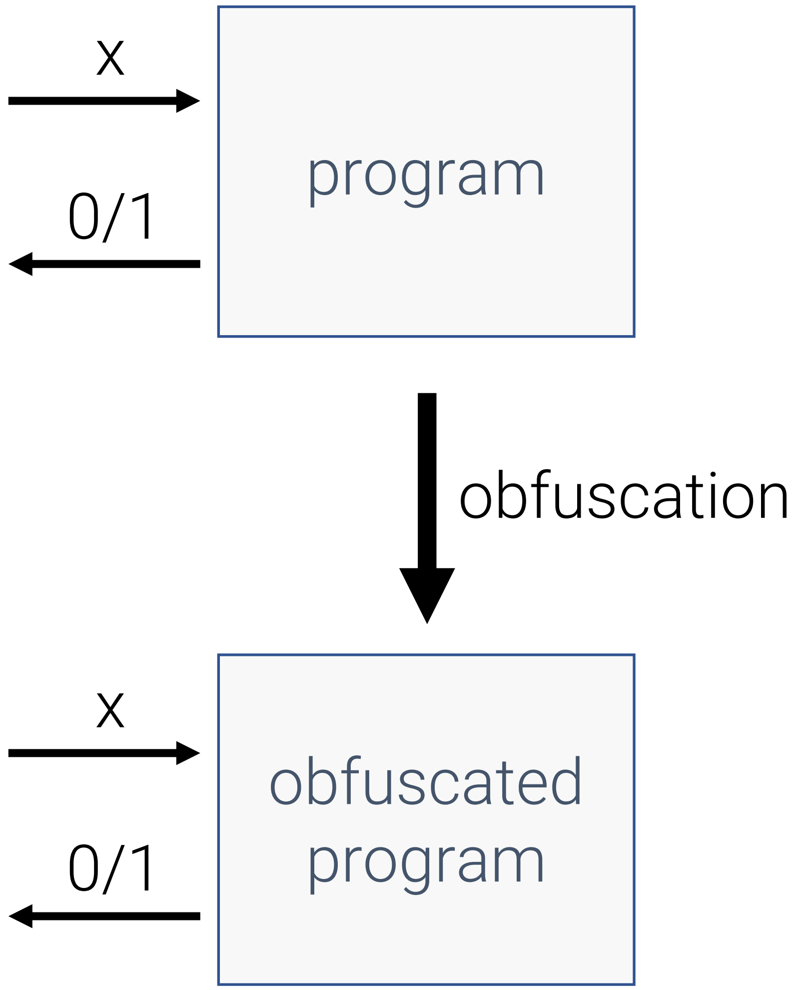
Affine Determinant Programs: A New Approach to Obfuscation

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]

$$\boxed{\text{iO for NC1}} \xrightarrow{\text{(assuming FHE)}} \boxed{\text{iO for all circuits}}$$

Takeaway: it suffices to consider NC1.

How do we build iO for NC1?

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

How do we build iO for NC1?

log-depth circuit C $\xrightarrow{\text{Barrington's Thm.}}$ 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	x_0	x_1
$M_{1,0}$	$M_{2,0}$	$M_{3,0}$	$M_{4,0}$
$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{4,1}$

How do we build iO for NC1?

