasser-Micali [GGM'84]
   - ✔ Based on any (doubling) PRG. $F_s(x_1 \cdots x_k) = G_{x_k}(\cdots G_{x_1}(s) \cdots)$
   - ✗ Inherently sequential: $\geq k$ iterations (circuit depth)

3. Naor-Reingold(-Rosen) [NR'95,NR'97,NRR'00]
   - ✔ Based on "synthesizers" or number theory (DDH, factoring)
   - ✔ Low-depth: NC^2, NC^1 or even TC^0 [$O(1)$ depth w/ threshold gates]
   - ✗ Huge circuits that need much preprocessing
   - ✗ No "post-quantum" construction under standard assumptions

PRFs from Lattices?

The Hope

- Lattices $\Rightarrow$ simple, highly parallel, practically efficient ... PRFs?

PRFs from Lattices?

The Hope

- Lattices $\Rightarrow$ simple, highly parallel, practically efficient . . . PRFs?

The Reality

✗ Only known PRF is generic GGM (not parallel or very efficient)

PRFs from Lattices?

The Hope

- Lattices $\Rightarrow$ simple, highly parallel, practically efficient . . . PRFs?

The Reality

✗ Only known PRF is generic GGM (not parallel or very efficient)

✗✗ We don't even have practical PRGs from lattices: biased errors

PRFs from Lattices?

The Hope

- Lattices ⇒ simple, highly parallel, practically efficient . . . PRFs?

The Reality

✗ Only known PRF is generic GGM (not parallel or very efficient)

✗✗ We don't even have practical PRGs from lattices: biased errors

New Results [BPR'12]

1. Low-depth, relatively small-circuit PRFs from lattices / (ring-)LWE

PRFs from Lattices?

The Hope

- Lattices $\Rightarrow$ simple, highly parallel, practically efficient ... PRFs?

The Reality

✗ Only known PRF is generic GGM (not parallel or very efficient)

✗✗ We don't even have practical PRGs from lattices: biased errors

New Results [BPR'12]

1. Low-depth, relatively small-circuit PRFs from lattices / (ring-)LWE
   - ⋆ Synthesizer-based PRF in $\mathsf{TC}^1 \subseteq \mathsf{NC}^2$ *a la* [NR'95]
   - ⋆ Direct construction in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$ analogous to [NR'97,NRR'00]

PRFs from Lattices?

The Hope

- Lattices $\Rightarrow$ simple, highly parallel, practically efficient ... PRFs?

The Reality

✗ Only known PRF is generic GGM (not parallel or very efficient)

✗✗ We don't even have practical PRGs from lattices: biased errors

New Results [BPR'12]

1. Low-depth, relatively small-circuit PRFs from lattices / (ring-)LWE
   - ⋆ Synthesizer-based PRF in $\mathsf{TC}^1 \subseteq \mathsf{NC}^2$ *a la* [NR'95]
   - ⋆ Direct construction in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$ analogous to [NR'97,NRR'00]
2. Main technique: "derandomization" of LWE: deterministic errors

Synthesizers and PRFs [NaorReingold'95]

Synthesizer

- A deterministic function $S\colon D \times D \to D$ s.t. for any $m =$ poly: for uniform $a_1, \ldots, a_m,\ b_1, \ldots, b_m \leftarrow D$,

$$\{\, S(a_i\,,\, b_j)\,\} \overset{c}{\approx} \text{Unif}(D^{m \times m}).$$

Synthesizers and PRFs [NaorReingold'95]

Synthesizer

- A deterministic function $S\colon D \times D \to D$ s.t. for any $m =$ poly: for uniform $a_1, \ldots, a_m,\ b_1, \ldots, b_m \leftarrow D$,

$$\{\, S(a_i\,,\, b_j) \,\} \overset{c}{\approx} \text{Unif}(D^{m \times m}).$$

	b_1	b_2	$\cdots$
a_1	$S(a_1, b_1)$	$S(a_1, b_2)$	$\cdots$
a_2	$S(a_2, b_1)$	$S(a_2, b_2)$	$\cdots$
$\vdots$		$\ddots$	

vs.

