Hiding secrets in public random functions

Yilei Chen

Dissertation committee
Adam Smith, Ran Canetti, Leonid Reyzin, Vinod Vaikuntanathan, Mayank Varia
> July 16, 2018, Boston, heavy snow.
> July 16, 2018, Boston, heavy snow. Alice finds a classical polynomial time algorithm for factoring.
July 16, 2018, Boston, heavy snow. Alice finds a classical polynomial time algorithm for factoring.

Instead of putting it in her thesis, she thinks it’s cool to write a program and post it on Github.
> Wait, let’s have some fun.
> Wait, let’s have some fun.

> obfuscate factoring.hs >> idontknowwhatimdoing.hs

XOpenDisplay( 0); z=RootWindow(e,0);
for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0));
scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++);
XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400, 0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval ......
> Maybe more?
> Maybe more?

> watermark idontknowwhatimdoing.hs

XOpenDisplay(0); z=RootWindow(e,0);
for (XSetForeground(e,k=XCreateGC(e,z,0,0),BlackPixel(e,0));
scanf("%lf%lf%lf",y+n,w+y,y+s)+1; y ++);
XSelectInput(e,z=XCreateSimpleWindow(e,z,0,0,400,400,0,0,WhitePixel(e,0)),KeyPressMask);
for(XMapWindow(e,z);
; T=sin(O)){ struct timeval ......}
... was a nice dream
> Alices~: factoring.hs  21

> Alices~: obfuscate factoring.hs

> Alices~: watermark idontknowwhatimdoing.hs
> Alices~: factoring.hs  21
21 = 1 x 21

> Alices~: obfuscate factoring.hs
How?

> Alices~: watermark idontknowwhatimdoing.hs
What?
My research:
How to achieve these advanced cryptographic capabilities based on hard mathematical problems (or break them)?

Watermarking
Obfuscation

How to factorize an integer?
The plan for the talk:

> Overview of our research

> Two specific works related to hiding secrets in public random functions
buzzwords in cryptography
A discrete subgroup of $\mathbb{R}^n$. 

Lattice
A discrete subgroup of $\mathbb{R}^n$. 

Find the shortest non-zero vector in a lattice?
A discrete subgroup of $\mathbb{R}^n$.

Find the shortest non-zero vector in a lattice:
Easy for 2 dimensional lattices
Lattice

A discrete subgroup of $\mathbb{R}^n$.

Find the shortest non-zero vector in a lattice:
Easy for 2 dimensional lattices

Find short vectors in n-dimensional lattices
A discrete subgroup of $\mathbb{R}^n$.

Find the shortest non-zero vector in a lattice:
   Easy for 2 dimensional lattices

Find short vectors in n-dimensional lattices
   $\Rightarrow$ Exponential approximation [Lenstra, Lenstra, Lovasz 82]
   …
Lattice

A discrete subgroup of $\mathbb{R}^n$.

Find the shortest non-zero vector in a lattice:
- Easy for 2-dimensional lattices

Find short vectors in n-dimensional lattices
- $\Rightarrow$ Exponential approximation [Lenstra, Lenstra, Lovasz 82]
- $\Rightarrow$ One-way function [Ajtai 96]
- $\Rightarrow$ Public-key encryption [Ajtai, Dwork 97, Regev 05]
- $\Rightarrow$ Homomorphic enc [Gentry 09, Brakerski, Vaikuntanathan 11]

...
A discrete subgroup of $\mathbb{R}^n$.

Find the shortest non-zero vector in a lattice:
Easy for 2 dimensional lattices

Find short vectors in n-dimensional lattices
=> Exponential approximation [ Lenstra, Lenstra, Lovasz 82 ]
...
=> One-way function [ Ajtai 96 ]
=> Public-key encryption [ Ajtai, Dwork 97, Regev 05 ]
=> Homomorphic enc [ Gentry 09, Brakerski, Vaikuntanathan 11 ]
...