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

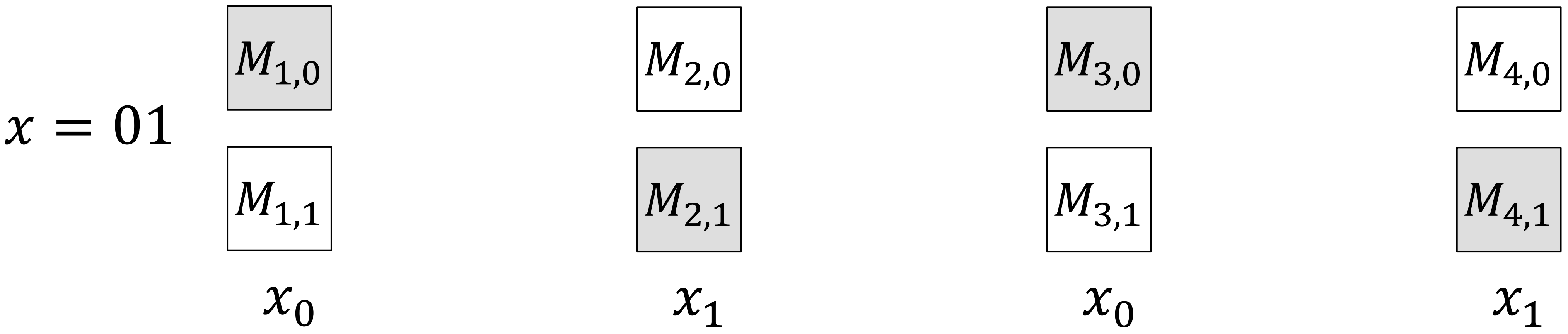
$x = 01$

$M_{1,0}$	$M_{2,0}$	$M_{3,0}$	$M_{4,0}$
$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{4,1}$
x_0	x_1	x_0	x_1

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



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

Sample random matrices

R_1, R_2, R_3

$$
\begin{array}{cccc}
M_{1,0} \cdot R_1 & R_1^{-1} \cdot M_{2,0} \cdot R_2 & R_2^{-1} \cdot M_{3,0} \cdot R_3 & R_3^{-1} \cdot M_{4,0} \\
M_{1,1} \cdot R_1 & R_1^{-1} \cdot M_{2,1} \cdot R_2 & R_2^{-1} \cdot M_{3,1} \cdot R_3 & R_3^{-1} \cdot M_{4,1} \\
x_0 & x_1 & x_0 & x_1
\end{array}
$$

What does the matrix branching program representation buy us?

"one-time" security by Kilian randomization

Sample random matrices

R_1 , R_2 , R_3

$\widehat{M}_{1,0}$	$\widehat{M}_{2,0}$	$\widehat{M}_{3,0}$	$\widehat{M}_{4,0}$
$\widehat{M}_{1,1}$	$\widehat{M}_{2,1}$	$\widehat{M}_{3,0}$	$\widehat{M}_{4,1}$
x_0	x_1	x_0	x_1

($\widehat{M}$ denotes M after applying Kilian randomization)

Kilian's Statistical Simulation Lemma:

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

$x = 01$				
	$\widehat{M}_{1,0}$	$\widehat{M}_{2,0}$	$\widehat{M}_{3,0}$	$\widehat{M}_{4,0}$
	$\widehat{M}_{1,1}$	$\widehat{M}_{2,1}$	$\widehat{M}_{3,0}$	$\widehat{M}_{4,1}$
	x_0	x_1	x_0	x_1

"grey matrices leak nothing beyond whether $BP(x) = 0$ or 1"

Kilian's Statistical Simulation Lemma:

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

Takeaway: Kilian-randomization yields "one-time" security.

Kilian's Statistical Simulation Lemma:

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

Takeaway: Kilian-randomization yields "one-time" security.

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

"one-time" secure

"many-time" secure

Multilinear maps enforce **input consistency**; without them, "mixed-input" attacks can break security!

$\widehat{M}_{1,0}$	$\widehat{M}_{2,0}$	$\widehat{M}_{3,0}$	$\widehat{M}_{4,0}$
$\widehat{M}_{1,1}$	$\widehat{M}_{2,1}$	$\widehat{M}_{3,0}$	$\widehat{M}_{4,1}$
x_0	x_1	x_0	x_1

Example: $\widehat{M}_{1,0} \times \widehat{M}_{2,0} \times \widehat{M}_{3,0} \times \widehat{M}_{4,0}$ is a mixed-input evaluation.

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

$\downarrow$

Kilian-randomized matrix branching program

$\downarrow$ encode in multilinear map

$Obf(C)$

[GGHRSW] approach to iO for NC1

NC1 circuit C

Barrington's Thm.

