Lattices
Multilinear Maps
Obfuscation

Yilei Chen
Visa Research

Lattices: Algorithms, Complexity, and Cryptography @ Simons Institute
What are Multilinear Maps?
> Discrete-log problem [ Diffie, Hellman 76 ]

Given $g, g^s \mod q$, finding $s$ is hard

> Bilinear maps from Weil pairing over elliptic curve groups
  [ Miller 86 ] How to compute Weil pairing
  [ Sakai, Ohgishi, Kasahara 00 ] Identity-based key-exchange
  [ Joux 00 ] Three-party non-interactive key-exchange
  [ Boneh, Franklin 02 ] Identity-base encryption

$$g^{S_1}, g^{S_2} \rightarrow g^{S_1S_2}_T$$

> Multilinear maps: motivated in [ Boneh, Silverberg 03 ] with the potential applications of constructing unique signature, broadcast encryption, etc.

$$g^{S_1}, g^{S_2}, g^{S_3}, \ldots \rightarrow g^{T\prod S}_T$$
> Discrete-log problem [Diffie, Hellman 76]

Given $g, g^s \mod q$, finding $s$ is hard

> Bilinear maps from Weil pairing over elliptic curve groups
  [Miller 86] How to compute Weil pairing
  [Sakai, Ohgishi, Kasahara 00] Identity-based key-exchange
  [Joux 00] Three-party non-interactive key-exchange
  [Boneh, Franklin 02] Identity-based encryption

$g^{S_1}, g^{S_2} \rightarrow g^{S_1S_2}_T$

> Multilinear maps: motivated in [Boneh, Silverberg 03] with the potential applications of constructing unique signature, broadcast encryption, etc.

$g^{S_1}, g^{S_2}, g^{S_3}, \ldots \rightarrow g^{\prod S}_T$
Where to find multilinear maps?

“If an n-multilinear map is computable, it is reasonable to expect it to come from geometry, as is the case for Weil and Tate pairings when n = 2.”

…

“If varieties giving rise to n-multilinear maps cannot be found for n > 2, one could at least hope that such maps might arise from motives.”

– Boneh, Silverberg, 2003

*New: Trilinear maps from abelian varieties [Huang 2019], requires further investigation.
What are multilinear maps?
Why from lattices?
Garg, Gentry, Halevi [GGH 13] propose a candidate based on a variant of the NTRU problem. No security reduction is given; cryptanalysis attempts are mentioned.

Think of as **homomorphic encryption + public zero-test**

i.e. everyone can test whether you get $g_T^0$ or $g_T$ non-zero

Coron, Lepoint, Tibouchi [CLT 13] propose a candidate based on a variant of approx-gcd.

Gentry, Gorbunov, Halevi [GGH 15] propose another candidate inspired by the FHE scheme of [Gentry, Sahai, Waters 13].
Multilinear maps

Applications overview

Private constrained PRFs

Witness encryption

Multilinear maps

GGH13, CLT13, GGH15

Indistinguishability obfuscation

Lockable obfuscation

(Compute-then-Compare obf.)

Multilinear maps

Witness encryption

Multiparty key agreement

Indistinguishability obfuscation

Privacy constrained PRFs

Witness encryption

Indistinguishability obfuscation

Deniable encryption

Functional encryption

Deniable encryption
Multilinear maps $\rightarrow$ Indistinguishability obfuscation

[ Garg, Gentry, Halevi, Raykova, Sahai, Waters 13 ]
Indistinguishability obfuscation

Defined by [ Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang 01 ]

Program Obfuscation: $P \Rightarrow \text{Obf}(P)$

Correctness: $\text{Obf}(P)$ preserves the functionality of $P$

Security: For two programs $P_0$ and $P_1$ with identical functionality

$$iO[ P_0 ] \approx iO[ P_1 ]$$
The big bang in crypto

- Private constrained PRFs
- Witness encryption
  - Multilinear maps
    - GGH13, CLT13, GGH15
  - Indistinguishability obfuscation
    - Multiparty key agreement
      - Fiat-Shamir
        - Hardness of Nash
    - Functional encryption
      - Deniable encryption

- Self-bilinear maps
The whiteboard on the 3rd floor of Simons Institute, in a sunny day in Summer 2015.

The big bang in crypto

Indistinguishability obfuscation

Functional encryption [ Waters 14 ]
Self-bilinear maps: \( g^{S_1}, g^{S_2} \rightarrow g^{S_1 S_2} \)

[Yamakawa, Yamada, Hanaoka, Kunihiro 14]: When the obfuscation is iO and \( N \) is an RSA modulus, the following idea works:

\[
\text{Encoding}(S) = \{ g^S \mod N, \text{Obf}[ f_S(x) = x^S \mod N ] \}
\]
Where are we right now?

Lattices
=> Multilinear maps
=> Obfuscation
=> …

The big bang in crypto
Multilinear maps & their friends

security overview

Private constrained PRFs

Witness encryption

Indistinguishability obfuscation

Multiparty key agreement

Multilinear maps

GGH13, CLT13, GGH15

Lockable obfuscation

(Compute-then-Compare obf.)

