Obfuscation

Jeremy Blackthorne

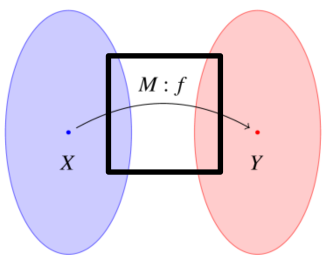
10/29/2014

Obfuscation

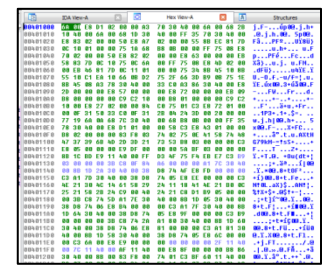
- What is it intuitively?
- Obfuscation vs. Cryptography

Spectrum of Abstraction

Functions



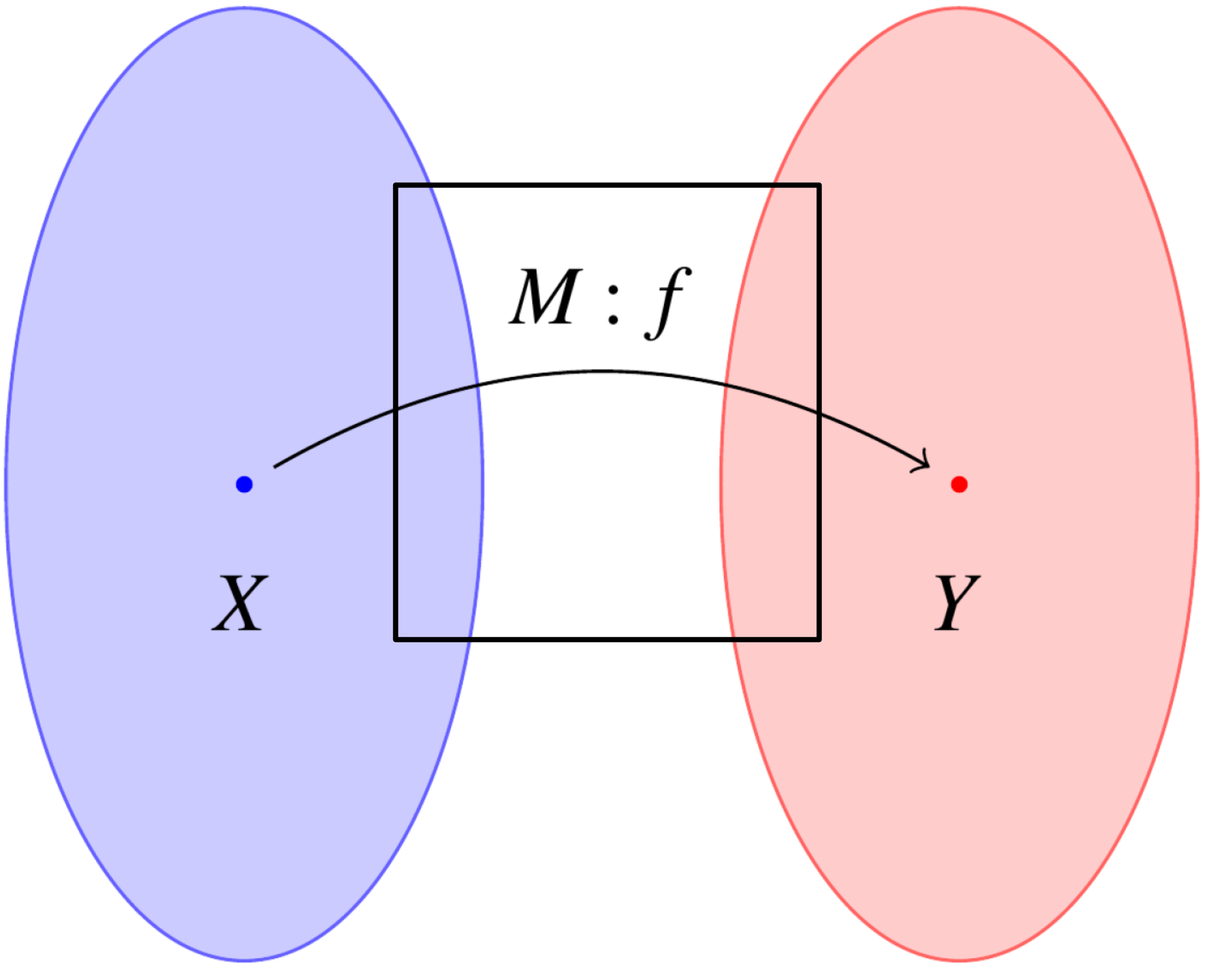
Programs



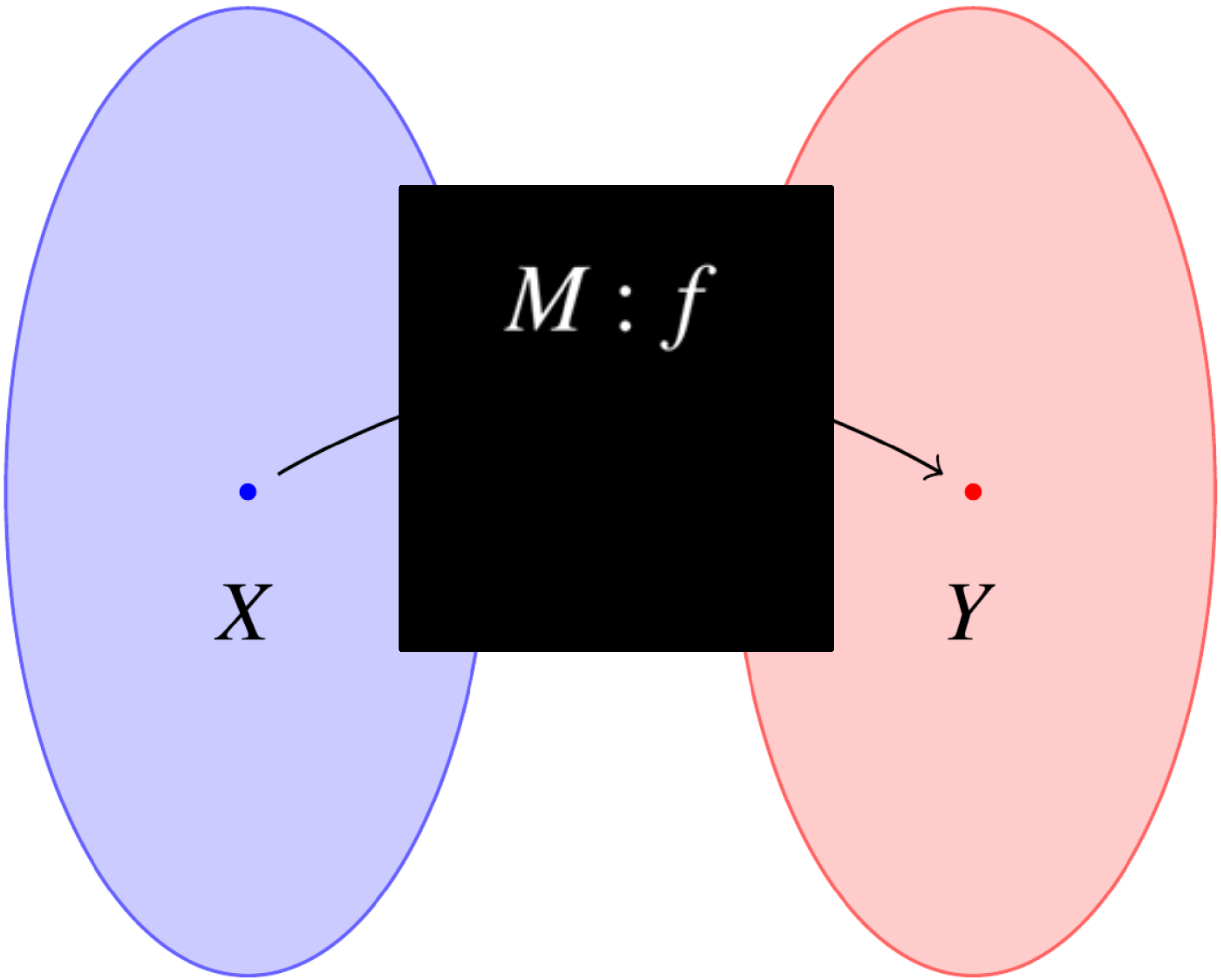
Ideal

Real World

Simplified Program Model



Virtual Black-box Obfuscation



VBB Definition

Definition 2.1 (TM obfuscator) *A probabilistic algorithm* $\mathcal{O}$ *is a* TM obfuscator *if the following three conditions hold:*

