Program obfuscation

Alice Pellet-Mary

LIP, ENS de Lyon

PhD seminar, CWI
July 12, 2019
Obfuscation

An obfuscator should:

- render the code of a program unintelligible;
- while preserving functionality and efficiency.
Obfuscation

An obfuscator should:

- render the code of a program unintelligible;
- while preserving functionality and efficiency.

Two kind of obfuscators:

- practical obfuscators (white box)
- theoretical obfuscators (iO)
Obfuscation

An obfuscator should:

- render the code of a program unintelligible;
- while preserving functionality and efficiency.

Two kind of obfuscators:

- practical obfuscators (white box)
- theoretical obfuscators (iO)
Overview of the talk

1 Definition

2 Candidates
   - Security
   - Practicability

3 Example of construction of an obfuscator
Outline of the talk

1. Definition

2. Candidates
   - Security
   - Practicability

3. Example of construction of an obfuscator
What is a program?

- C/C++/Python/⋯ code;
What is a program?

- C/C++/Python/⋯ code;
- Turing machine;

Notation

\[ C = \text{class of all polynomial size Boolean circuits} \]
What is a program?

- C/C++/Python/⋯ code;
- Turing machine;
- Boolean circuit;

\[ C = \text{class of all polynomial size boolean circuits} \]
What is a program?

- C/C++/Python/⋯ code;
- Turing machine;
- Boolean circuit;
- Branching programs;

\[
\bar{x} \lor (y \land z)
\]

**Notation**

\[
C = \text{class of all polynomial size boolean circuits}
\]
## Virtual Black Box (VBB) Obfuscation

### Recall

$\mathcal{C} =$ class of all polynomial size boolean circuits

A **VBB obfuscator** $\mathcal{O} : \mathcal{C} \rightarrow \mathcal{C}$ should satisfy

- **Functionality**: For all $C \in \mathcal{C}$, $\mathcal{O}(C) \equiv C$;
- **Efficiency**: $\mathcal{O}$ is PPT $\Rightarrow$ for all $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq \text{poly}(|C|)$;
- **Virtual Black Box security**: For all PPT $A$, there exists a PPT $\text{Sim}$ such that for all $C \in \mathcal{C}$, $\left| \text{Pr}[A(\mathcal{O}(C)) = 1] - \text{Pr}[\text{Sim}(C) = 1] \right| \leq \text{negl}$.

VBB obfuscation is impossible to achieve [BGI+01].
Virtual Black Box (VBB) obfuscation

Recall

\[ C = \text{class of all polynomial size boolean circuits} \]

A VBB obfuscator \( \mathcal{O} : C \rightarrow C \) should satisfy

- (Functionality) For all \( C \in C \), \( \mathcal{O}(C) \equiv C \);
Virtual Black Box (VBB) obfuscation

Recall

\( \mathcal{C} = \text{class of all polynomial size boolean circuits} \)

A VBB obfuscator \( \mathcal{O} : \mathcal{C} \rightarrow \mathcal{C} \) should satisfy

- (Functionality) For all \( C \in \mathcal{C} \), \( \mathcal{O}(C) \equiv C \);
- (Efficiency) \( \mathcal{O} \) is PPT \( \Rightarrow \) for all \( C \in \mathcal{C} \), \( |\mathcal{O}(C)| \leq \text{poly}(|C|) \);
Virtual Black Box (VBB) obfuscation

Recall

$\mathcal{C} = \text{class of all polynomial size boolean circuits}$

A VBB obfuscator $\mathcal{O} : \mathcal{C} \rightarrow \mathcal{C}$ should satisfy

- (Functionality) For all $C \in \mathcal{C}$, $\mathcal{O}(C) \equiv C$;
- (Efficiency) $\mathcal{O}$ is PPT $\Rightarrow$ for all $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq \text{poly}(|C|)$;
- (Virtual Black Box security) For all PPT $A$, there exists a PPT Sim s.t. for all $C \in \mathcal{C}$,

$$\left| \mathbb{P}[A(\mathcal{O}(C)) = 1] - \mathbb{P}[\text{Sim}^C(1^{|C|}) = 1] \right| \leq \text{negl.}$$
Virtual Black Box (VBB) obfuscation

Recall

$\mathcal{C} =$ class of all polynomial size boolean circuits

A VBB obfuscator $\mathcal{O} : \mathcal{C} \rightarrow \mathcal{C}$ should satisfy

- (Functionality) For all $C \in \mathcal{C}$, $\mathcal{O}(C) \equiv C$;
- (Efficiency) $\mathcal{O}$ is PPT $\Rightarrow$ for all $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq \text{poly}(|C|)$;
- (Virtual Black Box security) For all PPT $A$, there exists a PPT Sim s.t. for all $C \in \mathcal{C}$,

$$\left| \mathbb{P}[A(\mathcal{O}(C)) = 1] - \mathbb{P}[\text{Sim}^C(1^{|C|}) = 1] \right| \leq \text{negl.}$$

VBB obfuscation is impossible to achieve [BGI+01]

---

Indistinguishability Obfuscation (iO)

An indistinguishability obfuscator $O : C \rightarrow C$ should satisfy
Indistinguishability Obfuscation (iO)

An indistinguishability obfuscator $O : \mathcal{C} \rightarrow \mathcal{C}$ should satisfy

- (Functionality) For all $C \in \mathcal{C}$, $O(C) \equiv C$;
Indistinguishability Obfuscation (iO)