Functional encryption

Deniable encryption

With a reduction from LWE (via safe use of GGH15);

Candidates exist
Current status of multilinear maps and iO

https://malb.io/are-graded-encoding-schemes-broken-yet.html
https://sites.google.com/view/iostate-of-the-art/

Candidate constructions:
[Garg-Gentry-Halevi-Raykova-Sahai-Waters ‘13], [Barak-Garg-Kalai-Paneth-Sahai ‘14],
[Brakerski-Rothblum ‘14], [Pass-Seth-Telang ‘14], [Zimmerman ‘15], [Applebaum-Brakerski ‘15],
[Ananth-Jain ‘15], [Bitansky-Vaikuntanathan ‘15], [Gentry-Gorbunov-Halevi ‘15], [Lin ‘16], …

Cryptanalyses:
[Cheon-Han-Lee-Ryu-Stehle ‘15], [Coron et al ‘15], [Miles-Sahai-Zhandry ‘16], …
Open Problems, Cryptography, Summer 2015

Below is a list of open problems proposed during the Cryptography program at the Simons Institute for the Theory of Computing, compiled by Ron Rothblum and Alessandra Scafuro. Each problem comes with a symbolic cash prize.

1. One-way permutations from a worst-case lattice assumption ($100 from Vinod Vaikuntanathan).
2. Non-interactive zero-knowledge (NIZK) proofs (or even arguments) for NP from LWE ($100 from Vinod Vaikuntanathan).
3. IO from LWE ($100 from Amit Sahai). This result would also solve problems (1) and (2). For (1) see construction and limitations and for (2) see argument system and proof system.
4. Interactive proofs for languages computable in DTISP(t,s) (time t and space s), where the prover runs in time poly(t) and the verifier runs in time poly(s). The provers in known proofs of IP = PSPACE run in time exponential in \(2^{\text{poly(s)}}\) or \(2^{O(s)}\) ($100 from Yael Kalai).
5. $20 per broken password challenge (from Jeremiah Blocki).
6. (Dis)prove that scrypt requires amortized (space × time) = \(\Omega(n^2/\text{polylog(n)})\) per evaluation on a parallel machine ($100 from Joël).
7. A 3-linear map with unique encoding (i.e., without noise) for which “discrete log” is “plausibly hard” ($100 from Dan Boneh).
8. SZK = PZK, or in other words, transform any statistical zero-knowledge proof (SZK) into a perfect zero-knowledge proof (PZK) ($100 from Shafi Goldwasser).

Update: During the talk, Amit raised the award to $1000.
Multilinear maps
- GGH13, CLT13, GGH15
  - Lockable obfuscation (Compute-then-Compare obf.)
  - With a reduction from LWE (via safe use of GGH15); Candidates exist

Indistinguishability obfuscation
- Gentry, Gorbunov, Halevi (TCC 2015)
  "Graph-induced multilinear maps from lattices"

Private constrained PRFs

Today: Lattice behind the big bang in crypto
Private constrained PRFs

Multilinear maps
GGH13, CLT13, **GGH15**

Lockable obfuscation
(Compute-then-Compare obf.)

Indistinguishability obfuscation

Today: Lattice behind the big bang in crypto

- Multilinear maps with security based on LWE
- A new methodology of building lattice applications after “[GSW13]” and “[BGG+14]”

With a reduction from LWE (via safe use of GGH15); Candidates exists
Plan of today:

1. Introduction
2. GGH15: functionality and security overview
3. Applications: Obfuscators & Private constrained PRFs

Open problems will be mentioned during the talk
The arithmetic operations are just matrix operations in $\mathbb{Z}_q^{m \times m}$:

$$\text{neg}(p, D) := -D, \quad \text{add}(p, D, D') := D + D', \quad \text{and} \quad \text{mult}(p, D, D') := D \cdot D'.$$

To see that negation and addition maintain the right structure, let $D, D' \in \mathbb{Z}_q^{m \times m}$ be encodings relative to the same path $u \leadsto v$. Namely $D \cdot A_u = A_v \cdot S + E$ and $D' \cdot A_u = A_v \cdot S'$ with the matrices $D, D', E, E', S, S'$ all small. Then we have

$$-D \cdot A_u = A_v \cdot (-S) + (-E),$$
and
$$\text{and} \quad (D + D') \cdot A_u = (A_v \cdot S + E) + (A_v \cdot S' + E') = A_v \cdot (S + S') + (E + E'),$$

and all the matrices $-D, -S, -E, D + D', S + S', E + E'$ are still small. For multiplication consider encodings $D, D'$ relative to paths $v \leadsto w$ and $u \leadsto v$, respectively, then we have

$$(D \cdot D') \cdot A_u = D \cdot (A_v \cdot S' + E')$$

$$= (A_{uw} \cdot S + E) \cdot S' + D \cdot E' = A_{uw} \cdot (S \cdot S') + (E \cdot S' + D \cdot E'),$$

and the matrices $D \cdot D', S \cdot S', \text{ and } E''$ are still small.
The development of GGH15-like applications: 2015 - 2017

[ Gentry, Gorbunov, Halevi 15 ]: functionality, cryptanalytic attempts, candidate N-party key-exchange and iO.