- *(functionality) For every TM* M*, the string* $\mathcal{O}(M)$ *describes a TM that computes the same function as* M*.*
- *(polynomial slowdown) The description length and running time of* $\mathcal{O}(M)$ *are at most polynomially larger than that of* M*. That is, there is a polynomial* p *such that for every TM* M*,* $|\mathcal{O}(M)| \le p(|M|)$*, and if* M *halts in* t *steps on some input* x*, then* $\mathcal{O}(M)$ *halts within* $p(t)$ *steps on* x*.*
- *("virtual black box" property) For any PPT* A*, there is a PPT* S *and a negligible function* α *such that for all TMs* M

$$\left|\Pr\left[A(\mathcal{O}(M)) = 1\right] - \Pr\left[S^{\langle M \rangle}(1^{|M|}) = 1\right]\right| \le \alpha(|M|).$$

Obfuscation Results

Multilinear Maps

Indistinguishability Obfuscation

Virtual Black-Box Obfuscation

Hardness Assumptions

The following notation applies to all problems in this section. Let g be an element of prime order r in a group (with group operation written multiplicatively). Let $G = \langle g \rangle$ be the group generated by g.

1. DLP: discrete logarithm problem

Definition: Let notation be as above. Given $h \in G$ to compute x such that $h = g^x$.

5. DDH: decision Diffie-Hellman problem

Definition: Given $g^a, g^b, h \in G$ to determine whether or not $h = g^{ab}$.

Bilinear Hardness Assumption

1. DLP: discrete logarithm problem

Definition: Let notation be as above. Given $h \in G$ to compute x such that $h = g^x$.

5. DDH: decision Diffie-Hellman problem

Definition: Given $g^a, g^b, h \in G$ to determine whether or not $h = g^{ab}$.

DBDH: Decision Bilinear Diffie-Hellman problem

Definition: Given g^a,g^b,g^c,g^{ab},g^{ac},g^{bc}, h in G to determine whether or not $h = g^{abc}$

Multilinear Hardness Assumptions

Multilinear Discrete-log (MDL). The Multilinear Discrete-Log problem is hard for a scheme $\mathcal{MMP}$, if for all $\kappa > 1$, all $i \in [\kappa]$, and all probabilistic polynomial time algorithms, the discrete-logarithm advantage of $\mathcal{A}$,

$$\mathsf{AdvDlog}_{\mathcal{MMP},\mathcal{A},\kappa}(\lambda) \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(\mathsf{params}, i, g_i, \alpha \cdot g_i) = \alpha \;:\; (\mathsf{params}, g_1, \ldots, g_l) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa), \alpha \leftarrow \mathbb{Z}_p\right],$$

is negligible in λ

Multilinear DDH (MDDH). For a symmetric scheme $\mathcal{MMP}$ (with $G_1 = G_2 = \cdots$), the Multilinear Decision-Diffie-Hellman problem is hard for $\mathcal{MMP}$ if for any κ and every probabilistic polynomial time algorithms $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing between the following two distributions is negligible in λ:

$$\begin{aligned} &(\mathsf{params}, g, \alpha_0 g, \alpha_1 g, \ldots, \alpha_\kappa g, \; (\prod_{i=0}^{\kappa} \alpha_i) \cdot e(g \ldots, g)) \\ \text{and} \quad &(\mathsf{params}, g, \alpha_0 g, \alpha_1 g, \ldots, \alpha_\kappa g, \; \alpha \cdot e(g, \ldots, g)) \end{aligned}$$

where $(\mathsf{params}, g) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$ and $\alpha, \alpha_0, \alpha_1, \ldots, \alpha_\kappa$ are uniformly random in $\mathbb{Z}_p$.

Algebraic Structures

Standard:

- Sets
- Groups
- Rings
- Fields

Advanced:

- Polynomial Quotient Rings

iO Definition

Definition 1 (Indistinguishability Obfuscator ($i\mathcal{O}$)). A uniform PPT machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:

- For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$$

- For any (not necessarily uniform) PPT distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then

$$\Big| \Pr\left[D(i\mathcal{O}(\lambda, C_0)) = 1\right] - \Pr\left[D(i\mathcal{O}(\lambda, C_1)) = 1\right] \Big| \leq \alpha(\lambda)$$

Indistinguishability Obfuscation Process

Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

Indistinguishability Obfuscation Process

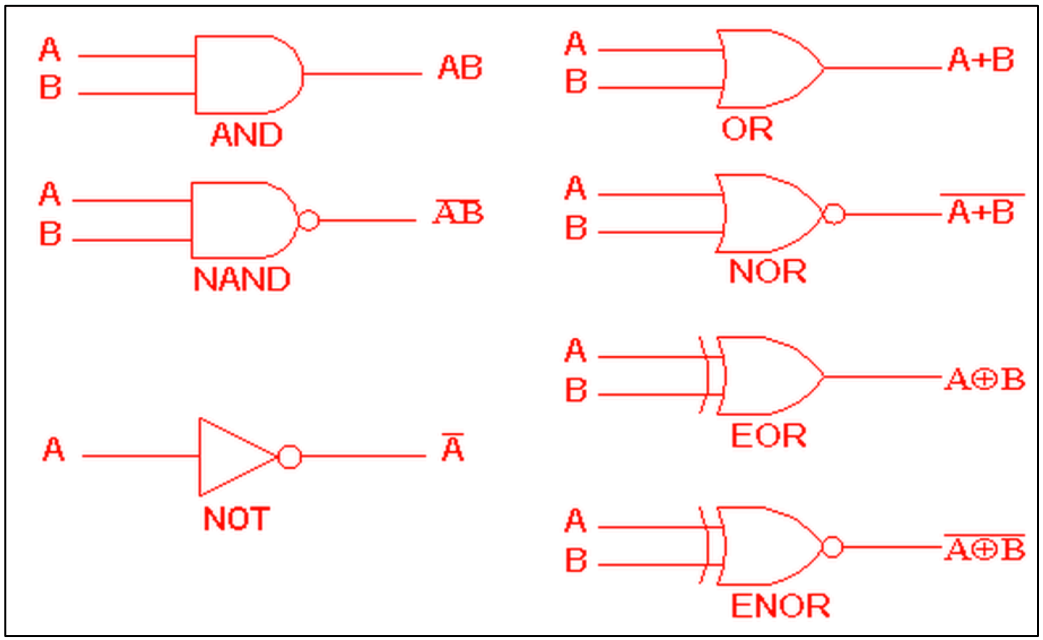
Functionality

1) **Boolean Formula/Circuit**
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

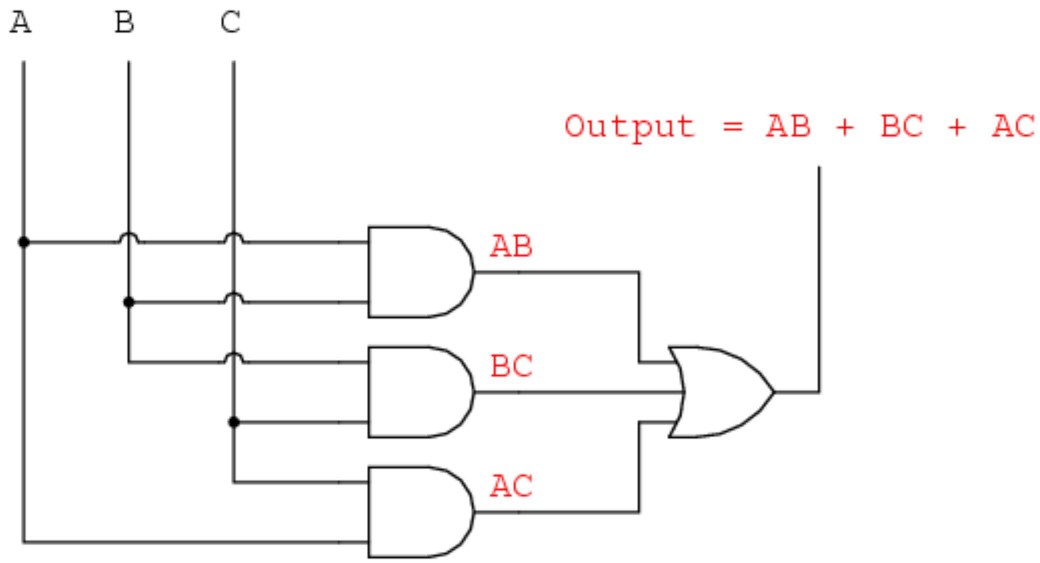
6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

1) Boolean Circuit/Formula



p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Circuit and Formula



Complexity Class

NC: Nick's Class

(Named in honor of Nick Pippenger.)