An indistinguishability obfuscator $\mathcal{O} : \mathcal{C} \rightarrow \mathcal{C}$ should satisfy

- (Functionality) For all $C \in \mathcal{C}$, $\mathcal{O}(C) \equiv C$;
- (Efficiency) $\mathcal{O}$ is PPT $\Rightarrow$ for all $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq \text{poly}(|C|)$;
Indistinguishability Obfuscation (iO)

An indistinguishability obfuscator $O : C \rightarrow C$ should satisfy

- (Functionality) For all $C \in C$, $O(C) \equiv C$;
- (Efficiency) $O$ is PPT $\Rightarrow$ for all $C \in C$, $|O(C)| \leq \text{poly}(|C|)$;
- (indistinguishability) For all $C_1, C_2 \in C$ with $C_1 \equiv C_2$,
  $$O(C_1) \simeq_c O(C_2).$$
If $P = NP$, then iOs exist, e.g.:

⇒ There exist inefficient iOs (even if $P \neq NP$)
If P = NP, then iOs exist, e.g.:

⇒ There exist inefficient iOs (even if P ≠ NP)

\( \mathcal{O}(C) = \text{smallest circuit computing the same function as } C \)
Why is iO useful (1)

iO achieves “best possible” obfuscation
Why is iO useful (1)

iO achieves “best possible” obfuscation

Proof:
- let $\mathcal{O}$ be an iO obfuscator and $\mathcal{O}'$ be another obfuscator

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

$\mathcal{O}(C)$ reveals less info than $\mathcal{O}'(C)$.
Why is iO useful (1)

iO achieves “best possible” obfuscation

Proof:

- let \( \mathcal{O} \) be an iO obfuscator and \( \mathcal{O}' \) be another obfuscator
- for any \( C \in \mathcal{C} \), \( \mathcal{O}(C) \sim_c \mathcal{O}(\mathcal{O}'(C)) \)
Why is iO useful (1)

iO achieves “best possible” obfuscation

Proof:

- let \( \mathcal{O} \) be an iO obfuscator and \( \mathcal{O}' \) be another obfuscator
- for any \( C \in \mathcal{C} \), \( \mathcal{O}(C) \sim_c \mathcal{O}(\mathcal{O}'(C)) \)
- \( \mathcal{O}(\mathcal{O}'(C)) \) reveals less info than \( \mathcal{O}'(C) \)
Why is iO useful (1)

iO achieves “best possible” obfuscation

Proof:

- let $O$ be an iO obfuscator and $O'$ be another obfuscator
- for any $C \in C$, $O(C) \simeq_c O(O'(C))$
- $O(O'(C))$ reveals less info than $O'(C)$
- $O(C)$ reveals less info than $O'(C)$
Why is iO useful (1)

iO achieves “best possible” obfuscation

**Proof:**

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

Informally: anything revealed by $\mathcal{O}(C)$ is revealed by any $C' \equiv C$
Why is iO useful (2)

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

Example: black box decryption (symmetric setting)
Recall: anything revealed by $O(C)$ is revealed by any $C' \equiv C$ taken $(\text{Setup}, \text{Enc}, \text{Dec})$ your favourite SKE scheme

\begin{align*}
\text{Setup'}: & \quad \sk_1 \leftarrow \text{Setup}(), \quad \sk_2 \leftarrow \text{Setup}() \\
\text{Enc'}(m, \sk') : & \quad c_1 \leftarrow \text{Enc}(m, \sk_1), \quad c_2 \leftarrow \text{Enc}(m, \sk_2) \\
\text{Dec'}: & \quad C_1(c_1, c_2) = \text{Dec}(\sk_1, c_1)(\text{sk}_1 \text{hardcoded in } C_1) \\
& \quad C_2(c_1, c_2) = \text{Dec}(\sk_2, c_2)(\text{sk}_2 \text{hardcoded in } C_2) \quad C_1 \equiv C_2 \Rightarrow C = O(C_1) \equiv O(C_2)
\end{align*}

A. Pellet-Mary

Program obfuscation

PhD seminar 10/23
Why is iO useful (2)

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

**Example:** black box decryption (symmetric setting)
Why is iO useful (2)

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

Example: black box decryption (symmetric setting)

Recall: anything revealed by $O(C)$ is revealed by any $C' \equiv C$
Why is iO useful (2)

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

**Example:** black box decryption (symmetric setting)

Recall: anything revealed by $O(C)$ is revealed by any $C' \equiv C$

- take $(\text{Setup}, \text{Enc}, \text{Dec})$ your favourite SKE scheme
Why is iO useful (2)

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

Example: black box decryption (symmetric setting)

Recall: anything revealed by $\mathcal{O}(C)$ is revealed by any $C' \equiv C$

- take $(\text{Setup}, \text{Enc}, \text{Dec})$ your favourite SKE scheme
- Setup':
  - $sk_1 \leftarrow \text{Setup}(), \ sk_2 \leftarrow \text{Setup}()$
  - output $sk' = (sk_1, sk_2)$
Why is iO useful (2)

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

**Example:** black box decryption (symmetric setting)

Recall: anything revealed by $O(C)$ is revealed by any $C' \equiv C$

- take $(\text{Setup,Enc,Dec})$ your favourite SKE scheme
- **Setup’:**
  - $sk_1 \leftarrow \text{Setup}()$, $sk_2 \leftarrow \text{Setup}()$
  - output $sk' = (sk_1, sk_2)$
- **Enc’($m, sk'$):**
  - $c_1 \leftarrow \text{Enc}(m, sk_1)$, $c_2 \leftarrow \text{Enc}(m, sk_2)$
  - output $(c_1, c_2)$
Why is iO useful (2)

