Quantum attack against some candidate obfuscators based on GGH13

Alice Pellet-Mary

LIP, ENS de Lyon

Séminaire C2
November 16, 2018

ENS DE LYON

European Research Council
Established by the European Commission

What is this talk about

Quantum attack against some candidate obfuscators built upon the GGH13 multilinear map [GGH13a]

[GGH13a] S. Garg, C. Gentry and S. Halevi. Candidate multilinear maps from ideal lattices, Eurocrypt.

What is this talk about

Quantum attack against some candidate obfuscators built upon the GGH13 multilinear map [GGH13a]

- GGH13 is known to be weak in quantum world

[GGH13a] S. Garg, C. Gentry and S. Halevi. Candidate multilinear maps from ideal lattices, Eurocrypt.

What is this talk about

Quantum attack against some candidate obfuscators built upon the GGH13 multilinear map [GGH13a]

- GGH13 is known to be weak in quantum world
- Transform this weakness into concrete attack on obfuscators

---

[GGH13a] S. Garg, C. Gentry and S. Halevi. Candidate multilinear maps from ideal lattices, Eurocrypt.

What is this talk about

Quantum attack against some candidate obfuscators built upon the GGH13 multilinear map [GGH13a]

- GGH13 is known to be weak in quantum world
- Transform this weakness into concrete attack on obfuscators
- Nothing quantum in this talk

[GGH13a] S. Garg, C. Gentry and S. Halevi. Candidate multilinear maps from ideal lattices, Eurocrypt.

Obfuscation

Obfuscator

An obfuscator O for a class of circuits $\mathcal{C}$ is an efficiently computable function over $\mathcal{C}$ such that

$$\forall C \in \mathcal{C}, \forall x, C(x) = O(C)(x)$$

In this talk, $\mathcal{C}$ = polynomial size circuits

Obfuscation

Obfuscator

An obfuscator O for a class of circuits $\mathcal{C}$ is an efficiently computable function over $\mathcal{C}$ such that

$$\forall C \in \mathcal{C}, \forall x, C(x) = O(C)(x)$$

In this talk, $\mathcal{C}$ = polynomial size circuits

Security.

- VBB: $O(C)$ acts as a black box computing C

Obfuscation

Obfuscator

An obfuscator O for a class of circuits $\mathcal{C}$ is an efficiently computable function over $\mathcal{C}$ such that

$$\forall C \in \mathcal{C}, \forall x, C(x) = O(C)(x)$$

In this talk, $\mathcal{C}$ = polynomial size circuits

Security.

- ~~VBB: $O(C)$ acts as a black box computing C~~ (impossible, [BGI$^+$01])

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang. On the (im) possibility of obfuscating programs, Crypto.

Obfuscation

Obfuscator

An obfuscator O for a class of circuits $\mathcal{C}$ is an efficiently computable function over $\mathcal{C}$ such that

$$\forall C \in \mathcal{C}, \forall x, C(x) = O(C)(x)$$

In this talk, $\mathcal{C}$ = polynomial size circuits

Security.

- ~~VBB: $O(C)$ acts as a black box computing C~~ (impossible, [BGI$^+$01])
- iO: $\forall C_1 \equiv C_2$, i.e. $C_1(x) = C_2(x)\ \forall x$,

$$O(C_1) \simeq_c O(C_2)$$

[BGI+01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang. On the (im) possibility of obfuscating programs, Crypto.

Why is iO interesting?

1 iO achieves "best possible" obfuscation

Why is iO interesting?

1 iO achieves "best possible" obfuscation

Proof:

- let O be an iO obfuscator and O' be another obfuscator

Why is iO interesting?

1 iO achieves "best possible" obfuscation

Proof:

- let O be an iO obfuscator and O' be another obfuscator
- for any $C \in \mathcal{C}$, $O(C) \simeq_c O(O'(C))$

Why is iO interesting?

1 iO achieves "best possible" obfuscation

Proof:

- let O be an iO obfuscator and O' be another obfuscator
- for any $C \in \mathcal{C}$, $O(C) \simeq_c O(O'(C))$
- $O(O'(C))$ reveals less info than $O'(C)$

Why is iO interesting?

1 iO achieves "best possible" obfuscation

Proof:

- let O be an iO obfuscator and O' be another obfuscator
- for any $C \in \mathcal{C}$, $O(C) \simeq_c O(O'(C))$
- $O(O'(C))$ reveals less info than $O'(C)$
- $O(C)$ reveals less info than $O'(C)$

Why is iO interesting?

1 iO achieves "best possible" obfuscation

Proof:

- let O be an iO obfuscator and O' be another obfuscator
- for any $C \in \mathcal{C}$, $O(C) \simeq_c O(O'(C))$
- $O(O'(C))$ reveals less info than $O'(C)$
- $O(C)$ reveals less info than $O'(C)$

2 Many cryptographic constructions from iO: functional encryption, deniable encryption, NIKZs, oblivious transfer, ...

Multilinear maps (mmaps) and iO

Observation

Almost all iO constructions for all circuits rely on multilinear maps (mmaps)

Three main candidate multilinear maps: GGH13, CLT13, GGH15

Multilinear maps (mmaps) and iO

Observation

Almost all iO constructions for all circuits rely on multilinear maps (mmaps)

Three main candidate multilinear maps: GGH13, CLT13, GGH15

Caution

All these candidate multilinear maps suffer from weaknesses
(e.g. encodings of zero, zeroizing attacks,. . .).
$\Rightarrow$ all current attacks against iO rely on the underlying mmap

Multilinear maps (mmaps) and iO

Observation

Almost all iO constructions for all circuits rely on multilinear maps (mmaps)

Three main candidate multilinear maps: **GGH13**, CLT13, GGH15

Caution

All these candidate multilinear maps suffer from weaknesses
(e.g. encodings of zero, zeroizing attacks,...).
$\Rightarrow$ all current attacks against iO rely on the underlying mmap

In this talk: we exploit known weakness of GGH13 to mount concrete attacks against some iO using it.