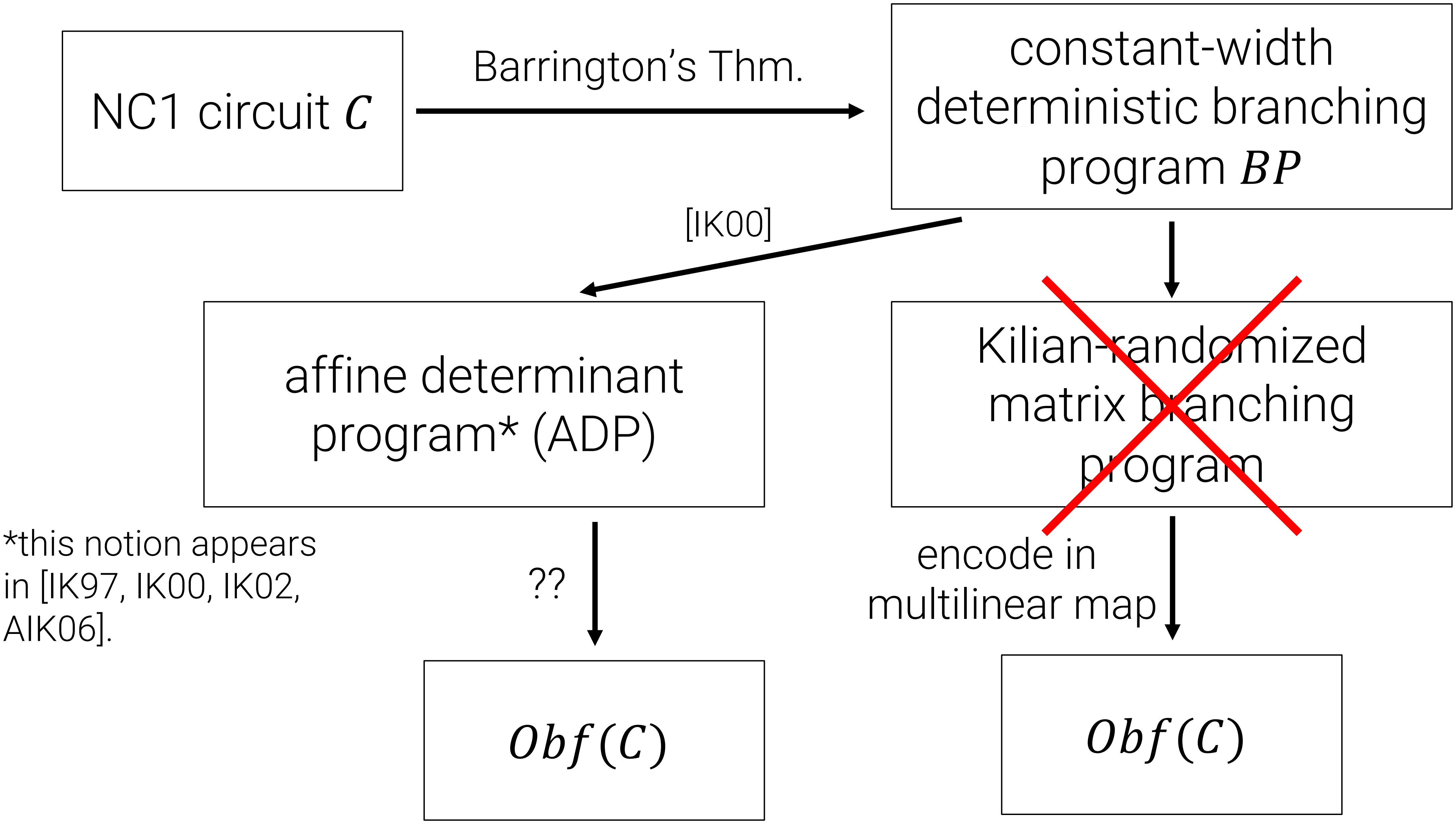
constant-width deterministic branching program BP

Kilian-randomized matrix branching program

encode in multilinear map

$Obf(C)$

Our goal: Avoid multilinear maps by using an alternative representation of C.


