Affine Determinant Programs: A New Approach to Obfuscation

James Bartusek (Princeton → UC Berkeley)
Yuval Ishai (Technion)
Aayush Jain (UCLA)
Fermi Ma (Princeton)
Amit Sahai (UCLA)
Mark Zhandry (Princeton + NTT Research)
Program Obfuscation

[BGIRSVY01]

• Scramble a program to hide implementation details

• Many possible security notions:
  • Virtual black box (VBB)
  • Indistinguishability obfuscation (iO)
Why did obfuscation ever need multilinear maps?
Why did obfuscation ever need multilinear maps?

A crash course in GGHRSW-style obfuscation

**Bootstrapping Theorem [GGHRSW]**

- iO for NC1
- (assuming FHE)
- iO for all circuits

**Takeaway:** it suffices to consider NC1.
How do we build iO for NC1?

log-depth circuit $\mathcal{C}$ \quad \rightarrow \quad \text{Barrington’s Thm.} \quad \rightarrow \quad \text{constant-width deterministic branching program } BP
How do we build iO for NC1?

log-depth circuit $\mathcal{C}$  \hspace{1cm} \text{Barrington’s Thm.} \hspace{1cm} \text{constant-width deterministic branching program } BP

Matrix branching program

$M_{1,0}$ \hspace{1cm} $M_{2,0}$ \hspace{1cm} $M_{3,0}$ \hspace{1cm} $M_{4,0}$

$M_{1,1}$ \hspace{1cm} $M_{2,1}$ \hspace{1cm} $M_{3,1}$ \hspace{1cm} $M_{4,1}$

$x_0$ \hspace{1cm} $x_1$ \hspace{1cm} $x_0$ \hspace{1cm} $x_1$
How do we build iO for NC1?

log-depth circuit $\mathcal{C}$  \[ \xrightarrow{\text{Barrington’s Thm.}} \] constant-width deterministic branching program $BP$

\[
\begin{align*}
x &= 01 \\
M_{1,0} & \quad M_{2,0} & \quad M_{3,0} & \quad M_{4,0} \\
M_{1,1} & \quad M_{2,1} & \quad M_{3,1} & \quad M_{4,1} \end{align*}
\]

$M_{0}$

$M_{1,1}$

$M_{2,1}$

$M_{3,1}$

$M_{4,1}$

$x = 01$

How do we build iO for NC1?
How do we build iO for NC1?

log-depth circuit $C$ \[ \xrightarrow{\text{Barrington's Thm.}} \] constant-width deterministic branching program $BP$

$x = 01$

Evaluation: $C(x) = 1$ if $\boxed{M_{1,0}} \times \boxed{M_{2,1}} \times \boxed{M_{3,0}} \times \boxed{M_{4,1}} = \boxed{F}$
What does the matrix branching program representation buy us?

“one-time” security by Kilian randomization

\[ x = 01 \]

\[
\begin{align*}
M_{1,0} & \quad M_{2,0} & \quad M_{3,0} & \quad M_{4,0} \\
M_{1,1} & \quad M_{2,1} & \quad M_{3,1} & \quad M_{4,1} \\
x_0 & \quad x_1 & \quad x_0 & \quad x_1
\end{align*}
\]

Evaluation: \( C(x) = 1 \) if \( M_{1,0} \times M_{2,1} \times M_{3,0} \times M_{4,1} = F \)
What does the matrix branching program representation buy us?

“one-time” security by Kilian randomization

Sample random matrices

\[ R_1, R_2, R_3 \]

\[
\begin{align*}
M_{1,0} \cdot R_1 & \quad R_1^{-1} \cdot M_{2,0} \cdot R_2 & \quad R_2^{-1} \cdot M_{3,0} \cdot R_3 & \quad R_3^{-1} \cdot M_{4,0} \\
M_{1,1} \cdot R_1 & \quad R_1^{-1} \cdot M_{2,1} \cdot R_2 & \quad R_2^{-1} \cdot M_{3,1} \cdot R_3 & \quad R_3^{-1} \cdot M_{4,1} \\
x_0 & \quad x_1 & \quad x_0 & \quad x_1
\end{align*}
\]
What does the matrix branching program representation buy us?