History (branching program obfuscators based on GGH13)

Some candidate iO for all circuits and attacks:

History (branching program obfuscators based on GGH13)

Some candidate iO for all circuits and attacks:

2013: [GGH+13b], first candidate

2014-2016: [AGIS14, BGK+14, BR14, MSW14, PST14, BMSZ16], with proofs in idealized models (the mmap is supposed to be somehow ideal)

History (branching program obfuscators based on GGH13)

Some candidate iO for all circuits and attacks:

2013: [GGH+13b], first candidate

2014-2016: [AGIS14, BGK+14, BR14, MSW14, PST14, BMSZ16], with proofs in idealized models (the mmap is supposed to be somehow ideal)

2016: [MSZ16], attack against all candidates above except [GGH+13b]

2016: [GMM+16], proof in a weaker idealized model (captures [MSZ16])

History (branching program obfuscators based on GGH13)

Some candidate iO for all circuits and attacks:

2013: [GGH+13b], first candidate

2014-2016: [AGIS14, BGK+14, BR14, MSW14, PST14, BMSZ16], with proofs in idealized models (the mmap is supposed to be somehow ideal)

2016: [MSZ16], attack against all candidates above except [GGH+13b]

2016: [GMM+16], proof in a weaker idealized model (captures [MSZ16])

2017: [CGH17], attack against [GGH+13b], in input-partitionable case

2017: [FRS17], prevent [CGH17] attack

History (branching program obfuscators based on GGH13)

Some candidate iO for all circuits and attacks:

2013: [GGH+13b], first candidate

2014-2016: [AGIS14, BGK+14, BR14, MSW14, PST14, BMSZ16], with proofs in idealized models (the mmap is supposed to be somehow ideal)

2016: [MSZ16], attack against all candidates above except [GGH+13b]

2016: [GMM+16], proof in a weaker idealized model (captures [MSZ16])

2017: [CGH17], attack against [GGH+13b], in input-partitionable case

2017: [FRS17], prevent [CGH17] attack

2018: [CHKL18], attack against all obfuscators, for specific choices of parameters

State of the art and contribution

iO (using GGH13) / Attacks	Branching program obfuscators				Circuit obfuscators
	[GGH+13b]	[BR14]	[AGIS14, MSW14] [PST14, BGK+14] [BMSZ16]	[GMM+16]	[Zim15, AB15] [DGG+16]
[MSZ16]		✓	✓		
[CGH17]*	✓				
[CHKL18]†	✓	✓	✓	✓	
This talk‡			✓	✓	✓

* for input-partitionable branching programs
† for specific choices of parameters
‡ in the quantum setting

State of the art and contribution

Attacks \ iO (using GGH13)	Branching program obfuscators				Circuit obfuscators
	[GGH$^+$13b]	[BR14]	[AGIS14, MSW14] [PST14, BGK$^+$14] [BMSZ16]	[GMM$^+$16]	[Zim15, AB15] [DGG$^+$16]
[MSZ16]		✓	✓		
[CGH17]⋆	✓				
[CHKL18]†	✓	✓	✓	✓	
This talk‡			✓	✓	✓

⋆ for input-partitionable branching programs

† for specific choices of parameters

‡ in the quantum setting

Outline of the talk

1. Simple obfuscator
2. The attack

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function inp : $\{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

A_0 $\quad$ $A_{1,1}$, $A_{1,0}$ $\quad$ $A_{2,1}$, $A_{2,0}$ $\quad$ $A_{3,1}$, $A_{3,0}$ $\quad$ $A_{4,1}$, $A_{4,0}$ $\quad$ $A_{5,1}$, $A_{5,0}$ $\quad$ $A_{6,1}$, $A_{6,0}$ $\quad$ A_7

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function inp : $\{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$$A_0 \quad \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \quad \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \quad \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \quad \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \quad \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \quad \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function inp : $\{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$\uparrow$

$$A_0 \times \begin{matrix} A_{1,1} & A_{2,1} & A_{3,1} & A_{4,1} & A_{5,1} & A_{6,1} \\ A_{1,0} & A_{2,0} & A_{3,0} & A_{4,0} & A_{5,0} & A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function inp : $\{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$\uparrow$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \quad \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \quad \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \quad \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \quad \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function $\text{inp} : \{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$\uparrow$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \times \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \quad \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \quad \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \quad \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \dots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function $\text{inp} : \{1, \dots, \ell\} \to \{1, \dots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
$\text{inp}(i)$	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$\uparrow$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \times \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \times \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \quad \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \quad \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function $\text{inp} : \{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$
$\uparrow$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \times \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \times \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \times \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \quad \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function inp : $\{1, \ldots, \ell\} \to \{1, \ldots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$\uparrow$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \times \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \times \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \times \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \times \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \quad A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \dots, \ell\}$ and $b \in \{0,1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function $\text{inp} : \{1, \dots, \ell\} \to \{1, \dots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
$\text{inp}(i)$	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \times \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \times \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \times \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \times \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \times A_7$$

Branching programs

A branching program is a way of representing a function (like a Turing machine, or a circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $A_{i,b}$ (for $i \in \{1, \dots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors A_0 and $A_{\ell+1}$,
- a function inp : $\{1, \dots, \ell\} \to \{1, \dots, r\}$ (where r is the size of the input).

i	1	2	3	4	5	6
inp(i)	1	1	2	1	3	2

$x = 0 \quad 1 \quad 1$

$$A_0 \times \begin{matrix} A_{1,1} \\ A_{1,0} \end{matrix} \times \begin{matrix} A_{2,1} \\ A_{2,0} \end{matrix} \times \begin{matrix} A_{3,1} \\ A_{3,0} \end{matrix} \times \begin{matrix} A_{4,1} \\ A_{4,0} \end{matrix} \times \begin{matrix} A_{5,1} \\ A_{5,0} \end{matrix} \times \begin{matrix} A_{6,1} \\ A_{6,0} \end{matrix} \times A_7 \begin{matrix} = 0 \to 0 \\ \neq 0 \to 1 \end{matrix}$$

Cryptographic multilinear maps

Definition: κ-multilinear map

Different levels of encodings, from 1 to κ.
Denote by $\mathrm{Enc}(a, i)$ a level-i encoding of the message a.

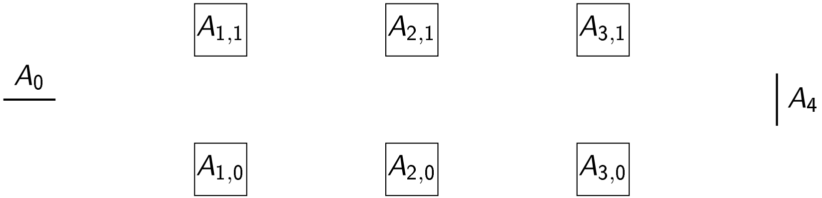
Addition: $\mathrm{Add}(\mathrm{Enc}(a_1, i), \mathrm{Enc}(a_2, i)) = \mathrm{Enc}(a_1 + a_2, i)$.

Multiplication: $\mathrm{Mult}(\mathrm{Enc}(a_1, i), \mathrm{Enc}(a_2, j)) = \mathrm{Enc}(a_1 \cdot a_2, i + j)$.

Zero-test: $\text{Zero-test}(\mathrm{Enc}(a, \kappa)) = \text{True iff } a = 0$.