	$U_{1,1}$	$U_{1,2}$	$\cdots$
	$U_{2,1}$	$U_{2,2}$	$\cdots$
		$\ddots$	

Synthesizers and PRFs [NaorReingold'95]

Synthesizer

- A deterministic function $S \colon D \times D \to D$ s.t. for any $m = \text{poly}$: for uniform $a_1, \ldots, a_m, \; b_1, \ldots, b_m \leftarrow D$,

$$\{ S(a_i , b_j) \} \stackrel{c}{\approx} \text{Unif}(D^{m \times m}).$$

	b_1	b_2	$\cdots$
a_1	$S(a_1, b_1)$	$S(a_1, b_2)$	$\cdots$
a_2	$S(a_2, b_1)$	$S(a_2, b_2)$	$\cdots$
$\vdots$		$\ddots$	

vs.

$U_{1,1}$	$U_{1,2}$	$\cdots$
$U_{2,1}$	$U_{2,2}$	$\cdots$
	$\ddots$	

- Alternative view: an (almost) length-squaring PRG with locality: maps $D^{2m} \to D^{m^2}$, and each output depends on only 2 inputs.

Synthesizers and PRFs [NaorReingold'95]

PRF from Synthesizer, Recursively

- Synthesizer $S \colon D \times D \to D$, where $\{ S(a_i , b_j) \} \stackrel{c}{\approx} \mathsf{Unif}(D^{m \times m})$.

Synthesizers and PRFs [NaorReingold'95]

PRF from Synthesizer, Recursively

- Synthesizer $S\colon D \times D \to D$, where $\{\, S(a_i\,,\, b_j) \,\} \overset{c}{\approx}$ Unif$(D^{m \times m})$.
- Base case: "one-bit" PRF $F_{s_0,s_1}(x) := s_x \in D$. ✔

Synthesizers and PRFs [NaorReingold'95]

PRF from Synthesizer, Recursively

- Synthesizer $S \colon D \times D \to D$, where $\{\, S(a_i\,,\, b_j)\,\} \overset{c}{\approx}$ Unif$(D^{m \times m})$.
- Base case: "one-bit" PRF $F_{s_0,s_1}(x) := s_x \in D$. ✔
- Input doubling: given k-bit PRF family $\mathcal{F} = \{F \colon \{0,1\}^k \to D\}$, define a $\{0,1\}^{2k} \to D$ function with seed $F_\ell, F_r \leftarrow \mathcal{F}$:

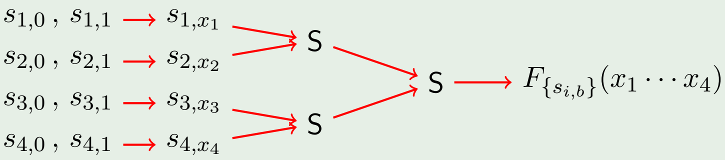
$$F_{(F_\ell,F_r)}(x_\ell\,,\, x_r) = S\big(\, F_\ell(x_\ell)\,,\, F_r(x_r)\,\big).$$

Synthesizers and PRFs [NaorReingold'95]

PRF from Synthesizer, Recursively

- Synthesizer $S\colon D \times D \to D$, where $\{\, S(a_i\,,\, b_j) \,\} \overset{c}{\approx} \mathsf{Unif}(D^{m \times m})$.
- Base case: "one-bit" PRF $F_{s_0,s_1}(x) := s_x \in D$. ✔
- Input doubling: given k-bit PRF family $\mathcal{F} = \{F\colon \{0,1\}^k \to D\}$, define a $\{0,1\}^{2k} \to D$ function with seed $F_\ell, F_r \leftarrow \mathcal{F}$:

$$F_{(F_\ell,F_r)}(x_\ell\,,\, x_r) = S\big(\, F_\ell(x_\ell)\,,\, F_r(x_r)\,\big).$$