NC1 circuit C
Barrington's Thm.
constant-width deterministic branching program BP
[IK00]
affine determinant program* (ADP)
Kilian-randomized matrix branching program
*this notion appears in [IK97, IK00, IK02, AIK06].
??
encode in multilinear map
$Obf(C)$
$Obf(C)$

Affine Determinant Programs (ADP)

Encode:

width w matrices over $\mathbb{Z}_q$

$$f:\{0,1\}^n \to \{0,1\} \longrightarrow \boxed{A}, \boxed{B_1}, \dots, \boxed{B_n}$$

Affine Determinant Programs (ADP)

Encode:

width w matrices over $\mathbb{Z}_q$

$$f : \{0,1\}^n \to \{0,1\} \longrightarrow \boxed{A}, \boxed{B_1}, \dots, \boxed{B_n}$$

Evaluate:

$$\boxed{M_x} := \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

Affine Determinant Programs (ADP)

Encode:

width w matrices over $\mathbb{Z}_q$

$$f:\{0,1\}^n \to \{0,1\} \longrightarrow \boxed{A}\,, \boxed{B_1}\,, \dots, \boxed{B_n}$$

Evaluate:

$$\boxed{M_x} := \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

$$f(x) = 1 \longleftrightarrow det(\boxed{M_x}) = 0$$

$$f(x) = 0 \longleftrightarrow det(\boxed{M_x}) \neq 0$$

$\boxed{M_x}$ rank deficient by 1 when $f(x) = 1$

Affine Determinant Programs (ADP)

Encode:

width w matrices over $\mathbb{Z}_q$

$$f:\{0,1\}^n \to \{0,1\} \longrightarrow \boxed{A}, \boxed{B_1}, \ldots, \boxed{B_n}$$

Evaluate:

$$\boxed{M_x} \coloneqq \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

$$f(x) = 1 \longleftrightarrow det(\boxed{M_x}) = 0$$

$$f(x) = 0 \longleftrightarrow det(\boxed{M_x}) \neq 0$$

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

$\boxed{M_x}$ rank deficient by 1 when $f(x) = 1$

Affine Determinant Programs (ADP)

Encode:

width w matrices over $\mathbb{Z}_q$

$$f : \{0,1\}^n \to \{0,1\} \longrightarrow \boxed{A}\,,\ \boxed{B_1}\,,\ \ldots,\ \boxed{B_n}$$

Evaluate:

$$\boxed{M_x} := \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

$$f(x) = 1 \longleftrightarrow det(\boxed{M_x}) = 0$$

$$f(x) = 0 \longleftrightarrow det(\boxed{M_x}) \neq 0$$

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.

$\boxed{M_x}$ rank deficient by 1 when $f(x) = 1$

Affine Determinant Programs (ADPs)

A , B_1 , …, 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)

$$\boxed{A}\,,\ \boxed{B_1}\,,\ \dots,\ \boxed{B_n}$$

Matrix Branching Programs (MBPs)

$$\begin{matrix} \boxed{M_{1,0}} & \boxed{M_{2,0}} & \boxed{M_{3,0}} & \boxed{M_{4,0}} \\ \boxed{M_{1,1}} & \boxed{M_{2,1}} & \boxed{M_{3,1}} & \boxed{M_{4,1}} \end{matrix}$$

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.

Affine Determinant Programs (ADPs)

$$\boxed{A}\,,\ \boxed{B_1}\,, \dots ,\ \boxed{B_n}$$

Matrix Branching Programs (MBPs)

$$\begin{array}{cccc} \boxed{M_{1,0}} & \boxed{M_{2,0}} & \boxed{M_{3,0}} & \boxed{M_{4,0}} \\ \boxed{M_{1,1}} & \boxed{M_{2,1}} & \boxed{M_{3,1}} & \boxed{M_{4,1}} \end{array}$$