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

Example: black box decryption (symmetric setting)

Recall: anything revealed by $O(C)$ is revealed by any $C' \equiv C$

- take $(\text{Setup}, \text{Enc}, \text{Dec})$ your favourite SKE scheme
- $\text{Setup}':$
  - $sk_1 \leftarrow \text{Setup}(), \ sk_2 \leftarrow \text{Setup}()$
  - output $sk' = (sk_1, sk_2)$
- $\text{Enc}'(m, sk')$:  
  - $c_1 \leftarrow \text{Enc}(m, sk_1), \ c_2 \leftarrow \text{Enc}(m, sk_2)$
  - output $(c_1, c_2)$
- $\text{Dec}'$:  
  - $C_1(c_1, c_2) = \text{Dec}(sk_1, c_1) \ (sk_1 \text{ hardcoded in } C_1)$
  - $C_2(c_1, c_2) = \text{Dec}(sk_2, c_2) \ (sk_2 \text{ hardcoded in } C_2)$
Why is iO useful (2)

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

Example: black box decryption (symmetric setting)
Recall: anything revealed by $\mathcal{O}(C)$ is revealed by any $C' \equiv C$

- take $(\text{Setup}, \text{Enc}, \text{Dec})$ your favourite SKE scheme
- $\text{Setup}'$:
  - $sk_1 \leftarrow \text{Setup}(), \ sk_2 \leftarrow \text{Setup}()$
  - output $sk' = (sk_1, sk_2)$
- $\text{Enc}'(m, sk')$:
  - $c_1 \leftarrow \text{Enc}(m, sk_1), \ c_2 \leftarrow \text{Enc}(m, sk_2)$
  - output $(c_1, c_2)$
- $\text{Dec}'$:
  - $C_1(c_1, c_2) = \text{Dec}(sk_1, c_1)$ ($sk_1$ hardcoded in $C_1$)
  - $C_2(c_1, c_2) = \text{Dec}(sk_2, c_2)$ ($sk_2$ hardcoded in $C_2$)

$C_1 \equiv C_2 \Rightarrow C = \mathcal{O}(C_1) \simeq_c \mathcal{O}(C_2)$ does not reveal $sk_1$ or $sk_2$
Outline of the talk

1. Definition

2. Candidates
   - Security
   - Practicability

3. Example of construction of an obfuscator
We only have candidate iO
(no construction based on standard cryptographic assumptions)
Three main categories

- Branching program obfuscators

- Circuit obfuscators

- Obfuscation via functional encryption
Three main categories

- Branching program obfuscators
  - needs bootstrapping via fully homomorphic encryption

- Circuit obfuscators
  - no need for bootstrapping
  - security proofs in some idealized models
  - but many attacks

- Obfuscation via functional encryption
  - try to find the weakest primitive implying iO
  - some attacks and impossibility results (not well understood yet)
  - most of them are not instantiable
Three main categories

- Branching program obfuscators
  - needs bootstrapping via fully homomorphic encryption
  - security proofs in some idealized models ...
  - ... but many attacks
Three main categories

- **Branching program obfuscators**
  - needs bootstrapping via fully homomorphic encryption
  - security proofs of VBB in some idealized models . . .
  - . . . but many attacks
Three main categories

- **Branching program obfuscators**
  - needs bootstrapping via fully homomorphic encryption
  - security proofs of VBB in some idealized models . . .
  - . . . but many attacks

- **Circuit obfuscators**
  - no need for bootstrapping
Three main categories

- **Branching program obfuscators**
  - needs bootstrapping via fully homomorphic encryption
  - security proofs of VBB in some idealized models . . .
  - … but many attacks

- **Circuit obfuscators**
  - no need for bootstrapping
  - security proofs of VBB in some idealized models . . .
  - … but many attacks
Three main categories

- **Branching program obfuscators**
  - needs bootstrapping via fully homomorphic encryption
  - security proofs of VBB in some idealized models ...
  - ... but many attacks

- **Circuit obfuscators**
  - no need for bootstrapping
  - security proofs of VBB in some idealized models ...
  - ... but many attacks

- **Obfuscation via functional encryption**
  - try to find the weakest primitive implying iO
  - some attacks and impossibility results (not well understood yet)
  - most of them are not instantiable
Branching program and circuit obfuscators use \textit{multilinear maps}. All the candidate multilinear maps we know suffer from weaknesses.
Branching program and circuit obfuscators use multilinear maps. All the candidate multilinear maps we know suffer from weaknesses.

