Walking the Edge between Structure and Randomness: The Quest for Indistinguishability Obfuscation

Shweta Agrawal
IIT Madras

Bootstrapping Obfuscation from Noisy Linear FE

Shweta Agrawal
IIT Madras
Indistinguishability Obfuscator $iO$ [BGI+01]

“Which one of two equivalent circuits $C_1 \equiv C_2$ is obfuscated?”

$C_1 \equiv C_2$, meaning
- Same size $|C_1| = |C_2|
- Same truth table $TB(C_1) = TB(C_2)$

\[
\{ iO(C_1) \} \approx \{ iO(C_2) \}
\]

**Quest:** Finding an efficient compiler $iO$
Does \textit{iO} exist?

- **Direct** Constructions
  - All based on multilinear maps [GGH13,CLT13,GGH15]
  - Same template in all works (all eggs in same basket?)
  - Many attacks, fixes, repeat: hard to understand security

- **Bootstrapping** based constructions
Bootsrapping Based Constructions: Reduce, Reduce, Reduce, Reduce

- What is the minimum functionality needed for iO?
- How much can we “clean up” assumptions?
- Much progress
- State of art: degree 3 multilinear maps and degree 3 “block local” pseudorandom generators [LT17].
- Not clear how to instantiate deg 3 maps
Can we base \(\mathcal{IO}\) on anything else?

- *Functional Encryption* supporting computation of degree \(\geq 3\) polynomials
- Should be good news except.....
  - All constructions of functional encryption themselves based on multilinear maps 😞
(mpk, msk) $\leftarrow$ \textbf{Setup}(1^n)

\textbf{Enc}(mpk, m):

\textbf{Kgen}(msk, C):

\textbf{Dec}(sk_C, ct):

y = C(m)
The State of Affairs

• Using bilinear maps, have FE for degree 2 polys
• For $iO$, need FE for degree (at least) 3 polys
• Where does degree 3 come from?
• Arithmetic degree required to compute a PRG.

Expansion/stretch: Difference in output and input lengths, i.e $|G(seed)| - |seed|$
The State of Affairs

• Need FE to compute $G(seed)$.

• Represent $G$ as polynomial.

• $G$ associated with pair $(L,E)$ where poly of degree $L$ is required to compute PRG of expansion $E$. 

$CT(seed) \Rightarrow sk_G \Rightarrow y = G(seed)$
Previously....

Lin-Tessaro, Crypto 17

Bootstrapping, determines Exp $\mathcal{E}$

FE for NC

iO for P/Poly

$[\text{AJ15, BV15, BNPW16}]$

FE for NC$_1$

Uses randomizing polys $[\text{AIK11, LV16}]$

FE for NC$_0$

$[\text{LT17, Lin17, AS17}]$

PRG of Deg $\mathcal{L}$, Exp $\mathcal{E}$

$(2, \mathcal{E})$ possible?

Recall, G associated with $(\mathcal{L}, \mathcal{E})$

FE for deg $\mathcal{L}$ poly

Exists for deg 2 using bilinear maps $[\text{Lin17, BCFG17}]$
The State of Affairs

• All works use randomizing polynomials for bootstrapping.
• Necessitates Boolean PRG with expansion $\mathcal{E}$.
• Previously, Lin-Tessaro conjecture PRG with degree 2 and expansion $\mathcal{E}$
• LV17, BBKK17 show that degree 2 impossible for expansion $\mathcal{E}$
• Narrow margin of expansion $\mathcal{E}'$ left open by attacks
• But not clear how to use this to build iO.
New Abstraction to build iO [A19]

Three different ways to instantiate abstraction.

- LWE/RLWE
  - Noisy Linear FE
    - FE for NC
      - Uses special “FE-compatible” FHE of [AR17]

- PK FE for PRG
  - Introduce non Boolean PRG
    - Boolean PRG lower expansion

- SK FH FE for CNG
  - Introduce CNG
    - New PRG families

- Direct Construction
  - New hardness conjectures

Sidesteps lower bound
Noisy Linear FE: The right abstraction?

