Attacks on GGH13-based obfuscation

Alice Pellet-Mary

ENS de Lyon

Opacity workshop
May 19, 2018

Indistinguishability Obfuscation (iO)

$$C \equiv C'$$

$C(x) = C'(x)$ for all x

$$\Downarrow$$

$$\mathcal{O}(C) \approx_c \mathcal{O}(C')$$

3 categories

- Branching program obfuscation
- Circuit obfuscation
- Obfuscation from functional encryption

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

3 categories

- Branching program obfuscation
- Circuit obfuscation
- Obfuscation from functional encryption

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

3 categories

- Branching program obfuscation
  - use multilinear maps (mmap)
  - security proofs if mmap is ideal

- Circuit obfuscation

- Obfuscation from functional encryption

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

3 categories

- Branching program obfuscation
  - use multilinear maps (mmap)
  - security proofs if mmap is ideal
  - attacks should use weaknesses of the mmap
- Circuit obfuscation
- Obfuscation from functional encryption

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

3 categories

- Branching program obfuscation ⇒ based on GGH13
  - use multilinear maps (mmap)
  - security proofs if mmap is ideal
  - attacks should use weaknesses of the mmap
- Circuit obfuscation
- Obfuscation from functional encryption

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

Overview of the talk

1. Simple obfuscator
2. Quantum attack
3. State-of-the-art

Outline of the talk

1. Simple obfuscator
2. Quantum attack
3. State-of-the-art

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \ldots, r\}^\ell$ (where r is the size of the input).

BP

	x_1	x_1	x_2	x_1	x_3	x_2	
M_0	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{4,1}$	$M_{5,1}$	$M_{6,1}$	M_7
	$M_{1,0}$	$M_{2,0}$	$M_{3,0}$	$M_{4,0}$	$M_{5,0}$	$M_{6,0}$	

Evaluation on $x = 0 \quad 1 \quad 1$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \ldots, r\}^\ell$ (where r is the size of the input).

								BP
	x_1	x_1	x_2	x_1	x_3	x_2		
M_0	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{4,1}$	$M_{5,1}$	$M_{6,1}$	M_7	
	$M_{1,0}$	$M_{2,0}$	$M_{3,0}$	$M_{4,0}$	$M_{5,0}$	$M_{6,0}$		

Evaluation on $x = 0 \quad 1 \quad 1$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \ldots, r\}^\ell$ (where r is the size of the input).

BP

		x_1	x_1	x_2	x_1	x_3	x_2	
M_0	$\times$	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{4,1}$	$M_{5,1}$	$M_{6,1}$	M_7
		$M_{1,0}$	$M_{2,0}$	$M_{3,0}$	$M_{4,0}$	$M_{5,0}$	$M_{6,0}$	

Evaluation on $x = 0 \quad 1 \quad 1$

$\uparrow$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \ldots, r\}^\ell$ (where r is the size of the input).

BP

	x_1	x_1	x_2	x_1	x_3	x_2	
$M_0 \times$	$M_{1,1}$ / $M_{1,0}$ $\times$	$M_{2,1}$ / $M_{2,0}$	$M_{3,1}$ / $M_{3,0}$	$M_{4,1}$ / $M_{4,0}$	$M_{5,1}$ / $M_{5,0}$	$M_{6,1}$ / $M_{6,0}$	M_7

Evaluation on $x = 0 \quad 1 \quad 1$

$\uparrow$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \ldots, r\}^\ell$ (where r is the size of the input).

BP

		x_1		x_1		x_2	x_1	x_3	x_2	
M_0	$\times$	$M_{1,1}$ / $M_{1,0}$	$\times$	$M_{2,1}$ / $M_{2,0}$	$\times$	$M_{3,1}$ / $M_{3,0}$	$M_{4,1}$ / $M_{4,0}$	$M_{5,1}$ / $M_{5,0}$	$M_{6,1}$ / $M_{6,0}$	M_7

Evaluation on $x = 0 \; 1 \; 1$

$\uparrow$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \dots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \dots, r\}^\ell$ (where r is the size of the input).

BP

		x_1		x_1		x_2		x_1	x_3	x_2	
M_0	$\times$	$M_{1,1}$ / $M_{1,0}$	$\times$	$M_{2,1}$ / $M_{2,0}$	$\times$	$M_{3,1}$ / $M_{3,0}$	$\times$	$M_{4,1}$ / $M_{4,0}$	$M_{5,1}$ / $M_{5,0}$	$M_{6,1}$ / $M_{6,0}$	M_7

Evaluation on $x = 0 \quad 1 \quad 1$

$\uparrow$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \dots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \dots, r\}^\ell$ (where r is the size of the input).

BP

$$\begin{array}{ccccccccccccc} & & x_1 & & x_1 & & x_2 & & x_1 & & x_3 & & x_2 & & \\ M_0 & \times & \begin{matrix} M_{1,1} \\ M_{1,0} \end{matrix} & \times & \begin{matrix} M_{2,1} \\ M_{2,0} \end{matrix} & \times & \begin{matrix} M_{3,1} \\ M_{3,0} \end{matrix} & \times & \begin{matrix} M_{4,1} \\ M_{4,0} \end{matrix} & \times & \begin{matrix} M_{5,1} \\ M_{5,0} \end{matrix} & & \begin{matrix} M_{6,1} \\ M_{6,0} \end{matrix} & & M_7 \end{array}$$

Evaluation on $x = 0 \;\; 1 \;\; 1$

$\uparrow$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \dots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \dots, r\}^\ell$ (where r is the size of the input).