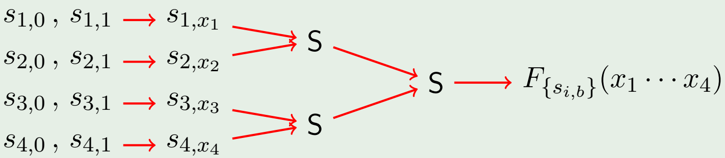


Synthesizers and PRFs [NaorReingold'95]

PRF from Synthesizer, Recursively

- Synthesizer $S \colon D \times D \to D$, where $\{ S(a_i , b_j) \} \stackrel{c}{\approx}$ Unif$(D^{m \times m})$.
- Base case: "one-bit" PRF $F_{s_0,s_1}(x) := s_x \in D$. ✔
- Input doubling: given k-bit PRF family $\mathcal{F} = \{F \colon \{0,1\}^k \to D\}$, define a $\{0,1\}^{2k} \to D$ function with seed $F_\ell, F_r \leftarrow \mathcal{F}$:

$$F_{(F_\ell,F_r)}(x_\ell , x_r) = S\big(F_\ell(x_\ell) , F_r(x_r) \big).$$



- Security: the queries $F_\ell(x_\ell)$ and $F_r(x_r)$ define (pseudo)random inputs $a_1, a_2, \ldots \in D$ and $b_1, b_2, \ldots \in D$ to synthesizer S.

LWE $\Rightarrow$ Synthesizer?

- Hard to distinguish pairs $(\mathbf{a}_i \in \mathbb{Z}_q^n \, , \, b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from $(\mathbf{a}_i \, , \, b_i)$.

LWE $\Rightarrow$ Synthesizer?

- Hard to distinguish pairs $(\mathbf{a}_i \in \mathbb{Z}_q^n \;,\; b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from $(\mathbf{a}_i \;,\; b_i)$.
- By hybrid argument, can't distinguish tuples

$$(\mathbf{A}_i \in \mathbb{Z}_q^{n \times n} \;,\; \mathbf{A}_i \cdot \mathbf{S}_1 + \mathbf{E}_{i,1} \in \mathbb{Z}_q^{n \times n} \;,\; \mathbf{A}_i \cdot \mathbf{S}_2 + \mathbf{E}_{i,2} \in \mathbb{Z}_q^{n \times n} \;,\; \ldots)$$

LWE $\Rightarrow$ Synthesizer?

- Hard to distinguish pairs $(\mathbf{a}_i \in \mathbb{Z}_q^n \ , \ b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from $(\mathbf{a}_i \ , \ b_i)$.
- By hybrid argument, can't distinguish tuples

$$(\mathbf{A}_i \in \mathbb{Z}_q^{n \times n} \ , \ \mathbf{A}_i \cdot \mathbf{S}_1 + \mathbf{E}_{i,1} \in \mathbb{Z}_q^{n \times n} \ , \ \mathbf{A}_i \cdot \mathbf{S}_2 + \mathbf{E}_{i,2} \in \mathbb{Z}_q^{n \times n} \ , \ \ldots)$$

An LWE-Based Synthesizer?

	$\mathbf{S}_1$	$\mathbf{S}_2$	$\cdots$
$\mathbf{A}_1$	$\mathbf{A}_1 \cdot \mathbf{S}_1 + \mathbf{E}_{1,1}$	$\mathbf{A}_1 \cdot \mathbf{S}_2 + \mathbf{E}_{1,2}$	$\cdots$
$\mathbf{A}_2$	$\mathbf{A}_2 \cdot \mathbf{S}_1 + \mathbf{E}_{2,1}$	$\mathbf{A}_2 \cdot \mathbf{S}_2 + \mathbf{E}_{2,2}$	$\cdots$
$\vdots$		$\ddots$	

LWE $\Rightarrow$ Synthesizer?

- Hard to distinguish pairs $(\mathbf{a}_i \in \mathbb{Z}_q^n\ ,\ b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from $(\mathbf{a}_i\ ,\ b_i)$.
- By hybrid argument, can't distinguish tuples

$$(\mathbf{A}_i \in \mathbb{Z}_q^{n\times n}\ ,\ \mathbf{A}_i \cdot \mathbf{S}_1 + \mathbf{E}_{i,1} \in \mathbb{Z}_q^{n\times n}\ ,\ \mathbf{A}_i \cdot \mathbf{S}_2 + \mathbf{E}_{i,2} \in \mathbb{Z}_q^{n\times n}\ ,\ \ldots)$$

An LWE-Based Synthesizer?