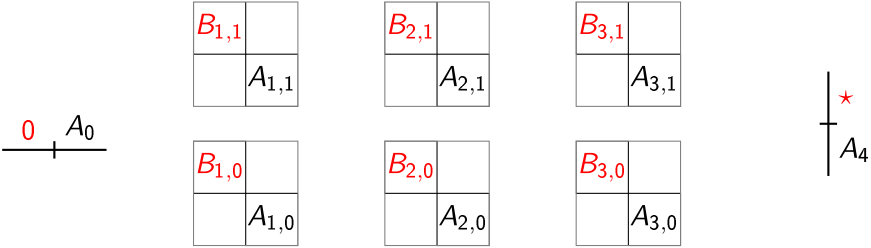
Simple obfuscator

- **Input:** A branching program
- Randomize the branching program
  - Add random diagonal blocks
  - Killian's randomization
  - Multiply by random (non zero) bundling scalars
- Encode the matrices using GGH13
- **Output:** The encoded matrices and vectors



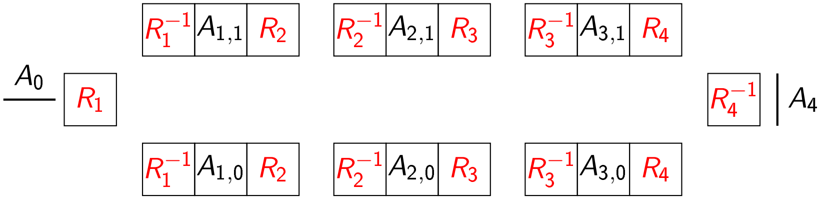
Simple obfuscator

- **Input:** A branching program
- Randomize the branching program
  - Add random diagonal blocks
  - Killian's randomization
  - Multiply by random (non zero) bundling scalars
- Encode the matrices using GGH13
- **Output:** The encoded matrices and vectors



Simple obfuscator

- **Input:** A branching program
- Randomize the branching program
  - Add random diagonal blocks
  - Killian's randomization
  - Multiply by random (non zero) bundling scalars
- Encode the matrices using GGH13
- **Output:** The encoded matrices and vectors



Simple obfuscator

- **Input:** A branching program
- Randomize the branching program
  - Add random diagonal blocks
  - Killian's randomization
  - Multiply by random (non zero) bundling scalars
- Encode the matrices using GGH13
- **Output:** The encoded matrices and vectors

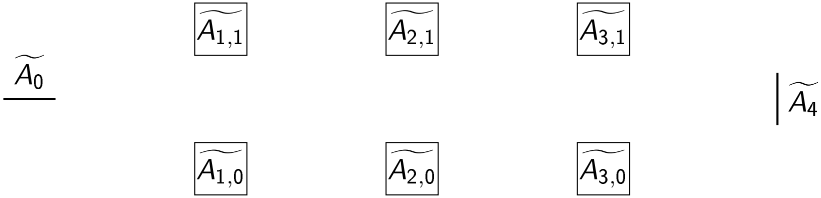
$$\alpha_{1,1} \times \boxed{A_{1,1}} \quad \alpha_{2,1} \times \boxed{A_{2,1}} \quad \alpha_{3,1} \times \boxed{A_{3,1}}$$

$$\underline{A_0} \qquad\qquad\qquad\qquad\qquad\qquad \Big| A_4$$

$$\alpha_{1,0} \times \boxed{A_{1,0}} \quad \alpha_{2,0} \times \boxed{A_{2,0}} \quad \alpha_{3,0} \times \boxed{A_{3,0}}$$

Simple obfuscator

- **Input:** A branching program
- Randomize the branching program
  - Add random diagonal blocks
  - Killian's randomization
  - Multiply by random (non zero) bundling scalars
- Encode the matrices using GGH13
- **Output:** The encoded matrices and vectors



Simple obfuscator

- **Input:** A branching program
- Randomize the branching program
  - Add random diagonal blocks
  - Killian's randomization
  - Multiply by random (non zero) bundling scalars
- Encode the matrices using GGH13
- **Output:** The encoded matrices and vectors

$\mathrm{Enc}(\widetilde{A_{1,1}})$ $\mathrm{Enc}(\widetilde{A_{2,1}})$ $\mathrm{Enc}(\widetilde{A_{3,1}})$

$\mathrm{Enc}(\widetilde{A_0})$ $\mathrm{Enc}(\widetilde{A_4})$

$\mathrm{Enc}(\widetilde{A_{1,0}})$ $\mathrm{Enc}(\widetilde{A_{2,0}})$ $\mathrm{Enc}(\widetilde{A_{3,0}})$

Outline of the talk

GGH13 in a quantum world

Reminder: κ-multilinear map

Different levels of encodings, from 1 to κ.
Denote by $\text{Enc}(a, i)$ a level-i encoding of the message a .

Addition: $\text{Add}(\text{Enc}(a_1, i), \text{Enc}(a_2, i)) = \text{Enc}(a_1 + a_2, i)$.

Multiplication: $\text{Mult}(\text{Enc}(a_1, i), \text{Enc}(a_2, j)) = \text{Enc}(a_1 \cdot a_2, i + j)$.

Zero-test: $\text{Zero-test}(\text{Enc}(a, \kappa)) = \text{True iff } a = 0$.

GGH13 in a quantum world

The GGH13 map

Different levels of encodings, from 1 to κ.
Denote by $\text{Enc}(a, i)$ a level-i encoding of the message $a \in \mathbb{Z}/p\mathbb{Z}$.

Addition: $\text{Add}(\text{Enc}(a_1, i), \text{Enc}(a_2, i)) = \text{Enc}(a_1 + a_2, i)$.

Multiplication: $\text{Mult}(\text{Enc}(a_1, i), \text{Enc}(a_2, j)) = \text{Enc}(a_1 \cdot a_2, i + j)$.

Zero-test: $\text{Zero-test}(\text{Enc}(a, \kappa)) = \text{True iff } a = 0 \mod p$.

GGH13 in a quantum world

The GGH13 map

Different levels of encodings, from 1 to κ.
Denote by $\mathrm{Enc}(a, i)$ a level-i encoding of the message $a \in \mathbb{Z}/p\mathbb{Z}$.

Addition: $\mathrm{Add}(\mathrm{Enc}(a_1, i), \mathrm{Enc}(a_2, i)) = \mathrm{Enc}(a_1 + a_2, i)$.

Multiplication: $\mathrm{Mult}(\mathrm{Enc}(a_1, i), \mathrm{Enc}(a_2, j)) = \mathrm{Enc}(a_1 \cdot a_2, i + j)$.

Zero-test: $\text{Zero-test}(\mathrm{Enc}(a, \kappa)) = \text{True iff } a = 0 \mod p$.