BP

$$M_0 \times \begin{matrix} x_1 \\ M_{1,1} \\ M_{1,0} \end{matrix} \times \begin{matrix} x_1 \\ M_{2,1} \\ M_{2,0} \end{matrix} \times \begin{matrix} x_2 \\ M_{3,1} \\ M_{3,0} \end{matrix} \times \begin{matrix} x_1 \\ M_{4,1} \\ M_{4,0} \end{matrix} \times \begin{matrix} x_3 \\ M_{5,1} \\ M_{5,0} \end{matrix} \times \begin{matrix} x_2 \\ M_{6,1} \\ M_{6,0} \end{matrix} \quad M_7$$

Evaluation on $x = 0 \quad 1 \quad 1$

$\uparrow$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1, \ldots, r\}^\ell$ (where r is the size of the input).

BP

$$M_0 \times \begin{matrix} x_1 \\ M_{1,1} \\ M_{1,0} \end{matrix} \times \begin{matrix} x_1 \\ M_{2,1} \\ M_{2,0} \end{matrix} \times \begin{matrix} x_2 \\ M_{3,1} \\ M_{3,0} \end{matrix} \times \begin{matrix} x_1 \\ M_{4,1} \\ M_{4,0} \end{matrix} \times \begin{matrix} x_3 \\ M_{5,1} \\ M_{5,0} \end{matrix} \times \begin{matrix} x_2 \\ M_{6,1} \\ M_{6,0} \end{matrix} \times M_7$$

Evaluation on $x = 0 \quad 1 \quad 1$

Branching programs

A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- 2ℓ matrices $M_{i,b}$ (for $i \in \{1,\ldots,\ell\}$ and $b \in \{0,1\}$),
- two vectors M_0 and $M_{\ell+1}$,
- a vector inp $\in \{1,\ldots,r\}^\ell$ (where r is the size of the input).

BP

$$\begin{matrix} & & x_1 & & x_1 & & x_2 & & x_1 & & x_3 & & x_2 & & & & \\ M_0 & \times & \begin{matrix} M_{1,1} \\ M_{1,0} \end{matrix} & \times & \begin{matrix} M_{2,1} \\ M_{2,0} \end{matrix} & \times & \begin{matrix} M_{3,1} \\ M_{3,0} \end{matrix} & \times & \begin{matrix} M_{4,1} \\ M_{4,0} \end{matrix} & \times & \begin{matrix} M_{5,1} \\ M_{5,0} \end{matrix} & \times & \begin{matrix} M_{6,1} \\ M_{6,0} \end{matrix} & \times & M_7 & \begin{matrix} = \\ \neq \end{matrix} & \begin{matrix} 0 \to 0 \\ 0 \to 1 \end{matrix} \end{matrix}$$

Evaluation on $x = 0 \quad 1 \quad 1$

Cryptographic multilinear maps (asymmetric setting)

Different levels of encodings, parametrized by sets $S \subseteq \{1, \cdots, \kappa\}$.

Definition: asymmetric multilinear map

$\mathrm{Enc}(a, S)$: encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$, maximum level.

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

Multiplication: if $S_1 \cap S_2 = \emptyset$,

$$\mathrm{Mult}(\mathrm{Enc}(a_1, S_1), \mathrm{Enc}(a_2, S_2)) = \mathrm{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

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

Cryptographic multilinear maps (asymmetric setting)

Different levels of encodings, parametrized by sets $S \subseteq \{1, \cdots, \kappa\}$.

Definition: asymmetric multilinear map

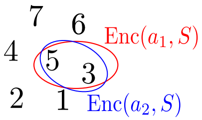
$\mathsf{Enc}(a, S)$: encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$, maximum level.

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

Multiplication: if $S_1 \cap S_2 = \emptyset$,

$$\mathsf{Mult}(\mathsf{Enc}(a_1, S_1), \mathsf{Enc}(a_2, S_2)) = \mathsf{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

Zero-test: $\mathsf{Zero\text{-}test}(\mathsf{Enc}(a, S^*)) = \mathsf{True}$ iff $a = 0$.



Cryptographic multilinear maps (asymmetric setting)

Different levels of encodings, parametrized by sets $S \subseteq \{1, \cdots, \kappa\}$.

Definition: asymmetric multilinear map

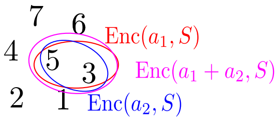
$\mathrm{Enc}(a, S)$: encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$, maximum level.

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

Multiplication: if $S_1 \cap S_2 = \emptyset$,

$$\mathrm{Mult}(\mathrm{Enc}(a_1, S_1), \mathrm{Enc}(a_2, S_2)) = \mathrm{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

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



Cryptographic multilinear maps (asymmetric setting)

Different levels of encodings, parametrized by sets $S \subseteq \{1, \cdots, \kappa\}$.