[ Brakerski, Vaikuntanathan, Wee, Wichs 16 ]: First proof methodology => obfuscating conjunctions

[ Coron, Lee, Lepoint, Tibouchi 16 ]: breaking the candidate N-party key exchange

[ Chen, Gentry, Halevi 17 ]: breaking iO for some parameters

[ Canetti, Chen 17 ]: Private Constrained PRF from LWE
[ Goyal, Koppula, Waters 17a ]: Circular security counterexample from LWE
[ Goyal, Koppula, Waters 17b ], [ Wichs, Zirdelis 17 ]: Lockable obfuscation, compute & compare obfuscation from LWE
[GGH15] Via a different view of the FHE scheme of Gentry, Sahai, Waters 13

Different motives/views of GGH15

[Alamati, Peikert 16], [Koppula, Waters 16], [Goyal, Koppula, Waters 17]
“cascaded products” or “telescoping cancelation”, motivated by showing circular security counterexamples.

Today: chaining LWE samples

GGH15 captures two lattice-based PRFs

[Canetti, Chen 17]

A generalization of Kilian randomization

[Chen, Vaikuntanathan, Wee 18]
Recall Learning with Errors
[ Regev 05 ]

\[ A \in \mathbb{Z}_q^{n \times m} \quad (m > n \log q) \]

Search LWE: Given \( A, Y = SA + E \), find \( S \).

Decisional LWE: Given \( A \), distinguish \( Y \) from random.
Recall Learning with Errors
[Regev 05]

\[ A \in Z_q^{n \times m} \quad (m > n \log q) \]

Search LWE: Given \( A, Y = SA + E \), find \( S \).
Decisional LWE: Given \( A \), distinguish \( Y \) from random.
Recall Learning with Errors [Regev 05]

\[ Y = S \times A + E \mod q \]

Entries of S from the error distribution
As hard as normal LWE [Applebaum, Cash, Peikert, Sahai 09]
> Multilinear maps: motivated in [ Boneh, Silverberg 2003 ]

\[
g, g^{S_1}, g^{S_2}, g^{S_3}, \ldots \rightarrow g^{\prod S}
\]

> (Ring)LWE analogy:

\[
A, S_1A+E_1, \ldots, S_kA+E_k \rightarrow \prod SA+E \mod q
\]

How to compute the map?
Multilinear maps: motivated in [Boneh, Silverberg 2003]

\( g, g^{S_1}, g^{S_2}, g^{S_3}, ... \rightarrow g_T^{\prod S} \)

(Ring)LWE analogy:

\( A, S_1 A + E_1, ..., S_k A + E_k \rightarrow \prod S A + E \mod q \)

Idea: using lattice trapdoor sampling to chain them together
The trapdoor for $A$ can be used to solve SIS and LWE.

Given an image $Y$, find a short vector $D$ s.t.

$$A \times D = Y \mod q$$
A lattice trapdoor is short and full rank in $\mathbb{Z}$.

\[
A \times T \equiv 0 \pmod{q}
\]
In a nutshell

\[ A, S_1 A + E_1, \ldots, S_k A + E_k \rightarrow \prod S A + E \mod q \]

> (Ring)LWE analogy:

> GGH15:

\[ A_0, S_1 A_1 + E_1, A_1, S_2 A_2 + E_2 \]
A, $S_1 A + E_1, ..., S_k A + E_k \rightarrow \prod S A + E \mod q$

> (Ring)LWE analogy:

$A_0 D_1 = S_1 A_1 + E_1$, $A_1 D_2 = S_2 A_2 + E_2 \mod q$

$D_i$ is sampled using the trapdoor of $A_{i-1}$

> GGH15:

in a nutshell
> (Ring)LWE analogy:

\[ A, S_1A+E_1, \ldots, S_kA+E_k \rightarrow \prod S_iA+E \mod q \]

> GGH15:

\[ A_0D_1 = S_1A_1+E_1, \quad A_1D_2 = S_2A_2+E_2 \mod q \]

\(D_i\) is sampled using the trapdoor of \(A_{i-1}\)
\[ A_0 D_1 = S_1 A_1 + E_1, \quad A_1 D_2 = S_2 A_2 + E_2 \mod q \]

\[ A_{i-1} D_i = s_i \times A_i + E_i \mod q \]

\( D_i \) is sampled using the trapdoor of \( A_{i-1} \)
in a nutshell

\[ A, S_1 A + E_1, \ldots, S_k A + E_k \rightarrow \prod S A + E \mod q \]

Publish \( A_0, D_1, D_2 \) as the encodings of \( S_1, S_2 \)
in a nutshell

> (Ring)LWE analogy:

\[ A, S_1 A + E_1, \ldots, S_k A + E_k \rightarrow \prod S A + E \mod q \]