“one-time” security by Kilian randomization

\[
\hat{M}_{1,0}, \hat{M}_{2,0}, \hat{M}_{3,0}, \hat{M}_{4,0}, \hat{M}_{1,1}, \hat{M}_{2,1}, \hat{M}_{3,0}, \hat{M}_{4,1}
\]

\[
x_0, x_1, x_0, x_1
\]

(Sample random matrices \( R_1, R_2, R_3 \))

(\( \hat{M} \) denotes \( M \) after applying Kilian randomization)
Kilian’s Statistical Simulation Lemma:

Can statistically simulate \( \hat{M}_{1,0}, \hat{M}_{2,1}, \hat{M}_{3,0}, \hat{M}_{4,1} \) given their product.

\[ x = 01 \]

\[ \hat{M}_{1,0} \quad \hat{M}_{2,0} \quad \hat{M}_{3,0} \quad \hat{M}_{4,0} \]

\[ \hat{M}_{1,1} \quad \hat{M}_{2,1} \quad \hat{M}_{3,0} \quad \hat{M}_{4,1} \]

\[ x_0 \quad x_1 \quad x_0 \quad x_1 \]

“grey matrices leak nothing beyond whether \( BP(x) = 0 \) or 1”
Kilian’s Statistical Simulation Lemma:

Can statistically simulate $\tilde{M}_{1,0}$, $\tilde{M}_{2,1}$, $\tilde{M}_{3,0}$, $\tilde{M}_{4,1}$ given their product.

Takeaway: Kilian-randomization yields “one-time” security.
Kilian’s Statistical Simulation Lemma:

Can statistically simulate \( \hat{M}_{1,0}, \hat{M}_{2,1}, \hat{M}_{3,0}, \hat{M}_{4,1} \) given their product.

**Takeaway:** Kilian-randomization yields “one-time” security.

Kilian-randomized matrix branching program \( \xrightarrow{\text{encode each matrix in multilinear map}} \text{Obf}(C) \)

“one-time” secure \( \xrightarrow{\text{“many-time” secure}} \)
Multilinear maps enforce input **consistency**; without them, “mixed-input” attacks can break security!

Example: \( \hat{M}_{1,0} \times \hat{M}_{2,0} \times \hat{M}_{3,0} \times \hat{M}_{4,0} \) is a mixed-input evaluation.
NC1 circuit $C$ \rightarrow \text{Barrington’s Thm.} \rightarrow \text{constant-width deterministic branching program $BP$} \rightarrow \text{Kilian-randomized matrix branching program} \rightarrow \text{encode in multilinear map} \rightarrow \text{Obf}(C)$

$[\text{GGHRSW}]$ approach to iO for NC1
Our goal: Avoid multilinear maps by using an alternative representation of $BP$. 

NC1 circuit $C$ \xrightarrow{\text{Barrington's Thm.}} \text{constant-width deterministic branching program } BP 

\xrightarrow{\text{Kilian-randomized matrix branching program}} \text{encode in multilinear map} 

$Obf(C)$
NC1 circuit $C$ \[\xrightarrow{\text{Barrington’s Thm.}}\] constant-width deterministic branching program $BP$ 

$\xrightarrow{[IK00]}$ affine determinant program* (ADP) 

$\xrightarrow{??}$ encode in multilinear map 

$Obf(C)$ 

*this notion appears in [IK97, IK00, IK02, AIK06].
Affine Determinant Programs (ADP)

Encode:

\[ f : \{0,1\}^n \to \{0,1\} \rightarrow A, B_1, \ldots, B_n \]

width \( w \) matrices over \( \mathbb{Z}_q \)
Affine Determinant Programs (ADP)

Encode:

\[ f: \{0,1\}^n \rightarrow \{0,1\} \rightarrow \begin{bmatrix} A \end{bmatrix}, \begin{bmatrix} B_1 \end{bmatrix}, \ldots, \begin{bmatrix} B_n \end{bmatrix} \]

Evaluate:

\[ M_x := A + \sum_{i \mid x_i = 1} B_i \]
Affine Determinant Programs (ADP)

Encode:

\[ f: \{0,1\}^n \rightarrow \{0,1\} \rightarrow A, B_1, \ldots, B_n \]