Definition: asymmetric multilinear map

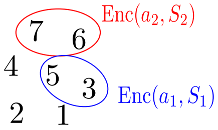
Enc(a, S): encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$, maximum level.

Addition: Add(Enc(a_1, S), Enc(a_2, S)) = Enc$(a_1 + a_2, S)$.

Multiplication: if $S_1 \cap S_2 = \emptyset$,

$$\text{Mult}(\text{Enc}(a_1, S_1), \text{Enc}(a_2, S_2)) = \text{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

Zero-test: Zero-test(Enc(a, S^*)) = True iff $a = 0$.



Cryptographic multilinear maps (asymmetric setting)

Different levels of encodings, parametrized by sets $S \subseteq \{1, \cdots, \kappa\}$.

Definition: asymmetric multilinear map

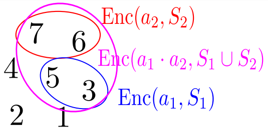
$\mathsf{Enc}(a, S)$: encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$, maximum level.

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

Multiplication: if $S_1 \cap S_2 = \emptyset$,

$$\mathsf{Mult}(\mathsf{Enc}(a_1, S_1), \mathsf{Enc}(a_2, S_2)) = \mathsf{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

Zero-test: $\text{Zero-test}(\mathsf{Enc}(a, S^*)) = \text{True}$ iff $a = 0$.



Cryptographic multilinear maps (asymmetric setting)

Different levels of encodings, parametrized by sets $S \subseteq \{1, \cdots, \kappa\}$.

Definition: asymmetric multilinear map

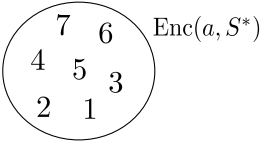
$\mathsf{Enc}(a, S)$: encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$, maximum level.

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

Multiplication: if $S_1 \cap S_2 = \emptyset$,

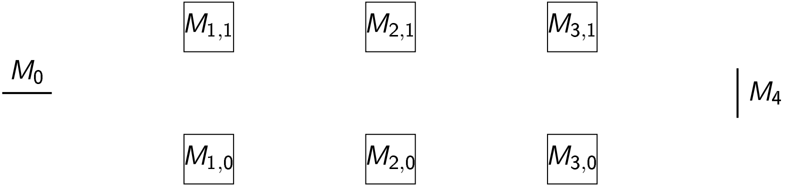
$$\mathsf{Mult}(\mathsf{Enc}(a_1, S_1), \mathsf{Enc}(a_2, S_2)) = \mathsf{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

Zero-test: $\mathsf{Zero\text{-}test}(\mathsf{Enc}(a, S^*)) = \mathsf{True}$ iff $a = 0$.



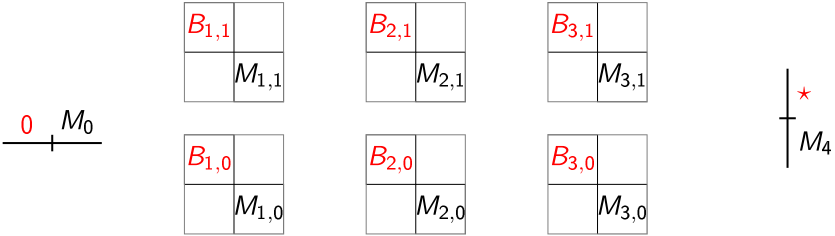
Simple obfuscator

- **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 mmap
- **Output:** The encoded matrices and vectors



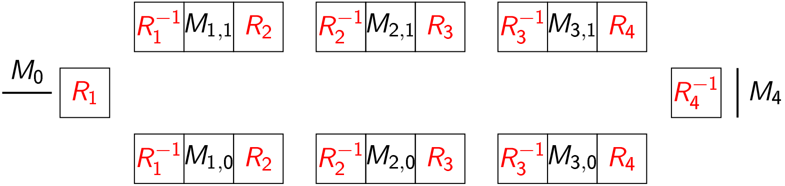
Simple obfuscator

- **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 mmap
- **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 mmap
- **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 mmap
- **Output:** The encoded matrices and vectors

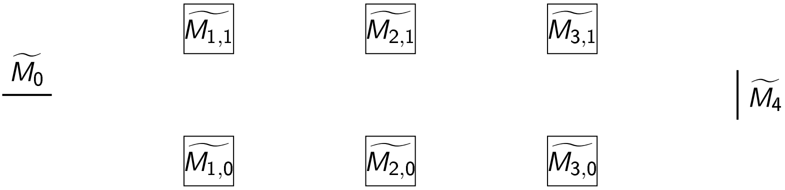
$\alpha_{1,1} \times \boxed{M_{1,1}}$ $\alpha_{2,1} \times \boxed{M_{2,1}}$ $\alpha_{3,1} \times \boxed{M_{3,1}}$

$\underline{M_0}$ $\big| M_4$

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

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 mmap
- **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 mmap
- **Output:** The encoded matrices and vectors

$S^* = \{1, 2, 3, 4, 5\}$

$\text{Enc}(\widetilde{M_{1,1}}, \{2\})$ $\text{Enc}(\widetilde{M_{2,1}}, \{3\})$ $\text{Enc}(\widetilde{M_{3,1}}, \{4\})$

$\text{Enc}(\widetilde{M_0}, \{1\})$ $\text{Enc}(\widetilde{M_4}, \{5\})$

$\text{Enc}(\widetilde{M_{1,0}}, \{2\})$ $\text{Enc}(\widetilde{M_{2,0}}, \{3\})$ $\text{Enc}(\widetilde{M_{3,0}}, \{4\})$

Outline of the talk

1 Simple obfuscator

2 Quantum attack

3 State-of-the-art

GGH13 in a quantum world

Reminder: asymmetric multilinear map

$\mathrm{Enc}(a, S)$ = encoding of a at level S.
$S^* = \{1, \cdots, \kappa\}$ maximum level.