	$\mathbf{S}_1$	$\mathbf{S}_2$	$\cdots$
$\mathbf{A}_1$	$\mathbf{A}_1 \cdot \mathbf{S}_1 + \mathbf{E}_{1,1}$	$\mathbf{A}_1 \cdot \mathbf{S}_2 + \mathbf{E}_{1,2}$	$\cdots$
$\mathbf{A}_2$	$\mathbf{A}_2 \cdot \mathbf{S}_1 + \mathbf{E}_{2,1}$	$\mathbf{A}_2 \cdot \mathbf{S}_2 + \mathbf{E}_{2,2}$	$\cdots$
$\vdots$		$\ddots$	

✔ $\{\mathbf{A}_i \cdot \mathbf{S}_j + \mathbf{E}_{i,j}\} \overset{c}{\approx}$ Uniform, but…

LWE $\Rightarrow$ Synthesizer?

- Hard to distinguish pairs $(\mathbf{a}_i \in \mathbb{Z}_q^n \ , \ b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$ from $(\mathbf{a}_i \ , \ b_i)$.
- By hybrid argument, can't distinguish tuples

$$(\mathbf{A}_i \in \mathbb{Z}_q^{n \times n} \ , \ \mathbf{A}_i \cdot \mathbf{S}_1 + \mathbf{E}_{i,1} \in \mathbb{Z}_q^{n \times n} \ , \ \mathbf{A}_i \cdot \mathbf{S}_2 + \mathbf{E}_{i,2} \in \mathbb{Z}_q^{n \times n} \ , \ \ldots)$$

An LWE-Based Synthesizer?

	$\mathbf{S}_1$	$\mathbf{S}_2$	$\cdots$
$\mathbf{A}_1$	$\mathbf{A}_1 \cdot \mathbf{S}_1 + \mathbf{E}_{1,1}$	$\mathbf{A}_1 \cdot \mathbf{S}_2 + \mathbf{E}_{1,2}$	$\cdots$
$\mathbf{A}_2$	$\mathbf{A}_2 \cdot \mathbf{S}_1 + \mathbf{E}_{2,1}$	$\mathbf{A}_2 \cdot \mathbf{S}_2 + \mathbf{E}_{2,2}$	$\cdots$
$\vdots$		$\ddots$	

✔ $\{\mathbf{A}_i \cdot \mathbf{S}_j + \mathbf{E}_{i,j}\} \overset{c}{\approx}$ Uniform, but. . .

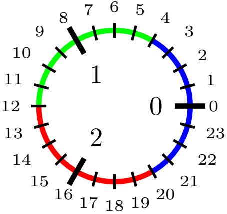
✗ What about $\mathbf{E}_{i,j}$?

Synthesizer must be deterministic. . .

"Learning With Rounding" (LWR) [BPR'12]

- IDEA: generate errors deterministically by rounding $\mathbb{Z}_q$ to a "sparse" subset (e.g. subgroup $\mathbb{Z}_p$).

  (Common in decryption to remove error.)

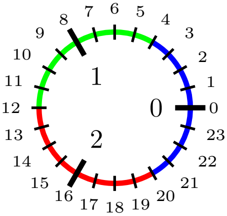


"Learning With Rounding" (LWR) [BPR'12]

- IDEA: generate errors deterministically by rounding $\mathbb{Z}_q$ to a "sparse" subset (e.g. subgroup $\mathbb{Z}_p$).

  (Common in decryption to remove error.)

  Let $p < q$ and define $\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p$.

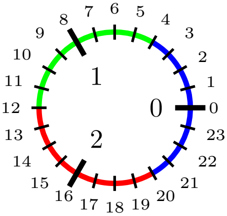


"Learning With Rounding" (LWR) [BPR'12]

- IDEA: generate errors deterministically by rounding $\mathbb{Z}_q$ to a "sparse" subset (e.g. subgroup $\mathbb{Z}_p$).