> GGH15:

\[ A_0 D_1 = S_1 A_1 + E_1, \quad A_1 D_2 = S_2 A_2 + E_2 \mod q \]

Publish \( A_0, D_1, D_2 \) as the encodings of \( S_1, S_2 \)

\[ A_0 D_1 D_2 = (S_1 A_1 + E_1)D_2 = S_1 A_1 D_2 + E_1 D_2 \]
\[ = S_1 (S_2 A_2 + E_2) + E_1 D_2 = S_1 S_2 A_2 + S_1 E_2 + E_1 D_2 \]

functionality  
small error
A typical evaluation pattern for GGH15: subset product

$$A_i = A_{i-1} \times D_{i, b} = S_{i, b} \times A_i + E_{i, b} \mod q$$
Subset product evaluation

\[ \text{Eval}(0110) = A_0 D_{1,0} D_{2,1} D_{3,1} D_{4,0} \]

\[ \leq \text{The input is a bit string that selects which } D_{i,b} \text{ to multiply} \]
Eval(0110)

= \( A_0D_{1,0}D_{2,1}D_{3,1}D_{4,0} \)

= \( (s_{1,0}A_1 + E_{1,0})D_{2,1}D_{3,1}D_{4,0} \)
Eval(0110)
= \text{A}_0D_{1,0}D_{2,1}D_{3,1}D_{4,0}
= (s_{1,0}\text{A}_1+E_{1,0})D_{2,1}D_{3,1}D_{4,0}
= s_{1,0}\text{A}_1D_{2,1}D_{3,1}D_{4,0} + \text{“small”}
Eval(0110)

= \( A_0 D_{1,0} D_{2,1} D_{3,1} D_{4,0} \)

= \( (s_{1,0} A_1 + E_{1,0}) D_{2,1} D_{3,1} D_{4,0} \)

= \( s_{1,0} A_1 D_{2,1} D_{3,1} D_{4,0} + \text{"small"} \)

= \( s_{1,0} (s_{2,1} A_2 + E_{2,1}) D_{3,1} D_{4,0} + \text{"small"} \)
Eval(0110)
= A_0 D_{1,0} D_{2,1} D_{3,1} D_{4,0}
= (s_{1,0} A_1 + E_{1,0}) D_{2,1} D_{3,1} D_{4,0}
= s_{1,0} A_1 D_{2,1} D_{3,1} D_{4,0} + “small”
= s_{1,0} (s_{2,1} A_2 + E_{2,1}) D_{3,1} D_{4,0} + “small”
= s_{1,0} s_{2,1} A_2 D_{3,1} D_{4,0} + “still small”
Eval(0110)

\[ = A_0D_{1,0}D_{2,1}D_{3,1}D_{4,0} \]

\[ = (s_{1,0}A_1 + E_{1,0})D_{2,1}D_{3,1}D_{4,0} \]

\[ = s_{1,0}A_1 D_{2,1}D_{3,1}D_{4,0} + \text{“small”} \]

\[ = s_{1,0}(s_{2,1}A_2 + E_{2,1})D_{3,1}D_{4,0} + \text{“small”} \]

\[ = s_{1,0}s_{2,1}A_2 D_{3,1}D_{4,0} + \text{“still small”} \]

\[ = s_{1,0}s_{2,1}s_{3,1}A_3 D_{4,0} + \text{“still smallish”} \]

\[ = s_{1,0}s_{2,1}s_{3,1}s_{4,0}A_4 + \text{“small”} \]

The “small” noise grows exponentially with #levels, becomes a problem in the efficiency.
Subset product evaluation

Evaluate

$A_4$ + “small”

$A_0$

$D_{1,1}$ $D_{2,1}$ $D_{3,1}$ $D_{4,1}$

$D_{1,0}$ $D_{2,0}$ $D_{3,0}$ $D_{4,0}$
Functionality: evaluate and test whether $\prod S$ is zero or not.

$(Designing \ GGH15\ applications: \ put\ structures\ in\ S_{i,b})$

$A_0, S_1A_1+E_1, \ldots, S_kA_k+E_k \rightarrow \prod S A_k+E \mod q$
Functionality and Security

Functionality: evaluate and test whether $\prod S$ is zero or not.
(Designing GGH15 applications: put structures in $S_{i,b}$)

Security (goal): hides $S_{i,b}$ for all $i, b$. But the reality is ...
complicated, depends on the structure inside $S_{i,b}$

Security (goal): hides $S_{i,b}$ for all $i, b$. But the reality is ...
What does “structure” in $S_{i,b}$ look like?
Toy example 1

Each $S_{i, b} = M_{i, b} \otimes s_{i, b}$

$\prod S A_2 + E$

$F(00) = 0$
$F(01) = 1$
$F(10) = 1$
$F(11) = 1$
Claim: this construction hides all the structures in the S matrices.