=> ???
Multilinear maps

A new beast
Encodings: [a], [b], [c], … so that we can
publicly Add ( [a], [b] ) -> [ a + b ]
publicly Mult ( [a], [b] ) -> [ ab ]
publicly Test ( [a] ) -> a = 0 or not

Ideal security: [a], [b], [c], … hide the plaintexts a, b, c, …
Multilinear maps

A new beast

Encodings: \([a], [b], [c], \ldots\) so that we can
publicly Add \(( [a], [b] ) \rightarrow [ a + b ]\)
publicly Mult \(( [a], [b] ) \rightarrow [ ab ]\)
publicly Test \(( [a] ) \rightarrow a = 0 \) or not

Ideal security: \([a], [b], [c], \ldots\) hide the plaintexts \(a, b, c, \ldots\)

**What we know:** Bilinear maps from elliptic curves \([ \text{Miller 1986} ]\)
**Motivation** of \(n\)-linear maps \([ \text{Boneh, Silverberg 2003} ]\)
Multilinear maps

A new beast

Encodings: [a], [b], [c], … so that we can
publicly Add ( [a], [b] ) -> [ a + b ]
publicly Mult ( [a], [b] ) -> [ ab ]
publicly Test ( [a] ) -> a = 0 or not

Ideal security: [a], [b], [c], … hide the plaintexts a, b, c, …

What we know: Bilinear maps from elliptic curves [ Miller 1986 ]
Motivation of n-linear maps [ Boneh, Silverberg 2003 ]

Candidates: Garg, Gentry, Halevi 2013 [ GGH 13 ]
Coron, Lepoint, Tibouchi 2013 [ CLT 13 ]
Gentry, Gorbunov, Halevi 2015 [ GGH 15 ]

All based on nonstandard use of lattices
Obfuscation:
- A compiler \( P \rightarrow P^* \)
- \( P^* = P \) in functionality
- \( P^* \) is “unintelligible”, “hides information” in \( P \)

\[
P = \text{factorize()}
\]

\[
P^* = \begin{align*}
\text{XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC(e,z,0,0),BlackPixel(e,0));} \\
\text{scanf(“%lf%lf%lf”,y +n,w+y, y+s)+1; y ++);} \\
\text{XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400, 0,0,WhitePixel(e,0) )},KeyPressMask); for(XMapWindow(e,z); ; T=sin(O))}{ \textbf{struct} \text{ timeval}}
\end{align*}
\]
Obfuscation:
- A compiler \( P \rightarrow P^* \)
- \( P^* = P \) in functionality
- \( P^* \) is “unintelligible”, “hides information” in \( P \)

Indistinguishability Obfuscation (iO):
- Defined by [Barak et al. 2001]
- Known to be the best-possible [Goldwasser, Rothblum 2007]
- First candidate (for general purpose obfuscation) based on multilinear maps [Garg et al. 2013]
Obfuscation:
- A compiler $P \rightarrow P^*$
- $P^*$ = $P$ in functionality
- $P^*$ is “unintelligible”, “hides information” in $P$

Indistinguishability Obfuscation (iO):
- Defined by [Barak et al. 2001]
- Known to be the best-possible [Goldwasser, Rothblum 2007]
- First candidate (for general purpose obfuscation) based on multilinear maps [Garg et al. 2013]

Oversimplified idea:
- Decompose $P$ into $a$, $b$, $c$, ... (e.g. $P(x) = (ax + b)c$)
- Use **Multilinear maps** to encode $[a]$, $[b]$, $[c]$.
- Plus other mechanisms to prevent illegal evaluations.
Obfuscation

Multilinear maps

From nonstandard use of lattices
2013 - 2016

Start the age of discovery in Cryptoland

Multilinear maps

Obfuscation
2013 - 2016

- Functional encryption
- Deniable encryption
- Obfuscation
- Multilinear maps
- Witness encryption
- Time-lock puzzle
- Instantiating random oracles
- Delegate RAM computation with privacy
- Watermarking
- Hardness of finding Nash Equilibrium