<table>
<thead>
<tr>
<th></th>
<th>number of candidates</th>
<th>still standing classically</th>
<th>still standing quantumly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branching program iO</td>
<td>$\approx 20$</td>
<td>$\approx 10$</td>
<td>3</td>
</tr>
<tr>
<td>Circuit iO</td>
<td>$\approx 8$</td>
<td>$\approx 8$</td>
<td>0</td>
</tr>
</tbody>
</table>

All attacks rely on the underlying multilinear map.
Restricted functionalities

VBB obfuscators based on RLWE for

A. Pellet-Mary

Program obfuscation

PhD seminar
Restricted functionalities

VBB obfuscators based on RLWE for

- point functions

\[ f_y(x) = 1 \text{ iff } x = y \]
Restricted functionalities

VBB obfuscators based on RLWE for

- point functions
  \[ f_y(x) = 1 \text{ iff } x = y \]

- conjunctions
  \[ f(x_1, \ldots, x_n) = \bigwedge_{i \in I} y_i \quad (\text{with } y_i = x_i \text{ or } \bar{x}_i) \]
Restricted functionalities

VBB obfuscators based on RLWE for

- point functions
  \[ f_y(x) = 1 \text{ iff } x = y \]

- conjunctions
  \[ f(x_1, \ldots, x_n) = \bigwedge_{i \in I} y_i \quad (\text{with } y_i = x_i \text{ or } \bar{x}_i) \]

- compute-and-compare functions
  \[ f_{g,y}(x) = 1 \text{ iff } g(x) = y \]
### Practicability

<table>
<thead>
<tr>
<th>function obfuscated</th>
<th>security parameter $\lambda$</th>
<th>size obfuscated program</th>
<th>obfuscation time</th>
<th>evaluation time</th>
<th>security assumption</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>128</td>
<td>18 700 TB</td>
<td></td>
<td>$10^{10}$ mults of $10^8$ bits integers</td>
<td>none -</td>
<td>[YLX17]</td>
</tr>
<tr>
<td>one-round key-exchange with 4 users</td>
<td>52</td>
<td>4.8 GB</td>
<td>2h20</td>
<td>$\leq 1$ min</td>
<td>none -</td>
<td>[CP18]</td>
</tr>
<tr>
<td>$A_1^{x_1} \times \cdots \times A_{20}^{x_{20}}$</td>
<td>80</td>
<td>80 h</td>
<td>25 min</td>
<td>none</td>
<td>[HHSSD17]</td>
<td></td>
</tr>
<tr>
<td>$x_1 \land \overline{x}<em>4 \land \cdots \land x</em>{32}$</td>
<td>53</td>
<td>6.2 min</td>
<td>32 ms</td>
<td>entropic RLWE</td>
<td>[CDCG⁺18]</td>
<td></td>
</tr>
<tr>
<td>$x_1 \land \overline{x}<em>4 \land \cdots \land x</em>{64}$</td>
<td>73</td>
<td>6.7h</td>
<td>2.4s</td>
<td>entropic RLWE</td>
<td>[CDCG⁺18]</td>
<td></td>
</tr>
</tbody>
</table>
Outline of the talk

1. Definition

2. Candidates
   - Security
   - Practicability

3. Example of construction of an obfuscator
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

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^{\ell}$ (where $r$ is the size of the input).

<table>
<thead>
<tr>
<th>$M_0$</th>
<th>$M_{1,1}$</th>
<th>$M_{2,1}$</th>
<th>$M_{3,1}$</th>
<th>$M_{4,1}$</th>
<th>$M_{5,1}$</th>
<th>$M_{6,1}$</th>
<th>$M_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{1,0}$</td>
<td>$M_{2,0}$</td>
<td>$M_{3,0}$</td>
<td>$M_{4,0}$</td>
<td>$M_{5,0}$</td>
<td>$M_{6,0}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Evaluation on $x = 0\ 1\ 1$
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

\[
\begin{array}{cccccccc}
M_0 & M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & M_7 \\
M_{1,0} & M_{1,1} & M_{2,0} & M_{2,1} & M_{3,0} & M_{3,1} & M_{4,0} & M_{4,1} \\
& M_{5,0} & M_{5,1} & M_{6,0} & M_{6,1} & M_7 \\
\end{array}
\]

Evaluation on $x = 0 \ 1 \ 1$
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

![BP Diagram]

Evaluation on $x = 0 \ 1 \ 1$
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

**Evaluation on**

$x = 0 \ 1 \ 1$

\[\uparrow\]

\[
\begin{array}{cccccccc}
M_0 & \times & M_{1,1} & \times & M_{2,1} & M_{3,1} & M_{4,1} & M_{5,1} & M_{6,1} & M_7 \\
M_{1,0} & \times & M_{2,0} & M_{3,0} & M_{4,0} & M_{5,0} & M_{6,0} & & \\
\end{array}
\]
Branching programs

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

A Branching Program (BP) is a collection of

- 2\(\ell\) 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 \(\text{inp} \in \{1, \ldots, r\}^\ell\) (where \(r\) is the size of the input).

\[
\begin{array}{cccccc}
M_0 & \times & M_{1,1} & \times & M_{2,1} & \times & M_{3,1} & M_{4,1} & M_{5,1} & M_{6,1} & M_7 \\
M_{1,0} & \times & M_{2,0} & \times & M_{3,0} & & M_{4,0} & M_{5,0} & M_{6,0} & & \\
\end{array}
\]

Evaluation on \(x = 0 \ 1 \ 1\)
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

Evaluation on $x = 0 \ 1 \ 1$
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

![Branching Program Diagram]

Evaluation on $x = 0 1 1$
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

Evaluation on $x = 0 \ 1 \ 1$
Branching programs

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

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

Evaluation on $x = 0 \ 1 \ 1$
A branching program represents a function (cf Turing machine, or circuit).