Toy example 2

\[ \prod S A_2 + E \]

\[
F(00) = 1 \\
F(01) = 1 \\
F(10) = 1 \\
F(11) = 1
\]
Recall decisional LWE

\[ A, S \times A \approx \text{computational} \]

Permutation - LWE:

\[ A(1), A(2), A(3) \approx \text{computational} \]

\[ U \]
Claim: this construction hides all the structures in the $S$ matrices.
Permutation LWE

Functionality & Security
toy examples

$A_0 \xrightarrow{D_{1,1}} A_1 + E$

$A_0 \xrightarrow{D_{1,0}} A_1 + E$

$A_1 \xrightarrow{D_{2,1}} U_{2,1}$

$A_1 \xrightarrow{D_{2,0}} U_{2,0}$
For random images, there is a way to sample the preimage \textbf{without} revealing the trapdoor.
For random images, there is a way to sample the preimage without revealing the trapdoor.

Real: \[ A \quad U \quad D \quad \text{s.t.} \quad A \quad x \quad D = U \quad \text{mod} \quad q \]

\[ \approx \text{statistical} \]

Simulated: \[ A \quad D \quad U \quad \text{s.t.} \quad A \quad x \quad D = U \quad \text{mod} \quad q \]
Functionality & Security

toy examples

Turn off the trapdoor using GPV
Functionality & Security
toy examples

Permutation LWE
Turn off the trapdoor using GPV
Looks simple to achieve security based on LWE!
How do the insecure examples look like?
For example, let $S_2 = 0$ in

$$A_0 \ D_1 = S_1A_1+E_1, \quad A_1 \ D_2 = S_2A_2+E_2 \mod q$$
For example, let $S_2 = 0$ in

$A_0 \cdot D_1 = S_1 \cdot A_1 + E_1,$ \hspace{1cm} $A_1 \cdot D_2 = S_2 \cdot A_2 + E_2 \mod q$

$D_2$ becomes a “weak trapdoor” of $A_1$, then $S_1$ is in danger
For example, let $S_2 = 0$ in

$$A_0 D_1 = S_1 A_1 + E_1, \quad A_1 D_2 = S_2 A_2 + E_2 \quad \text{mod } q$$

$D_2$ becomes a “weak trapdoor” of $A_1$, then $S_1$ is in danger

$$\text{Eval} = A_0 D_1 D_2 = (S_1 A_1 + E_1) D_2 = S_1 E_2 + E_1 D_2 \quad \text{mod } q$$

Recover $S_1 E_2 + E_1 D_2$ over integers, can do many things.
Compared to other lattice application frameworks

“Regev-like schemes” [Regev 05]

Public key: $A$, $SA+E$; secret key: $S$; message: $(SA+E)*R + m*(q/2)$

“Dual-Regev-like schemes” [Gentry, Peikert, Vaikuntanathan 08]

Public key: $A_0$, $A_1$, ..., $A_d$, (master) secret key: the trapdoor of $A_0$

“GGH15-like” $A_0$, $S_1A_1+E_1$, ..., $S_kA_k+E_k \rightarrow \prod S A_k+E$

Both the message/function to be hidden are in the LWE secret terms
Plan of today:

1. Introduction
2. GGH15: functionality and security overview
3. Applications: Obfuscators & Private constrained PRFs

Open problems will be mentioned during the talk
Multilinear maps

GGH13, CLT13, **GGH15**

1. Private Constrained PRFs
   
   [Canetti, Chen 17]

2. General-purpose obfuscation

   [Gentry, Gorbunov, Halevi 15], …

With a reduction from LWE (via safe use of GGH15); Candidates exists
Private Constrained PRFs
Private Constrained Pseudorandom Function in 3 slides
Private Constrained Pseudorandom Function in 3 slides

[ Goldreich, Goldwasser, Micali 86 ]

PRF

A truly random function

With oracle access to either left or right

adv
Private **Constrained** Pseudorandom Function in 3 slides

[ Boneh, Waters 13 ], [ Kiayias, Papadopoulos, Triandopoulos, Zacharias 13 ], [ Boyle, Goldwasser, Ivan 14 ]

original key

modified key
Private Constrained Pseudorandom Function in 3 slides

[ Boneh, Lewi, Wu 17 ]

original key
privately modified key
either the original key or the modified one

Private key owner
What are private constrained PRFs?
What is the motive?
Two-key secure PCPRF (for a circuit class C) implies obfuscation (for C)

Obf = \{ K[ C ], K[ original ] \}

Eval( Obf, x ): Compare K[ C ](x) and K[ original ](x)
[Canetti Chen 17]: Two-key secure PCPRF (for a circuit class C) implies obfuscation (for C)

\[
\text{Obf} = \{ \text{K}[\text{C}], \text{K}[\text{original}] \} \\
\text{Eval}(\text{Obf}, x): \text{Compare K}[\text{C}](x) \text{ and K}[\text{original}](x)
\]

But if two constrained keys are published, then we don’t know how to prove constraint-hiding based on LWE.
[Canetti, Chen 17] 1-key PCPRF implies 1-key private-key functional encryption (a.k.a. reusable garbled circuits).

