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 $C$
- Barrington’s Thm.
- constant-width deterministic branching program $BP$
How do we build iO for NC1?

log-depth circuit $\mathcal{C}$

Barrington’s Thm.

constant-width deterministic branching program $BP$

$M_{1,0}$ $M_{2,0}$ $M_{3,0}$ $M_{4,0}$

$M_{1,1}$ $M_{2,1}$ $M_{3,1}$ $M_{4,1}$

$x_0$ $x_1$ $x_0$ $x_1$

matrix branching program
How do we build iO for NC1?

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

$x = 01$

$x_0$

$x_1$
How do we build iO for NC1?

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

Evaluation: $C(x) = 1$ if $M_{1,0} \times M_{2,1} \times M_{3,0} \times M_{4,1} = F$

$x = 01$

$M_{1,0}$ $M_{2,0}$ $M_{3,0}$ $M_{4,0}$

$M_{1,1}$ $M_{2,1}$ $M_{3,1}$ $M_{4,1}$

$x_0$ $x_1$ $x_0$ $x_1$
What does the matrix branching program representation buy us?

“one-time” security by Kilian randomization

\[ x = 01 \]

\[ x_0 \]

\[ x_1 \]

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

\[ R_1, R_2, R_3 \]

\[
\begin{align*}
M_{1,0} \cdot R_1 &\quad R_1^{-1} \cdot M_{2,0} \cdot R_2 \\
M_{1,1} \cdot R_1 &\quad R_1^{-1} \cdot M_{2,1} \cdot R_2 \\
x_0 &\quad x_1
\end{align*}
\]

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

“one-time” security by Kilian randomization

Sample random matrices

\[ R_1, R_2, R_3 \]

\( \tilde{M}_{1,0}, \tilde{M}_{1,1} \)
\( \tilde{M}_{2,0}, \tilde{M}_{2,1} \)
\( \tilde{M}_{3,0}, \tilde{M}_{4,0} \)
\( \tilde{M}_{4,1} \)

\( x_0, x_1 \)

(\( \tilde{M} \) 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.

\[
x = 01
\]

\[
\begin{align*}
\hat{M}_{1,0} & \quad \hat{M}_{2,0} & \quad \hat{M}_{3,0} & \quad \hat{M}_{4,0} \\
\hat{M}_{1,1} & \quad \hat{M}_{2,1} & \quad \hat{M}_{3,1} & \quad \hat{M}_{4,1} \\
\begin{array}{l}
x_0 \\
\end{array} & \begin{array}{l}
x_1 \\
\end{array} & \begin{array}{l}
x_0 \\
\end{array} & \begin{array}{l}
x_1 \\
\end{array}
\end{align*}
\]

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

Can statistically simulate $\tilde{M}_1, \tilde{M}_2, \tilde{M}_3, \tilde{M}_4$ 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: $\tilde{M}_{1,0} \times \tilde{M}_{2,0} \times \tilde{M}_{3,0} \times \tilde{M}_{4,0}$ is a mixed-input evaluation.
NC1 circuit $C$ \[\Rightarrow\] Barrington’s Thm. \[\Rightarrow\] constant-width deterministic branching program $BP$

\[\downarrow\]

Kilian-randomized matrix branching program

\[\downarrow\]

encode in multilinear map

\[\downarrow\]

$Obf(C)$

[GGHRSW] approach to iO for NC1
Our goal: Avoid multilinear maps by using an alternative representation of $C$. 

NC1 circuit $C$ \[\rightarrow\] Barrington’s Thm. \[\rightarrow\] constant-width deterministic branching program $BP$ 

\[\downarrow\] Kilian-randomized matrix branching program 

\[\downarrow\] encode in multilinear map 

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

\begin{align*}
\text{affine determinant program* (ADP)} & \quad \xrightarrow{\text{[IK00]}} \quad \text{Kilian-randomized matrix branching program} \\
\text{Obf}(C) & \quad \xrightarrow{??} \quad \text{encode in multilinear map} \\
\text{Obf}(C)
\end{align*}

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

Encode:

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

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

Encode:

\[ f: \{0,1\}^n \rightarrow \{0,1\} \]

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

Evaluate:

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

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

Encode:

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

Evaluate:

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

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

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

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

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

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

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

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

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

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

Encode:

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

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.
Affine Determinant Programs (ADPs) are an “additive” analogue of Matrix Branching Programs (MBPs).

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

**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 \( \mathbb{Z}_q \).

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

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

\[
U \quad \text{secret random rank } w = 2 \text{ matrix}
\]

\[
\begin{bmatrix}
1 \\
B_1 \\
\end{bmatrix}
\begin{bmatrix}
1 \\
B_2 \\
\end{bmatrix}
\begin{bmatrix}
0 \\
B_4 \\
\end{bmatrix}
\]

random \( u_1 v_1^T \)
random \( u_2 v_2^T \)
random \( u_4 v_4^T \)
$s = 11^*0^*$ of length $n = 5$, $w = 2$ wildcards, width $w + 1 = 3$ square matrices over $\mathbb{Z}_q$.