A Branching Program (BP) is a collection of

- $2\ell$ 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 $\text{inp} \in \{1, \ldots, r\}^\ell$ (where $r$ is the size of the input).

\[
\begin{array}{cccccccc}
\times & x_1 & x_1 & x_2 & x_1 & x_3 & x_2 \\
M_0 & M_{1,1} & M_{2,1} & M_{3,1} & M_{4,1} & M_{5,1} & M_{6,1} & M_7 \\
M_{1,0} & M_{2,0} & M_{3,0} & M_{4,0} & M_{5,0} & M_{6,0}
\end{array}
\]

Evaluation on $x = 0 \ 1 \ 1$

$\text{BP}$

$M_7 = 0 \rightarrow 0$

$\neq 0 \rightarrow 1$
Cryptographic multilinear maps

**Definition: \( \kappa \)-multilinear map**

Different levels of encodings, from 1 to \( \kappa \).
Write \( \text{Enc}(a, i) \) 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 \( a = 0 \).
Simple obfuscator

[GGH^{+}13, BR14, BGK^{+}14, PST14, AGIS14, MSW14, GMM^{+}16]

- **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 a multilinear map
- **Output**: The encoded matrices and vectors

\[
\begin{align*}
A_0 & \\
A_{1,0} & A_{2,0} & A_{3,0} \\
A_{1,1} & A_{2,1} & A_{3,1} \\
\end{align*}
\]
Simple obfuscator

[GGH\(^+\)13, BR14, BGK\(^+\)14, PST14, AGIS14, MSW14, GMM\(^+\)16]

- **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 a multilinear map
- **Output**: The encoded matrices and vectors

\[
\begin{align*}
B_{1,0} & \quad B_{2,0} & \quad B_{3,0} \\
A_{1,0} & \quad A_{2,0} & \quad A_{3,0} \\
B_{1,1} & \quad B_{2,1} & \quad B_{3,1} \\
A_{1,1} & \quad A_{2,1} & \quad A_{3,1} \\
\end{align*}
\]

\[0 \quad A_0 \quad A_4 \quad \star \]
Simple obfuscator

[GGH$^{+}$13, BR14, BGK$^{+}$14, PST14, AGIS14, MSW14, GMM$^{+}$16]

- **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 a multilinear map
- **Output:** The encoded matrices and vectors

\[
\begin{align*}
R_1^{-1} & \ A_{1,1} \ R_2 \\
R_2^{-1} & \ A_{2,1} \ R_3 \\
R_3^{-1} & \ A_{3,1} \ R_4 \\
A_0 & \ R_1 \\
R_4^{-1} & \ A_4 \\
R_1^{-1} & \ A_{1,0} \ R_2 \\
R_2^{-1} & \ A_{2,0} \ R_3 \\
R_3^{-1} & \ A_{3,0} \ R_4
\end{align*}
\]
Simple obfuscator

[GGH$^{+}$13, BR14, BGK$^{+}$14, PST14, AGIS14, MSW14, GMM$^{+}$16]

- **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 a multilinear map**
- **Output:** The encoded matrices and vectors

\[
\begin{align*}
\alpha_{1,1} \times A_{1,1} & \quad \alpha_{2,1} \times A_{2,1} & \quad \alpha_{3,1} \times A_{3,1} \\
\alpha_{1,0} \times A_{1,0} & \quad \alpha_{2,0} \times A_{2,0} & \quad \alpha_{3,0} \times A_{3,0} \\
A_0 & \quad & A_4
\end{align*}
\]
Simple obfuscator

\[ \text{Input: A branching program} \]
\[ \text{Randomize the branching program} \]
\[ \quad \rightarrow \text{Add random diagonal blocks} \]
\[ \quad \rightarrow \text{Killian’s randomization} \]
\[ \quad \rightarrow \text{Multiply by random (non zero) bundling scalars} \]
\[ \text{Encode the matrices using a multilinear map} \]
\[ \text{Output: The encoded matrices and vectors} \]

\[ \tilde{A}_0, \tilde{A}_{1,0}, \tilde{A}_{1,1}, \tilde{A}_{2,0}, \tilde{A}_{2,1}, \tilde{A}_{3,0}, \tilde{A}_{3,1}, \tilde{A}_4 \]
Simple obfuscator

[GGH⁺13, BR14, BGK⁺14, PST14, AGIS14, MSW14, GMM⁺16]

- **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 a multilinear map
- **Output**: The encoded matrices and vectors

\[
\begin{align*}
\text{Enc}(\overline{A_0}) & \quad \text{Enc}(\overline{A_{1,1}}) & \quad \text{Enc}(\overline{A_{3,1}}) \\
\text{Enc}(\overline{A_{1,0}}) & \quad \text{Enc}(\overline{A_{2,0}}) & \quad \text{Enc}(\overline{A_{3,0}}) \\
\text{Enc}(\overline{A_{4}}) &
\end{align*}
\]
Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\tilde{A}_{i,b}$ after randomisation
- $\hat{A}_{i,b}$ after encoding with a multilinear map (output of the iO)

\[ \hat{A}_0, \hat{A}_{1,1}, \hat{A}_{2,1}, \hat{A}_{3,1}, \hat{A}_{1,0}, \hat{A}_{2,0}, \hat{A}_{3,0}, \hat{A}_{4} \]
Mixed-input attack