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):** $A, B_1, \dots, B_n$ 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 $s = \mathbf{11{*}0{*}}$, output

$$\boxed{A} \quad \boxed{B_1} \cdots \boxed{B_n}$$

Evaluation: Input x matches s if

$$det\left(\boxed{A} + \sum_{i|x_i=1} \boxed{B_i}\right) = 0$$

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

$\boxed{U}$ 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

1	1	0
B_1	B_2	B_4
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_4 v_4^\top$

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

$\boxed{U}$ secret random rank $w = 2$ matrix

1	1	*	0	*
$\boxed{B_1}$	$\boxed{B_2}$	$\boxed{B_3}$	$\boxed{B_4}$	$\boxed{B_5}$
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(\boxed{U})$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(\boxed{U})$

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

$\boxed{U}$ secret random rank $w = 2$ matrix

$$\boxed{A} = \boxed{U} - \sum_{i \mid s_i = 1} \boxed{B_i}$$

1	1	*	0	*
$\boxed{B_1}$	$\boxed{B_2}$	$\boxed{B_3}$	$\boxed{B_4}$	$\boxed{B_5}$
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(\boxed{U})$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(\boxed{U})$

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

U secret random rank $w = 2$ matrix

$$A = U - B_1 - B_2$$

1	1	*	0	*
B_1	B_2	B_3	B_4	B_5
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(U)$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(U)$

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

$\boxed{U}$ secret random rank $w = 2$ matrix

$\boxed{A} = \boxed{U} - \boxed{B_1} - \boxed{B_2}$

1	1	*	0	*
$\boxed{B_1}$	$\boxed{B_2}$	$\boxed{B_3}$	$\boxed{B_4}$	$\boxed{B_5}$
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(\boxed{U})$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(\boxed{U})$

Evaluation:

On input $x = 11010$

$\boxed{A} + \boxed{B_1} + \boxed{B_2} + \boxed{B_4}$

$= \boxed{U} + \boxed{B_4}$

(rank 3 w.h.p.)

$s = 11{*}0{*}$ of length $n = 5$, $\mathrm{w} = 2$ 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	1	*	0	*
B_1	B_2	B_3	B_4	B_5
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(U)$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(U)$

Evaluation:
On input $x = 01000$

$$A + B_2$$

$$= U - B_1$$

(rank 3 w.h.p.)

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

$\boxed{U}$ secret random rank $w = 2$ matrix

$$\boxed{A} = \boxed{U} - \boxed{B_1} - \boxed{B_2}$$

1	1	*	0	*
$\boxed{B_1}$	$\boxed{B_2}$	$\boxed{B_3}$	$\boxed{B_4}$	$\boxed{B_5}$
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(\boxed{U})$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(\boxed{U})$

Evaluation:
On input $x = 11000$

$$\boxed{A} + \boxed{B_1} + \boxed{B_2}$$

$$= \boxed{U}$$

(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

$A = U - B_1 - B_2$

1	1	*	0	*
B_1	B_2	B_3	B_4	B_5
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(U)$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(U)$

Evaluation:

On input $x = 11100$

$A + B_1 + B_2 + B_3$

$= U + B_3$

rank 2 since

$col(B_3) \subset col(U)$

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

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

$\boxed{U}$ secret random rank $w = 2$ matrix

$$\boxed{A} = \boxed{U} - \boxed{B_1} - \boxed{B_2}$$

1	1	*	0	*
$\boxed{B_1}$	$\boxed{B_2}$	$\boxed{B_3}$	$\boxed{B_4}$	$\boxed{B_5}$
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(\boxed{U})$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(\boxed{U})$

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

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

$\boxed{U}$ secret random rank $w = 2$ matrix

$\approx_s$ uniformly random matrix

$$\boxed{A} = \boxed{U} - \boxed{\boxed{B_1} - \boxed{B_2}}$$

1	1	*	0	*
B_1	B_2	B_3	B_4	B_5
random $u_1 v_1^\top$	random $u_2 v_2^\top$	random $u_3 v_3^\top$ with $u_3 \leftarrow col(\boxed{U})$	random $u_4 v_4^\top$	random $u_5 v_5^\top$ with $u_5 \leftarrow col(\boxed{U})$