- Recall **Linear FE** [ABDP15,ALS16]: Enc(x), Keygen(y), Decrypt to get <x,y>.
- Possible from standard assumptions – DDH, LWE, QR
- Noisy Linear FE: Enc(x), Keygen(y), Decrypt to get <x,y> **plus noise**
- Where does noise come from?
- What security properties does it need to satisfy?
- Going in circles?

---

- Noise must satisfy only mild statistical properties
- Need not be Boolean, as no need for randomized enc
- A key new observation: Computing a noise term is easier as exact value not important

Weakening requirements on PRGs
Advantage 1: Relax requirements on PRGs

• Boolean: make do with lower expansion $E'$ which is not ruled out

• Non-Boolean PRGs: Two new classes of randomness generators
  • Need not be Boolean: save degree blowup (caused by arithmetization)
  • Satisfy much weaker property than pseudorandomness
  • May be computable with lower degree

• How to instantiate these PRGs?

Won’t have time to talk about this
Advantage 2: Permits Mixed Assumptions

• Noisy linear FE is used **black box** in bootstrapping

• Bootstrapping uses LWE.

• Noisy linear FE may use **any assumption**
  • In one instance, *mix of pairings and lattices*. Uses best of both.

Garuda: Mythical Indian character
Half eagle, half man.
Advantage 3: More Efficient

• Previously FE for degree $L \rightarrow$ FE for NC$_0$ $\rightarrow$ FE for NC$_1$ $\rightarrow$ iO

• Can bootstrap to FE for NC$_1$ directly.

  Noisy Linear FE $\rightarrow$ FE for NC$_0$ $\rightarrow$ FE for NC$_1$ $\rightarrow$ iO

  $\Rightarrow$ FE for degree $L \rightarrow$ FE for NC$_0$ $\rightarrow$ FE for NC$_1$ $\rightarrow$ iO

• More efficient
Advantage 4: Permits New Direct Constructions

- Previously, all direct constructions use same template:
  - Fix plaintext matrix branching program
  - Add randomization layers: diagonal padding, random scalars, Kilian randomization
  - Encode randomized matrices using one of three mmap families
- All eggs in same basket
- Noisy linear FE can be constructed without multilinear, or even bilinear maps!
Advantage 4: Permits New Direct Constructions
A key new observation: Old grandma advice!

If you cannot have what you want, you must learn to want what you can have
A key new observation: Relax requirement on correctness!

If you cannot compute what you can use, you must learn to use what you can compute.
A key new observation: Relax requirement on correctness!

• Only \(<x,y>\) needs to be correct! \(G(\text{seed})\) is allowed some corruption
• So far: Assume polynomial is PRG and insist on computing it exactly
• Here: Compute whatever can be computed and check if it can satisfy PRG like properties
Advantage 4: Permits New Direct Constructions

• Extend LWE based Linear FE ofALS16 to Noisy Linear FE using new hardness conjectures on lattices.
• Unrelated to multilinear map assumptions (modulo mathematical structure)
• May be more robust. May be post-quantum.
• Much simpler to analyse than mmap based direct constructions: no need for straddling sets, Kilian randomization etc used by all prior work
• First construction of nonlinear FE without any maps.
• Philosophically similar idea used in follow-up [JLMS19]
Bootstrapping Functional Encryption

Noisy Linear FE $\rightarrow$ FE for $NC_0$ $\rightarrow$ FE for $NC_1$ $\rightarrow$ iO
Ring Learning with Errors Problem

Let ring

\[ R_q = \mathbb{Z}_q[x]/ < x^n + 1 > \]

\textbf{DISTRIBUTION 1}

Sample \( s \) uniformly in \( R_q \)

\[ \begin{align*}
  a_1, b_1 &= a_1 s + \text{err}_1 \\
  a_2, b_2 &= a_2 s + \text{err}_2 \\
  &\vdots \\
  a_m, b_m &= a_m s + \text{err}_m \\
\end{align*} \]