NC^i is the class of decision problems solvable by a uniform family of Boolean circuits, with polynomial size, depth $O(\log^i(n))$, and fan-in 2.

Complexity Class

NC: Nick's Class

(Named in honor of Nick Pippenger.)

NC^i is the class of decision problems solvable by a uniform family of Boolean circuits, with polynomial size, depth $O(\log^i(n))$, and fan-in 2.

We are working in NC^1

Indistinguishability Obfuscation Process

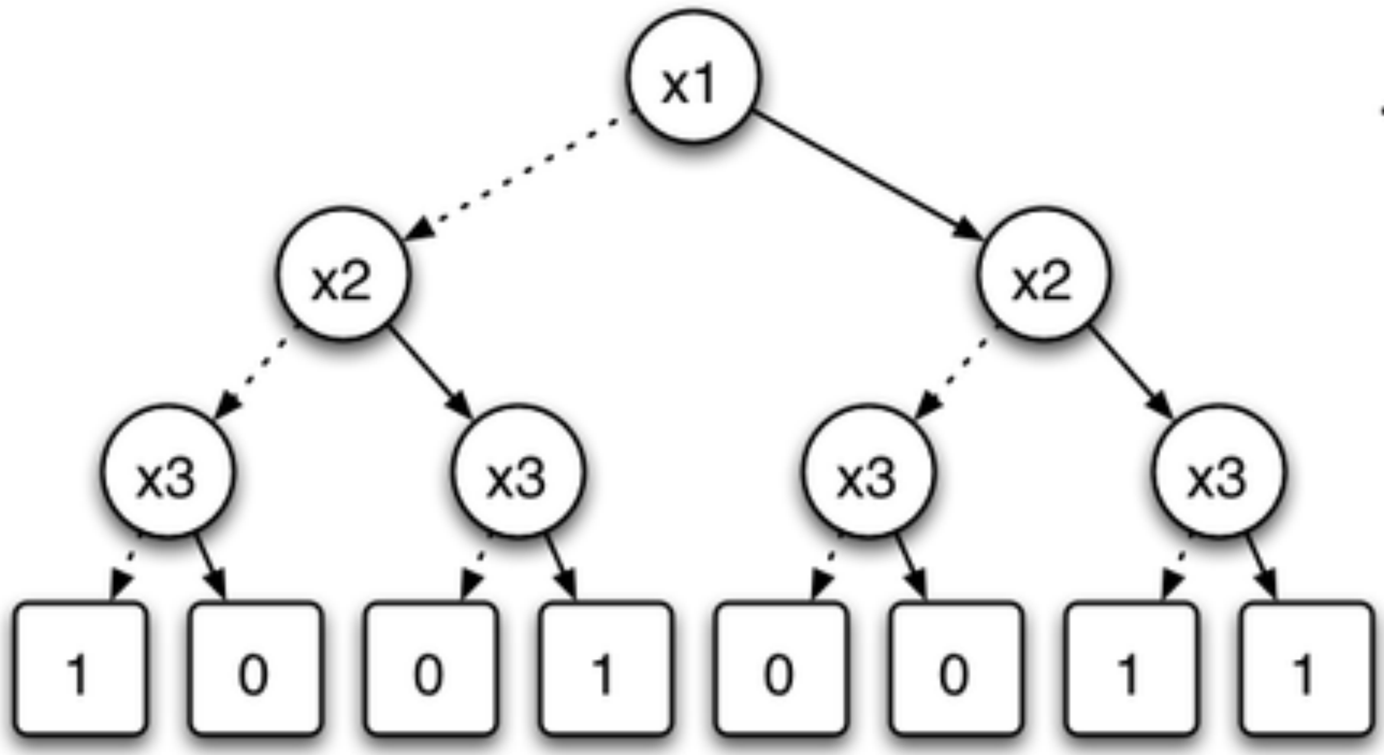
Functionality

1) Boolean Formula/Circuit
2) **Binary Decision Diagram**
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

2) Binary Decision Diagram



x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Binary Decision Diagram

A binary decision diagram is often a convenient way to store a boolean function.

Definition 12.1. *A* binary decision diagram *of size s is a directed acyclic graph on s vertices with the following constraints:*

- *Each vertex is either an internal vertex or a final vertex.*
- *Internal vertices are labelled with one of $\{x_1, \ldots, x_n\}$. They have outdegree 2, and one of the outgoing edges is labelled with 0, the other one with 1.*
- *Final vertices are labelled with an element of some set S, and have outdegree 0.*
- *There is a unique starting vertex.*

Indistinguishability Obfuscation Process

Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) **Branching Program (BP)**
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

3) Branching Program

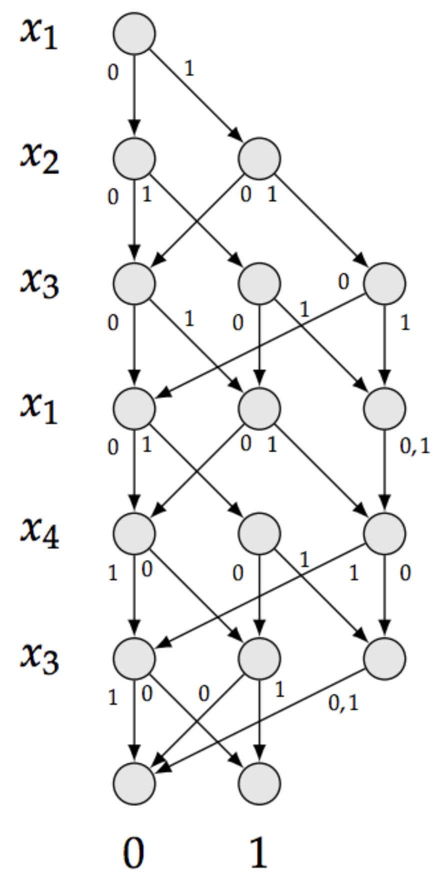


Figure 2: A branching program of width 3 and 6 layers computing a function $f : \{0,1\}^4 \to \{0,1\}$. In order to evaluate $f(x_1, x_2, x_3, x_4)$ one starts at the top, and follows the arrows whose label corresponds to x_i. For example, $f(0010) = 1$.