Evaluate:

\[
M_x := A + \sum_{i \mid x_i = 1} B_i
\]

\[ f(x) = 1 \quad \text{if} \quad det(M_x) = 0 \]

\[ f(x) = 0 \quad \text{if} \quad det(M_x) \neq 0 \]

\[ M_x \text{ rank deficient by 1 when } f(x) = 1 \]
Affine Determinant Programs (ADP)

Encode:

\[ f: \{0,1\}^n \rightarrow \{0,1\} \rightarrow A, B_1, \ldots, B_n \]

Evaluate:

\[
M_x := A + \sum_{i \mid x_i = 1} B_i
\]

\[ f(x) = 1 \iff \text{det}(M_x) = 0 \]

\[ f(x) = 0 \iff \text{det}(M_x) \neq 0 \]

Lemma 1 [IK00]: Any deterministic branching program can be written as a poly-size ADP.

\[ \text{rank deficient by 1 when } f(x) = 1 \]
**Affine Determinant Programs (ADP)**

**Encode:**

\[ f: \{0,1\}^n \to \{0,1\} \mapsto A, B_1, \ldots, B_n \]

**Evaluate:**

\[ M_x := A + \sum_{i \mid x_i = 1} B_i \]

\[ f(x) = 1 \iff \det(M_x) = 0 \]

\[ f(x) = 0 \iff \det(M_x) \neq 0 \]

---

**Lemma 1 [IK00]:** Any deterministic branching program can be written as a poly-size ADP.

**Lemma 2 [IK00]:** By left and right re-randomizing, ADPs can be made “one-time” secure.

\[ M_x \quad \text{rank deficient by 1} \quad \text{when } f(x) = 1 \]
Affine Determinant Programs (ADPs)

\[ A, B_1, \ldots, B_n \]

Matrix Branching Programs (MBPs)

\[ M_{1,0}, M_{2,0}, M_{3,0}, M_{4,0}, M_{1,1}, M_{2,1}, M_{3,1}, M_{4,1} \]

ADPs are an “additive” analogue of MBPs

- MBPs require multilinear maps to enforce input consistency.
- ADPs only read each input bit once!
Affine Determinant Programs (ADPs) | Matrix Branching Programs (MBPs)

\[ A, B_1, \ldots, B_n \]

\[ M_{1,0} \quad M_{2,0} \quad M_{3,0} \quad M_{4,0} \]

\[ M_{1,1} \quad M_{2,1} \quad M_{3,1} \quad M_{4,1} \]

ADPs are an “additive” analogue of MBPs

- MBPs require multilinear maps to enforce input consistency.
- ADPs only read each input bit once!

**Takeaway:** It seems plausible that we could build “many-time” secure ADPs without multilinear maps.
Until recently, all known ADPs were only “one-time” secure.

- “one-time” security: only release one evaluation of $A + \sum_{i \mid x_i=1} B_i$.
- “many-time” security (obfuscation): $A, B_1, \ldots, B_n$ can be public.
The rest of this talk:

- (if time permits) provably secure many-time secure ADP for conjunctions [BLMZ19]

- candidate many-time secure ADPs for NC1.
Conjunctions

Program has a hard-coded string $s = 11*0^*$. Accepts iff input matches on every 0/1 bits.

Example: $s = 11*0^*$

$$f_s(11000) = 1$$
$$f_s(11101) = 1$$
$$f_s(00010) = 0$$
$$f_s(01000) = 0$$
[BLMZ19] Obfuscation Construction:
On length $n$ string $s = 11^*0^*$, output

\[
\begin{array}{cccc}
A & B_1 & \ldots & B_n
\end{array}
\]

Evaluation: Input $x$ matches $s$ if

\[
det \left( A + \sum_{i | x_i = 1} B_i \right) = 0
\]
\( s = 11^{*0^*} \) of length \( n = 5 \), \( w = 2 \) wildcards, width \( w + 1 = 3 \) square matrices over \( \mathbb{Z}_q \).

\[
U \quad \text{secret random rank } w = 2 \text{ matrix}
\]
\[
s = 11^*0^* \text{ of length } n = 5, \ w = 2 \text{ wildcards, width } w + 1 = 3 \text{ square matrices over } \mathbb{Z}_q.
\]

\[
\begin{array}{ccc}
1 & 1 & 0 \\
B_1 & B_2 & B_4 \\
\text{random } u_1v_1^T & \text{random } u_2v_2^T & \text{random } u_4v_4^T \\
\end{array}
\]

secret random rank \( w = 2 \) matrix
$s = 11^*0^*$ of length $n = 5$, $w = 2$ wildcards, width $w + 1 = 3$ square matrices over $\mathbb{Z}_q$.