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

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

U secret random rank $w = 2$ matrix

$$A = U - B_1 - B_2$$

$\approx_s$ uniformly random matrix

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

1	1	*	0	*
B_1	B_2	B_3	B_4	B_5
random $u_1 v_1^{\top}$	random $u_2 v_2^{\top}$	random $u_3 v_3^{\top}$ with $u_3 \leftarrow col(U)$	random $u_4 v_4^{\top}$	random $u_5 v_5^{\top}$ with $u_5 \leftarrow col(U)$

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


CAUTION
WORK IN
PROGRESS
EURO SIGNS

Candidate Many-Time Secure ADPs for NC1

branching program
$BP(x)$

$\xrightarrow{\text{[IK00]}}$

"one-time secure"
$A^*, B_1^*, \dots, B_n^*$

$\downarrow$ + add determinant-preserving noise

"many-time secure"
$A, B_1, \dots, B_n$

Obfuscated program

Approach 1
(not today)

Candidate Many-Time Secure ADPs for NC1

log-depth boolean formula $f(x)$ $\xrightarrow{\text{encode } f(x) \text{ gate-by-gate as ADP}}$

"many-time secure"

$\boxed{A}, \boxed{B_1}, \dots, \boxed{B_n}$

Obfuscated program

Approach 2

Candidate Many-Time Secure ADPs for NC1

log-depth
boolean formula
$f(x)$

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

"many-time secure"

$A, B_1, \dots, B_n$

Obfuscated program

- Positive/Negative Input-wire ADPs
- AND Gates
- OR Gates

Affine Determinant Programs (ADP)

Encode:

width w matrices over $\mathbb{Z}_q$

$$f : \{0,1\}^n \to \{0,1\} \longrightarrow \boxed{A}, \boxed{B_1}, \ldots, \boxed{B_n}$$

Evaluate:

$$\boxed{M_x} := \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

$$f(x) = 1 \longleftrightarrow det(\boxed{M_x}) = 0$$

$$f(x) = 0 \longleftrightarrow det(\boxed{M_x}) \neq 0$$

$\boxed{M_x}$ rank deficient by 1 when $f(x) = 1$

Positive Input Wire

$f(x_1, \dots, x_n) = x_i$

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

$$\boxed{A} = u, \quad \boxed{B_i} = -u, \quad \boxed{B_j} = 0 \quad (\forall j \neq i)$$

Positive Input Wire

$f(x_1, \ldots, x_n) = x_i$

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

$$\boxed{A} = u, \quad \boxed{B_i} = -u, \quad \boxed{B_j} = 0 \quad (\forall j \neq i)$$

Correctness

$$\boxed{M_x} \coloneqq \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

- If $x_i = 1$, then $\boxed{M_x} = 0$
- If $x_i = 0$, then $\boxed{M_x} = u$

(determinant of a scalar is itself)

Negative Input Wire

$f(x_1, \dots, x_n) = \neg x_i$

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

$$\boxed{A} = 0, \quad \boxed{B_i} = u, \quad \boxed{B_j} = 0 \quad (\forall j \neq i)$$

Negative Input Wire

$f(x_1, \dots, x_n) = \neg x_i$

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

$$\boxed{A} = 0, \quad \boxed{B_i} = u, \quad \boxed{B_j} = 0 \quad (\forall j \neq i)$$

Correctness

$$\boxed{M_x} \coloneqq \boxed{A} + \sum_{i \mid x_i = 1} \boxed{B_i}$$

- If $x_i = 1$, then $\boxed{M_x} = u$
- If $x_i = 0$, then $\boxed{M_x} = 0$

(determinant of a scalar is itself)