  (Common in decryption to remove error.)

  Let $p < q$ and define $\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p$.



- LWR problem: distinguish any $m =$ poly pairs

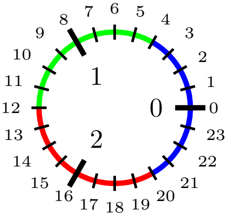
$$\left(\mathbf{a}_i\ ,\ \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p\right) \in \mathbb{Z}_q \times \mathbb{Z}_p \quad \text{from uniform}$$

"Learning With Rounding" (LWR) [BPR'12]

- IDEA: generate errors deterministically by rounding $\mathbb{Z}_q$ to a "sparse" subset (e.g. subgroup $\mathbb{Z}_p$).

  (Common in decryption to remove error.)

  Let $p < q$ and define $\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p$.



- LWR problem: distinguish any $m = \text{poly}$ pairs

$$\left(\mathbf{a}_i \ , \ \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p \right) \in \mathbb{Z}_q \times \mathbb{Z}_p \quad \text{from uniform}$$

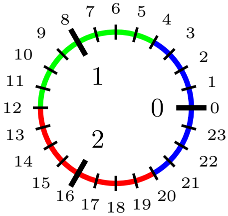
  Interpretation: LWE conceals low-order bits by adding small random error. LWR just discards those bits instead.

"Learning With Rounding" (LWR) [BPR'12]

- IDEA: generate errors deterministically by rounding $\mathbb{Z}_q$ to a "sparse" subset (e.g. subgroup $\mathbb{Z}_p$).

  (Common in decryption to remove error.)

  Let $p < q$ and define $\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p$.



- LWR problem: distinguish any $m =$ poly pairs

$$\left(\mathbf{a}_i \,,\ \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p\right) \in \mathbb{Z}_q \times \mathbb{Z}_p \quad \text{from uniform}$$

  Interpretation: LWE conceals low-order bits by adding small random error. LWR just discards those bits instead.

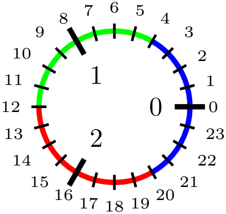
- We prove LWE $\leq$ LWR for $q \geq p \cdot n^{\omega(1)}$ [but it seems 2^n-hard for $q \geq p\sqrt{n}$]

“Learning With Rounding” (LWR) [BPR’12]

- IDEA: generate errors deterministically by rounding $\mathbb{Z}_q$ to a “sparse” subset (e.g. subgroup $\mathbb{Z}_p$).

  (Common in decryption to remove error.)

  Let $p < q$ and define $\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p$.



- LWR problem: distinguish any $m =$ poly pairs

$$\left(\mathbf{a}_i\ ,\ \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p\right) \in \mathbb{Z}_q \times \mathbb{Z}_p \quad \text{from uniform}$$

  Interpretation: LWE conceals low-order bits by adding small random error. LWR just discards those bits instead.

- We prove LWE $\leq$ LWR for $q \geq p \cdot n^{\omega(1)}$ [but it seems 2^n-hard for $q \geq p\sqrt{n}$]

  Proof idea: w.h.p., $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s} \rangle + e \rceil_p) = (\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p)$

  and $(\mathbf{a}, \lfloor \text{Unif}(\mathbb{Z}_q) \rceil_p) = (\mathbf{a}, \text{Unif}(\mathbb{Z}_p))$

LWR-Based Synthesizer & PRF

- Synthesizer $S\colon \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_p^{n\times n}$ is $\quad S(\mathbf{A}, \mathbf{S}) = \lfloor \mathbf{A} \cdot \mathbf{S} \rceil_p$.

  (Note: range $\mathbb{Z}_p$ is slightly smaller than domain $\mathbb{Z}_q$. Only limits composition.)

LWR-Based Synthesizer & PRF

- Synthesizer $S\colon \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_p^{n\times n}$ is $S(\mathbf{A},\mathbf{S}) = \lfloor \mathbf{A}\cdot\mathbf{S} \rceil_p$.