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

Multiplication: if $S_1 \cap S_2 = \emptyset$,

$$\mathrm{Mult}(\mathrm{Enc}(a_1, S_1), \mathrm{Enc}(a_2, S_2)) = \mathrm{Enc}(a_1 \cdot a_2, S_1 \cup S_2).$$

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

GGH13 in a quantum world

The GGH13 map

$\text{Enc}(a, S)$ = encoding of $a \in \mathbb{Z}/p\mathbb{Z}$ at level S.
$S^* = \{1, \cdots, \kappa\}$ maximum level.

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

Multiplication: for any S_1, S_2,

$$\text{Mult}(\text{Enc}(a_1, S_1), \text{Enc}(a_2, S_2)) = \text{Enc}(a_1 \cdot a_2, S_1 \uplus S_2).$$

Zero-test: $\text{Zero-test}(\text{Enc}(a, S^*))$ = True iff $a = 0 \bmod p$.

e.g. $\{1, 3, 4\} \uplus \{2, 3\} = \{1, 2, 3, 3, 4\}$

GGH13 in a quantum world

The GGH13 map

$\text{Enc}(a, S)$ = encoding of $a \in \mathbb{Z}/p\mathbb{Z}$ at level S.
$S^* = \{1, \cdots, \kappa\}$ maximum level.

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

Multiplication: for any S_1, S_2,

$$\text{Mult}(\text{Enc}(a_1, S_1), \text{Enc}(a_2, S_2)) = \text{Enc}(a_1 \cdot a_2, S_1 \uplus S_2).$$

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

$2S^* = S^* \uplus S^* = \{1, 1, 2, 2, \cdots, \kappa, \kappa\}$

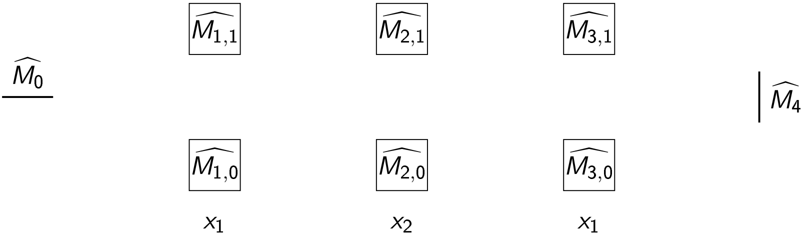
With a quantum computer

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

Mixed-input attack

Notations

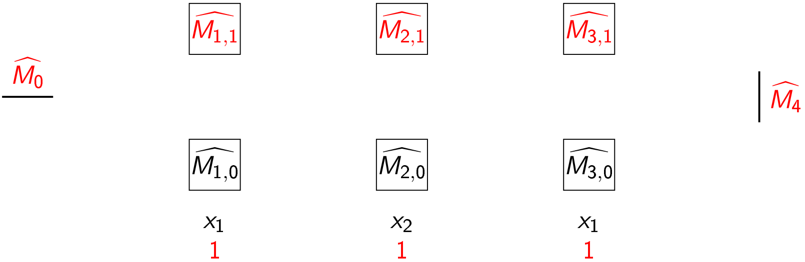
- $M_{i,b}$ input branching program
- $\widetilde{M_{i,b}}$ after randomisation
- $\widehat{M_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

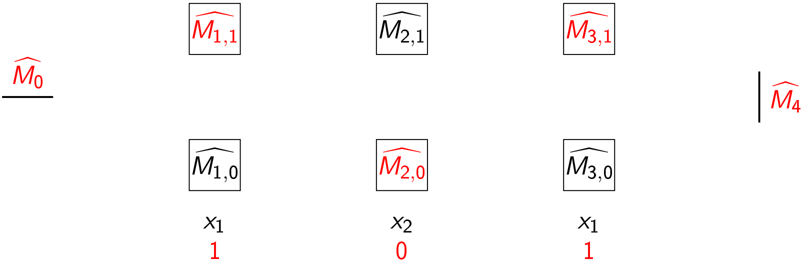
- $M_{i,b}$ input branching program
- $\widetilde{M_{i,b}}$ after randomisation
- $\widehat{M_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

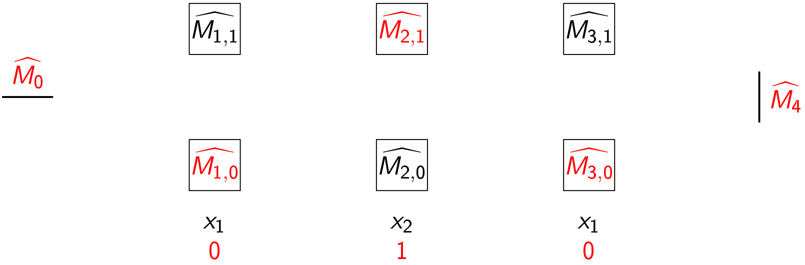
- $M_{i,b}$ input branching program
- $\widetilde{M_{i,b}}$ after randomisation
- $\widehat{M_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