Notations
- $A_{i,b}$ input branching program
- $\tilde{A}_{i,b}$ after randomisation
- $\hat{A}_{i,b}$ after encoding with a multilinear map (output of the iO)

\[
\begin{align*}
\hat{A}_0 & \quad \hat{A}_{1,1} & \quad \hat{A}_{2,1} & \quad \hat{A}_{3,1} \\
\quad \quad \hat{A}_{1,0} & \quad \hat{A}_{2,0} & \quad \hat{A}_{3,0} \\
1 & \quad 1 & \quad 1
\end{align*}
\]
Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\tilde{A}_{i,b}$ after randomisation
- $\hat{A}_{i,b}$ after encoding with a multilinear map (output of the iO)

\[ \hat{A}_0 \]

\[ \hat{A}_{1,0} \]
\[ x_1 \]
\[ 1 \]

\[ \hat{A}_{1,1} \]

\[ \hat{A}_{2,1} \]

\[ \hat{A}_{3,1} \]

\[ \hat{A}_{2,0} \]
\[ x_2 \]
\[ 0 \]

\[ \hat{A}_{3,0} \]

\[ x_1 \]
\[ 1 \]

\[ \hat{A}_4 \]
Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\tilde{A}_{i,b}$ after randomisation
- $\hat{A}_{i,b}$ after encoding with a multilinear map (output of the iO)

\[
\begin{align*}
\hat{A}_0 & \quad \hat{A}_{1,1} & \quad \hat{A}_{2,1} & \quad \hat{A}_{3,1} \\
\quad \hat{A}_{1,0} & \quad \hat{A}_{2,0} & \quad \hat{A}_{3,0} \\
\quad x_1 & \quad x_2 & \quad x_1 \\
\quad 0 & \quad 1 & \quad 0
\end{align*}
\]
Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\tilde{A}_{i,b}$ after randomisation
- $\hat{A}_{i,b}$ after encoding with a multilinear map (output of the iO)

\[ \hat{A}_0 \]

\[ \hat{A}_1,1 \quad \hat{A}_2,1 \quad \hat{A}_3,1 \]

\[ \hat{A}_1,0 \quad \hat{A}_2,0 \quad \hat{A}_3,0 \]

\[ x_1 \quad x_2 \quad x_1 \]

\[ 0 \quad 0 \quad 0 \]
Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $A_{i,b}$ after randomisation
- $A_{i,b}$ after encoding with a multilinear map (output of the iO)

\[\hat{A}_0\]

\[\hat{A}_{1,1}\] \[\hat{A}_{2,1}\] \[\hat{A}_{3,1}\] \[\hat{A}_4\]

\[\hat{A}_{1,0}\] \[\hat{A}_{2,0}\] \[\hat{A}_{3,0}\]

\[x_1\] \[x_2\] \[x_1\]

0 \[0\] \[1\]
Mixed-input attack

Notations

- $A_{i,b}$ input branching program
- $\tilde{A}_{i,b}$ after randomisation
- $\hat{A}_{i,b}$ after encoding with a multilinear map (output of the iO)

\[
\begin{align*}
\text{Enc}(\tilde{A}_{1,1}, 1) & \quad \text{Enc}(\tilde{A}_{2,1}, 1) & \quad \text{Enc}(\tilde{A}_{3,1}, 1) \\
\text{Enc}(\tilde{A}_{0,1}) & & & \quad \text{Enc}(\tilde{A}_{4,1}) \\
\text{Enc}(\tilde{A}_{1,0}, 1) & \quad \text{Enc}(\tilde{A}_{2,0}, 1) & \quad \text{Enc}(\tilde{A}_{3,0}, 1) \\
\end{align*}
\]

\[
\begin{align*}
& x_1 \\
& 0 \\
& x_2 \\
& 0 \\
& x_1 \\
& 1 \\
\end{align*}
\]
Preventing mixed-input attacks

- In the randomization phase ⇒ not in this talk \([\text{GGH}^+_{13}, \text{BR14}]\)
- Using the mmap ⇒ straddling set system
  \([\text{BGK}^+_{14}, \text{PST14}, \text{AGIS14}, \text{MSW14}, \text{GMM}^+_{16}]\)
Preventing mixed-input attacks

- In the randomization phase ⇒ not in this talk [GGH\textsuperscript{+13}, BR\textsuperscript{14}]
- Using the mmap ⇒ straddling set system
  [BGK\textsuperscript{+14}, PST\textsuperscript{14}, AG\textsuperscript{14}, MSW\textsuperscript{14}, GMM\textsuperscript{+16}]

**Mmap degree:** $\kappa = 5$

\[
\begin{align*}
\text{Enc}(\widetilde{A}_0, 1) & \quad \text{Enc}(\widetilde{A}_1, 1) & \quad \text{Enc}(\widetilde{A}_2, 1) & \quad \text{Enc}(\widetilde{A}_3, 1)
\end{align*}
\]

\[
\begin{align*}
\text{Enc}(\widetilde{A}_1, 0) & \quad \text{Enc}(\widetilde{A}_2, 0) & \quad \text{Enc}(\widetilde{A}_3, 0) & \quad \text{Enc}(\widetilde{A}_4, 1)
\end{align*}
\]

$\times_1 \quad \times_2 \quad \times_1$

Total level: 7 ⇒ cannot zero-test
Preventing mixed-input attacks

- In the randomization phase $$\Rightarrow$$ not in this talk [GGH$^+$13, BR14]
- Using the mmap $$\Rightarrow$$ straddling set system
  [BGK$^+$14, PST14, AGIS14, MSW14, GMM$^+$16]

Mmap degree: $$\kappa = 6$$
Preventing mixed-input attacks

- In the randomization phase ⇒ not in this talk [GGH^{+}13, BR14]
- Using the mmap ⇒ straddling set system [BGK^{+}14, PST14, AGIS14, MSW14, GMM^{+}16]