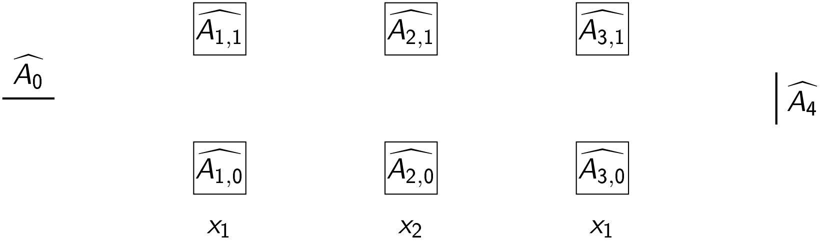
With a quantum computer

$$\text{Double-zero-test}(\mathrm{Enc}(a, 2\kappa)) = \text{True iff } a = 0 \mod p^2$$

Mixed-input attack

Notations

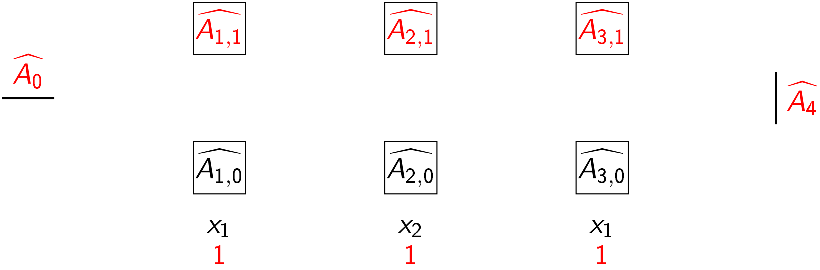
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

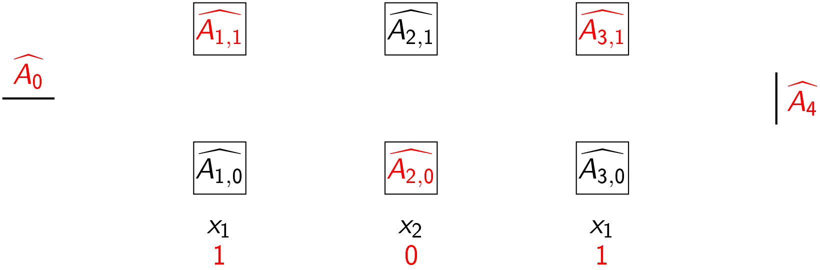
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

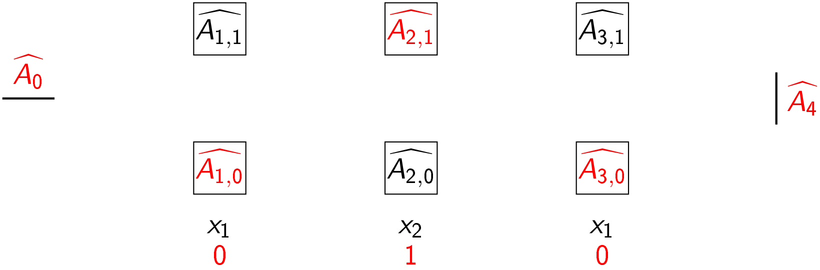
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

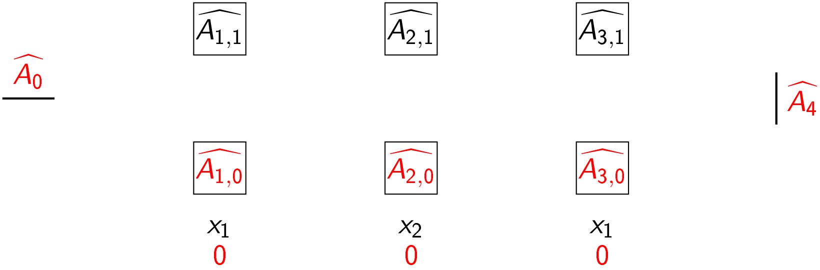
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

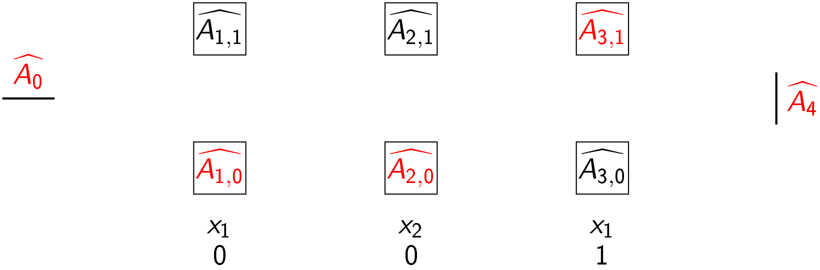
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

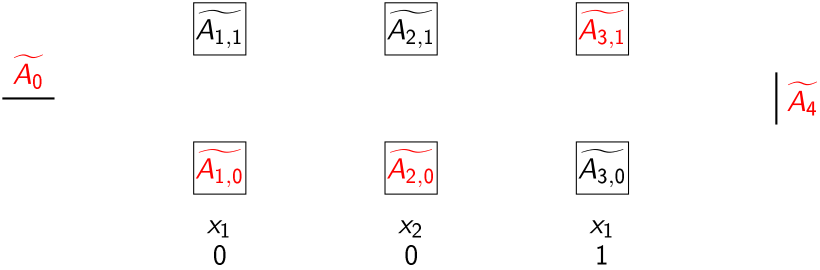
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

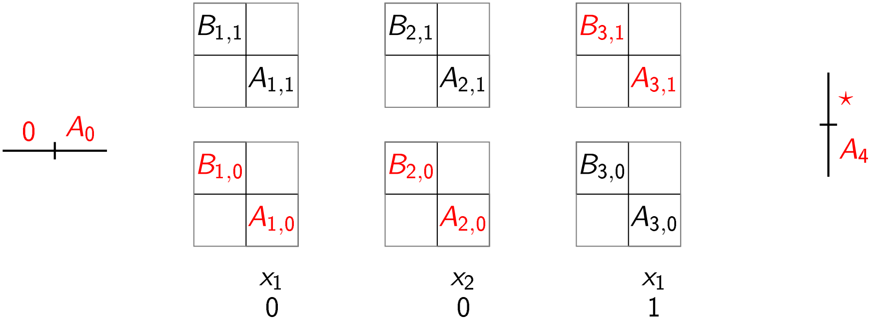
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

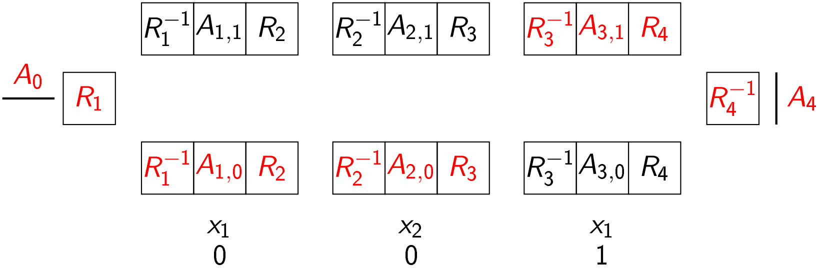
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