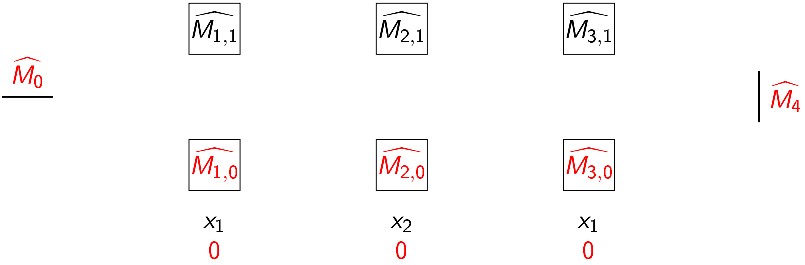
- $M_{i,b}$ input branching program
- $\widetilde{M_{i,b}}$ after randomisation
- $\widehat{M_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

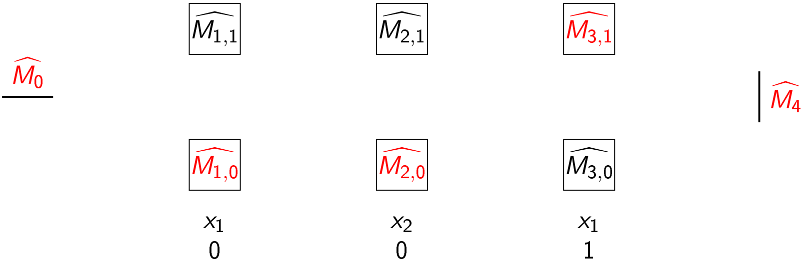
- $M_{i,b}$ input branching program
- $\widetilde{M_{i,b}}$ after randomisation
- $\widehat{M_{i,b}}$ after encoding with GGH13 map (output of the iO)



Mixed-input attack

Notations

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



Mixed-input attack

Notations

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

	$\text{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\})$	$\text{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\})$	$\text{Enc}(\boxed{\widetilde{M_{3,1}}},\{4\})$	
$\text{Enc}(\widetilde{M_0},\{1\})$				$\text{Enc}(\widetilde{M_4},\{5\})$
	$\text{Enc}(\boxed{\widetilde{M_{1,0}}},\{2\})$	$\text{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\})$	$\text{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$	
	x_1	x_2	x_1	
	0	0	1	

Preventing mixed-input attacks

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

Preventing mixed-input attacks

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

Preventing mixed-input attacks

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

Mmap degree: $S^* = \{1, 2, 3, 4, 5\}$

	$\mathrm{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\})$	$\mathrm{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\})$	$\mathrm{Enc}(\boxed{\widetilde{M_{3,1}}},\{4\})$	
$\mathrm{Enc}(\widetilde{M_0},\{1\})$				$\mathrm{Enc}(\widetilde{M_4},\{5\})$
	$\mathrm{Enc}(\boxed{\widetilde{M_{1,0}}},\{2\})$	$\mathrm{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\})$	$\mathrm{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$	
	x_1	x_2	x_1	

Preventing mixed-input attacks

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

Mmap degree: $S^* = \{1, 2, 3, 4, 5, 6\}$

$\mathrm{Enc}(\widetilde{M_0}, \{1\})$

$\mathrm{Enc}(\boxed{\widetilde{M_{1,1}}}, \{2\})$ $\mathrm{Enc}(\boxed{\widetilde{M_{2,1}}}, \{3\})$ $\mathrm{Enc}(\boxed{\widetilde{M_{3,1}}}, \{4, 6\})$

$\mathrm{Enc}(\boxed{\widetilde{M_{1,0}}}, \{2, 6\})$ $\mathrm{Enc}(\boxed{\widetilde{M_{2,0}}}, \{3\})$ $\mathrm{Enc}(\boxed{\widetilde{M_{3,0}}}, \{4\})$

x_1 x_2 x_1

$\mathrm{Enc}(\widetilde{M_4}, \{5\})$

Preventing mixed-input attacks

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

Mmap degree: $S^* = \{1, 2, 3, 4, 5, 6\}$

$\mathsf{Enc}(\widetilde{M_0}, \{1\})$

$\mathsf{Enc}(\boxed{\widetilde{M_{1,1}}}, \{2\})$ $\mathsf{Enc}(\boxed{\widetilde{M_{2,1}}}, \{3\})$ $\mathsf{Enc}(\boxed{\widetilde{M_{3,1}}}, \{4, 6\})$

$\mathsf{Enc}(\widetilde{M_4}, \{5\})$

$\mathsf{Enc}(\boxed{\widetilde{M_{1,0}}}, \{2, 6\})$ $\mathsf{Enc}(\boxed{\widetilde{M_{2,0}}}, \{3\})$ $\mathsf{Enc}(\boxed{\widetilde{M_{3,0}}}, \{4\})$