Age of discovery
On the correlation intractability of obfuscated pseudorandom functions
Ran Canetti, Yilei Chen, Leonid Reyzin
Theory of Cryptography Conference 2016-A

Instantiating random oracles

Obfuscation

Adaptive succinct garbled RAM, or How to delegate your database
Ran Canetti, Yilei Chen, Justin Holmgren, Mariana Raykova
Theory of Cryptography Conference 2016-B

Delegate RAM computation with privacy

Age of discovery
Source of inspirations of cryptography

Multilinear maps

Witness encryption

Obfuscation

Time-lock puzzle

Delegate RAM computation with privacy

Hardness of finding Nash Equilibrium

Age of discovery
Functional encryption
[ Waters 14 ]

Obfuscation

Memory from the glorious old days
How about the security of mmaps / iO candidates themselves?

Multilinear maps

Obfuscation
Candidate program obfuscators:

Since [Garg, Gentry, Halevi, Raykova, Sahai, Waters ‘13], around 20 variants [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], [Lin, Vaikuntanathan ‘16], [Garg, Miles, Mukherjee, Sahai, Srinivasan, Zhandry ‘16] ... 

So far, all based on n-linear maps (n>2).

Candidate multilinear maps for n>2: GGH13, CLT13, GGH15

All make non-standard uses of lattices.
You never know ...

GGH13, CLT13, GGH15

[Marg, 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], [Lin, Vaikuntanathan ‘16], [Garg, Miles, Mukherjee, Sahai, Srinivasan, Zhandry ‘16] ...
<table>
<thead>
<tr>
<th>Type of program (branching programs)</th>
<th>Simple (read-once BP)</th>
<th>Complex (read-many BP)</th>
<th>Very Complex (dual-input BP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGH13</td>
<td>Standing</td>
<td>Standing</td>
<td>Standing</td>
</tr>
<tr>
<td>CLT13</td>
<td>[ Cheon et al. 15 ]</td>
<td>Standing</td>
<td>Standing</td>
</tr>
<tr>
<td>[ Coron et al. 15 ]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GGH15</td>
<td>Standing</td>
<td>Standing</td>
<td>Standing</td>
</tr>
</tbody>
</table>
### Status of indistinguishability obfuscators
under the framework of [Garg et al. 2013]

<table>
<thead>
<tr>
<th>Type of program (branching programs)</th>
<th>Simple (read-once BP)</th>
<th>Complex (read-many BP)</th>
<th>Very Complex (dual-input BP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGH13</td>
<td>[ CGH 17 ]</td>
<td>Standing</td>
<td>Standing</td>
</tr>
<tr>
<td>CLT13</td>
<td>[ Cheon et al. 15 ]</td>
<td>[ Coron et al. 17 ]</td>
<td>Standing</td>
</tr>
<tr>
<td>GGH15</td>
<td>[ CGH 17 ]*</td>
<td>Standing</td>
<td>Standing</td>
</tr>
</tbody>
</table>

Cryptanalysis of candidate branching program obfuscators
Yilei Chen, Craig Gentry, Shai Halevi
Eurocrypt 2017
### Status of indistinguishability obfuscators under the framework of [Garg et al. 2013]

<table>
<thead>
<tr>
<th>Type of program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple (read-once BP)</td>
</tr>
<tr>
<td>GGH13</td>
</tr>
<tr>
<td>CLT13</td>
</tr>
<tr>
<td>GGH15</td>
</tr>
</tbody>
</table>