$U$ secret random rank $w = 2$ matrix

$$
\begin{align*}
B_1 &:= \begin{bmatrix} 1 \\ \end{bmatrix} &
& \text{random } u_1v_1^T \\
B_2 &:= \begin{bmatrix} 1 \\ \end{bmatrix} &
& \text{random } u_2v_2^T \\
B_3 &:= \begin{bmatrix} * \\ \end{bmatrix} &
& \text{random } u_3v_3^T \text{ with } \begin{bmatrix} U \\ \end{bmatrix} \\
B_4 &:= \begin{bmatrix} 0 \\ \end{bmatrix} &
& \text{random } u_4v_4^T \\
B_5 &:= \begin{bmatrix} * \\ \end{bmatrix} &
& \text{random } u_5v_5^T \text{ with } \begin{bmatrix} U \\ \end{bmatrix}
\end{align*}
$$
s = 11*0* of length n = 5, w = 2 wildcards, width w + 1 = 3 square matrices over \( \mathbb{Z}_q \).

\[
\begin{align*}
U & \quad \text{secret random rank } w = 2 \text{ matrix} \\
A & = U - \sum_{i \mid s_i=1} B_i \\
B_1 & = \begin{bmatrix} 1 \\ \end{bmatrix} \text{ random } u_1^T v_1 \\
B_2 & = \begin{bmatrix} 1 \\ \end{bmatrix} \text{ random } u_2^T v_2 \\
B_3 & = \begin{bmatrix} * \\ \end{bmatrix} \text{ random } u_3^T v_3 \text{ with } u_3 \leftarrow \text{col}(U) \\
B_4 & = \begin{bmatrix} 0 \\ \end{bmatrix} \text{ random } u_4^T v_4 \\
B_5 & = \begin{bmatrix} * \\ \end{bmatrix} \text{ random } u_5^T v_5 \text{ with } u_5 \leftarrow \text{col}(U)
\end{align*}
\]
\[ s = 11^*0^* \] of length \( n = 5 \), \( w = 2 \) wildcards, width \( w + 1 = 3 \) square matrices over \( \mathbb{Z}_q \).

- Secret random rank \( w = 2 \) matrix \( U \)

\[
A = U - B_1 - B_2
\]

- \( 1 \) random \( u_1v_1^T \)
- \( 1 \) random \( u_2v_2^T \)
- Random \( u_3v_3^T \) with \( U \)
- Random \( u_4v_4^T \)
- Random \( u_5v_5^T \) with \( U \)

\[ u_3 \leftarrow \text{col}(U) \]
\[ u_5 \leftarrow \text{col}(U) \]
\[ s = 11*0* \text{ of length } n = 5, \ w = 2 \text{ wildcards, width } w + 1 = 3 \text{ square matrices over } \mathbb{Z}_q. \]

**Secret Random Rank**

- **Matrix** \( U \)
- Rank \( w = 2 \)

**Evaluation**:

On input \( x = 11010 \)

\[
\begin{align*}
A &= U - B_1 - B_2 \\
1 &\quad 1 \\
B_1 &\quad B_2
\end{align*}
\]

\[
\begin{align*}
&\text{random } u_1v_1^T \quad \text{random } u_2v_2^T \\
&\quad \text{random } u_3v_3^T \text{ with } u_3 \leftarrow \text{col}(U) \\
&\quad \text{random } u_4v_4^T \text{ with } u_5 \leftarrow \text{col}(U)
\end{align*}
\]

\[
\begin{align*}
&\quad 0 \\
B_3 &\quad B_4
\end{align*}
\]

\[
\begin{align*}
&\quad * \\
B_5 &
\end{align*}
\]

\[
\begin{align*}
A + B_1 + B_2 + B_4 &= U + B_4 \\
\text{(rank 3 w.h.p.)}
\end{align*}
\]
\[ s = 11*0* \text{ of length } n = 5, w = 2 \text{ wildcards, } \]
\[ \text{width } w + 1 = 3 \text{ square matrices over } \mathbb{Z}_q. \]