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: $S^* = \{1, 2, 3, 4, 5, 6\}$

$\text{Enc}(\widetilde{M_0}, \{1\})$

$\text{Enc}(\boxed{\widetilde{M_{1,1}}}, \{2\})$ $\quad$ $\text{Enc}(\boxed{\widetilde{M_{2,1}}}, \{3\})$ $\quad$ $\text{Enc}(\boxed{\widetilde{M_{3,1}}}, \{4, 6\})$

$\text{Enc}(\boxed{\widetilde{M_{1,0}}}, \{2, 6\})$ $\quad$ $\text{Enc}(\boxed{\widetilde{M_{2,0}}}, \{3\})$ $\quad$ $\text{Enc}(\boxed{\widetilde{M_{3,0}}}, \{4\})$

$\text{Enc}(\widetilde{M_4}, \{5\})$

x_1	x_2	x_1
0	0	1

Total level: $\{1, 2, 3, 4, 5, 6, 6\} \Rightarrow$ cannot zero-test

Preventing mixed-input attacks

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

Mmap degree: $S^* = \{1, 2, 3, 4, 5, 6\}$

$\mathrm{Enc}(\widetilde{M_{1,1}}, \{2\})$ $\mathrm{Enc}(\widetilde{M_{2,1}}, \{3\})$ $\mathrm{Enc}(\widetilde{M_{3,1}}, \{4, 6\})$

$\mathrm{Enc}(\widetilde{M_0}, \{1\})$ $\mathrm{Enc}(\widetilde{M_4}, \{5\})$

$\mathrm{Enc}(\widetilde{M_{1,0}}, \{2, 6\})$ $\mathrm{Enc}(\widetilde{M_{2,0}}, \{3\})$ $\mathrm{Enc}(\widetilde{M_{3,0}}, \{4\})$

x_1	x_2	x_1
0	0	1

Generalisation: $\{1\}, \{2,3\}, \{4,5\}, \{6,7\}$
$\{1,2\}, \{3,4\}, \{5,6\}, \{7\}$

Attack idea: double mixed input

Reminder

In quantum world, we have

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

Attack idea: double mixed input

Reminder

In quantum world, we have

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

	$\text{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\})$	$\text{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\})$	$\text{Enc}(\boxed{\widetilde{M_{3,1}}},\{4,6\})$		
$\underline{\text{Enc}(\widetilde{M_0},\{1\})}$				$\text{Enc}(\widetilde{M_4},\{5\})$	$\Rightarrow$ Level $\{1,2,3,4,5,6,6\}$
	$\text{Enc}(\boxed{\widetilde{M_{1,0}}},\{2,6\})$	$\text{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\})$	$\text{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$		
	x_1	x_2	x_1		

Attack idea: double mixed input

Reminder

In quantum world, we have

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

$\text{Enc}(\widetilde{M_0}, \{1\})$

$\text{Enc}(\widetilde{M_{1,1}}, \{2\})$ $\text{Enc}(\widetilde{M_{2,1}}, \{3\})$ $\text{Enc}(\widetilde{M_{3,1}}, \{4,6\})$

$\text{Enc}(\widetilde{M_{1,0}}, \{2,6\})$ $\text{Enc}(\widetilde{M_{2,0}}, \{3\})$ $\text{Enc}(\widetilde{M_{3,0}}, \{4\})$

x_1 x_2 x_1

$\text{Enc}(\widetilde{M_4}, \{5\})$ $\Rightarrow$ Level $\{1,2,3,4,5,6,6\}$

$\text{Enc}(\widetilde{M_0}, \{1\})$

$\text{Enc}(\widetilde{M_{1,1}}, \{2\})$ $\text{Enc}(\widetilde{M_{2,1}}, \{3\})$ $\text{Enc}(\widetilde{M_{3,1}}, \{4,6\})$

$\text{Enc}(\widetilde{M_{1,0}}, \{2,6\})$ $\text{Enc}(\widetilde{M_{2,0}}, \{3\})$ $\text{Enc}(\widetilde{M_{3,0}}, \{4\})$

x_1 x_2 x_1

$\text{Enc}(\widetilde{M_4}, \{5\})$ $\Rightarrow$ Level $\{1,2,3,4,5\}$

Attack idea: double mixed input

Reminder

In quantum world, we have

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

$\text{Enc}(\widetilde{M_0},\{1\})$

$\text{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\})$ $\text{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\})$ $\text{Enc}(\boxed{\widetilde{M_{3,1}}},\{4,6\})$

$\text{Enc}(\boxed{\widetilde{M_{1,0}}},\{2,6\})$ $\text{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\})$ $\text{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$

x_1 x_2 x_1

$\Big| \text{Enc}(\widetilde{M_4},\{5\})$ $\Rightarrow$ Level $\{1,2,3,4,5,6,6\}$

$\times$

$\text{Enc}(\widetilde{M_0},\{1\})$

$\text{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\})$ $\text{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\})$ $\text{Enc}(\boxed{\widetilde{M_{3,1}}},\{4,6\})$