Construction:

\[
\text{Enc}(m;r): \quad \text{ct} = \text{Enc}_{\text{Sym.K}}(m;r); \quad \text{tag} = \text{PRF.K}[ \text{original} ](\text{ct})
\]

\text{Functional_SK[Sym.K, PRF.K, C]}:

A private constrained key for the “decryption and eval” functionality

\[
\text{PRF.K}[ C( \text{Dec}_{\text{Sym.K}}( . ) ) ]
\]

Eval: compute \( \text{PRF.K}[ C( \text{Dec}_{\text{Sym.K}}( . ) ) ](\text{ct}) \), and compare with \text{tag}
Applications of Private Constrained PRFs: Obfuscation (if it is 2-key secure)*
Reusable garbled circuits
Privately-detectable watermarking
With key homomorphism => traitor tracing
Maybe more …

original key
privately modified key
What are private constrained PRFs?
What is the motive?
How to construct from *lattices*?
Private Constrained PRFs from Lattices?

Step 1: Start from a lattice PRF.  
[Banerjee, Peikert, Rosen 12]

Step 2: Embed a constraint.  
[Barrington 86]

Step 3: Do Step 2 privately.  
[GGH15]
Key: $s_{1,1} \quad s_{2,1} \quad \ldots \quad s_{n,1}$

$A \mod q$

Eval: $F(x) = \{ \prod s_{i,xi} A \}_2$

$s_{i,b}$ are LWE secrets from low-norm distributions
Rounding: \{t\}_p: Z_q \rightarrow Z_p

Compute \( t \cdot p/q \), then round to the nearest integer

In this talk, \( p=2 \), \( q/p > \exp(L) \), \( q/p \sim \text{super-polynomial} \)

Amount of noise
Main observation: After rounding, can inject noises without changing the functionality with high probability.

$$F(x) = \{ \prod_{i} s_{i,x_i} A \}_2$$
\[ F(x) = \{ \prod_{s_i, x_i} A \} \pmod{q} \]

\[
F(0110) = \{ s_{1,0}s_{2,1}s_{3,1}s_{4,0} A \} \pmod{2}
\]
\[ F(0110) = \{ s_{1,0}s_{2,1}s_{3,1}s_{4,0} A \}_2 \]
\[ \approx_s \{ s_{1,0}s_{2,1}s_{3,1}(s_{4,0} A+E_{4,0}) \}_2 \]

\[ F(x) = \{ \prod s_{i,x_i} A \}_2 \]
\[ F(x) = \{ \prod_{i} s_{i,x_i} A \} \mod q \]

\[
\begin{align*}
F(0110) &= \{ s_{1,0}s_{2,1}s_{3,1}s_{4,0} A \} \mod q \\
&\approx_s \{ s_{1,0}s_{2,1}s_{3,1}(s_{4,0} A+E_{4,0}) \} \mod q \\
&\approx_c \{ s_{1,0}s_{2,1}s_{3,1}Y_{0} \} \mod q
\end{align*}
\]

[ Banerjee, Peikert, Rosen 12 ]
F(x) = \{ \prod_{i=0}^{4} s_{i,x_i} A \}_{2}

\begin{align*}
F(0110) &= \{ s_{1,0}s_{2,1}s_{3,1}s_{4,0} A \}_{2} \\
&\approx_s \{ s_{1,0}s_{2,1}s_{3,1}(s_{4,0}A+E_{4,0}) \}_{2} \\
&\approx_c \{ s_{1,0}s_{2,1}s_{3,1}Y^{**0} \}_{2} \\
&\approx_s \{ s_{1,0}s_{2,1}(s_{3,1}Y^{**0}+E_{3,1}) \}_{2}
\end{align*}

\text{mod q}
\[ F(x) = \{ \prod_{i} s_{i, x_i} A \}_2 \]

\[
F(0110) = \{ s_{1,0}s_{2,1}s_{3,1}s_{4,0} A \}_2 \\
\approx_s \{ s_{1,0}s_{2,1}s_{3,1}(s_{4,0} A+E_{4,0}) \}_2 \\
\approx_c \{ s_{1,0}s_{2,1}s_{3,1}Y_{**0} \}_2 \\
\approx_s \{ s_{1,0}s_{2,1}(s_{3,1}Y_{**0}+E_{3,1}) \}_2 \\
\approx_c \{ s_{1,0}s_{2,1}Y_{**10} \}_2 \\
\approx ... \approx \{ Y_{0110} \}_2
\]
Exercise: show that taking matrix subset-product without rounding does not give a PRF.
Open problem: prove or disprove that when $q$ is a polynomial, the construction is a PRF.
The distribution of the $S$ matrices can be uniformly from $\mathbb{Z}_q$
Private Constrained PRFs from Lattices?