Branching Program

Definition 12.2. *A* branching program *of width W is a binary decision diagram with the following constraints:*

- *The vertices are partitioned in layers L_i, $|L_i| \leq W$.*
- *All vertices in a layer have the same label x_j.*
- *All edges starting in L_i end in L_{i+1}.*
- *All final vertices are in the same layer.*

Indistinguishability Obfuscation Process

Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) **Oblivious Linear Branching Program through Barrington's Theorem**
5) Branching Program represented with permutation matrices

Security

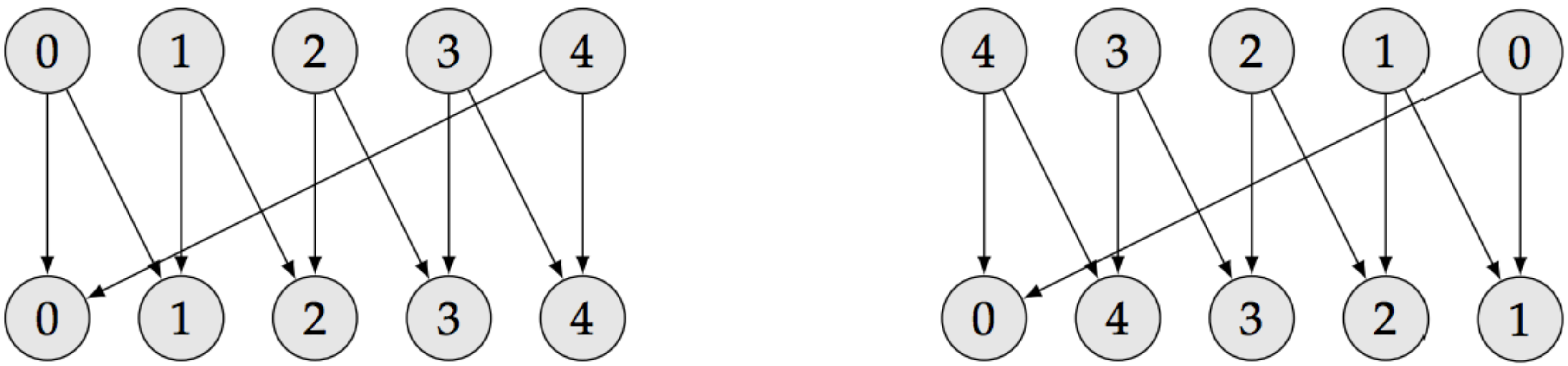
6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

4) Barrinton's Theorem

Theorem 12.4 (Barrington). *Let C be a circuit of depth d and 1 bit of output. There exists a branching program of width 5 and with 4^d layers which computes the same function as C.*

Lemma 12.5. *There exist cyclic permutations α and β over $\{0,1,2,3,4\}$ such that* $\beta^{-1} \circ \alpha^{-1} \circ \beta \circ \alpha = (0\,1\,2\,3\,4)$.

Not-Gate



$(0\,1\,2\,3\,4)^{g_j(x)}$ $\qquad$ $(0\,1\,2\,3\,4)^{\neg g_j(x)}$

Figure 4: Relabelling the vertices to implement a ¬-gate.

Barrington's Commutator

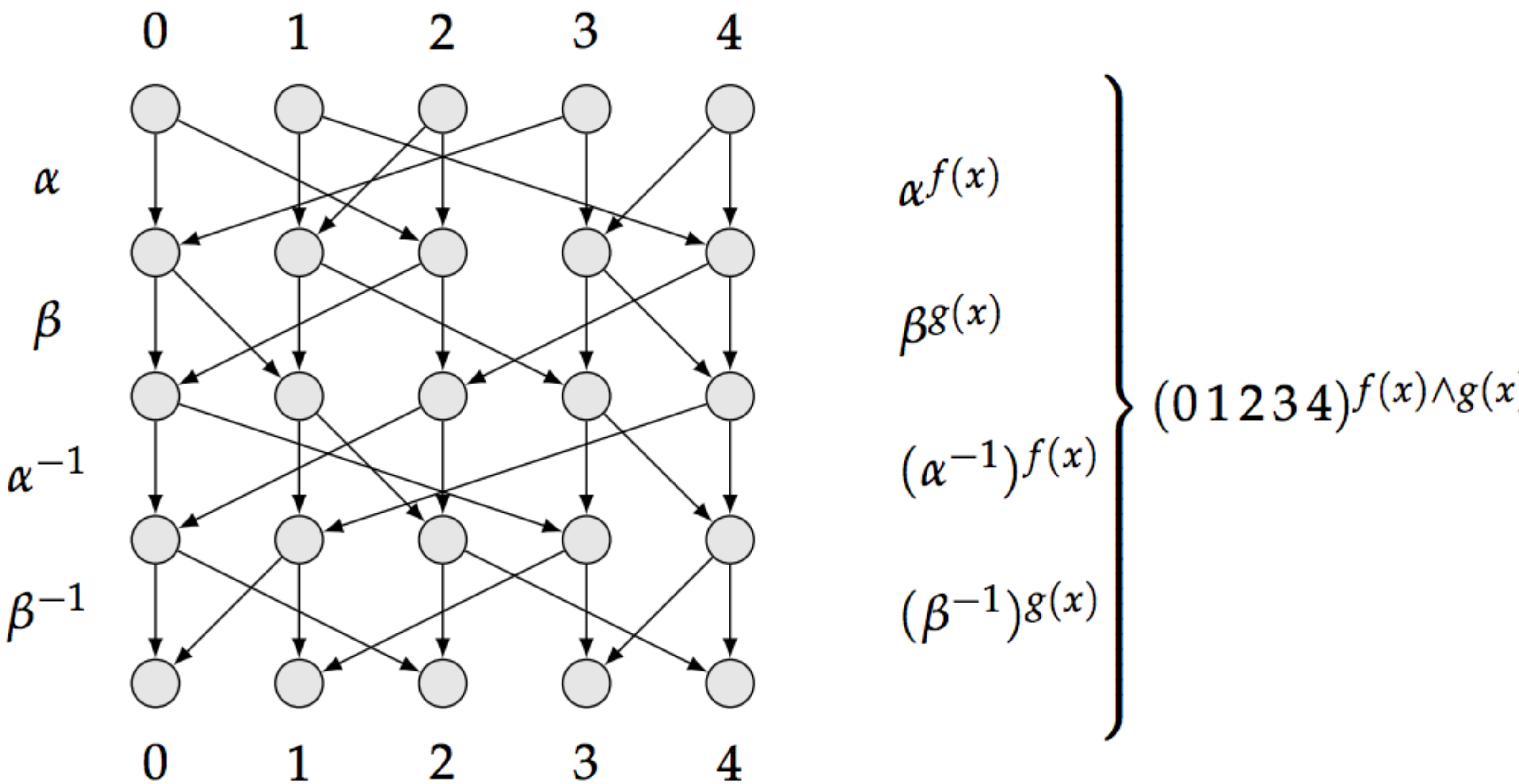


Figure 3: Permutations used in the proof of Barrington's theorem.

Oblivious Linear Branching Program

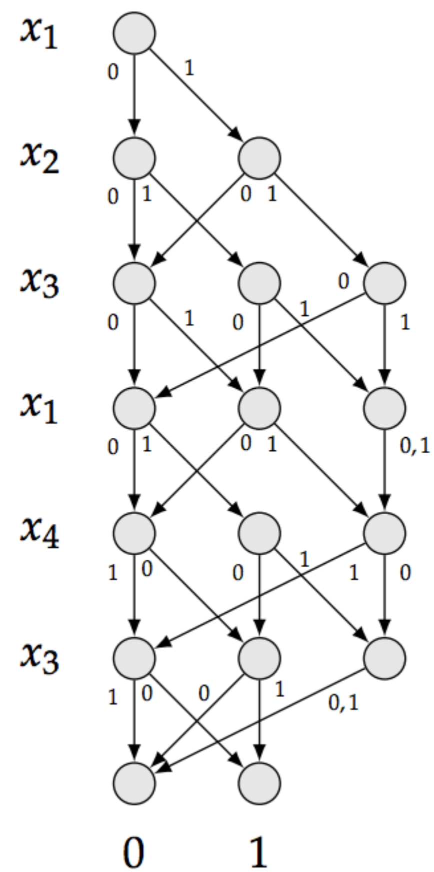


Figure 2: A branching program of width 3 and 6 layers computing a function $f : \{0,1\}^4 \to \{0,1\}$. In order to evaluate $f(x_1, x_2, x_3, x_4)$ one starts at the top, and follows the arrows whose label corresponds to x_i. For example, $f(0010) = 1$.