  (Note: range $\mathbb{Z}_p$ is slightly smaller than domain $\mathbb{Z}_q$. Only limits composition.)

PRF on Domain $\{0,1\}^{k=2^d}$

- "Tower" of public moduli $q_d > q_{d-1} > \cdots > q_0$.
- Secret key is $2k$ square matrices $\mathbf{S}_{i,b}$ over $\mathbb{Z}_{q_d}$ for $i \in [k]$, $b \in \{0,1\}$.

LWR-Based Synthesizer & PRF

- Synthesizer $S \colon \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_p^{n\times n}$ is $S(\mathbf{A}, \mathbf{S}) = \lfloor \mathbf{A} \cdot \mathbf{S} \rceil_p$.

  (Note: range $\mathbb{Z}_p$ is slightly smaller than domain $\mathbb{Z}_q$. Only limits composition.)

PRF on Domain $\{0,1\}^{k=2^d}$

- "Tower" of public moduli $q_d > q_{d-1} > \cdots > q_0$.
- Secret key is $2k$ square matrices $\mathbf{S}_{i,b}$ over $\mathbb{Z}_{q_d}$ for $i \in [k]$, $b \in \{0,1\}$.
- Depth $d = \lg k$ tree of LWR synthesizers:

$$F_{\{\mathbf{S}_{i,b}\}}(x_1 \cdots x_8) =$$
$$\left\lfloor \left\lfloor \lfloor \mathbf{S}_{1,x_1} \cdot \mathbf{S}_{2,x_2} \rceil_{q_2} \cdot \lfloor \mathbf{S}_{3,x_3} \cdot \mathbf{S}_{4,x_4} \rceil_{q_2} \right\rceil_{q_1} \cdot \left\lfloor \lfloor \mathbf{S}_{5,x_5} \cdot \mathbf{S}_{6,x_6} \rceil_{q_2} \cdot \lfloor \mathbf{S}_{7,x_7} \cdot \mathbf{S}_{8,x_8} \rceil_{q_2} \right\rceil_{q_1} \right\rceil_{q_0}$$

Shallower? More Efficient?

- Synth-based PRF is $\log k$ levels of NC^1 synthesizers $\Rightarrow$ NC^2.

Shallower? More Efficient?

- Synth-based PRF is $\log k$ levels of NC^1 synthesizers $\Rightarrow$ NC^2.
- [NR'97,NRR'00]: direct PRFs from DDH / factoring, in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$.

$$F_{g,s_1,\ldots,s_k}(x_1 \cdots x_k) = g^{\prod s_i^{x_i}}$$

(Computing this in TC^0 needs huge circuits, though...)

Shallower? More Efficient?

- Synth-based PRF is $\log k$ levels of NC^1 synthesizers $\Rightarrow$ NC^2.
- [NR'97,NRR'00]: direct PRFs from DDH / factoring, in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$.

$$F_{g,s_1,\ldots,s_k}(x_1 \cdots x_k) = g^{\prod s_i^{x_i}}$$

(Computing this in TC^0 needs huge circuits, though. . .)

Direct LWE-Based Construction

- Public moduli $q > p$.
- Secret key is uniform $\mathbf{A}$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$ over $\mathbb{Z}_q$.

Shallower? More Efficient?

- Synth-based PRF is $\log k$ levels of NC^1 synthesizers $\Rightarrow$ NC^2.
- [NR'97,NRR'00]: direct PRFs from DDH / factoring, in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$.

$$F_{g,s_1,\ldots,s_k}(x_1 \cdots x_k) = g^{\prod s_i^{x_i}}$$

(Computing this in TC^0 needs huge circuits, though. . .)

Direct LWE-Based Construction

- Public moduli $q > p$.
- Secret key is uniform $\mathbf{A}$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$ over $\mathbb{Z}_q$.
- "Rounded subset-product" function:

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A} \cdot \prod_{i=1}^{k} \mathbf{S}_i^{x_i} \bmod q \right\rceil_p$$

Shallower? More Efficient?

