Limits on the Power of Indistinguishability Obfuscation

Gilad Asharov
Gil Segev

IBM

THE HEBREW UNIVERSITY OF JERUSALEM
Limits on the Power of iO

• Limits on the Power of Indistinguishability Obfuscation (and Functional Encryption)
  • FOCS 2015

• On Constructing One-Way Permutations from Indistinguishability Obfuscation
  • TCC 2016A
Obfuscation

• Makes a program “unintelligible” while preserving its functionality

```javascript
for (i=0; i < M.length; i++) {
    // Adjust position of clock hands
    var ML=(ns)?document.layers['nsMinutes'+i]:ieMinutes[i].style;
    ML.top=y[i]+HandY+(i*HandHeight)*Math.sin(min)+scrll;
    ML.left=x[i]+HandX+(i*HandWidth)*Math.cos(min);
}
```

```javascript
for(O79=0;O79<l6x.length;O79++){var O63=(l70)?document.layers
["nsM\151\156u\164\145s"+O79]:ieMinutes[O79].style;
O63.top=161[O79]+O76+(O79*O75)*Math.sin(O51)+173;
O63.left=175[O79]+177+(O79*176)*Math.cos(O51);}
```
Obfuscation

• [Barak Goldreich Impagliazzo Rudich Sahai Vadhan Yang01] :
  • Virtual black-box obfuscation (VBB)
    Obfuscated program reveals no more than a black box implementing the program
    Impossible
  • Indistinguishability obfuscation (iO)
    Obfuscations of any two functionally-equivalent programs be computationally indistinguishable
    May be possible?

• [Garg Gentry Halevi Raykova Sahai Waters12] :
  A candidate indistinguishability obfuscator (iO)
Indistinguishability Obfuscation

• An efficient algorithm $iO$
  Receives a circuit $C$, outputs an obfuscated circuit $\hat{C}$

• **Preserves functionality**: $C(x) = \hat{C}(x)$ for all $x$

• **Indistinguishability**: For every PPT distinguisher $D$, for every pair of functionally-equivalent circuits $C_1$ and $C_2$

  $$\left| \Pr[D(iO(C_1)) = 1] - \Pr[D(iO(C_2)) = 1] \right| < \text{negl}(n)$$

• What can be constructed using $iO$?
The Power of Indistinguishability Obfuscation

• Public-key encryption, short “hash-and-sign” signatures, CCA-secure public-key encryption, non-interactive zero-knowledge proofs, injective trapdoor functions, oblivious transfer [SW14]
• Deniable encryption scheme [SW14]
• One-way functions [KMN+14]
• Trapdoor permutations [BPW15]
• Multiparty key exchange [BZ14]
• Efficient traitor tracing [BZ14]
• Full-domain hash without random oracles [HSW14]
• Multi-input functional encryption [GGG+14, AJ15]

• Functional encryption for randomized functionalities [GJK+15]
• Adaptively-secure multiparty computation [GGH+14a, CGP15, DMR15, GP15]
• Communication-efficient secure computation [HW15]
• Adaptively-secure functional encryption [Wat14]
• Polynomially-many hardcore bits for any one-way function [BST14]
• ZAPs and non-interactive witness-indistinguishable proofs [BP15]
• Constant-round zero-knowledge proofs [CLP14]
• Fully-homomorphic encryption [CLT+15]
• Cryptographic hardness for the complexity class PPAD [BPR14]

(Last update: April 2015)
The Power of Indistinguishability Obfuscation
Is there a natural task that **cannot** be solved using indistinguishability obfuscation?
Yes

(probably…)
Black-Box Separations

• The main technique for proving lower bound in cryptography [IR89]:
  **Black Box Separations**

• The vast majority of constructions in cryptography are “black box”

  "**Building a primitive X from any implementation of a primitive Y**"

  • The construction and security proof rely only on the input-output behavior of Y and of X's adversary
  • The construction ignores the internal structure of Y

• **Examples:**
  • PRF from PRG [GGM86], PRG from OWFs [HILL93]
Black-Box Separations

• Impossibility of black-box constructions

• Typically, show impossibility of “X ⇒ Y” by:

  “There exists an oracle relative to which Y exists but X does not exist”

• Examples:
  • No key agreement from OWFs [IR89]
  • No CRHF from OWFs [Sim98]
Our Challenge: Non-Black-Box Constructions

• Constructions that are based on \(iO\), almost always have some non-black-box ingredient