$\text{Enc}(\boxed{\widetilde{M_{1,0}}},\{2,6\})$ $\text{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\})$ $\text{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$

x_1 x_2 x_1

$\Big| \text{Enc}(\widetilde{M_4},\{5\})$ $\Rightarrow$ Level $\{1,2,3,4,5\}$

Attack idea: double mixed input

Reminder

In quantum world, we have

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

$$\underline{\text{Enc}(\widetilde{M_0},\{1\})}$$

$$\text{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\}) \quad \text{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\}) \quad \text{Enc}(\boxed{\widetilde{M_{3,1}}},\{4,6\})$$

$$\text{Enc}(\boxed{\widetilde{M_{1,0}}},\{2,6\}) \quad \text{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\}) \quad \text{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$$

$$x_1 \quad x_2 \quad x_1$$

$$\Big| \text{Enc}(\widetilde{M_4},\{5\}) \quad \Rightarrow \text{Level } \{1,2,3,4,5,6,6\}$$

$\times$

$$\underline{\text{Enc}(\widetilde{M_0},\{1\})}$$

$$\text{Enc}(\boxed{\widetilde{M_{1,1}}},\{2\}) \quad \text{Enc}(\boxed{\widetilde{M_{2,1}}},\{3\}) \quad \text{Enc}(\boxed{\widetilde{M_{3,1}}},\{4,6\})$$

$$\text{Enc}(\boxed{\widetilde{M_{1,0}}},\{2,6\}) \quad \text{Enc}(\boxed{\widetilde{M_{2,0}}},\{3\}) \quad \text{Enc}(\boxed{\widetilde{M_{3,0}}},\{4\})$$

$$x_1 \quad x_2 \quad x_1$$

$$\Big| \text{Enc}(\widetilde{M_4},\{5\}) \quad \Rightarrow \text{Level } \{1,2,3,4,5\}$$

Product level: $S^* \uplus S^* = 2S^*$

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

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

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

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

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

- $C_1 \equiv C_2$

One example of C_1 and C_2

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

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

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

One example of C_1 and C_2

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

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

- $C_1 \equiv C_2$
- 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

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

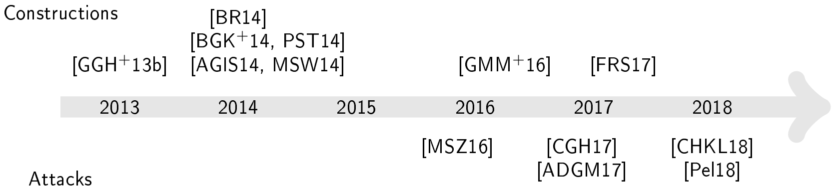
$$C_2: \quad \begin{pmatrix}1 & 0\end{pmatrix} \quad \begin{matrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\\ \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} & \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} & \begin{pmatrix}0 &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

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

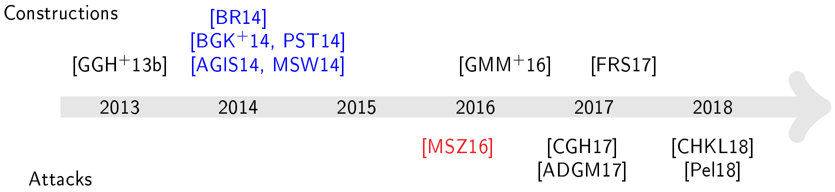
We can distinguish $\mathcal{O}(C_1)$ from $\mathcal{O}(C_2)$

Outline of the talk

History (GGH13-based branching program obfuscation)

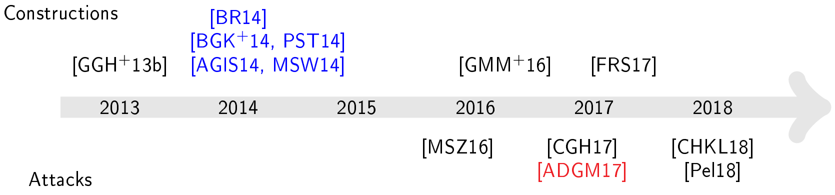


History (GGH13-based branching program obfuscation)



[MSZ16]: all constructions without diagonal blocks

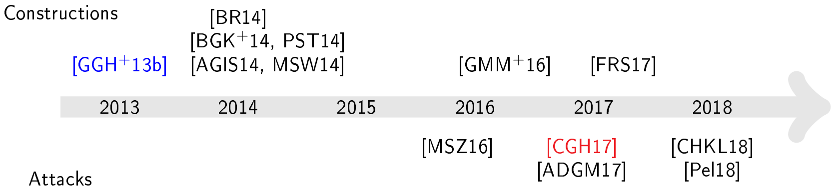
History (GGH13-based branching program obfuscation)



[MSZ16]: all constructions without diagonal blocks

[ADGM17]: idem MSZ but from circuits

History (GGH13-based branching program obfuscation)

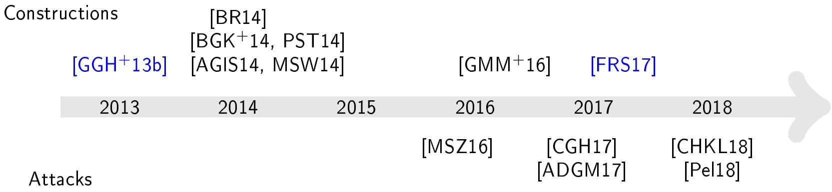


[MSZ16]: all constructions without diagonal blocks

[ADGM17]: idem MSZ but from circuits

[CGH17]: use input-partitionability (cf CLT13)