**Mmap degree**: \( \kappa = 6 \)

\[
\begin{align*}
\text{Enc}(\overline{A}_1, 1) & \quad \text{Enc}(\overline{A}_2, 1) & \quad \text{Enc}(\overline{A}_3, 2) \\
\text{Enc}(\overline{A}_0, 1) & \quad & \\
\text{Enc}(\overline{A}_1, 2) & \quad \text{Enc}(\overline{A}_2, 1) & \quad \text{Enc}(\overline{A}_3, 1) \\
X_1 & \quad X_2 & \quad X_1 \\
\text{0} & \quad \text{0} & \quad \text{1}
\end{align*}
\]

\text{Total level: 7} ⇒ cannot zero-test
Preventing mixed-input attacks

- In the randomization phase ⇒ not in this talk \cite{GGH13, BR14}
- Using the mmap ⇒ straddling set system
  \cite{BGK+14, PST14, AGIS14, MSW14, GMM+16}

**Mmap degree:** $\kappa = 6$

\[
\begin{align*}
\text{Enc}(\overline{A_0}, 1) & \quad \text{Enc}(\overline{A_1, 1}, 1) & \quad \text{Enc}(\overline{A_2, 1}, 1) & \quad \text{Enc}(\overline{A_3, 1}, 2) \\
\text{Enc}(\overline{A_1, 0}, 2) & \quad \text{Enc}(\overline{A_2, 0}, 1) & \quad \text{Enc}(\overline{A_3, 0}, 1) & \quad \text{Enc}(\overline{A_4}, 1)
\end{align*}
\]

$X_1 = 0 \quad X_2 = 0 \quad X_1 = 1$

**Total level:** 7 ⇒ cannot zero-test
What to remember

+ iO would be very useful (at least for theory) ...
What to remember

+ iO would be very useful (at least for theory) …

− … but no constructions from standard assumptions yet
What to remember

+ iO would be very useful (at least for theory) ...

− ... but no constructions from standard assumptions yet

− ... even insecure constructions are very inefficient
What to remember

+ iO would be very useful (at least for theory) ...

− ... but no constructions from standard assumptions yet

− ... even insecure constructions are very inefficient

+ maybe for restricted class of functions efficiency and security are possible
What to remember

+ iO would be very useful (at least for theory) ...

− ... but no constructions from standard assumptions yet

− ... even insecure constructions are very inefficient

+ maybe for restricted class of functions efficiency and security are possible

Questions?
Benny Applebaum and Zvika Brakerski.
Obfuscating circuits via composite-order graded encoding.

Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee.
Cryptanalysis of indistinguishability obfuscations of circuits over ggh13.

Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai.
Optimizing obfuscation: Avoiding barrington’s theorem.

On the (im) possibility of obfuscating programs.

Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai.
Protecting obfuscation against algebraic attacks.

Zvika Brakerski and Guy N. Rothblum.
Virtual black-box obfuscation for all circuits via generic graded encoding.
References II

David Bruce Cousins, Giovanni Di Crescenzo, Kamil Doruk Gür, Kevin King, Yuriy Polyakov, Kurt Rohloff, Gerard W Ryan, and Erkay Savas.
Implementing conjunction obfuscation under entropic ring lwe.

Yilei Chen, Craig Gentry, and Shai Halevi.
Cryptanalyses of candidate branching program obfuscators.

Jung Hee Cheon, Minki Hhan, Jiseung Kim, and Changmin Lee.
Cryptanalyses of branching program obfuscations over ggh13 multilinear map from the ntru problem.

Jean-Sébastien Coron and Hilder VL Pereira.
On kilian’s randomization of multilinear map encodings.
ePrint, 2018.

Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee.
Obfuscation from low noise multilinear maps.

Rex Fernando, Peter MR Rasmussen, and Amit Sahai.
Preventing clt attacks on obfuscation with linear overhead.

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


Dingfeng Ye, Peng Liu, and Jun Xu. How fast can we obfuscate using ideal graded encoding schemes. ePrint, 2017.
## History (GGH13-based branching program obfuscation)

<table>
<thead>
<tr>
<th>Constructions</th>
<th>[BR14]</th>
<th>[BGK(^+)14, PST14]</th>
<th>[AGIS14, MSW14]</th>
<th>[GMM(^+)16]</th>
<th>[FRS17]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GGH(^+)13]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>2014</td>
<td>2015</td>
<td>2016</td>
<td>2017</td>
<td>2018</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attacks</th>
<th>[MSZ16]</th>
<th>[CGH17]</th>
<th>[CHKL18]</th>
<th>[Pel18]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MSZ16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[CGH17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ADGM17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Pe18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Rightarrow \]\text{prevented by [FRS17]}
### History (GGH13-based branching program obfuscation)

#### Constructions

<table>
<thead>
<tr>
<th>Year</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>[GGH+13]</td>
</tr>
<tr>
<td>2014</td>
<td>[BR14]</td>
</tr>
<tr>
<td>2015</td>
<td>[BGK+14, PST14]</td>
</tr>
<tr>
<td>2016</td>
<td>[AGIS14, MSW14]</td>
</tr>
<tr>
<td>2017</td>
<td>[GMM+16]</td>
</tr>
<tr>
<td>2018</td>
<td>[FRS17]</td>
</tr>
</tbody>
</table>