Step 1: Start from a lattice PRF.
[Banerjee, Peikert, Rosen 12]

Step 2: Embed a constraint.
[Barrington 86]

Step 3: Do Step 2 privately.
[GGH15]
Example: how to represent an AND gate
Barrington 1986: log-depth circuit $\rightarrow$ matrix branching program

Example: how to represent an AND gate $0$ and $0$
Example: how to represent an AND gate

0 and 1

Barrington 1986: log-depth circuit => matrix branching program
Example: how to represent an AND gate

1 and 0

Barrington 1986: log-depth circuit => matrix branching program
Example: how to represent an AND gate

PQP⁻¹Q⁻¹ = C ≠ 1

Barrington 1986: log-depth circuit => matrix branching program
Private Constrained PRFs from Lattices?

Step 1: Start from a lattice PRF.
[Banerjee, Peikert, Rosen 12]

Step 2: Embed a constraint.
[Barrington 86]

Step 3: Do Step 2 privately.
[GGH15]
Embed the permutation matrices in the LWE secret $B_{i,b} \otimes s_{i,b}$

E.g. $I \otimes s = \begin{bmatrix}
S & S & S & S \\
S & S & S & S \\
S & S & S & S \\
S & S & S & S \\
\end{bmatrix}$

$P \otimes s = \begin{bmatrix}
S & S & S & S \\
S & S & S & S \\
S & S & S & S \\
S & S & S & S \\
\end{bmatrix}$
Embed the permutation matrices in the LWE secret $B_{i,b} \otimes s_{i,b}$

Constrained key: the GGH15 encoding
How to prove the branching program is hidden by GGH15 encoding?

The real constrained key
Claim: this construction hides all the structures in the $S$ matrices.

Recall Toy example 2
Recall Toy example 2

Perm-LWE + Turning off the trapdoor using GPV
The real constrained key

The simulated constrained key
Takeaway from the Private Constrained PRF:
It is safe to use GGH15 to encode permutation matrices, and make it useful.
Genealogy of Lattices-based PRFs

[BPR12] -- the first lattice-based PRF
[BLMR13] -- key homomorphic
* [BP14] -- better key homomorphic, embed a tree
* [BFPPS15] -- [BP14] is puncturable
* [BV15] -- embed a circuit, constrained for P
* [BKM17] -- puncture privately, built from [BV15]
[CC17] -- constrained privately for NC1, influenced by GGH15 mmaps
* [BTVW17] -- constrained privately for all P, built from [BV15]
* [PS18] -- constrained and program privately for all P, built from [BV15]
[CVW18] -- constrained privately for BP, influenced by GGH15 mmaps

* uses gadget matrix G, adapted from the lattices-based FHE, ABE, PE

Open Question: Is there a transformation between Dual-Regev-based homomorphic schemes and GGH15-based ones?
Multilinear maps
GGH13, CLT13, **GGH15**

1. Private Constrained PRFs
[ Canetti, Chen 17 ]

2. General-purpose obfuscation
[ Gentry, Gorbunov, Halevi 15 ], …
Recall [ Canetti Chen 17 ]

“Obfuscation is almost private constrained PRF with two keys: One for the constraint C, the other one for all 1.”
Recall [Canetti Chen 17]
“Private constrained PRF is almost [GGHRSW 13] + [GGH 15] obfuscator with only one branch.”

The more “historically correct” view

Recall [Canetti Chen 17]
“Obfuscation is almost private constrained PRF with two keys: One for the constraint C, the other one for all 1.”
Recall [Canetti Chen 17]

“Private constrained PRF is almost [GGHRSW 13] + [GGH 15] obfuscator with only one branch.”

The constrained key for C

The constrained key for all 1
Claim 1: the proof strategy mentioned does not work.
Claim 2: when a sufficient amount of evaluation-to-0 is available, we can break the obfuscation scheme.
Claim 1: the proof strategy mentioned does not work.

Recall Toy example 2
Claim 1: the proof strategy mentioned does not work.

In the GGH15 obfuscator, it looks like …
In the GGH15 obfuscator, it looks like ...

Correlated

Can apply LWE, but don’t know how to use GPV

Claim 1: the proof strategy mentioned does not work.
Claim 2: when a sufficient amount of evaluation-to-0 is available, we can break the obfuscation scheme.

For x such that \( C(x) = 0 \), \( \text{Eval}(x) = \ldots = S_1E_2 + E_1D_2 \mod q \)

Recover \( S_1E_2 + E_1D_2 \) over integers, can do many things.

[ Cheon, Han, Lee, Ryu, Stehle 15], [ Coron, Lee, Lepoint, Tibouchi 16 ], [ Chen, Gentry, Halevi 17 ]
Claim 2: when a sufficient amount of evaluation-to-0 is available, we can break the obfuscation scheme.

For $x$ such that $C(x) = 0$, $\text{Eval}(x) = \ldots = S_1E_2 + E_1D_2 \mod q$

Recover $S_1E_2 + E_1D_2$ over integers, can do many things.