\[ U \]
secret random rank \( w = 2 \) matrix

\[
\begin{bmatrix}
1 \\
B_1
\end{bmatrix}
\quad \begin{bmatrix}
1 \\
B_2
\end{bmatrix}
\quad \begin{bmatrix}
* \\
B_3
\end{bmatrix}
\quad \begin{bmatrix}
0 \\
B_4
\end{bmatrix}
\quad \begin{bmatrix}
* \\
B_5
\end{bmatrix}
\]

random \( u_1 v_1^T \)
random \( u_2 v_2^T \)
random \( v_3^T \) \( v_3 \) with
random \( u_4 v_4^T \)
random \( u_5 v_5^T \) \( v_5 \) with
\[ u_3 \leftarrow \text{col}(U) \]
\[ u_5 \leftarrow \text{col}(U) \]

Evaluation:
On input \( x = 01000 \)

\[
\begin{bmatrix}
A \\
B_2
\end{bmatrix}
\]

\[ = \]
\[
\begin{bmatrix}
U \\
B_1
\end{bmatrix}
\]

(rank 3 w.h.p.)
Evaluation:

On input $x = 11000$

$$A = U - B_1 - B_2$$

$$A + B_1 + B_2 = U \quad \text{(rank 2)}$$

$s = 11^*0^*$ of length $n = 5$, $w = 2$ wildcards, width $w + 1 = 3$ square matrices over $\mathbb{Z}_q$.

$U$ secret random rank $w = 2$ matrix

$$1 \quad 1 \quad * \quad 0 \quad *$$

$B_1$ $\left[ u_1v_1^T \right]$ random $u_1v_1^T$

$B_2$ $\left[ u_2v_2^T \right]$ random $u_2v_2^T$

$B_3$ $\left[ u_3v_3^T \right]$ random with $u_3 \leftarrow \text{col}(U)$

$B_4$ $\left[ u_4v_4^T \right]$ random $u_4v_4^T$

$B_5$ $\left[ u_5v_5^T \right]$ random with $u_5 \leftarrow \text{col}(U)$
\( s = 11^{*}0^{*} \) of length \( n = 5 \), \( w = 2 \) wildcards, width \( w + 1 = 3 \) square matrices over \( \mathbb{Z}_q \).

\[
\begin{align*}
A &= U - B_1 - B_2 \\
1 &\quad \quad 1 \\
B_1 &\quad \quad B_2 \\
\text{random } u_1v_1^T &\quad \quad \text{random } u_2v_2^T \\
B_3 &\quad \quad B_4 \\
\text{random } u_3v_3^T \text{ with } &\quad \quad \text{random } u_4v_4^T \text{ with } \\
0 &\quad \quad 0 \\
B_5 &\quad \quad B_6 \\
\text{random } u_5v_5^T \text{ with } &\quad \quad \\
* &\quad \quad * \\
\end{align*}
\]

Evaluation:
On input \( x = 11100 \)

\[
\begin{align*}
A + B_1 + B_2 + B_3 &= U + B_3 \\
\text{rank 2 since } &\quad \quad \text{rank 2 since } \\
\text{col( } B_3 \text{ ) } &\subset \text{col( } U \text{ )} \\
\text{col( } B_3 \text{ ) } &\subset \text{col( } U \text{ )} \\
\end{align*}
\]
Claim [BLMZ19]: $A, B_1, \ldots, B_n$ statistically hides $s$ if $s$ has sufficient entropy on 0/1 bits.

$s = 11*0*$ of length $n = 5$, $w = 2$ wildcards, width $w + 1 = 3$ square matrices over $\mathbb{Z}_q$.