Candidate AND Gates

width k

$A^{(f)}$ $B_1^{(f)}$ ... $B_n^{(f)}$ Evaluation on x is $M_x^{(f)}$

width k

$A^{(g)}$ $B_1^{(g)}$ ... $B_n^{(g)}$ Evaluation on x is $M_x^{(g)}$

Candidate AND Gates

width k

$A^{(f)}$ $B_1^{(f)}$ … $B_n^{(f)}$

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

width k

$A^{(g)}$ $B_1^{(g)}$ … $B_n^{(g)}$

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

$$\underset{(2k-1)\times(2k-1)}{M_x^{(f\wedge g)}} = \underset{(2k-1)\times 2k}{\underset{\text{random}}{R}} \times \underset{2k\times 2k}{\begin{bmatrix} M_x^{(f)} & 0 \\ 0 & M_x^{(g)} \end{bmatrix}} \times \underset{2k\times(2k-1)}{\underset{\text{random}}{S}}$$

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)

$$\overset{(2k-1)\times(2k-1)}{M_x^{(f\wedge g)}} = \overset{(2k-1)\times 2k}{\underset{\text{random}}{R}} \times \overset{2k\times 2k}{\begin{bmatrix} M_x^{(f)} & 0 \\ 0 & M_x^{(g)} \end{bmatrix}} \times \overset{2k\times(2k-1)}{\underset{\text{random}}{S}}$$

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)

$$\overset{(2k-1)\times(2k-1)}{M_x^{(f\wedge g)}} = \overset{(2k-1)\times 2k}{\underset{\text{random}}{R}} \times \overset{2k\times 2k}{\begin{bmatrix} M_x^{(f)} & 0 \\ 0 & M_x^{(g)} \end{bmatrix}} \times \overset{2k\times(2k-1)}{\underset{\text{random}}{S}}$$

Claim: For appropriately-designed “input wire ADPs”, applying these AND gates recovers the [BLMZ19] conjunction obfuscator.

$$\overset{(2k-1)\times(2k-1)}{M_x^{(f\wedge g)}} = \overset{(2k-1)\times 2k}{\underset{\text{random}}{R}} \times \overset{2k\times 2k}{\begin{bmatrix} M_x^{(f)} & 0 \\ 0 & M_x^{(g)} \end{bmatrix}} \times \overset{2k\times(2k-1)}{\underset{\text{random}}{S}}$$