• Typical example
  From private-key to public-key encryption [SW14] (simplified)
    • Private-key scheme: \(\text{Enc}(K,m) = (r, \text{PRF}(K,r) \oplus m)\)
    • Public-key scheme: \(SK = K, \ PK = iO(\text{Enc}(K,:))\)

Non-black-box ingredient:
Need the specific evaluation circuit of the PRF

How can one reason about such non-black-box techniques?
Our Solution

• Overcome this challenge by considering $iO$ for a richer class of circuits:

**oracle-aided circuits**

(circuits with oracle gates)
Our Solution

- Transform *almost all* iO-based constructions from non-black-box to black-box
  \[ \text{iO}(r, \text{PRF}(K, r) \oplus m) \]
  \[ \text{iO}(r, C^{\text{OWF}}(K, r) \oplus m) \]
  (possible due to [GGM86]+[HILL89])

- Constructing iO for *oracle-aided* circuits is clearly *as hard as then* constructing iO for *standard circuits*

- Limits on the power of iO for *oracle-aided* circuits thus *imply* limits on the power of iO for *standard circuits*
Techniques We Don’t Capture

• Constructions that use NIZK proofs for languages that are defined relative to a computational primitive

• **NIZK proof** \[ L = \{(d,r) \mid \exists r \text{ s.t. } d = Enc(i;r)\} \]
  - Uses Cook-Levin reduction to SAT
  - This reduction uses the circuit for deciding \( L \) (representing its computation state as boolean formula) - **non-black-box**

• [BKSY11] seems as a promising approach for extending our framework to capture such constructions

• Other (less common) techniques (so far not used with iO)
On Constructing One-Way Permutations from Indistinguishability Obfuscation
One-Way Permutation

- One of the most fundamental primitives in cryptography
- Enabling elegant constructions of a wide variety of cryptographic primitives
  - Universal one-way hash function
  - Pseudorandom generators
One-Way Permutation

• **One-Way Functions:** Many candidates
• **One-Way Permutations:** Only few candidates
  • Based on hardness of problems related to discrete logarithms and factoring

• [Rudich88,…]:
  No black-box construction of a one-way permutation from a one-way function
TDP from iO+OWF

[BitanskyPanethWichs15]

(i, PRF_K(i)) → (i+1, PRF_K(i+1))

Elements:

(i, PRF_K(i))
TDP from iO+OWF

[BitanskyPanethWichs15]

Next(x):
If x=(i,PRF_K(i))
  Output (i+1,PRF_K(i+1))
Output ⊥
TDP from iO+OWF

(BitanskyPanethWichs15)

Next(x):
If \( X = (i, \text{PRF}_K(i)) \)
Output \((i+1, \text{PRF}_K(i+1))\)
Output ⊥
Question 1:

Can we construct a *single* one-way permutation over \( \{0,1\}^n \) from iO+OWF?
The domain depends on the specific PRF. For the same K, different underlying PRF - different domain!
Question 2:

Can we construct a family where the domain does not depend on the underlying building blocks (iO+OWF)?

We call a construction where the domain does not depend on the underlying building blocks as “domain invariant”
Back to [Rudich88,...]

- Separation of OWP from OWF
- Rules out only a **single domain-invariant** permutation
  - Rudich assumes that the domain is independent of the OWF
Question 3:

Can we construct a non-domain-invariant OWP (family) from a OWF?
Our Results

Can we construct a *single* one-way permutation over \( \{0,1\}^n \) from iO+OWF?

**NO.**

Can we construct a *family* where the domain *does not depend* on the underlying building blocks (iO+OWF)?

**NO.**

Can we construct a *non-domain-invariant* OWP (family) from a OWF?

**NO.**
Theorem 1:
There is no fully black-box construction of a domain-invariant one-way permutation family from
• a one-way function $f$ and
• an indistinguishability obfuscator for all oracle-aided circuits $C_f$

Unless with an exponential security loss (rules out sub-exponential hardness as well!)
• **Theorem 2:** There is no fully black-box construction of a non-domain-invariant one-way permutation family from
  • a one-way function \( f \)

• Unless with an **exponential** security loss (rules out sub-exponential hardness as well!)
So.. What do we have?