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

$B_1$ random $u_1v_1^T$

$B_2$ random $u_2v_2^T$

$B_3$ random $u_3v_3^T$ with $U$

$B_4$ random $u_4v_4^T$

$B_5$ 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 \( w = 2 \) matrix

\[
A = U - \sum_{i \mid s_i=1} B_i
\]

- \( B_1 \) random \( u_1v_1^T \)
- \( B_2 \) random \( u_2v_2^T \)
- \( B_3 \) random \( u_3v_3^T \) with \( U \)
- \( B_4 \) random \( u_4v_4^T \)
- \( B_5 \) random \( u_5v_5^T \) with \( U \)

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

\[
\begin{align*}
U & \quad \text{secret random rank $w=2$ matrix} \\
A & = U - B_1 - B_2 \\
B_1 & = \begin{bmatrix} 1 \\ \end{bmatrix} \\
B_2 & = \begin{bmatrix} 1 \\ \end{bmatrix} \\
B_3 & = \begin{bmatrix} * \\ \end{bmatrix} \\
B_4 & = \begin{bmatrix} 0 \\ \end{bmatrix} \\
B_5 & = \begin{bmatrix} * \\ \end{bmatrix}
\end{align*}
\]

random $u_1v_1^T$, random $u_2v_2^T$, random $u_3v_3^T$ with $u_3 \leftarrow \text{col}(U)$, random $u_4v_4^T$, random $u_5v_5^T$ with $u_5 \leftarrow \text{col}(U)$.
\(s = 11^*0^*\) of length \(n = 5\), \(w = 2\) wildcards, width \(w + 1 = 3\) square matrices over \(\mathbb{Z}_q\).

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

\[
\begin{align*}
A &= U - B_1 - B_2 \\
1 &\quad 1 &\quad * &\quad 0 &\quad * \\
B_1 &\quad B_2 &\quad B_3 &\quad B_4 &\quad B_5
\end{align*}
\]

\(B_1, B_2\) random \(u_1v_1^T\), \(u_2v_2^T\)  
\(B_3\) random \(u_3v_3^T\) with \(u_3 \leftarrow \text{col}(U)\)  
\(B_4, B_5\) random \(u_4v_4^T, u_5v_5^T\) with \(u_5 \leftarrow \text{col}(U)\)

**Evaluation:**

On input \(x = 11010\)

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

**Evaluation:**

On input \( x = 01000 \)

\[
A = U - B_1 - B_2
\]

\[ A + B_2 = U - B_1 \]

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

\[
\begin{bmatrix}
U
\end{bmatrix} \text{ secret random rank } w = 2 \text{ matrix}
\]

\[
A = U - B_1 - B_2
\]

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

\begin{align*}
\text{random } u_1 v_1^T & \quad \text{random } u_2 v_2^T & \quad \text{random with } u_3 v_3^T \quad & \quad \text{random } u_4 v_4^T \quad & \quad \text{random } u_5 v_5^T \\
\text{with } u_3 \leftarrow \text{col}(U) & \quad & \quad \text{with } u_5 \leftarrow \text{col}(U)
\end{align*}

Evaluation:
On input \( x = 11000 \)

\[
A + B_1 + B_2 = U
\]

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

**Evaluation:**

On input \( x = 11100 \)

\[
A + B_1 + B_2 + B_3 = U + B_3
\]

rank 2 since \( \text{col}(B_3) \subseteq \text{col}(U) \)
$s = 11^*0^*$ of length $n = 5$, $w = 2$ wildcards, width $w + 1 = 3$ square matrices over $\mathbb{Z}_q$.

Claim [BLMZ19]: $A, B_1, ..., B_n$ statistically hides $s$ if $s$ has sufficient entropy on 0/1 bits.

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

\[
A = U - B_1 - B_2
\]

\[
\begin{array}{cccc}
1 & 1 & * & 0 & * \\
B_1 & B_2 & B_3 & B_4 & B_5 \\
\text{random} & \text{random} & \text{random with} & \text{random} & \text{random} \\
\begin{bmatrix} u_1 v_1^T \end{bmatrix} & \begin{bmatrix} u_2 v_2^T \end{bmatrix} & \begin{bmatrix} u_3 v_3^T \end{bmatrix} & \begin{bmatrix} u_4 v_4^T \end{bmatrix} & \begin{bmatrix} u_5 v_5^T \end{bmatrix} \\
\end{array}
\]

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

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

\[ A = U - B_1 - B_2 \]

\[ \begin{align*}
1 & \quad 1 \\
B_1 & \quad B_2
\end{align*} \]
random \( u_1 v_1^T \) \quad random \( u_2 v_2^T \)

\[ \begin{align*}
* & \quad 0 \\
B_3 & \quad B_4 \quad B_5
\end{align*} \]
random \( u_3 v_3^T \) \quad random \( u_4 v_4^T \) \quad random \( u_5 v_5^T \)

\[ \begin{align*}
u_3 & \leftarrow \text{col}(U) \\
u_5 & \leftarrow \text{col}(U)
\end{align*} \]