$U$ secret random rank $w = 2$ matrix

$A = U - B_1 - B_2$

$1 \quad 1 \quad * \quad 0 \quad *$

$B_1 \quad B_2 \quad B_3 \quad B_4 \quad B_5$

random $u_1 v_1^T$ random $u_2 v_2^T$ random $u_3 v_3^T$ with random $u_4 v_4^T$ random $u_5 v_5^T$ with

$u_3 \leftarrow \text{col}(U)$ $u_5 \leftarrow \text{col}(U)$
Claim [BLMZ19]: \(A, B_1, \ldots, B_n\) statistically hides \(s\) if \(s\) has sufficient entropy on 0/1 bits.

\[s = 11*0*\text{ of length } n = 5, w = 2 \text{ wildcards, width } w + 1 = 3 \text{ square matrices over } \mathbb{Z}_q.\]

\[
\begin{align*}
U &\quad \text{secret random rank } w = 2 \text{ matrix} \\
A &= U - B_1 - B_2 \\
1 &\quad \text{random } u_1v_1^T \\
B_1 &\quad \text{random } u_2v_2^T \\
1 &\quad \text{random } u_3v_3^T \text{ with } u_3 \leftarrow \text{col}(U) \\
B_2 &\quad \text{random } u_4v_4^T \\
* &\quad \text{random } u_5v_5^T \text{ with } u_5 \leftarrow \text{col}(U) \\
B_3 &\quad 0 \\
B_4 &\quad * \\
B_5 &
\]

\(\approx_s\) uniformly random matrix
\( s = 11^*0^* \) of length \( n = 5 \), \( w = 2 \) wildcards, width \( w + 1 = 3 \) square matrices over \( \mathbb{Z}_q \).

\[
A = U - B_1 - B_2
\]

\[
B_1 = \begin{pmatrix} 1 & \\ B_2 & \end{pmatrix}
\]

\[
u_1^T v_1^T \text{ random}
\]

\[
u_2^T v_2^T \text{ random}
\]

\[
B_3 = \begin{pmatrix} * & \\ \end{pmatrix}
\]

\[
B_4 = \begin{pmatrix} 0 & \\ \end{pmatrix}
\]

\[
B_5 = \begin{pmatrix} * & \\ \end{pmatrix}
\]

\[
B_i \approx_s \text{ uniformly random rank 1 matrix for all } i
\]

\[
u_3 \leftarrow \text{col}(U)
\]

\[
u_5 \leftarrow \text{col}(U)
\]

Claim [BLMZ19]: \( A, B_1, ..., B_n \) statistically hides \( s \) if \( s \) has sufficient entropy on \( 0/1 \) bits.
Candidate Many-Time Secure ADPs for NC1

Approach 1
(not today)

branching program $BP(x)$ $\xrightarrow{[IK00,AIK04]}$ "one-time secure" $A^* , B_1^* , ..., B_n^*$

+ add determinant-preserving noise

"many-time secure" $A , B_1 , ..., B_n$

Obfuscated program
Candidate Many-Time Secure ADPs for NC1

Approach 2

log-depth boolean formula $f(x)$

encode $f(x)$ gate-by-gate as ADP

Obfuscated program

“many-time secure” $A$, $B_1$, ..., $B_n$
Candidate Many-Time Secure ADPs for NC1

- Positive/Negative Input-wire ADPs
- AND Gates
- OR Gates
Affine Determinant Programs (ADP)

Encode:

\[ f: \{0,1\}^n \rightarrow \{0,1\} \rightarrow A, B_1, \ldots, B_n \]

Evaluate:

\[ M_x := A + \sum_{i \mid x_i = 1} B_i \]

\[ f(x) = 1 \iff \det(M_x) = 0 \]

\[ f(x) = 0 \iff \det(M_x) \neq 0 \]

\[ M_x \text{ rank deficient by 1 when } f(x) = 1 \]
Positive Input Wire

\[ f(x_1, ..., x_n) = x_i \]

1) Draw random \( u \leftarrow \mathbb{Z}_q \)
2) Construct width-1 ADP:

\[
\begin{align*}
A &= u, \\
B_i &= -u, \\
B_j &= 0 \quad (\forall j \neq i)
\end{align*}
\]
Positive Input Wire

\[ f(x_1, \ldots, x_n) = x_i \]

1) Draw random \( u \leftarrow \mathbb{Z}_q \)
2) Construct width-1 ADP:

\[
\begin{align*}
A &= u, \quad B_i = -u, \quad B_j = 0 \quad (\forall j \neq i)
\end{align*}
\]

Correctness

- If \( x_i = 1 \), then \( M_x = 0 \)
- If \( x_i = 0 \), then \( M_x = u \)

(determinant of a scalar is itself)
Negative Input Wire

\[ f(x_1, \ldots, x_n) = \neg x_i \]