- Synth-based PRF is $\log k$ levels of NC^1 synthesizers $\Rightarrow \mathsf{NC}^2$.
- [NR'97,NRR'00]: direct PRFs from DDH / factoring, in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$.

$$F_{g,s_1,\ldots,s_k}(x_1 \cdots x_k) = g^{\prod s_i^{x_i}}$$

(Computing this in TC^0 needs huge circuits, though...)

Direct LWE-Based Construction

- Public moduli $q > p$.
- Secret key is uniform $\mathbf{A}$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$ over $\mathbb{Z}_q$.
- "Rounded subset-product" function:

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A} \cdot \prod_{i=1}^{k} \mathbf{S}_i^{x_i} \bmod q \right\rceil_p$$

Ring variant has small(ish) TC^0 circuit, practical implementation

Proof Sketch

- Seed is uniform $\mathbf{A}$ over $\mathbb{Z}_q$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$.

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A}\mathbf{S}_1^{x_1} \cdots \mathbf{S}_k^{x_k} \bmod q \right\rceil_p$$

Proof Sketch

- Seed is uniform $\mathbf{A}$ over $\mathbb{Z}_q$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$.

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A}\mathbf{S}_1^{x_1} \cdots \mathbf{S}_k^{x_k} \bmod q \right\rceil_p$$

- Like the LWE $\leq$ LWR proof, but "souped up" to handle queries.

Proof Sketch

- Seed is uniform $\mathbf{A}$ over $\mathbb{Z}_q$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$.

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A}\mathbf{S}_1^{x_1} \cdots \mathbf{S}_k^{x_k} \bmod q \right\rceil_p$$

- Like the LWE $\leq$ LWR proof, but "souped up" to handle queries.

  Thought experiment: answer queries with

$$\tilde{F}(x) := \left\lfloor (\mathbf{A}\mathbf{S}_1^{x_1} + x_1\mathbf{E})\mathbf{S}_2^{x_2} \cdots \mathbf{S}_k^{x_k} \right\rceil_p = \left\lfloor \mathbf{A}\prod_{i=1}^{k} \mathbf{S}_i^{x_i} + x_1\mathbf{E}\prod_{i=2}^{k} \mathbf{S}_i^{x_i} \right\rceil_p$$

  W.h.p., $\tilde{F}(x) = F(x)$ on all queries due to "small" error & rounding.

Proof Sketch

- Seed is uniform $\mathbf{A}$ over $\mathbb{Z}_q$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$.

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A}\mathbf{S}_1^{x_1} \cdots \mathbf{S}_k^{x_k} \bmod q \right\rceil_p$$

- Like the LWE $\leq$ LWR proof, but "souped up" to handle queries.

  Thought experiment: answer queries with

$$\tilde{F}(x) := \lfloor (\mathbf{A}\mathbf{S}_1^{x_1} + x_1\mathbf{E})\mathbf{S}_2^{x_2} \cdots \mathbf{S}_k^{x_k} \rceil_p = \left\lfloor \mathbf{A}\prod_{i=1}^{k}\mathbf{S}_i^{x_i} + x_1\mathbf{E}\prod_{i=2}^{k}\mathbf{S}_i^{x_i} \right\rceil_p$$

  W.h.p., $\tilde{F}(x) = F(x)$ on all queries due to "small" error & rounding.

- Using LWE, replace $(\mathbf{A}, \mathbf{A}\mathbf{S}_1 + \mathbf{E})$ with uniform $(\mathbf{A}_0, \mathbf{A}_1)$

  $\Rightarrow$ New function $F'(x) = \lfloor \mathbf{A}_{x_1}\mathbf{S}_2^{x_2} \cdots \mathbf{S}_k^{x_k} \rceil_p$.

Proof Sketch

- Seed is uniform $\mathbf{A}$ over $\mathbb{Z}_q$ and short $\mathbf{S}_1, \ldots, \mathbf{S}_k$.

$$F_{\mathbf{A},\mathbf{S}_1,\ldots,\mathbf{S}_k}(x_1 \cdots x_k) = \left\lfloor \mathbf{A}\mathbf{S}_1^{x_1} \cdots \mathbf{S}_k^{x_k} \bmod q \right\rceil_p$$

- Like the LWE $\leq$ LWR proof, but "souped up" to handle queries.