Claim [BLMZ19]: \( A, B_1, \ldots, B_n \) statistically hides \( s \) if \( s \) has sufficient entropy on 0/1 bits.
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

$$A = U - B_1 - B_2$$

$$1 \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$ random $u_1v_1^T$ random $u_2v_2^T$

$$\approx_s$$ uniformly randomly matrix

$$u_3, u_5$$ from (hidden) random 2-dimensional subspace

$B_i \approx_s$ uniformly randomly rank 1 matrix for all $i$

$$\begin{bmatrix} * \\ 0 \end{bmatrix}$$ random $u_3v_3^T$ with random $u_4v_4^T$

$$\begin{bmatrix} * \\ \end{bmatrix}$$ random $u_5v_5^T$ with random $U$

$$u_3 \leftarrow \text{col}(U)$$ $$u_5 \leftarrow \text{col}(U)$$
CAUTION

WORK IN PROGRESS
Candidate Many-Time Secure ADPs for NC1

Approach 1
(not today)

branching program $BP(x)$

[IK00]

"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

"many-time secure"

Obfuscated program

$A$, $B_1$, ..., $B_n$
Candidate Many-Time Secure ADPs for NC1

- Log-depth boolean formula \( f(x) \)
- Encode \( f(x) \) gate-by-gate as ADP
- Obfuscated program

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

**Encode:**

$f: \{0,1\}^n \rightarrow \{0,1\} \rightarrow A, B_1, ..., B_n$

**Evaluate:**

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

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

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

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

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

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

\[
A = u, \quad B_i = -u, \quad B_j = 0 \quad (\forall j \neq i)
\]
Positive Input Wire

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

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

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

**Correctness**

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

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

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

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

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

Correctness

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

- 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 $M_x^{(g)}$

$(2k-1) \times (2k-1)$ \hspace{1cm} $(2k-1) \times 2k$ \hspace{1cm} $2k \times 2k$ \hspace{1cm} $2k \times (2k-1)$

$M_x^{(f \land g)} = \begin{array}{c} R \\ \text{random} \end{array} \times \begin{array}{c} M_x^{(f)} \\ 0 \end{array} \times S$

$S$ \hspace{1cm} \text{random}$
• If \( f(x) \) and \( g(x) \) are both 1, then \( M_x^{(f)} \) and \( M_x^{(g)} \) are both rank \( k - 1 \), so \( M_x^{(f \wedge g)} \) is rank \( 2k - 2 \) (rank deficient)

**AND Gate Correctness**

\[
\begin{align*}
(2k - 1) \times (2k - 1) &\quad \times \quad (2k - 1) \times 2k &\quad \times \quad 2k \times 2k &\quad \times \quad 2k \times (2k - 1) \\
M_x^{(f \wedge g)} &\quad = \quad R &\quad \times \quad 0 &\quad \times \quad S \\
\text{random} &\quad \text{random} &\quad \text{random}
\end{align*}
\]
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 $M_x^{(f \wedge g)}$ 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 $M_x^{(f \wedge g)}$ is rank $2k - 1$ (full rank)

\[
M_x^{(f \wedge g)} = \begin{pmatrix} (2k - 1) \times (2k - 1) \\ \end{pmatrix}
= \begin{pmatrix} (2k - 1) \times 2k \\ \end{pmatrix}
\times \begin{pmatrix} 2k \times 2k \\ \end{pmatrix}
\times \begin{pmatrix} 2k \times (2k - 1) \\ \end{pmatrix}
\]

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

$R$ is random
Claim: For appropriately-designed “input wire ADPs”, applying these AND gates recovers the $[\text{BLMZ19}]$ conjunction obfuscator.
Candidate OR Gates

width $k$

$A^{(f)}$ \quad $B^{(f)}_1$ \quad ... \quad $B^{(f)}_n$

Evaluation on $x$ is $M^{(f)}_x$

width $k$

$A^{(g)}$ \quad $B^{(g)}_1$ \quad ... \quad $B^{(g)}_n$

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

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

Evaluation on $x$ is $M_x^{(g)}$
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 $M_x^{(f \vee g)}$ is rank $2k - 1$ (rank deficient)
- If neither $f(x)$ and $g(x)$ are 1, then $M_x^{(f \wedge g)}$ is rank $2k$ (full rank)

$$
\begin{bmatrix}
2k \times 2k \\
M_x^{(f \vee g)}
\end{bmatrix}
= 
\begin{bmatrix}
2k \times 2k \\
R
\end{bmatrix}
\times
\begin{bmatrix}
2k \times 2k \\
M_x^{(f)} \\
0 \\
M_x^{(g)}
\end{bmatrix}
\times
\begin{bmatrix}
2k \times 2k \\
S
\end{bmatrix}
\text{random ADP}
$$
Attacks and Defenses

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

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

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/~fermim/