- Domain-invariant OWP
- Domain-invariant OWP family
- OWF
- iO + OWF
- Thm. 1.1
- Thm. 1.2
- [BPW15]
- [Rud88, …]
Proof Sketch

• Builds upon and generalizes [Rudich88, MatsudaMatsuura11, AsharovSegev15]

• We define an oracle $\mathcal{I}$ such that relative to it:
  1. There exists a one-way function $f$
  2. There exists an indistinguishability obfuscator for all oracle-aided circuits $\mathcal{C}_f$
  3. There does not exist a domain-invariant one-way permutation family
The one-way function $f$

$$f = \{f_n\}_n,$$ where each $f_n : \{0,1\}^n \rightarrow \{0,1\}^n$ is a uniformly chosen function

O and Eval

$$O = \{O_n\}_{n \in \mathbb{N}},$$ where each $O_n$ is a uniformly chosen injective function $\{0,1\}^{2n} \rightarrow \{0,1\}^{10n}$

$Eval(\tilde{C}, a)$ with $|\tilde{C}| = 10n$, $|a| = n$

Looks for the pair $(C, r) \in \{0,1\}^{2n}$ such that $O_n(C, r) = \tilde{C}$
If exists, returns $C^f(a)$
Otherwise, returns $\bot$

- **We implement iO as follows:**
  $$\hat{C}(\cdot) = iO(C)$$
  - On input oracle-aided circuit $C$ (with $|C| = n$), choose a random $r$
  - Outputs $\tilde{C} = O_n(C, r)$
We Need to Show

• We define an oracle \( \Phi \) such that relative to it:
  1. There exists a one-way function \( f \)
     (somewhat similar to [AS15])
  2. There exists an indistinguishability obfuscator
     for all oracle-aided circuits \( C_f \)
     (somewhat similar to [AS15])
  3. There does not exist a domain-invariant one-way permutation family
Warm-up: Rudich's Attack in the Random-Oracle Model

$\mathbf{f}$ Random oracle

$\mathbf{P}^f$ One-Way Permutation over domain $\mathcal{D}$ for every function $\mathbf{f}$

**Theorem:**

There exists an oracle-aided adversary $\mathcal{A}$ that makes polynomially many queries, such that for every $\mathbf{f}, \mathbf{x}^*$

$$\Pr[\mathcal{A}^f(\mathbf{y}^*) = \mathbf{x}^*] = 1$$

where $\mathbf{y}^* = \mathbf{P}^f(\mathbf{x}^*)$
The Adversary

• **Input:** some element $y^* \in D$
• **Oracle access:** the random oracle $f$
  • Initializes a set of queries $Q$
    (initially empty. always consistent with $f$)
  • Repeats the following for polynomially many times:
    • **Simulation:** $\mathcal{A}$ finds an input $x' \in D$ and a set of oracle/queries $f'$ that is consistent with $Q$, such that $P^{f'}(x')=y^*$
    • **Evaluation:** $\mathcal{A}$ evaluates $P^f(x')$. If $y^*$ - found!
    • **Update:** $\mathcal{A}$ asks $f$ for all queries in $f'$ that are not in $Q$, and update $Q$
The Claim

- In every iteration, one of the following:
  - \( A \) finds \( x^* \), (i.e., \( x' = x^* \) where \( P^f(x^*) = y^* \)) or
  - In the update phase, \( A \) queries \( f \) with at least one query that is made in the computation of \( P^f(x^*) = y^* \)

- **Input:** some element \( y^* \in D \)
- **Oracle access:** \( f \)
  - Initializes a set of queries \( Q \) (initially empty, always consistent with \( f \))
  - Repeats the following for polynomially many times:
    - **Simulation:** \( A \) finds an input \( x' \in D \) and a set of oracle/queries \( f' \) that is consistent with \( Q \), such that \( P^f(x') = y^* \)
    - **Evaluation:** \( A \) evaluates \( P(x') \). If \( y^* \) - found!
    - **Update:** \( A \) asks \( f \) for all queries in \( f' \) that are not in \( Q \), and update \( Q \)
Otherwise

\[ P^{f'}(x') = y^* \]

\[ P^f(x^*) = y^* \]

\[ \alpha \text{ in } Q: \quad f''(\alpha) := f(\alpha) \]

\[ \alpha \text{ appears in } P^{f'}(x'): \quad f''(\alpha) := f'(\alpha) \]

\[ \alpha \text{ appears in } P^f(x^*): \quad f''(\alpha) := f(\alpha) \]

\[ P^{f''}(x') = y^* \]

\[ P^{f''}(x^*) = y^* \]

- In every iteration, one of the following:
  - \( A \) finds \( x^* \), or
  - In the update phase, \( A \) queries \( f \) with at least one query that is made in the computation of \( P^f(x^*) = y^* \)
Otherwise

\[ Pf'(x') = y^* \]
\[ Pf(x^*) = y^* \]