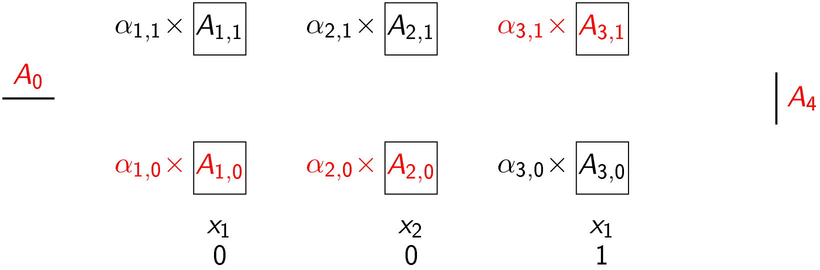
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

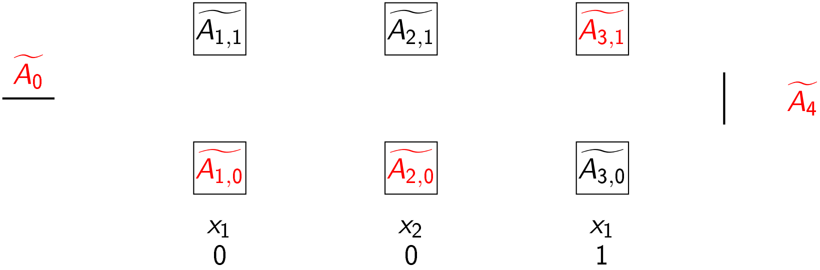
- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\widetilde{A_{i,b}}$ after randomisation
- $\widehat{A_{i,b}}$ after encoding with GGH13 map (output of the iO)

$$
\underline{\mathrm{Enc}(\widetilde{A_0},1)} \quad
\begin{matrix}
\mathrm{Enc}(\boxed{\widetilde{A_{1,1}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{2,1}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,1}}},1) \\
\mathrm{Enc}(\boxed{\widetilde{A_{1,0}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{2,0}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,0}}},1) \\
x_1 & x_2 & x_1 \\
0 & 0 & 1
\end{matrix}
\quad \Big| \, \mathrm{Enc}(\widetilde{A_4},1)
$$

Preventing mixed-input attacks

- In the randomization phase $\Rightarrow$ not in this talk

Preventing mixed-input attacks

- In the randomization phase $\Rightarrow$ not in this talk
- Using the mmap $\Rightarrow$ straddling set system

Preventing mixed-input attacks

- In the randomization phase ⇒ not in this talk
- Using the mmap ⇒ straddling set system

Mmap degree: $\kappa = 5$

$$\mathrm{Enc}(\widetilde{A_0},1) \quad \begin{matrix} \mathrm{Enc}(\boxed{\widetilde{A_{1,1}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{2,1}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,1}}},1) \\ \mathrm{Enc}(\boxed{\widetilde{A_{1,0}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{2,0}}},1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,0}}},1) \\ x_1 & x_2 & x_1 \end{matrix} \quad \Big| \mathrm{Enc}(\widetilde{A_4},1)$$

Preventing mixed-input attacks

- In the randomization phase ⇒ not in this talk
- Using the mmap ⇒ straddling set system

Mmap degree: $\kappa = 6$

$$\underline{\mathrm{Enc}(\widetilde{A_0}, 1)} \quad \begin{matrix} \mathrm{Enc}(\boxed{\widetilde{A_{1,1}}}, 1) & \mathrm{Enc}(\boxed{\widetilde{A_{2,1}}}, 1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,1}}}, 2) \\ \mathrm{Enc}(\boxed{\widetilde{A_{1,0}}}, 2) & \mathrm{Enc}(\boxed{\widetilde{A_{2,0}}}, 1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,0}}}, 1) \\ x_1 & x_2 & x_1 \end{matrix} \quad \Big| \mathrm{Enc}(\widetilde{A_4}, 1)$$

Preventing mixed-input attacks

- In the randomization phase $\Rightarrow$ not in this talk
- Using the mmap $\Rightarrow$ straddling set system

Mmap degree: $\kappa = 6$

$\mathrm{Enc}(\widetilde{A_{1,1}}, 1)$ $\mathrm{Enc}(\widetilde{A_{2,1}}, 1)$ $\mathrm{Enc}(\widetilde{A_{3,1}}, 2)$

$\mathrm{Enc}(\widetilde{A_0}, 1)$ $\mathrm{Enc}(\widetilde{A_4}, 1)$

$\mathrm{Enc}(\widetilde{A_{1,0}}, 2)$ $\mathrm{Enc}(\widetilde{A_{2,0}}, 1)$ $\mathrm{Enc}(\widetilde{A_{3,0}}, 1)$

x_1 x_2 x_1

0 0 1

Preventing mixed-input attacks

- In the randomization phase $\Rightarrow$ not in this talk
- Using the mmap $\Rightarrow$ straddling set system

Mmap degree: $\kappa = 6$

$$\mathrm{Enc}(\widetilde{A_0}, 1) \qquad \begin{matrix} \mathrm{Enc}(\boxed{\widetilde{A_{1,1}}}, 1) & \mathrm{Enc}(\boxed{\widetilde{A_{2,1}}}, 1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,1}}}, 2) \\ \mathrm{Enc}(\boxed{\widetilde{A_{1,0}}}, 2) & \mathrm{Enc}(\boxed{\widetilde{A_{2,0}}}, 1) & \mathrm{Enc}(\boxed{\widetilde{A_{3,0}}}, 1) \\ x_1 & x_2 & x_1 \\ 0 & 0 & 1 \end{matrix} \qquad \mathrm{Enc}(\widetilde{A_4}, 1)$$

Total level: 7 $\Rightarrow$ cannot zero-test

Attack idea: double mixed input

Reminder

In quantum world, we have

$$\text{Double-zero-test}(\text{Enc}(a, 2\kappa)) = \text{True iff } a = 0 \mod p^2$$

Attack idea: double mixed input

Reminder

In quantum world, we have

$$\text{Double-zero-test}(\text{Enc}(a, 2\kappa)) = \text{True iff } a = 0 \mod p^2$$

$\text{Enc}(\widetilde{A_0}, 1)$

	x_1	x_2	x_1
1	$\text{Enc}(\boxed{\widetilde{A_{1,1}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{2,1}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{3,1}}}, 2)$
0	$\text{Enc}(\boxed{\widetilde{A_{1,0}}}, 2)$	$\text{Enc}(\boxed{\widetilde{A_{2,0}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{3,0}}}, 1)$

$\text{Enc}(\widetilde{A_4}, 1)$ $\Rightarrow$ Level 7

Attack idea: double mixed input

Reminder

In quantum world, we have

$$\text{Double-zero-test}(\text{Enc}(a, 2\kappa)) = \text{True iff } a = 0 \mod p^2$$

$\text{Enc}(\widetilde{A_0}, 1)$

	x_1	x_2	x_1	
1	$\text{Enc}(\boxed{\widetilde{A_{1,1}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{2,1}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{3,1}}}, 2)$	
0	$\text{Enc}(\boxed{\widetilde{A_{1,0}}}, 2)$	$\text{Enc}(\boxed{\widetilde{A_{2,0}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{3,0}}}, 1)$	