Indistinguishability Obfuscation Process

Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) **Branching Program represented with permutation matrices**

Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

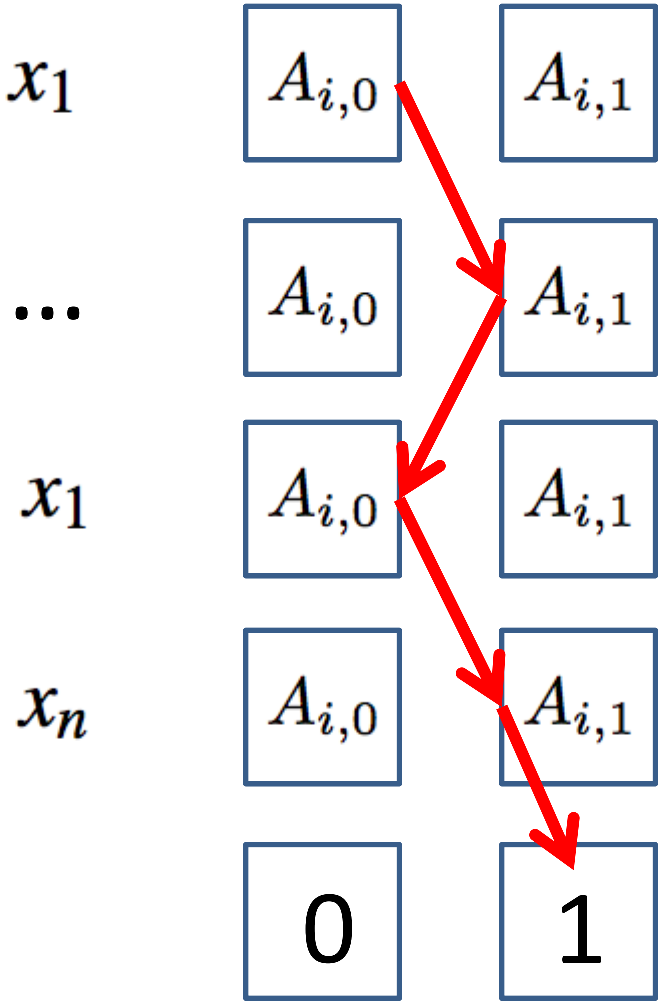
5) Oblivious Linear Branching Program with Matrices

$$BP = \{(\mathsf{inp}(i), A_{i,0}, A_{i,1}) : i \in [n], \mathsf{inp}(i) \in [\ell], A_{i,b} \in \{0,1\}^{5\times 5}\},$$

x_1	$A_{i,0}$	$A_{i,1}$
...	$A_{i,0}$	$A_{i,1}$
x_1	$A_{i,0}$	$A_{i,1}$
x_n	$A_{i,0}$	$A_{i,1}$
	0	1

Oblivious Linear Branching Program

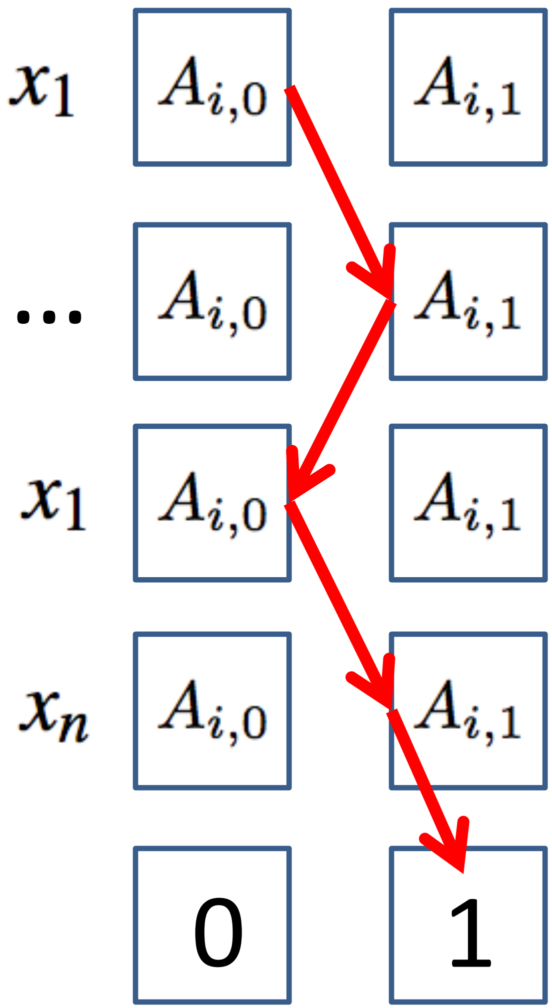
$$BP = \{(\mathsf{inp}(i), A_{i,0}, A_{i,1}) : i \in [n], \mathsf{inp}(i) \in [\ell], A_{i,b} \in \{0,1\}^{5\times 5}\},$$



Input = 0101

Oblivious Linear Branching Program

$$BP = \{(\mathsf{inp}(i), A_{i,0}, A_{i,1}) : i \in [n], \mathsf{inp}(i) \in [\ell], A_{i,b} \in \{0,1\}^{5\times 5}\},$$



Input = 0101

$$\boxed{A_{i,0}} \times \boxed{A_{i,1}} \times \boxed{A_{i,0}} \times \boxed{A_{i,1}} = \boxed{1}$$

Indistinguishability Obfuscation Process

Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

6) **Dummy Program and Random Scalar multiplication of $A_{i,b}$**
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

6) Dummy Branching Program

x_1	$A_{i,0}$	$A_{i,1}$	I	I
…	$A_{i,0}$	$A_{i,1}$	I	I
c x_1	$A_{i,0}$	$A_{i,1}$	I	I
x_n	$A_{i,0}$	$A_{i,1}$	I	I
	0	1	0	1

Dummy Program with Scalar Multiplication

$\{\alpha_{i,b_1,b_2} \in \mathbb{Z}_p$ $\quad \prod_{\mathsf{inp}(i)=j} \alpha_{i,b} = \prod_{\mathsf{inp}(i)=j} \alpha'_{i,b}$ for all $j \in [\ell],\ b \in \{0,1\}.$

x_1	3x	$A_{i,0}$	$A_{i,1}$	7x	I	I
…		$A_{i,0}$	$A_{i,1}$		I	I
x_1	7x	$A_{i,0}$	$A_{i,1}$	3x	I	I
x_n		$A_{i,0}$	$A_{i,1}$		I	I
		0	1		0	1

Small Matrix in Big Matrix

$\boxed{A_{i,0}}$ $\boxed{A_{i,1}}$ $\boxed{I}$ $\boxed{I}$

Indistinguishability Obfuscation Process

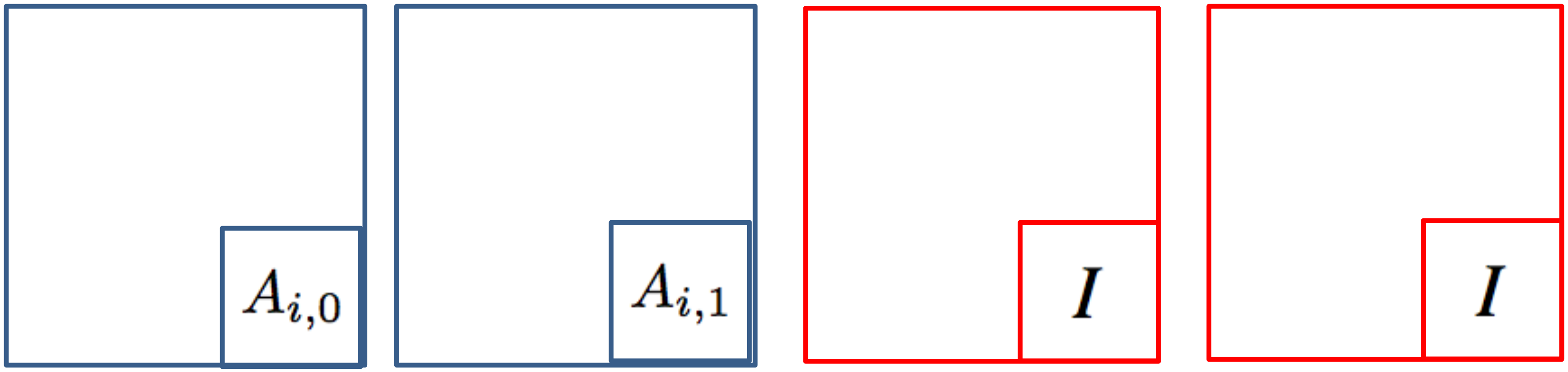
Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