\( a_i \text{ uniform } \in R_q, \ e_i \sim \varphi \in R \)

\textbf{DISTRIBUTION 2}

\[ \begin{align*}
  a'_1, b'_1 \\
  a'_2, b'_2 \\
  &\vdots \\
  a'_m, b'_m \\
\end{align*} \]

\( a_i, b_i \text{ uniform } \in R_q \)
Regev Public Key Encryption

Finding short \( \tilde{e} \) such that \( \langle \tilde{a}; \tilde{e} \rangle = u \) is hard

- **SK**: \( \tilde{e} \)  
  **PK**: \( \tilde{a}, u \)

- **Encrypt** (PK, m):
  \[ c_0 = \tilde{a} \cdot s + \tilde{err}_1 \]
  \[ c_1 = u \cdot s + err_2 + m \left\lfloor \frac{q}{2} \right\rfloor \]

- **Decrypt** (SK):
  \[ c_1 - \langle \tilde{e}; c_0 \rangle \]
  \[ = u \cdot s + err_2 + m \left\lfloor \frac{q}{2} \right\rfloor - u \cdot s - \langle \tilde{e}; \tilde{err}_1 \rangle \]
  \[ = m \left\lfloor \frac{q}{2} \right\rfloor + \text{error} \]

Pseudorandom
By R-LWE

Small only if e is small
Regev Public Key Encryption

- **SK**: $\bar{e}$
- **PK**: $\bar{a}, u$

**Encrypt (PK, m):**

- $c_0 = \bar{a} \cdot s + \bar{err}_1$
- $c_1 = u \cdot s + err_2 + m \left\lfloor \frac{q}{2} \right\rfloor$

**Decrypt (SK):**

- $c_1 - \langle \bar{e}; c_0 \rangle$
- $= u \cdot s + err_2 + m \left\lfloor \frac{q}{2} \right\rfloor - u \cdot s - \langle \bar{e}; \bar{err}_1 \rangle$
- $= m \left\lfloor \frac{q}{2} \right\rfloor + error$

**Randomness encoding**

**Message encoding**
What’s special about this PKE?

Lends Itself to Fully Homomorphic Encryption!
Encrypt \((s, x_1, x_2)\):
Sample \(u_1, u_2\) randomly in ring. Sample \(err_1, err_2\).
Compute:
\[
\begin{align*}
c_1 &= u_1 s + err_1 + x_1 \\
c_2 &= u_2 s + err_2 + x_2 
\end{align*}
\]
Evaluate \((c_1, c_2, f = x_1 x_2)\):

Want: Use \(c_1, c_2\) to compute product ciphertext \(c_{12}\) that encrypts \(x_1 x_2\)
FHE Evaluation

We may write:

\[ x_1 \approx c_1 - u_1 s \]
\[ x_2 \approx c_2 - u_2 s \]
\[ \therefore x_1 x_2 \approx c_1 c_2 - (c_1 u_2 + c_2 u_1) s + u_1 u_2 s^2 \]

Let \( c^{\text{mult}} = (c_1 \ c_2, \ c_1 \ u_2 + c_2 \ u_1, \ u_1 u_2) \)

Given secret key \( s \), and ciphertext \( c^{\text{mult}} \), decryptor can recover message up to noise.
Onwards to Functional Encryption

• FHE secret key $s$ permits decryptor to also decrypt original messages $x_1$ and $x_2$.

• Wish to constrain decryption so that key holder learns $x_1x_2$ but not individual $x_1$ and $x_2$.
Recall....

Regev PKE Ciphertext:

\[
\tilde{c} = \tilde{a} \cdot s + \tilde{err}_1
\]

\[
c_1 = u_1 \cdot s + err_2 + x_1
\]

\[
c_2 = u_2 \cdot s + err_2 + x_2
\]

Observe:

• Ciphertext can be split into randomness carrier “c” and message carrier \(c_1, c_2\).