$\text{Enc}(\widetilde{A_4}, 1) \Rightarrow$ Level 7

$\text{Enc}(\widetilde{A_0}, 1)$

	x_1	x_2	x_1
1	$\text{Enc}(\boxed{\widetilde{A_{1,1}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{2,1}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{3,1}}}, 2)$
0	$\text{Enc}(\boxed{\widetilde{A_{1,0}}}, 2)$	$\text{Enc}(\boxed{\widetilde{A_{2,0}}}, 1)$	$\text{Enc}(\boxed{\widetilde{A_{3,0}}}, 1)$

$\text{Enc}(\widetilde{A_4}, 1) \Rightarrow$ Level 5

Attack idea: double mixed input

Reminder

In quantum world, we have

$$\text{Double-zero-test}(\text{Enc}(a, 2\kappa)) = \text{True iff } a = 0 \mod p^2$$

$\underline{\text{Enc}(\widetilde{A_0}}, 1)$ $\quad$ $\text{Enc}(\boxed{\widetilde{A_{1,1}}}, 1)$ $\text{Enc}(\boxed{\widetilde{A_{2,1}}}, 1)$ $\text{Enc}(\boxed{\widetilde{A_{3,1}}}, 2)$ $\quad$ $|\ \text{Enc}(\widetilde{A_4}, 1)$ $\Rightarrow$ Level 7

$\text{Enc}(\boxed{\widetilde{A_{1,0}}}, 2)$ $\text{Enc}(\boxed{\widetilde{A_{2,0}}}, 1)$ $\text{Enc}(\boxed{\widetilde{A_{3,0}}}, 1)$

x_1 $\quad$ x_2 $\quad$ x_1

$\times$ $\quad$ $\underline{\text{Enc}(\widetilde{A_0}}, 1)$ $\quad$ $\text{Enc}(\boxed{\widetilde{A_{1,1}}}, 1)$ $\text{Enc}(\boxed{\widetilde{A_{2,1}}}, 1)$ $\text{Enc}(\boxed{\widetilde{A_{3,1}}}, 2)$ $\quad$ $|\ \text{Enc}(\widetilde{A_4}, 1)$ $\Rightarrow$ Level 5

$\text{Enc}(\boxed{\widetilde{A_{1,0}}}, 2)$ $\text{Enc}(\boxed{\widetilde{A_{2,0}}}, 1)$ $\text{Enc}(\boxed{\widetilde{A_{3,0}}}, 1)$

x_1 $\quad$ x_2 $\quad$ x_1

Attack idea: double mixed input

Reminder

In quantum world, we have

$$\text{Double-zero-test}(\text{Enc}(a, 2\kappa)) = \text{True iff } a = 0 \mod p^2$$

$\text{Enc}(\widetilde{A_0}, 1)$

$\text{Enc}(\widetilde{A_{1,1}}, 1)$ $\text{Enc}(\widetilde{A_{2,1}}, 1)$ $\text{Enc}(\widetilde{A_{3,1}}, 2)$

$\text{Enc}(\widetilde{A_{1,0}}, 2)$ $\text{Enc}(\widetilde{A_{2,0}}, 1)$ $\text{Enc}(\widetilde{A_{3,0}}, 1)$

x_1 x_2 x_1

$\text{Enc}(\widetilde{A_4}, 1)$ $\Rightarrow$ Level 7

× $\text{Enc}(\widetilde{A_0}, 1)$

$\text{Enc}(\widetilde{A_{1,1}}, 1)$ $\text{Enc}(\widetilde{A_{2,1}}, 1)$ $\text{Enc}(\widetilde{A_{3,1}}, 2)$

$\text{Enc}(\widetilde{A_{1,0}}, 2)$ $\text{Enc}(\widetilde{A_{2,0}}, 1)$ $\text{Enc}(\widetilde{A_{3,0}}, 1)$

x_1 x_2 x_1

$\text{Enc}(\widetilde{A_4}, 1)$ $\Rightarrow$ Level 5

Product level: $12 = 2\kappa$

iO distinguishing attack

Reminder: iO

$$\forall C_1 \equiv C_2,\ O(C_1) \simeq_c O(C_2)$$

iO distinguishing attack

Reminder: iO

$$\forall C_1 \equiv C_2,\ O(C_1) \simeq_c O(C_2)$$

Objective: Find $C_1 \equiv C_2$ s.t. double mixed input product is 0 on C_1 and $\neq 0$ on C_2, e.g.

- the two mixed-input are $0 \mod p$ for C_1
  $\Rightarrow$ product is $0 \mod p^2$
- the two mixed-input are $\neq 0 \mod p$ for C_2
  $\Rightarrow$ product is $\neq 0 \mod p^2$

One example of C_1 and C_2

$$\boldsymbol{C_1}: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_2}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \begin{pmatrix}0\\1\end{pmatrix} \qquad \Rightarrow \forall x,\ C_1(x) = 0$$

One example of C_1 and C_2

$$\boldsymbol{C_1}: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_2}{\begin{matrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\end{matrix}} \quad \begin{pmatrix}0\\ 1\end{pmatrix} \quad \Rightarrow \forall x,\ C_1(x) = 0$$

$$\boldsymbol{C_2}: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \color{red}{\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}}\end{matrix}} \quad \underset{x_2}{\begin{matrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\\ \color{red}{\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}}\end{matrix}} \quad \begin{pmatrix}0\\ 1\end{pmatrix} \quad \Rightarrow \forall x,\ C_2(x) = 0$$

One example of C_1 and C_2

$$C_1: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_2}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \begin{pmatrix}0\\1\end{pmatrix} \quad \Rightarrow \forall x,\ C_1(x) = 0$$

$$C_2: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\end{matrix}} \quad \underset{x_2}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\end{matrix}} \quad \underset{x_1}{\begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\end{matrix}} \quad \begin{pmatrix}0\\1\end{pmatrix} \quad \Rightarrow \forall x,\ C_2(x) = 0$$

- $C_1 \equiv C_2$

One example of C_1 and C_2

$$C_1: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{pmatrix}1&0\\0&1\end{pmatrix} \atop \begin{pmatrix}1&0\\0&1\end{pmatrix}} \quad \underset{x_2}{\begin{pmatrix}1&0\\0&1\end{pmatrix} \atop \begin{pmatrix}1&0\\0&1\end{pmatrix}} \quad \underset{x_1}{\begin{pmatrix}1&0\\0&1\end{pmatrix} \atop \begin{pmatrix}1&0\\0&1\end{pmatrix}} \quad \begin{pmatrix}0\\1\end{pmatrix} \quad \Rightarrow \forall x,\ C_1(x) = 0$$

$$C_2: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \underset{x_1}{\begin{pmatrix}1&0\\0&1\end{pmatrix} \atop \begin{pmatrix}0&1\\1&0\end{pmatrix}} \quad \underset{x_2}{\begin{pmatrix}1&0\\0&1\end{pmatrix} \atop \begin{pmatrix}1&0\\0&1\end{pmatrix}} \quad \underset{x_1}{\begin{pmatrix}1&0\\0&1\end{pmatrix} \atop \begin{pmatrix}0&1\\1&0\end{pmatrix}} \quad \begin{pmatrix}0\\1\end{pmatrix} \quad \Rightarrow \forall x,\ C_2(x) = 0$$