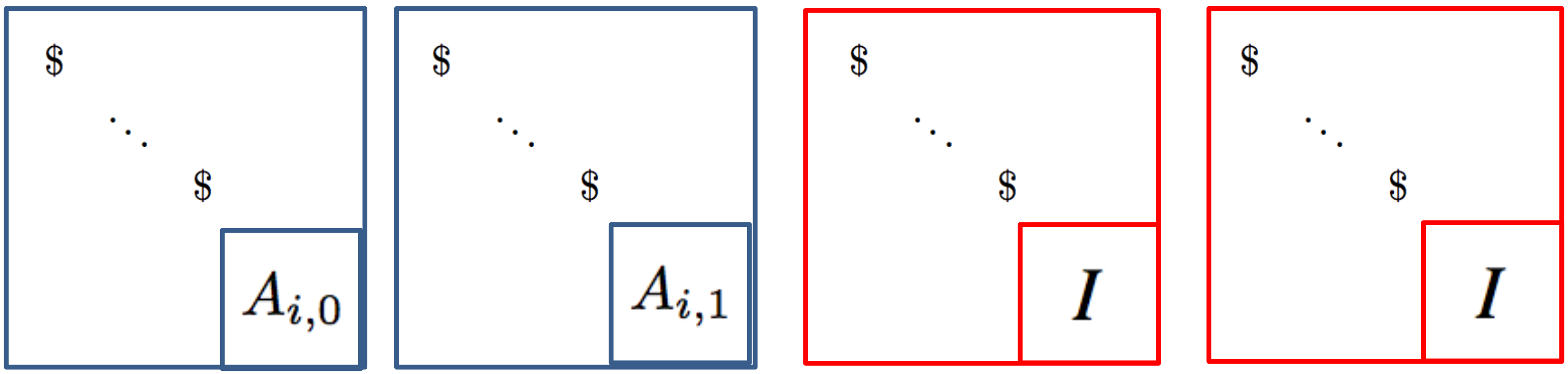
6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) **Small Matrix in Big Matrix**
8) Sandwich Vectors
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

7) Small Matrix in Big Matrix



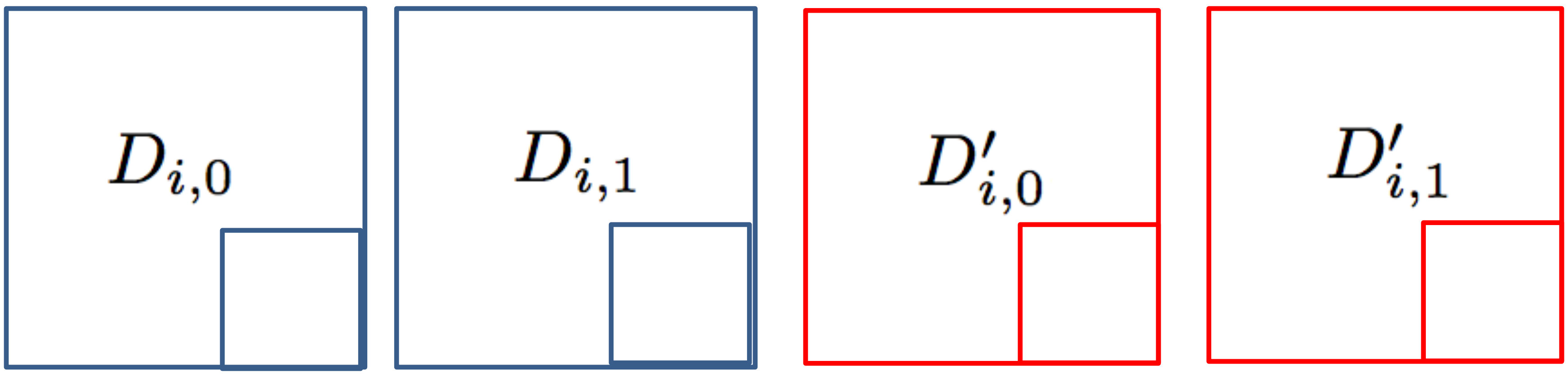
$$(2m+5) \times (2m+5)$$

Small Matrix in Big Matrix



$$(2m + 5) \times (2m + 5)$$

Small Matrix in Big Matrix



Indistinguishability Obfuscation Process

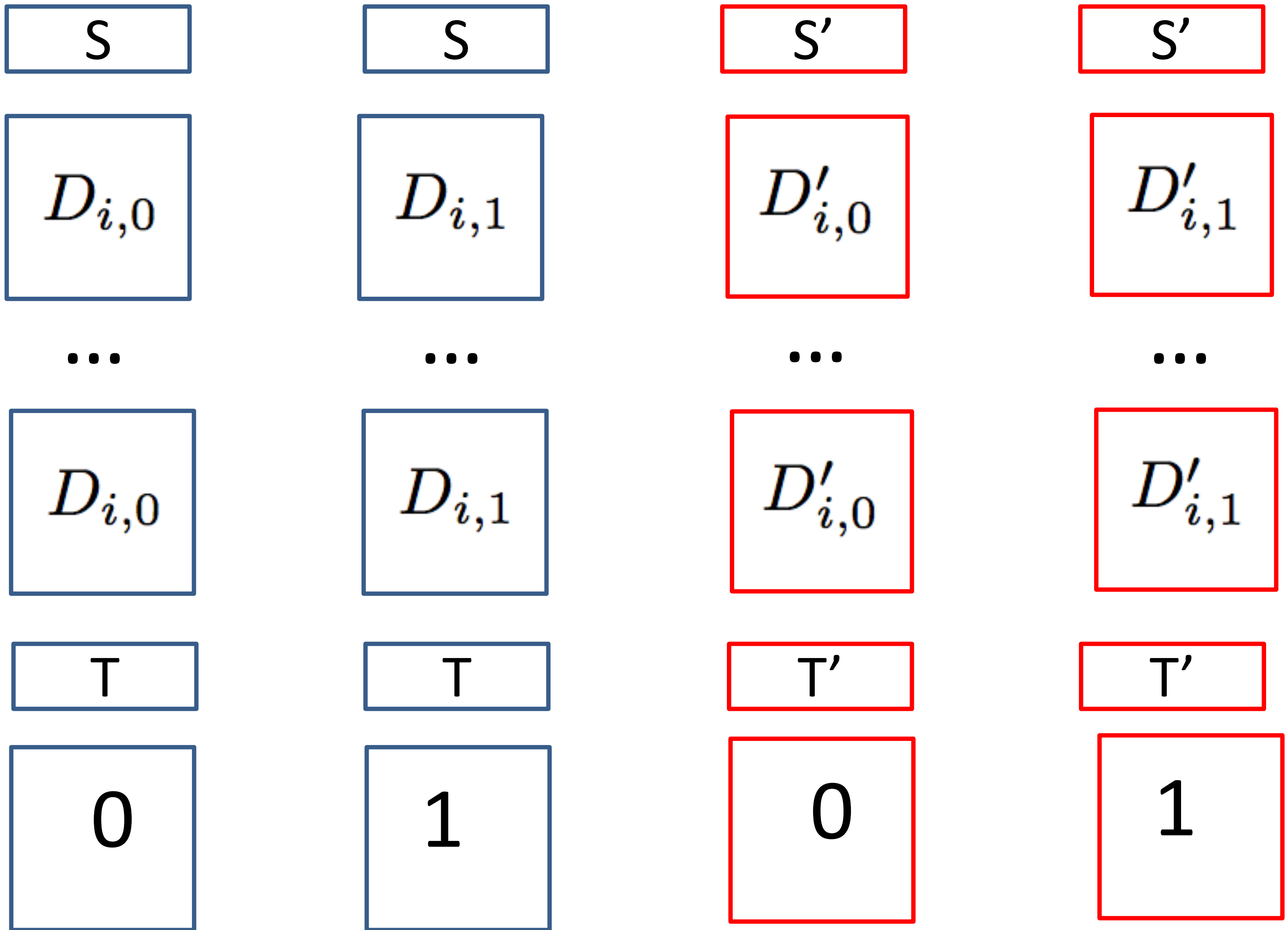
Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington’s Theorem
5) Branching Program represented with permutation matrices

Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) **Sandwich Vectors**
9) Flank by Random Matrices
10) Encode all Matrix elements in multilinear map

8) Sandwich Vectors



Sandwich Vectors

$$(2m + 5)$$

S

S’

T

T’