Candidate OR Gates

width k

$A^{(f)}$ $B_1^{(f)}$ ... $B_n^{(f)}$ Evaluation on x is $M_x^{(f)}$

width k

$A^{(g)}$ $B_1^{(g)}$ ... $B_n^{(g)}$ Evaluation on x is $M_x^{(g)}$

Candidate OR Gates

width k

$A^{(f)}$ $B_1^{(f)}$ ... $B_n^{(f)}$

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

width k

$A^{(g)}$ $B_1^{(g)}$ ... $B_n^{(g)}$

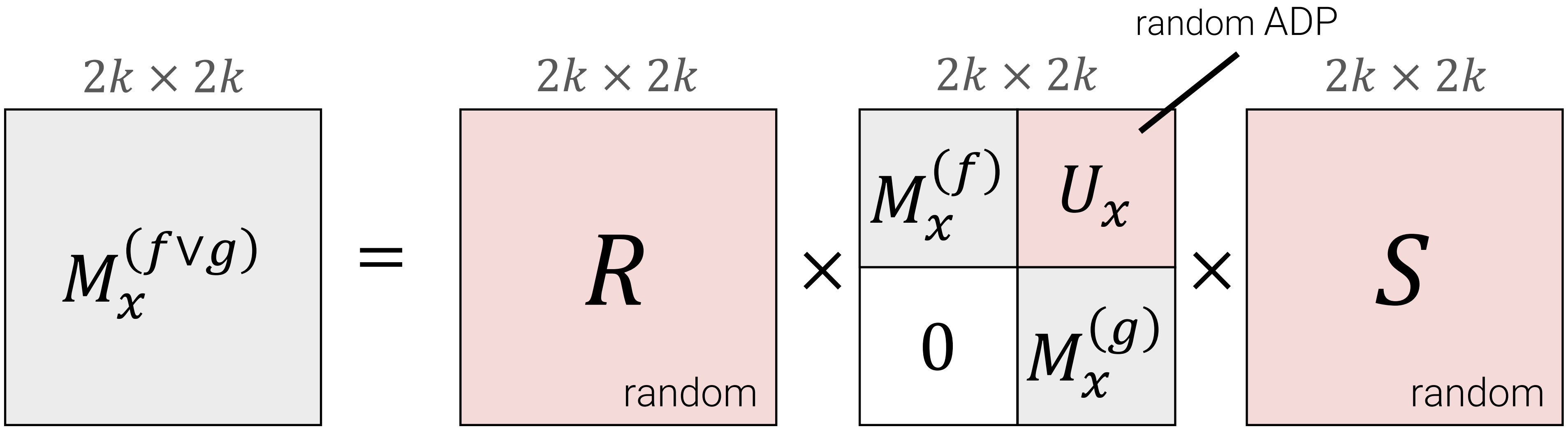
Evaluation on x is $M_x^{(g)}$

$$\underbrace{M_x^{(f \vee g)}}_{2k \times 2k} = \underbrace{R}_{\substack{2k \times 2k \\ \text{random}}} \times \underbrace{\begin{pmatrix} M_x^{(f)} & U_x \\ 0 & M_x^{(g)} \end{pmatrix}}_{2k \times 2k} \times \underbrace{S}_{\substack{2k \times 2k \\ \text{random}}}$$

random ADP

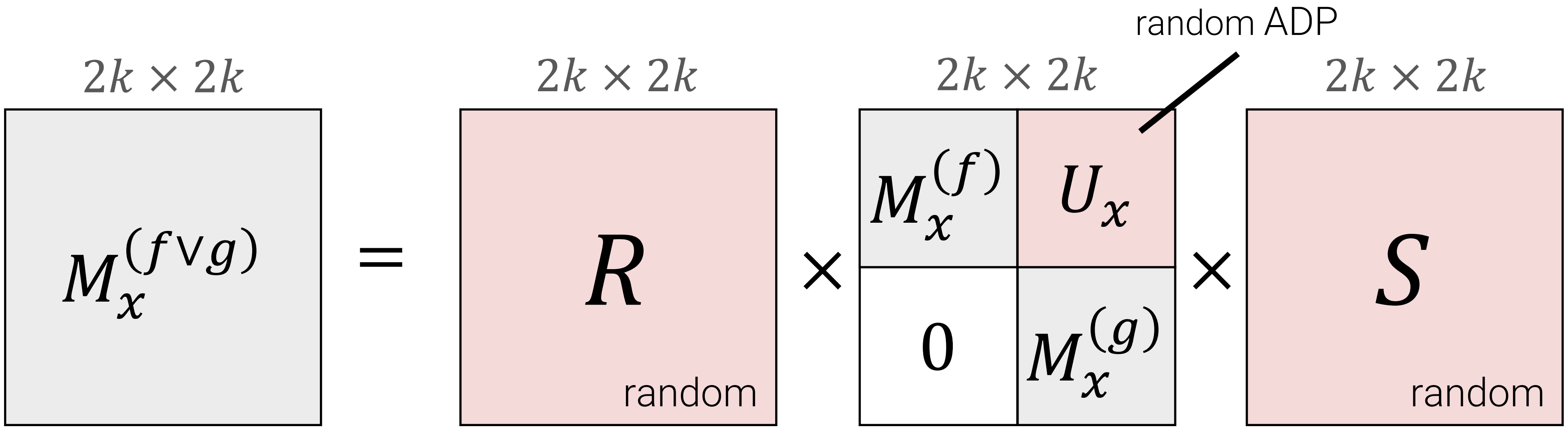
OR Gate Correctness

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



OR Gate Correctness

- If at least one of $f(x)$ and $g(x)$ is 1, then $M_x^{(f\wedge g)}$ 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)



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 $M_{x_1}, M_{x_2}, \dots, M_{x_k}$ from accepting inputs $x_1, x_2, \dots, x_k$.

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/