- $C_1 \equiv C_2$
- the two mixed-input products are 0 for C_1

One example of C_1 and C_2

$$C_1: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ x_1 & x_2 & x_1\end{matrix} \quad \begin{pmatrix}0\\1\end{pmatrix} \quad \Rightarrow \forall x,\ C_1(x) = 0$$

$$C_2: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \begin{matrix}\color{blue}{\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}} & \color{blue}{\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}} & \color{red}{\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}}\\ \color{red}{\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}} & \color{red}{\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}} & \color{blue}{\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}}\\ x_1 & x_2 & x_1\end{matrix} \quad \begin{pmatrix}0\\1\end{pmatrix} \quad \Rightarrow \forall x,\ C_2(x) = 0$$

- $C_1 \equiv C_2$
- the two mixed-input products are 0 for C_1
- the two mixed-input products are $\neq 0$ for C_2

One example of C_1 and C_2

$$C_1: \quad \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \underset{x_1}{\begin{matrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix}} \quad \underset{x_2}{\begin{matrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix}} \quad \underset{x_1}{\begin{matrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix}} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \Rightarrow \forall x, \ C_1(x) = 0$$

$$C_2: \quad \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \underset{x_1}{\begin{matrix} \color{blue}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \\ \color{red}{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}} \end{matrix}} \quad \underset{x_2}{\begin{matrix} \color{blue}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \\ \color{red}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \end{matrix}} \quad \underset{x_1}{\begin{matrix} \color{red}{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}} \\ \color{blue}{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}} \end{matrix}} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \Rightarrow \forall x, \ C_2(x) = 0$$

- $C_1 \equiv C_2$
- the two mixed-input products are 0 for C_1
- the two mixed-input products are $\neq 0$ for C_2

> We can distinguish $O(C_1)$ from $O(C_2)$

Conclusion (1/2)

Counter-intuitive remark

This attack works only against the recent schemes
(with stronger security proofs)

Conclusion (1/2)

Counter-intuitive remark

This attack works only against the recent schemes
(with stronger security proofs)

Why?

- Previous schemes prevent mixed-input attack using the randomization phase
  - difficult to get a security proof

Conclusion (1/2)

Counter-intuitive remark

This attack works only against the recent schemes
(with stronger security proofs)

Why?

- Previous schemes prevent mixed-input attack using the randomization phase
  - difficult to get a security proof
- New schemes use the mmap
  - easy to get a proof (in idealized model)

Conclusion (1/2)

Counter-intuitive remark

This attack works only against the recent schemes
(with stronger security proofs)

Why?

- Previous schemes prevent mixed-input attack using the randomization phase
  - difficult to get a security proof
- New schemes use the mmap
  - easy to get a proof (in idealized model)
- GGH13 mmap is not ideal
  - easier for an attacker to exploit its weakness

Conclusion (2/2)

Remarks

- Quantum poly time or classical $2^{O(\sqrt{n})}$ time

Conclusion (2/2)

Remarks

- Quantum poly time or classical $2^{O(\sqrt{n})}$ time
- Double mixed input attacks can be extended to circuit obfuscators

Conclusion (2/2)

iO (using GGH13) \ Attacks	Branching program obfuscators				Circuit obfuscators
	[GGH+13b]	[BR14]	[AGIS14, MSW14] [PST14, BGK+14] [BMSZ16]	[GMM+16]	[Zim15, AB15] [DGG+16]
[MSZ16]		✓	✓		
[CGH17]*	✓				
[CHKL18]†	✓	✓	✓	✓	
This talk‡			✓	✓	✓

* for input-partitionable branching programs

† for specific choices of parameters

‡ in the quantum setting

Conclusion (2/2)

Remarks

- Quantum poly time or classical $2^{O(\sqrt{n})}$ time
- Double mixed input attacks can be extended to circuit obfuscators
- [GGH$^+$13b]: only BP/circuit obfuscator currently standing in quantum

[GGH$^+$13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits, FOCS.

Conclusion (2/2)

Remarks

- Quantum poly time or classical $2^{O(\sqrt{n})}$ time
- Double mixed input attacks can be extended to circuit obfuscators
- [GGH$^+$13b]: only BP/circuit obfuscator currently standing in quantum

Open problems

- Quantum attack against [GGH$^+$13b]

[GGH$^+$13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits, FOCS.

Conclusion (2/2)

Remarks

- Quantum poly time or classical $2^{O(\sqrt{n})}$ time
- Double mixed input attacks can be extended to circuit obfuscators
- [GGH$^+$13b]: only BP/circuit obfuscator currently standing in quantum

Open problems

- Quantum attack against [GGH$^+$13b]
- Obfuscation for evasive functions

[GGH$^+$13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits, FOCS.

Conclusion (2/2)

Remarks

- Quantum poly time or classical $2^{O(\sqrt{n})}$ time
- Double mixed input attacks can be extended to circuit obfuscators
- [GGH$^+$13b]: only BP/circuit obfuscator currently standing in quantum

Open problems

- Quantum attack against [GGH$^+$13b]
- Obfuscation for evasive functions

Questions?

[GGH$^+$13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits, FOCS.

References I

Benny Applebaum and Zvika Brakerski.
Obfuscating circuits via composite-order graded encoding.
In TCC 2015, pages 528–556, 2015.

Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai.
Optimizing obfuscation: Avoiding barrington's theorem.
In CCS 2014, pages 646–658. ACM, 2014.

Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang.
On the (im) possibility of obfuscating programs.
In Crypto 2001, pages 1–18. Springer, 2001.

Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai.
Protecting obfuscation against algebraic attacks.
In Eurocrypt 2014, pages 221–238, 2014.

Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry.
Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits.
In Eurocrypt 2016, pages 764–791, 2016.

Zvika Brakerski and Guy N Rothblum.
Obfuscating conjunctions.
Crypto 2014, 2014.

Jean-François Biasse and Fang Song.
Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields.
In SODA 2016, pages 893–902. Society for Industrial and Applied Mathematics, 2016.

References II

Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev.
Recovering short generators of principal ideals in cyclotomic rings.
In Eurocrypt 2016, pages 559–585, 2016.

Yilei Chen, Craig Gentry, and Shai Halevi.
Cryptanalyses of candidate branching program obfuscators.
In Eurocrypt 2017, pages 278–307. Springer, 2017.

Jung Hee Cheon, Minki Hhan, Jiseung Kim, and Changmin Lee.
Cryptanalyses of branching program obfuscations over ggh13 multilinear map from the ntru problem.
In Crypto 2018, pages 184–210. Springer, 2018.

Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee.
Obfuscation from low noise multilinear maps.
ePrint, Report 2016/599, 2016.

Rex Fernando, Peter Rasmussen, and Amit Sahai.
Preventing CLT attacks on obfuscation with linear overhead.
In Asiacrypt 2017, pages 242–271, 2017.

Sanjam Garg, Craig Gentry, and Shai Halevi.
Candidate multilinear maps from ideal lattices.
In Eurocrypt 2013, pages 1–17. Springer, 2013.

Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters.
Candidate indistinguishability obfuscation and functional encryption for all circuits.
FOCS 2013, 2013.

References III

Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry.
Secure obfuscation in a weak multilinear map model.
In TCC 2016, pages 241–268, 2016.

Eric Miles, Amit Sahai, and Mor Weiss.
Protecting obfuscation against arithmetic attacks.
ePrint, Report 2014/878, 2014.

Eric Miles, Amit Sahai, and Mark Zhandry.
Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13.
In Crypto 2016, pages 629–658, 2016.

Rafael Pass, Karn Seth, and Sidharth Telang.
Indistinguishability obfuscation from semantically-secure multilinear encodings.
In Crypto 2014, pages 500–517, 2014.

Joe Zimmerman.
How to obfuscate programs directly.
In Eurocrypt 2015, pages 439–467, 2015.

The GGH13 multilinear map

- Define $R = \mathbb{Z}[X]/(X^n + 1)$ with $n = 2^k$.

The GGH13 multilinear map

- Define $R = \mathbb{Z}[X]/(X^n + 1)$ with $n = 2^k$.
- Sample g a "small" element in R.
  $\Rightarrow$ the plaintext space is $\mathcal{P} = R/\langle g \rangle$.

The GGH13 multilinear map

- Define $R = \mathbb{Z}[X]/(X^n + 1)$ with $n = 2^k$.
- Sample g a "small" element in R.
  $\Rightarrow$ the plaintext space is $\mathcal{P} = R/\langle g \rangle$.
- Sample q a "large" integer.
  $\Rightarrow$ the encoding space is $R_q = R/(qR) = \mathbb{Z}_q[X]/(X^n + 1)$.

Notation

We write $[r]_q$ or $[r]$ the elements in R_q.

The GGH13 multilinear map: encodings

- Sample z uniformly in R_q.
- **Encoding:** An encoding of a at level i is

$$u = \left[\frac{a + rg}{z^i}\right]_q$$

where $a + rg$ is a small element in $a + \langle g \rangle$.

The GGH13 multilinear map: encodings

- Sample z uniformly in R_q.
- **Encoding:** An encoding of a at level i is

$$u = \left[\frac{a + rg}{z^i} \right]_q$$

where $a + rg$ is a small element in $a + \langle g \rangle$.

Addition and multiplication

Addition:

$$\left[\frac{a_1 + r_1 g}{z^i} \right]_q + \left[\frac{a_2 + r_2 g}{z^i} \right]_q = \left[\frac{a_1 + a_2 + r' g}{z^i} \right]_q .$$

Multiplication:

$$\left[\frac{a_1 + r_1 g}{z^i} \right]_q \cdot \left[\frac{a_2 + r_2 g}{z^j} \right]_q = \left[\frac{a_1 \cdot a_2 + r' g}{z^{i+j}} \right]_q .$$

The GGH13 multilinear map: zero-test

- Sample h in R of the order of $q^{1/2}$.
- Define

$$p_{zt} = [z^{\kappa} h g^{-1}]_q.$$

The GGH13 multilinear map: zero-test

- Sample h in R of the order of $q^{1/2}$.
- Define
$$p_{zt} = [z^\kappa h g^{-1}]_q.$$

Zero-test

To test if $u = [c/z^\kappa]$ is an encoding of zero (i.e. $c = 0 \mod g$), compute

$$[u \cdot p_{zt}]_q = [chg^{-1}]_q.$$

This is small iff c is a small multiple of g.

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

- Get multiple top-level encoding of zero $u_i = [c_i g / z^\kappa]_q$

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

- Get multiple top-level encoding of zero $u_i = [c_i g / z^\kappa]_q$
- Zero-test them $\Rightarrow [u_i p_{zt}]_q = c_i h$

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

- Get multiple top-level encoding of zero $u_i = [c_i g / z^\kappa]_q$
- Zero-test them $\Rightarrow [u_i p_{zt}]_q = c_i h$
- Recover ideal $\langle h \rangle$ from the $c_i h$

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

- Get multiple top-level encoding of zero $u_i = [c_i g / z^\kappa]_q$
- Zero-test them $\Rightarrow [u_i p_{zt}]_q = c_i h$
- Recover ideal $\langle h \rangle$ from the $c_i h$
- Recover h from $\langle h \rangle$ (quantum poly time [BS16, CDPR16])

[BS16] J.-F. Biasse and F. Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields, SODA.

[CDPR16] R. Cramer, L. Ducas, C. Peikert and O.Regev. Recovering Short Generators of Principal Ideals in Cyclotomic Rings, Eurocrypt.

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

- Get multiple top-level encoding of zero $u_i = [c_i g / z^\kappa]_q$
- Zero-test them $\Rightarrow [u_i p_{zt}]_q = c_i h$
- Recover ideal $\langle h \rangle$ from the $c_i h$
- Recover h from $\langle h \rangle$ (quantum poly time [BS16, CDPR16])
- Create $p'_{zt} = [p_{zt}^2 / h^2]_q = [z^{2\kappa} g^{-2}]_q$

[BS16] J.-F. Biasse and F. Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields, SODA.

[CDPR16] R. Cramer, L. Ducas, C. Peikert and O.Regev. Recovering Short Generators of Principal Ideals in Cyclotomic Rings, Eurocrypt.

Quantum double-zero-test

Reminder

Zero-test: $p_{zt} = [z^\kappa h g^{-1}]_q$.

- Get multiple top-level encoding of zero $u_i = [c_i g / z^\kappa]_q$
- Zero-test them $\Rightarrow [u_i p_{zt}]_q = c_i h$
- Recover ideal $\langle h \rangle$ from the $c_i h$
- Recover h from $\langle h \rangle$ (quantum poly time [BS16, CDPR16])
- Create $p'_{zt} = [p_{zt}^2 / h^2]_q = [z^{2\kappa} g^{-2}]_q$

$$[u p'_{zt}]_q \text{ small } \Leftrightarrow u = [cg^2/z^{2\kappa}]_q \text{ for some small } c$$
$$\Leftrightarrow u \text{ is a double zero at level } 2\kappa$$

[BS16] J.-F. Biasse and F. Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields, SODA.

[CDPR16] R. Cramer, L. Ducas, C. Peikert and O.Regev. Recovering Short Generators of Principal Ideals in Cyclotomic Rings, Eurocrypt.