Sandwich Vectors

- Choose two pairs of random 5-vectors $\mathbf{s}^*$ and $\mathbf{t}^*$, and $\mathbf{s}'^*$ and $\mathbf{t}'^*$, such that $\langle \mathbf{s}^*, \mathbf{t}^* \rangle = \langle \mathbf{s}'^*, \mathbf{t}'^* \rangle$. The last 5 entries in $\mathbf{s}$ are set to $\mathbf{s}^*$, the last 5 entries in $\mathbf{t}$ are set to $\mathbf{t}^*$. The last 5 entries in $\mathbf{s}'$ are set to $\mathbf{s}'^*$, the last 5 entries in $\mathbf{t}'$ are set to $\mathbf{t}'^*$.

$$\mathbf{s} \sim (0 \ldots 0 \; \$ \ldots \$ \;\; \text{-}\; \mathbf{s}^* \text{-}\;), \qquad \mathbf{t} \sim (\$ \ldots \$ \;\; 0 \ldots 0 \;\; \text{-}\; \mathbf{t}^* \text{-}\;)^T$$
$$\mathbf{s}' \sim (0 \ldots 0 \; \$ \ldots \$ \;\; \text{-}\; \mathbf{s}'^* \text{-}\;), \qquad \mathbf{t}' \sim (\$ \ldots \$ \;\; 0 \ldots 0 \;\; \text{-}\; \mathbf{t}'^* \text{-}\;)^T.$$

Sandwich Vectors

$$(2m + 5)$$

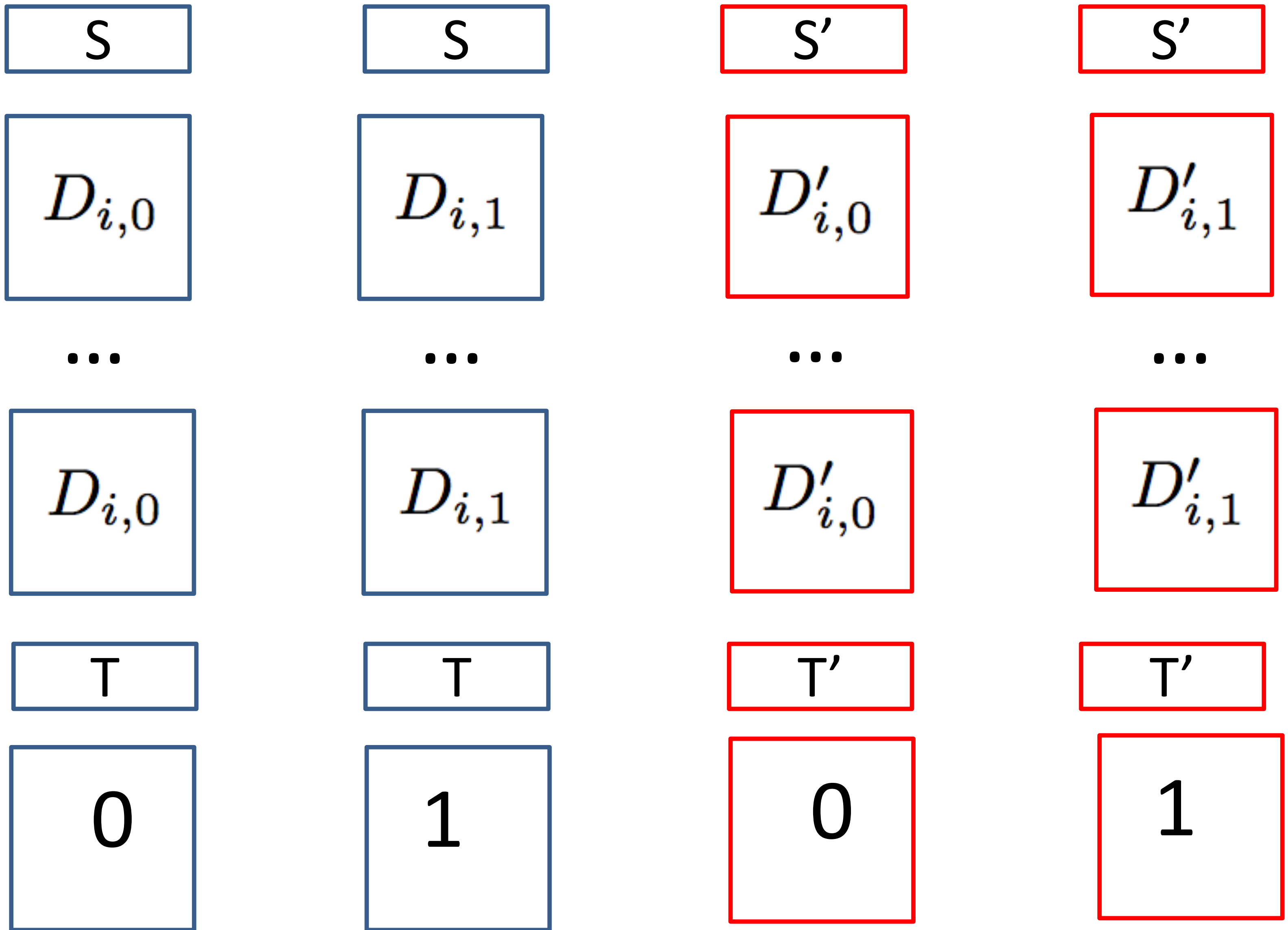
$...$ 0...0 S* S* S* S* S*

$...$0...0 S*' S*' S*' S*' S*'

0...0$$...$ T* T* T* T* T*

0...0$$...$T*'T*'T*'T*'T*'

Sandwich Vectors



Indistinguishability Obfuscation Process

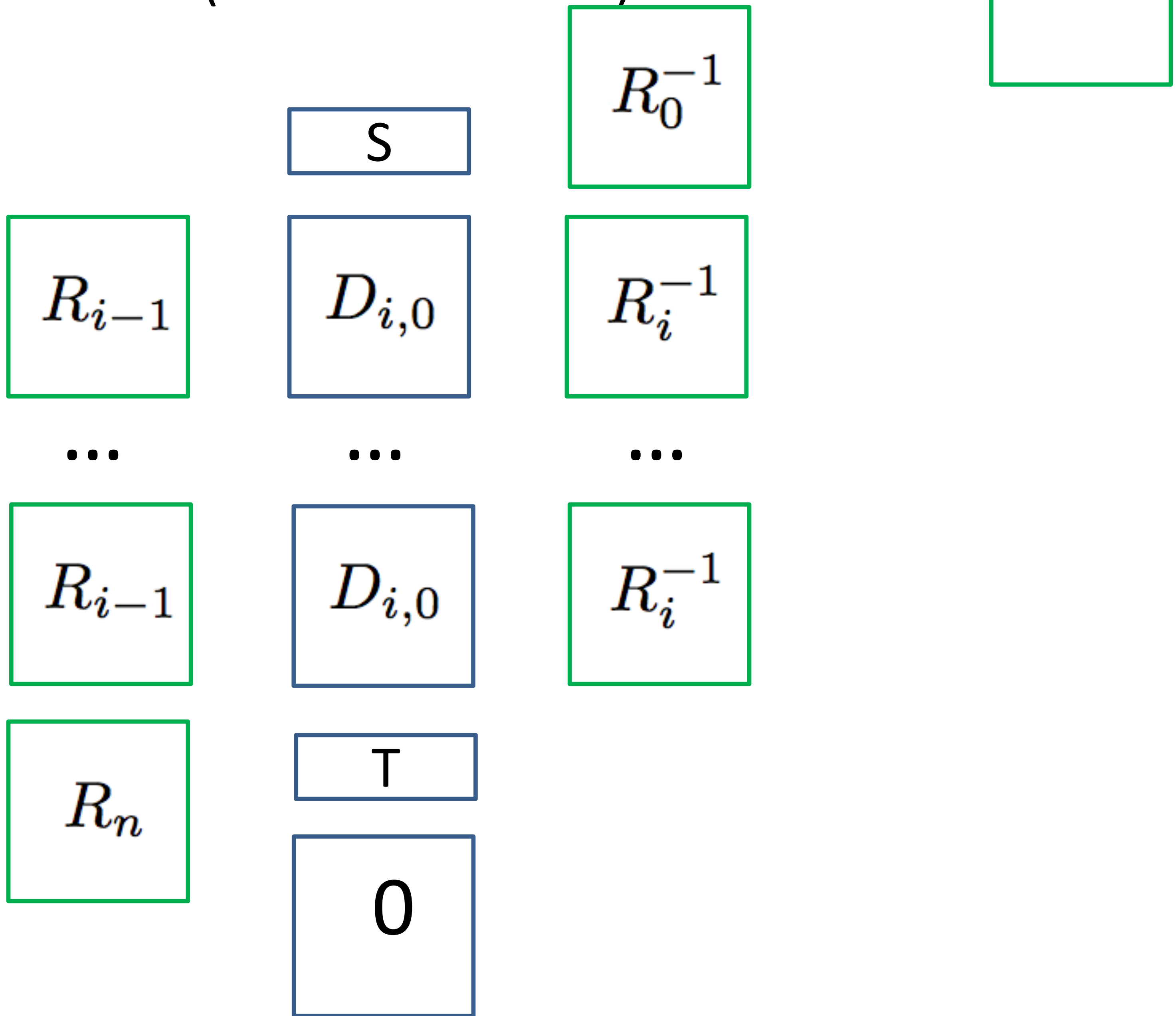
Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