  Thought experiment: answer queries with

$$\tilde{F}(x) := \lfloor (\mathbf{A}\mathbf{S}_1^{x_1} + x_1\mathbf{E})\mathbf{S}_2^{x_2} \cdots \mathbf{S}_k^{x_k} \rceil_p = \left\lfloor \mathbf{A}\prod_{i=1}^{k} \mathbf{S}_i^{x_i} + x_1\mathbf{E}\prod_{i=2}^{k} \mathbf{S}_i^{x_i} \right\rceil_p$$

  W.h.p., $\tilde{F}(x) = F(x)$ on all queries due to "small" error & rounding.

- Using LWE, replace $(\mathbf{A}, \mathbf{A}\mathbf{S}_1 + \mathbf{E})$ with uniform $(\mathbf{A}_0, \mathbf{A}_1)$

  $\Rightarrow$ New function $F'(x) = \lfloor \mathbf{A}_{x_1}\mathbf{S}_2^{x_2} \cdots \mathbf{S}_k^{x_k} \rceil_p$.

- Repeat for $\mathbf{S}_2, \mathbf{S}_3, \ldots$ to get $F'''''''(x) = \lfloor \mathbf{A}_x \rceil_p = U(x)$. $\square$

Open Questions

- Better hardness for LWR, e.g. for $q/p = \sqrt{n}$?

  (The proof from LWE relies on approx factor and modulus $= n^{\omega(1)}$.)

Open Questions

- Better hardness for LWR, e.g. for $q/p = \sqrt{n}$?

  (The proof from LWE relies on approx factor and modulus $= n^{\omega(1)}$.)

- Synth-based PRF relies on approx factor and modulus $n^{\Theta(\log k)}$.

  Direct construction relies on approx factor and modulus $n^{\Theta(k)}$.

Open Questions

- Better hardness for LWR, e.g. for $q/p = \sqrt{n}$?

  (The proof from LWE relies on approx factor and modulus $= n^{\omega(1)}$.)

- Synth-based PRF relies on approx factor and modulus $n^{\Theta(\log k)}$.

  Direct construction relies on approx factor and modulus $n^{\Theta(k)}$.

  Conjecture (?): direct PRF is secure for integral $q/p = \text{poly}(n)$.

Open Questions

- Better hardness for LWR, e.g. for $q/p = \sqrt{n}$?
  (The proof from LWE relies on approx factor and modulus $= n^{\omega(1)}$.)
- Synth-based PRF relies on approx factor and modulus $n^{\Theta(\log k)}$.
  Direct construction relies on approx factor and modulus $n^{\Theta(k)}$.
  Conjecture (?): direct PRF is secure for integral $q/p = \mathrm{poly}(n)$.
- Efficient PRFs from subset-sum/LPN?

Open Questions

- Better hardness for LWR, e.g. for $q/p = \sqrt{n}$?

  (The proof from LWE relies on approx factor and modulus $= n^{\omega(1)}$.)

- Synth-based PRF relies on approx factor and modulus $n^{\Theta(\log k)}$.

  Direct construction relies on approx factor and modulus $n^{\Theta(k)}$.

  Conjecture (?): direct PRF is secure for integral $q/p = \text{poly}(n)$.

- Efficient PRFs from subset-sum/LPN?

Selected bibliography for this talk:

NR'95 M. Naor, O. Reingold, "Synthesizers and Their Applications to the Parallel Construction of Pseudorandom Functions," FOCS'95 / JCSS'99.

NR'97 M. Naor, O. Reingold, "Number-theoretic constructions of efficient pseudorandom functions," FOCS'97 / JACM'04.

NRR'00 M. Naor, O. Reingold, A. Rosen, "Pseudorandom functions and factoring," STOC'00 / SICOMP'02.

BPR'12 A. Banerjee, C. Peikert, A. Rosen, "Pseudorandom Functions and Lattices," Eurocrypt'12.