History (GGH13-based branching program obfuscation)

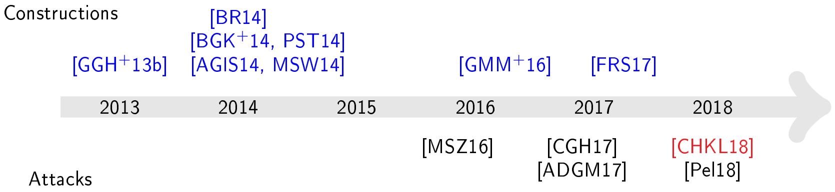


[MSZ16]: all constructions without diagonal blocks

[ADGM17]: idem MSZ but from circuits

[CGH17]: use input-partitionability (cf CLT13) $\Rightarrow$ prevented by [FRS17]

History (GGH13-based branching program obfuscation)



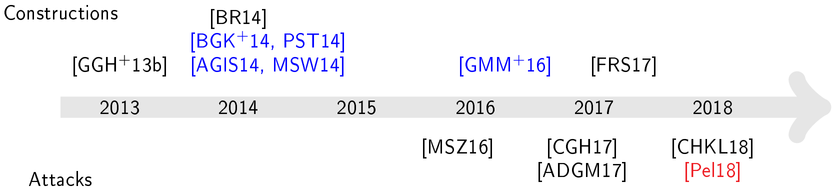
[MSZ16]: all constructions without diagonal blocks

[ADGM17]: idem MSZ but from circuits

[CGH17]: use input-partitionability (cf CLT13) $\Rightarrow$ prevented by [FRS17]

[CHKL18]: NTRU attack for specific choices of parameters

History (GGH13-based branching program obfuscation)



[MSZ16]: all constructions without diagonal blocks

[ADGM17]: idem MSZ but from circuits

[CGH17]: use input-partitionability (cf CLT13) $\Rightarrow$ prevented by [FRS17]

[CHKL18]: NTRU attack for specific choices of parameters

[Pel18]: quantum attack

Current status

Attacks \ iOs	[GGH+13b]	[BR14, BGK+14, PST14, AGIS14, MSW14]	[GMM+16]	circuit obfuscators [Zim15, AB15, DGG+18]
[MSZ16]		fully broken		
[CGH17]	input-partitionable			
[CHKL18]	some parameters		some parameters	
[Pel18]			quantum	quantum

Still standing classically:

- [GGH+13b]+[FRS17]
- [GMM+16]
- all circuit obfuscators

Still standing quantumly:

- [GGH+13b]+[FRS17]

Current status

Attacks \ iOs	[GGH+13b]	[BR14, BGK+14, PST14, AGIS14, MSW14]	[GMM+16]	circuit obfuscators [Zim15, AB15, DGG+18]
[MSZ16]		fully broken		
[CGH17]	input-partitionable			
[CHKL18]	some parameters		some parameters	
[Pel18]			quantum	quantum

Still standing classically:

- [GGH+13b]+[FRS17]
- [GMM+16]
- all circuit obfuscators

Still standing quantumly:

- [GGH+13b]+[FRS17]

Questions?

References I

Benny Applebaum and Zvika Brakerski.
Obfuscating circuits via composite-order graded encoding.
In Theory of Cryptography Conference, pages 528–556. Springer, 2015.

Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee.
Cryptanalysis of indistinguishability obfuscations of circuits over ggh13.
In 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai.
Optimizing obfuscation: Avoiding barrington's theorem.
In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 646–658. ACM, 2014.

Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai.
Protecting obfuscation against algebraic attacks.
In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 221–238. Springer, 2014.

Zvika Brakerski and Guy N. Rothblum.
Virtual black-box obfuscation for all circuits via generic graded encoding.
In Theory of Cryptography Conference, pages 1–25. Springer, 2014.

Yilei Chen, Craig Gentry, and Shai Halevi.
Cryptanalyses of candidate branching program obfuscators.
In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 278–307. Springer, 2017.

References II

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

Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee.
Obfuscation from low noise multilinear maps.
In International Conference in Cryptology in India, pages 329–352. Springer, 2018.

Rex Fernando, Peter MR Rasmussen, and Amit Sahai.
Preventing clt attacks on obfuscation with linear overhead.
In International Conference on the Theory and Application of Cryptology and Information Security, pages 242–271. Springer, 2017.

Sanjam Garg, Craig Gentry, and Shai Halevi.
Candidate multilinear maps from ideal lattices.
In Annual International Conference on the Theory and Applications of Cryptographic Techniques, 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.
pages 40–49, 2013.

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

Eric Miles, Amit Sahai, and Mor Weiss.
Protecting obfuscation against arithmetic attacks.
IACR Cryptology ePrint Archive, 2014:878, 2014.

References III

Eric Miles, Amit Sahai, and Mark Zhandry.
Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13.
In Annual International Cryptology Conference, pages 629–658. Springer, 2016.

Alice Pellet-Mary.
Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model.
In Annual International Cryptology Conference, pages 153–183. Springer, 2018.

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

Joe Zimmerman.
How to obfuscate programs directly.
In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 439–467. Springer, 2015.