1) Draw random \( u \leftarrow \mathbb{Z}_q \)

2) Construct width-1 ADP:

\[
\begin{align*}
A &= 0, \\
B_i &= u, \\
B_j &= 0 \quad (\forall j \neq i)
\end{align*}
\]
Negative Input Wire

\[ f(x_1, \ldots, x_n) = -x_i \]

1) Draw random \( u \leftarrow \mathbb{Z}_q \)

2) Construct width-1 ADP:

\[
\begin{align*}
A &= 0, \quad B_i = u, \quad B_j = 0 \quad (\forall j \neq i)
\end{align*}
\]

Correctness

\[
M_x := A + \sum_{i \mid x_i = 1} B_i
\]

- If \( x_i = 1 \), then \( M_x = u \)
- If \( x_i = 0 \), then \( M_x = 0 \)

(determinant of a scalar is itself)
Candidate AND Gates

Evaluation on $x$ is $M_x^{(f)}$

Evaluation on $x$ is $M_x^{(g)}$
Candidate AND Gates

Evaluation on $x$ is $M_x^{(f)}$

Evaluation on $x$ is $M_x^{(g)}$

$$
M_x^{(f \wedge g)} = R \times \begin{pmatrix} M_x^{(f)} & 0 \\ 0 & M_x^{(g)} \end{pmatrix} \times S
$$

where $R$ is random.
• If $f(x)$ and $g(x)$ are both 1, then $M_x^{(f)}$ and $M_x^{(g)}$ are both rank $k - 1$, so $M_x^{(f \land g)}$ is rank $2k - 2$ (rank deficient)
AND Gate Correctness

- If $f(x)$ and $g(x)$ are both 1, then $M_x^{(f)}$ and $M_x^{(g)}$ are both rank $k - 1$, so $M_x^{(f \land g)}$ is rank $2k - 2$ (rank deficient)

- If at least one of $f(x)$ and $g(x)$ is 0, then at least one of $M_x^{(f)}$ and $M_x^{(g)}$ is rank $k$, so $M_x^{(f \land g)}$ is rank $2k - 1$ (full rank)
Claim: For appropriately-designed “input wire ADPs”, applying these AND gates recovers the [BLMZ19] conjunction obfuscator.
Candidate OR Gates

width $k$

$A^{(f)} \quad B_1^{(f)} \quad \ldots \quad B_n^{(f)}$

Evaluation on $x$ is $M_x^{(f)}$

width $k$

$A^{(g)} \quad B_1^{(g)} \quad \ldots \quad B_n^{(g)}$

Evaluation on $x$ is $M_x^{(g)}$
Candidate OR Gates

Evaluation on $x$ is $M_x^{(f)}$

Evaluation on $x$ is $M_x^{(g)}$

$M_x^{(f \lor g)} = R \times \begin{bmatrix} M_x^{(f)} & U_x \\ 0 & M_x^{(g)} \end{bmatrix} \times S$

random ADP
OR Gate
Correctness

- If at least one of $f(x)$ and $g(x)$ is 1, then $M_x^{(f \lor g)}$ is rank $2k - 1$ (rank deficient)
OR Gate

Correctness

- If at least one of $f(x)$ and $g(x)$ is 1, then $M_x^{(f \lor g)}$ is rank $2k - 1$ (rank deficient)

- If neither $f(x)$ and $g(x)$ are 1, then $M_x^{(f \land g)}$ is rank $2k$ (full rank)
Attacks and Defenses

All attacks so far are “kernel attacks”, which exploit linear relationships between kernels of $M_{x_1}, M_{x_2}, \ldots, M_{x_k}$ from accepting inputs $x_1, x_2, \ldots, x_k$. 
Attacks and Defenses

All attacks so far are “kernel attacks”, which exploit linear relationships between kernels of $M_{x_1}, M_{x_2}, \ldots, M_{x_k}$ from accepting inputs $x_1, x_2, \ldots, x_k$.

Future Directions:

1. Design new input wires to resist kernel attacks.
2. Security for null/evasive circuits?
3. Post-processing strategies, e.g., compute the AND of $k$ independent ADP obfuscations of $f$. 
Thank you!
Questions?

slides available at cs.princeton.edu/~fermin/