• Message carrier exactly resembles FHE Symmetric key CT
The Hope

• **Switch View to FHE:**
  • Interpret PKE message carrier components as FHE symmetric key ciphertexts

• **FHE Computation:**
  • Evaluate \( f \) on encrypted data using FHE.
  • Recover encryption of \( f(x) \)

\[
\begin{align*}
  c_f &= u_f \cdot s + err_f + f(\bar{x}) \\
  \bar{c} &= \bar{a} \cdot s + \bar{err}_1
\end{align*}
\]

• **Switch view back to Regev PKE:**
  Given randomness carrier and message carrier for \( f(x) \), decrypt as in original Regev PKE

\[
\begin{align*}
  \bar{c} &= \bar{a} \cdot s + \bar{err}_1 \\
  c_f &= u_f \cdot s + err_f + f(\bar{x})
\end{align*}
\]
The News (Good and Bad)

• The Bad: Too good to be true
• Want FHE computation to result in a CT

\[ c_f = u_f \cdot s + err_f + f(\bar{x}) \]

• Instead, CT looks like

\[ c_f = u_{f,ct} \cdot s + err_f + f(\bar{x}) \]

• The Good: This blueprint works for linear functions
• Given CT(x), SK(y), decryption outputs <x,y>
• We’ll leverage linear functions to support deeper circuits
Generalizing to Quadratic (AR17)

• Recall FHE multiplication:

\[ x_1 \approx c_1 - u_1 s \]
\[ x_2 \approx c_2 - u_2 s \]
\[ \therefore x_1 x_2 \approx c_1 c_2 - (c_1 u_2 + c_2 u_1) s + u_1 u_2 s^2 \]

• What if we group the terms differently?

\[ \therefore x_1 x_2 \approx c_1 c_2 - u_2 (c_1 s) - u_1 (c_2 s) + u_1 u_2 (s^2) \]
Functional Encryption for Quadratic polynomials $P(x) = x_1x_2$

$\text{Enc}(\text{mpk}, \tilde{x} = (x_1 \ldots x_n))$: 
\[\{c_i = u_i \cdot s + err_i + x_i\}_{i \in [n]}\]

$\text{LinearFE.Enc}(s^2, c_1 s, \ldots c_n s)$

$\text{Dec}(\text{sk}_P, \text{ct}_x)$:

LinearFE.Dec

$\langle (s^2, c_1 s, \ldots, c_n s), (u_1 u_2, -u_2, -u_1, 0, \ldots, 0) \rangle$

$= u_1 u_2 (s^2) - u_2 (c_1 s) - u_1 (c_2 s)$

$= c_1 c_2 - u_2 c_1 s - u_1 c_2 s + u_1 u_2 s^2 - c_1 c_2$

$\approx x_1 x_2 - c_1 c_2$

Decryptor can compute $c_1 c_2$ itself.
Is this secure?

• **Attack**: Easily recover $s$ given exact linear equation

\[ u_2(c_1s) + u_1(c_2s) + u_1u_2(s^2) \]

• Intuition for **fix**: Exact linear equations trivial, noisy linear equations intractable

**Motto**: Add noise!

• Replace linear FE by **noisy linear FE**.
• This is not a proof!
• New proof technique to show this works
Wrapping it up

• Generalizes to NC₁ via induction
  • Very technical!
  • Need encryptor to provide some extra encodings as “advice”
  • Need to only compute linear function plus noise, i.e. noisy linear FE

• Can use FE for PRG or direct construction to generate noise

• Concurrent work by AJS18 identify similar classes of PRG, incomparable results

• Follow up work by LM18 improves assumption on PRG by handling leakage caused by polynomial bounded PRG
Walking the Edge between Structure and Randomness: The Quest for Indistinguishability Obfuscation

Shweta Agrawal, IIT Madras

Thank You for your attention 😊

Image Credits: Jackson Pollock, who solves similar problems in a different space!