#### Attacks

<table>
<thead>
<tr>
<th>Year</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>[MSZ16]</td>
</tr>
<tr>
<td>2017</td>
<td>[CGH17]</td>
</tr>
<tr>
<td>2018</td>
<td>[CHKL18]</td>
</tr>
<tr>
<td>2018</td>
<td>[Pel18]</td>
</tr>
</tbody>
</table>

[MSZ16]: all constructions without diagonal blocks
History (GGH13-based branching program obfuscation)

Constructions

<table>
<thead>
<tr>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GGH+13]</td>
<td>[BR14]</td>
<td>[BGK+14, PST14]</td>
<td>[AGIS14, MSW14]</td>
<td>[GMM+16]</td>
<td>[FRS17]</td>
</tr>
</tbody>
</table>

Attacks

<table>
<thead>
<tr>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MSZ16]</td>
<td>[CGH17]</td>
<td>[CHKL18]</td>
<td>[ADGM17]</td>
<td>[Pel18]</td>
<td></td>
</tr>
</tbody>
</table>

[MSZ16]: all constructions without diagonal blocks

[ADGM17]: idem MSZ but from circuits
## History (GGH13-based branching program obfuscation)

<table>
<thead>
<tr>
<th>Constructions</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGH+13</td>
<td></td>
<td>BR14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BGK+14, PST14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AGIS14, MSW14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMM+16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRS17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attacks</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSZ16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGH17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHKL18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pel18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **[MSZ16]**: all constructions without diagonal blocks
- **[ADGM17]**: idem MSZ but from circuits
- **[CGH17]**: use input-partitionability (cf CLT13)
History (GGH13-based branching program obfuscation)

Constructions

<table>
<thead>
<tr>
<th>Year</th>
<th>[GGH+13]</th>
<th>[BR14]</th>
<th>[BGK+14, PST14]</th>
<th>[AGIS14, MSW14]</th>
<th>[GMM+16]</th>
<th>[FRS17]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>[GGH+13]</td>
<td>[BR14]</td>
<td>[BGK+14, PST14]</td>
<td>[AGIS14, MSW14]</td>
<td>[GMM+16]</td>
<td>[FRS17]</td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td>[GMM+16]</td>
<td>[FRS17]</td>
<td>[CHKL18]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>[FRS17]</td>
<td>[CHKL18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>[FRS17]</td>
<td>[CHKL18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Attacks

<table>
<thead>
<tr>
<th>Year</th>
<th>[MSZ16]</th>
<th>[CGH17]</th>
<th>[CHKL18]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td>[MSZ16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>[CGH17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td>[ADGM17]</td>
<td>[CGH17]</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>[CHKL18]</td>
<td>[ADGM17]</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>[Pe18]</td>
<td>[CHKL18]</td>
<td>[Pe18]</td>
</tr>
</tbody>
</table>

[MSZ16]: all constructions without diagonal blocks
[ADGM17]: idem MSZ but from circuits
[CGH17]: use input-partitionability (cf CLT13) ⇒ prevented by [FRS17]
### History (GGH13-based branching program obfuscation)

<table>
<thead>
<tr>
<th>Constructions</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GGH$^+$13]</td>
<td></td>
<td>[BR14]</td>
<td></td>
<td>[GMM$^+$16]</td>
<td></td>
<td>[FRS17]</td>
</tr>
<tr>
<td>[BGK$^+$14, PST14]</td>
<td></td>
<td>[AGIS14, MSW14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[MSZ16]</td>
<td>[CGH17]</td>
<td>[CHKL18]</td>
<td>[Pel18]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Constructions

- **[GGH$^+$13]**: all constructions without diagonal blocks
- **[BGK$^+$14, PST14]**: idem MSZ but from circuits
- **[AGIS14, MSW14]**: use input-partitionability (cf CLT13) \(\Rightarrow\) prevented by [FRS17]
- **[MSZ16]**: all constructions without diagonal blocks
- **[CGH17]**: use input-partitionability (cf CLT13) \(\Rightarrow\) prevented by [FRS17]
- **[CHKL18]**: NTRU attack for specific choices of parameters

#### Attacks

- **[FRS17]**: quantum attack
### History (GGH13-based branching program obfuscation)

<table>
<thead>
<tr>
<th>Year</th>
<th>Constructions</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>[GGH^{+13}] [BR14] [BGK^{+14}, PST14] [AGIS14, MSW14]</td>
<td>[MSZ16] [CGH17] [ADGM17] [CHKL18]</td>
</tr>
<tr>
<td>2014</td>
<td>[GMM^{+16}] [FRS17]</td>
<td>[Pel18]</td>
</tr>
<tr>
<td>2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \[MSZ16\]: 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

<table>
<thead>
<tr>
<th>Attacks</th>
<th>iOs</th>
<th>[GGH$^{+13}$]</th>
<th>[BR14, BGK$^{+14}$, PST14, AGIS14, MSW14]</th>
<th>[GMM$^{+16}$]</th>
<th>Circuit obfuscators</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MSZ16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[CGH17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fully broken</td>
</tr>
<tr>
<td>[CHKL18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>input-partitionable</td>
</tr>
<tr>
<td>[Pel18]</td>
<td>some parameters</td>
<td></td>
<td>some parameters</td>
<td>quantum</td>
<td>quantum</td>
</tr>
</tbody>
</table>

Still standing classically:
- [GGH$^{+13}$]+[FRS17]
- [GMM$^{+16}$]
- all circuit obfuscators

Still standing quantumly:
- [GGH$^{+13}$]+[FRS17]