**Cryptanalysis of candidate branching program obfuscators**

Yilei Chen, Craig Gentry, Shai Halevi

Eurocrypt 2017

**GGH15 beyond permutation branching programs**

Yilei Chen, Vinod Vaikuntanathan, Hoeteck Wee

In submission 2018
Functional encryption

Witness encryption

Multilinear maps

Obfuscation

Time-lock puzzle

Hardness of finding Nash Equilibrium

Delegation

Instantiating random oracles

Watermarking

Deniable encryption

[CGH 17]
Functional encryption
Deniable encryption
Witness encryption
Instantiating random oracles
Multilinear maps
Obfuscation
Time-lock puzzle
Delegation
Hardness of finding Nash Equilibrium

[CVW 18]
Where do we stand today?
Obfuscation

Multilinear maps

Obfuscation
Construct (or fix the existing) mmaps and iO: open problems

Award prices see
https://simons.berkeley.edu/crypto2015/open-problems
The plan for the rest of the talk:

> Hiding secrets in public random functions
   Based on short vector problems on lattices
Construct **private constrained PRFs** from lattices

Constraint-hiding PRFs for NC1 circuits from LWE
Ran Canetti, Yilei Chen
Eurocrypt 2017

GGH15 beyond permutation branching programs
Yilei Chen, Vinod Vaikuntanathan, Hoeteck Wee
In submission 2018
Private Constrained PRF in 3 mins
PRF = Pseudorandom function [ Goldwasser, Goldreich, Micali 84 ]
Private Constrained PRF in a nutshell

original key  

constrained key (a modified key)

Constrained PRF
Private Constrained PRF in a nutshell

original key

constrained key (a modified key)

Constraint-hiding: hide where it is modified
Constrained PRF (not hiding)
Puncturable PRF from [GGM 84]
Constrained PRF (not hiding)
Puncturable PRF from [GGM 84]
Constrained PRF (not hiding)
Puncturable PRF from [GGM 84]

The punctured key reveals the point $x^*$
Constrained PRF
Private Constrained PRF
Private Constrained PRF [ Boneh, Lewi, Wu 17 ]
Private Constrained PRF [ Boneh, Lewi, Wu 17 ]

? = embed a secret message
=> watermarking [BLW17]

\[
\text{CK}_{\{x^*\}} = \begin{cases} 
  \text{“Alice”} , & \text{if } x = x^* \\
  F(x) , & \text{else}
\end{cases}
\]

Watermarked key
Private Constrained PRF [Boneh, Lewi, Wu 17]

\( ? = \) embed a secret message
\( \Rightarrow \) watermarking [BLW17]

\( ? = \) modify the key according to a function \( F \)
\( \Rightarrow \) functional encryption [CC17]

\[ \text{CK_{\{F,sk\}} = F[ Dec_{sk}(x) ]} \]
Private Constrained PRF [ Boneh, Lewi, Wu 17 ]

? = embed a secret message
=> watermarking [BLW17]

? = modify the key according to a function F
=> functional encryption [CC17]

? = if you can achieve 2-key security
=> Obfuscation [CC17]
Private Constrained PRF [Boneh, Lewi, Wu 17]

? = embed a secret message
   => watermarking [BLW17]

? = modify the key according to a function F
   => functional encryption [CC17]

? = if you can achieve 2-key security
   => Obfuscation [CC17]

? =
Watermarking

Functional encryption

Indistinguishability obfuscation?
(if PCPRF is two-key secure)

How to build PCPRF?
Main construction [CC17]

Private constrained PRF for NC1 with 1-key security from Learning With Errors.
Learning with errors [ Regev 05 ]

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

A small amount of noise

secret vector

public matrix (coefficients)

LWE: given A, Y, find s.
Learning with errors [ Regev 05 ]

A small amount of noise

secret vector

public matrix (coefficients)

LWE: given A, Y, find s.