[ Cheon, Han, Lee, Ryu, Stehle 15 ], [ Coron, Lee, Lepoint, Tibouchi 16 ], [ Chen, Gentry, Halevi 17 ]

[ Chen, Vaikuntanathan, Wee 18 ] shows a classical polynomial attack, works as long as the inputs repeat for at most constant times.
First compute a matrix,

\[
W_{1,1} \ldots W_{1,k} \\
\ldots \quad \ldots \quad \ldots \\
W_{j,1} \ldots W_{j,k}
\]

\[=\]

Results on many inputs that eval to small
First compute a matrix, then compute the rank of the matrix.

\[
\begin{bmatrix}
W_{1,1} & \ldots & W_{1,k} \\
\vdots & \ddots & \vdots \\
W_{j,1} & \ldots & W_{j,k}
\end{bmatrix}
\]

\[ = \]

\[
\begin{bmatrix}
S_{1,1} & E_{1,1} & \ldots & D_{2,1} \\
S_{1,2} & E_{1,2} & \ldots & D_{2,k} \\
\vdots & \vdots & \ddots & \vdots \\
S_{1,j} & E_{1,j} & \ldots & D_{2,1}
\end{bmatrix}
\]

Heuristically random

[Chen, Vaikuntanathan, Wee 18]
Survey of iO candidates related to GGH15:

[ Gentry, Gorbunov, Halevi 15 ]: translate GGHRSW13 into GGH15
[ Chen, Gentry, Halevi 17 ]: quantum attack for simple branching program
[ Chen, Vaikuntanathan, Wee 18 ]: Break GGH15 with constant repetition, propose a candidate that enforce repetitions, non-commutative scalars, etc.
[ Bartusek, Guan, Ma, Zhandry 18 ]: Another candidate, proof in the idealized model
[ Cheon, Cho, Hhan, Kim, Lee 19 ]: Statistical attack on CVW18 for polynomial noise

Short summary:
Take [ Gentry, Gorbunov, Halevi 15 ], or [ Chen, Vaikuntanathan, Wee 18 ], using branching programs with super-constant repetitions, super-polynomial noise, no attacks are known, even quantum ones.
What to play next?
**Lockable obfuscation**
(Compute-then-Compare obf.)

**Private constrained PRFs**

Permutation branching program, almost always output 1 (random)

Witness encryption

Open Problem 4: classify

General evasive function obfuscators

Output 0 (small) very often

Multi-party key agreement

Indistinguishability obfuscation
Thought 1: on the proof technique
Thought 1: on the proof technique
Proof works when $A_1$ and $A_2$ are public, but they don’t have to be public ...
Lockable obfuscation  
(Compute-then-Compare obf.)

Private constrained PRFs

Permutation branching program, almost always output 1 (random)

[Chen, Vaikuntanathan, Wee 18]: proof beyond permutation BPs, using the fact that A matrices are hidden, but the S matrices are public

Still, witness encryption and general evasive function obfuscators are open

Output 0 (small) very often

Indistinguishability obfuscation
Thought 2: need new hard problems "without mod q"
LWE + low-degree “PRG”

[ Barak, Hopkins, Jain, Kothari, Sahai 19 ], [ Jain, Lin, Matt, Sahai 19 ]

LWE + degree 3 functions over \( \mathbb{Z} \):

\[
A, \quad s^T A + e^T \mod q, \quad \{ Q_i, Q_i(x, y, e) \}, \quad i = 1 \text{ to } N
\]

The adversary is asked to recover \( e \). Here \( x, y, e \) are small and of dimension \( m \), \( Q_i \) are degree-3 “small” polynomials over \( \mathbb{Z} \), \( N = m^{1.01} \)

Bilinear maps + LWE + low-degree “PRG”

\( \Rightarrow \) Succinct Functional Encryption for low-degree function

\( \Rightarrow \) iO

Open Problem 5: break it.

Open Problem 6: if not, build iO from it directly.
The efficiency of GGH15

Eval(0110)

= \(A_0D_{1,0}D_{2,1}D_{3,1}D_{4,0}\)

= \((s_{1,0}A_1+E_{1,0})D_{2,1}D_{3,1}D_{4,0}\)

= \(s_{1,0}A_1D_{2,1}D_{3,1}D_{4,0} + \text{"small"}\)

= \(s_{1,0}(s_{2,1}A_2+E_{2,1})D_{3,1}D_{4,0} + \text{"small"}\)

= \(s_{1,0}s_{2,1}A_2D_{3,1}D_{4,0} + \text{"still small"}\)

= \(s_{1,0}s_{2,1}s_{3,1}A_3D_{4,0} + \text{"still smallish"}\)

= \(s_{1,0}s_{2,1}s_{3,1}s_{4,0}A_4 + \text{"small"}\)

The “small” noise grows exponentially with #levels, becomes a problem in the efficiency.
Open Problem 7: construct PCPRF or LO based on GGH13 or CLT13, prove security from a concrete assumption, like NTRU or approx-gcd.

Likely to give new insights on GGH13 and CLT13, and improve efficiency.
LWE => iO = $100

#7 with further investigation

Update: During the talk, Amit raised the award for “iO from LWE” to $1000.
Happy lunar new year!