In every iteration, one of the following:
- \( A \) finds \( x^* \), or
- In the update phase, \( A \) queries \( f \) with at least one query that is made in the computation of \( Pf(x^*) = y^* \)

\[ P f''(x') = y^* \]
\[ P f''(x^*) = y^* \]

\[ x' \neq x^* \]
In Our Setting

- **Challenges:**
  - Family and not just a single permutation
  - Our oracle $\Gamma$ is much more structured than just a random oracle

- **$\Gamma$ consists of:**
  - Length preserving function $f$
  - *Injective* length-increasing function $O$
  - “Evaluation” oracle $Eval$

---

**Recall [BPW15]:**
Relative to $\Gamma$ there exists a construction of a non-domain invariant one-way permutation family!!
Regarding $O$

- $\Gamma$ consists of:
  - length preserving function $f$
  - injective length-increasing function $O$
  - "evaluation" oracle $Eval$

\[ Q, P^\Gamma(x') = y^*, P^\Gamma(x^*) = y^* \]

\[ O'(\alpha) = \beta, \quad O(\delta) = \beta \]

\[ O''(\alpha) = \beta, \quad O''(\delta) = \beta \]

Non-injective!
Regarding O and Eval

- Γ consists of:
  - length preserving function f
  - injective length-increasing function O
  - “evaluation” oracle Eval

\[
Q \quad P^Γ'(x') = y^* \quad P^Γ(x^*) = y^*
\]

\[
O'(C, r) = \hat{C} \quad \text{Eval}(\hat{C}, d) = \bot
\]

\[
O''(C, r) = \hat{C} \quad \text{Eval}''(\hat{C}, d) = C^f(d)
\]

incorrect!
The Proof

• Very subtle

• Carefully define the dependencies between oracles in order to avoid the above scenarios

• Regarding $O$: choose the oracle $O'$ uniformly at random from the set of all oracles that are consistent with $Q$
  • We show that with high probability
    • $O'$ avoids the image of $O$
    • $O'$ avoids the invalid $Eval$ calls
    • It is possible to construct the hybrid oracle $\Gamma$
    • Relies on the fact that $O$ is length-increasing

Further details: see the paper
Theorem:
There is no fully black-box construction of a non-domain-invariant one-way permutation family from
• a one-way function f

Unless with an exponential security loss (rules out sub-exponential hardness as well!)
Non-Domain-Invariant Family

\[ \alpha \leftarrow \text{Gen}^f(1^n) \quad x \leftarrow \text{Samp}^f(\alpha) \quad y \leftarrow \text{P}^f(\alpha, x) \]

Different \( f \): completely different set of indices (different family)

The domain \( D_{\alpha}^f \): depends both on \( \alpha \), \( f \)

Careful! \( \alpha \) may be invalid w.r.t \( f \)
\( x \) may not be in \( D_{\alpha}^f \)

Example [BPW15]
A non-domain-invariant family (uses both OWF and iO):
The index depends on iO+OWF
The domain depends on OWF only (and not on the index)
Challenges: Constructing the Hybrid Oracle

\[ P_{f'}(\alpha, x') = y^* \quad Q \quad P_{f}(\alpha, x^*) = y^* \]

**Define** \( f'' \):

- \( \alpha \) in Q: \( f''(\alpha) := f(\alpha) \)
- \( \alpha \) appears in \( P_{f'}(\alpha, x') \): \( f''(\alpha) := f'(\alpha) \)
- \( \alpha \) appears in \( P_{f}(\alpha, x^*) \): \( f''(\alpha) := f(\alpha) \)

1. No guarantee that \( \alpha \) is a valid index relative to \( f'' \)
2. No guarantee that \( y^* \) is in the domain of \( D_{\alpha}f'' \)
3. The same for \( x' \) and \( x^* \)
Solutions

• Adversary is given $\alpha$, $y^*$
  • Sample in addition to $f'$:
    • A “certificate” that $\alpha$ is a valid index respectively to $f'$
    • A “certificate” that $x'$ is a valid element in the domain of $\alpha$ respective to $f'$
  • For $\alpha$, $x^*$ there also exist certificates such that
    • $\alpha$ is a valid index respectively to $f$
    • $x^*$ is a valid element in the domain of $\alpha$ respective to $f$
    • Using these certificate, build $f''$
      • Guarantees that $\alpha$, $x'$, $x^*$, $y^*$ are valid respective to $f''$

Further details: see the paper
Conclusions

Thank You!