Proved to be as hard as short-vector problems in lattices [ Regev 05 ]. Conjectured to be hard even for quantum computers.
Starting point: a plain PRF by [Banerjee, Peikert, Rosen 12]

Key:

\[
\begin{array}{ccc}
S_{1,1} & S_{2,1} & \ldots & S_{n,1} \\
S_{1,0} & S_{2,0} & \ldots & S_{n,0}
\end{array}
\]

Eval:

\[F(x) = \{ \prod_{i} S_{i,xi} A \}_2\]
Starting point: a plain PRF by [Banerjee, Peikert, Rosen 12]

Key:

\[
\begin{array}{ccc}
  s_{1,1} & s_{2,1} & \ldots & s_{n,1} \\
  s_{1,0} & s_{2,0} & \ldots & s_{n,0}
\end{array}
\]

mod q

Eval:

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

What we need in addition to build a PCPRF:

+ Embed structures in the secret terms to perform functionality
+ A proper public mode of the function
Step 1: Embed **structures**: permutation branching programs
Step 1: Embed structures: permutation branching programs

Key:

\[
\begin{array}{c}
S_{1,1} & S_{2,1} & \cdots & S_{n,1} \\
S_{1,0} & S_{2,0} & \cdots & S_{n,0}
\end{array}
\]

\(A \mod q\)
Step 2: **Encode** the structured key?
Step 2: *Encode* the structured key?

Key:

\[
\begin{align*}
S_{1,1} & \quad S_{2,1} & \cdots & \quad S_{n,1} \\
S_{1,0} & \quad S_{2,0} & \cdots & \quad S_{n,0}
\end{align*}
\]

Constrained Key:

\[
\begin{align*}
D_{1,1} & \quad D_{2,1} & \cdots & \quad D_{n,1} \\
D_{1,0} & \quad D_{2,0} & \cdots & \quad D_{n,0}
\end{align*}
\]

\[A \mod q\]
Short explanation of the technical challenges
Short explanation of the technical challenges

- GGH15 encoding uses LWE with lattice trapdoors, trapdoor sampling of arbitrary plaintext can be dangerous (as revealed by cryptanalytic attacks)
Short explanation of the technical challenges

- GGH15 encoding uses LWE with lattice trapdoors, trapdoor sampling of arbitrary plaintext can be dangerous (as revealed by cryptanalytic attacks)
- [CC 17] discovers a “safe mode” related to permutation matrices.
Short explanation of the technical challenges

- GGH15 encoding uses LWE with lattice trapdoors, trapdoor sampling of arbitrary plaintext can be dangerous (as revealed by cryptanalytic attacks)

- [CC 17] discovers a “safe mode” related to permutation matrices.

- [CVW 18] shows “more general safe modes”. As a result, it improves the efficiency of e.g. private puncturable PRFs.
Demo: 2-bit PRF that is punctured on $x^* = 11$:

$[\text{CVW 18}]$ uses diagonal matrices (low-rank).

$[\text{CC 17}]$ uses permutation matrices.
Performance?

Implementations of GGH15-based schemes:
1. BPobfus
2. PALISADE

Current status: (for 80-bit security)

<table>
<thead>
<tr>
<th>Multilinearity</th>
<th>generate the encodings</th>
<th>storage</th>
<th>time per evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1 min</td>
<td>1 GB</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>8</td>
<td>8 mins</td>
<td>8 GB</td>
<td>&lt; 4 sec</td>
</tr>
<tr>
<td>16</td>
<td>120 mins</td>
<td>300 GB</td>
<td>&lt; 100 sec</td>
</tr>
</tbody>
</table>
Performance?

Implementations of GGH15-based schemes:
1. BPobfus
2. PALISADE

Current status: (for 80-bit security)

<table>
<thead>
<tr>
<th>Multilinearity</th>
<th>generate the encodings</th>
<th>storage</th>
<th>time per evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1 min</td>
<td>1 GB</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>8</td>
<td>8 mins</td>
<td>8 GB</td>
<td>&lt; 4 sec</td>
</tr>
<tr>
<td>16</td>
<td>120 mins</td>
<td>300 GB</td>
<td>&lt; 100 sec</td>
</tr>
</tbody>
</table>

Estimation of 16 bit private puncturable PRF using [CVW 18]:
use multilinearity = 4, wordsize of $2^4 = 16$,

16 mins, 16 GB storage, 1 sec per eval.
Watermarking

Functional encryption

Indistinguishability obfuscation?
(if PCPRF is two-key secure)

Lattices

What’s more?
Watermarking

Functional encryption

Indistinguishability obfuscation?
(if PCPRF is two-key secure)

Correlation Intractable functions

Lattices
Watermarking

Functional encryption

Indistinguishability obfuscation?
(if PCPRF is two-key secure)

Lattices
Constructing cryptographic hash functions:

- On the correlation intractability of obfuscated pseudorandom functions
  Ran Canetti, Yilei Chen, Leonid Reyzin
  Theory of Cryptography Conference 2016-A

- Fiat-Shamir from strong KDM encryption schemes
  Ran Canetti, Yilei Chen, Leonid Reyzin, Ron Rothblum
  Eurocrypt 2018
A **public** function

\[ h: \{0,1\}^l \rightarrow \{0,1\}^m \]

behaves like a random function?
One of the desirable property of a public random function is

**Correlation Intractability**

“infeasibility of finding ‘rare’ input-output relations”
Sparse Relations

“For each input \((x)\),
the fraction of outputs \((y)\) in the relation is \textit{negligible}”
Sparse Relations

“For each input \( x \),
the fraction of outputs \( y \) in the relation is negligible”

**Correlation intractability** [Canetti, Goldreich, Halevi ‘98]

For all sparse relations \( R \):

Adversary \rightarrow \text{Challenger}

\( x, \text{ (as a result, } y=h(x)) \)

Adversary wins if \( R(x, y)=1 \)
\[ H(???...?) = 000000....XYZ3d83h \]
Correlation intractability for moderately sparse relations [ CCRR 18 ]
What we know about CI:

[ Canetti, Goldreich, Halevi 98 ] some parameters are impossible.

[ Goldwasser, Kalai 03 ] more parameters are impossible.

[ Bitanski et al. 13 ] Impossible to prove based on black-box reduction to efficiently falsifiable assumptions. (difficult to prove)

Before 2015: sometime cited as unconditionally impossible
What we know about CI:

...

SHA256 (and others) are heuristical candidates.

We want to base CI on clear mathematical problems.

Intuitive difficulty: how to handle “weird relations” like

$$R(x, y) = 1 \text{ iff the “2nd ~ 2n-th bits of } (3^x + x^2) = y$$
Bounded correlation intractability from iO( Puncturable.PRF )
Canetti, Chen, Reyzin (TCC 2016-A)

**Bounded correlation intractability from iO( Puncturable.PRF )**

Canetti, Chen, Reyzin, Rothblum (Eurocrypt 2018)

**CI for every sparse relation from exponentially KDM secure encryption schemes.**

(with candidates from LWE and discrete-log like problems + KDM assumption)
Main idea in the analysis:
1. Starting from a relation $R$ and a random function $F$
2. Moving to $F_R$ that is indistinguishable from $F$
3. Finally, argue the correlation intractability of $F_R$
Main idea in the analysis:
1. Starting from a relation $R$ and a random function $F$
2. Moving to $F_R$ that is indistinguishable from $F$
3. Finally, argue the correlation intractability of $F_R$

Reasoning about moderate hardness is still an open problem.
3 more mins … Acknowledgements
Ran watching thrillers in a restaurant (Jan 2015)
Rare footage of Leo running the main office (Mar 2015)
Vinod looking for the most beautiful short non-zero vector in the desert. (Jan 2016)
Boston University
Security Group
Coauthors:

Craig Gentry,
Shai Halevi,
Justin Holmgren,
Mariana Raykova,
Ron Rothblum,
Hoeteck Wee.
Thanks for your time!