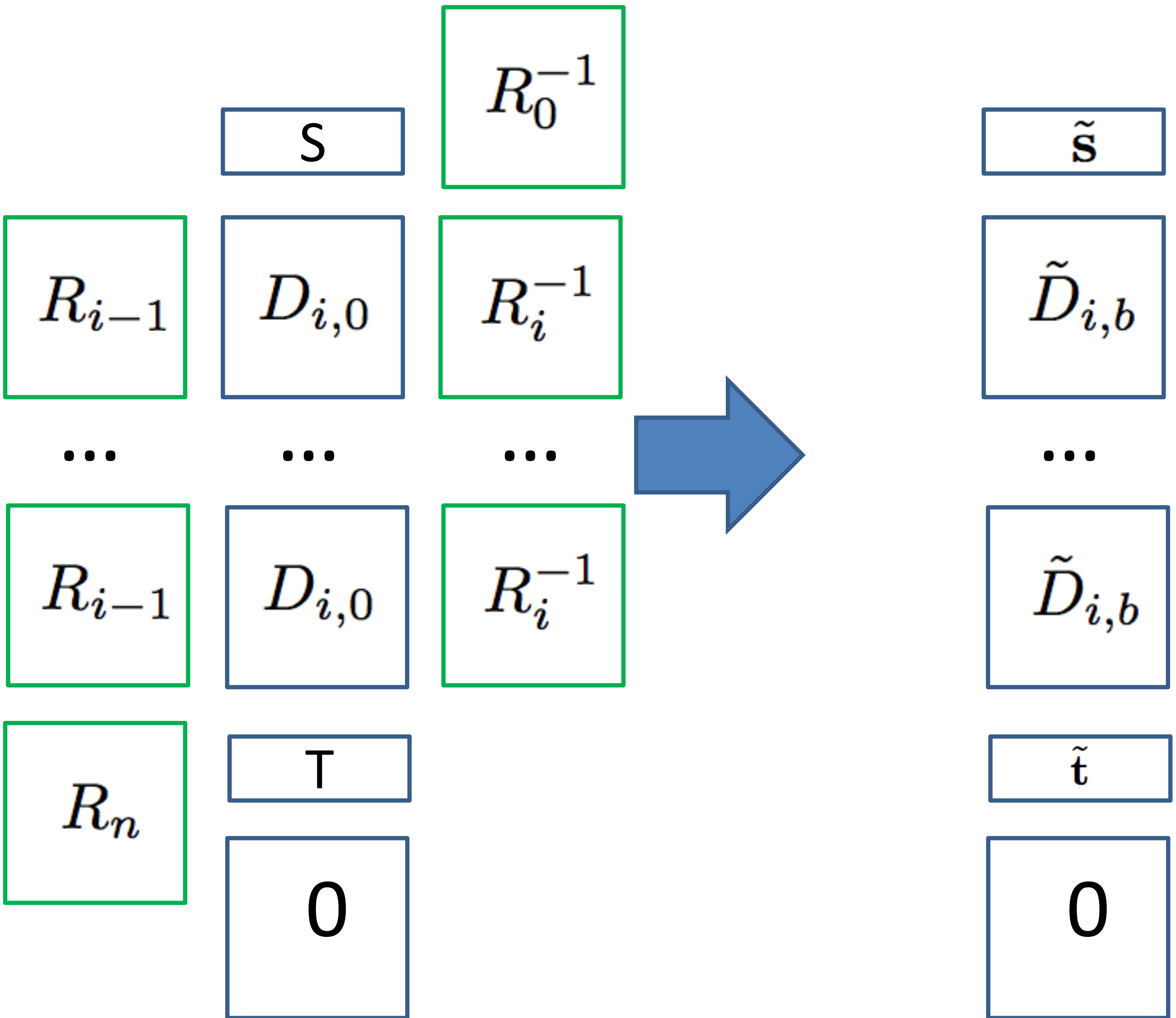
Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) **Flank by Random Matrices**
10) Encode all Matrix elements in multilinear map

9) Flanked by Random Matrices (Killian's Protocol)



Flanked by Random Matrices



Randomized Branching Program

$$\mathcal{RND}_p(BP) =$$

$$\left\{ \begin{array}{l} \tilde{\mathbf{s}} = \mathbf{s}R_0^{-1},\ \tilde{\mathbf{t}} = R_n\mathbf{t}, \\ \{\tilde{D}_{i,b} = R_{i-1}D_{i,b}R_i^{-1} : i \in [n], b \in \{0,1\}\}, \\ \\ \tilde{\mathbf{s}}' = \mathbf{s}'(R_0')^{-1},\ \tilde{\mathbf{t}}' = R_n'\mathbf{t}' \\ \{\tilde{D}_{i,b}' = R_{i-1}'D_{i,b}'(R_i')^{-1} : i \in [n], b \in \{0,1\}\} \end{array} \right\}$$

Indistinguishability Obfuscation Process

Functionality

1) Boolean Formula/Circuit
2) Binary Decision Diagram
3) Branching Program (BP)
4) Oblivious Linear Branching Program through Barrington's Theorem
5) Branching Program represented with permutation matrices

Security

6) Dummy Program and Random Scalar multiplication of $A_{i,b}$
7) Small Matrix in Big Matrix
8) Sandwich Vectors
9) Flank by Random Matrices
10) **Encode all Matrix elements in multilinear map**

10) Encode in Multilinear Map

g^a	$\tilde{\mathbf{s}}$	$\tilde{\mathbf{s}}$	$\tilde{\mathbf{s}}'$	$\tilde{\mathbf{s}}'$
g^b	$\tilde{D}_{i,b}$	$\tilde{D}_{i,b}$	$\tilde{D}'_{i,b}$	$\tilde{D}'_{i,b}$
	…	…	…	…
g^y	$\tilde{D}_{i,b}$	$\tilde{D}_{i,b}$	$\tilde{D}'_{i,b}$	$\tilde{D}'_{i,b}$
g^z	$\tilde{\mathbf{t}}$	$\tilde{\mathbf{t}}$	$\tilde{\mathbf{t}}'$	$\tilde{\mathbf{t}}'$
	0	1	0	1

Encode in Multilinear Map

g^a	$\tilde{\mathbf{s}}$	$\tilde{\mathbf{s}}$	$\tilde{\mathbf{s}}'$	$\tilde{\mathbf{s}}'$
g^b	$\tilde{D}_{i,b}$	$\tilde{D}_{i,b}$	$\tilde{D}'_{i,b}$	$\tilde{D}'_{i,b}$
	…	…	…	…
g^y	$\tilde{D}_{i,b}$	$\tilde{D}_{i,b}$	$\tilde{D}'_{i,b}$	$\tilde{D}'_{i,b}$
g^z	$\tilde{\mathbf{t}}$	$\tilde{\mathbf{t}}$	$\tilde{\mathbf{t}}'$	$\tilde{\mathbf{t}}'$
$g^{abc\ldots z}$	0	1	0	1

Encoding in Multilinear Map

- First, she reduces a modulo (g) to create a small polynomial $\hat{a} \equiv a \pmod{g}$ in R.
- Then, she computes in R_q the following:

$$\mathsf{Encode}_S(a) = \frac{\hat{a} + e \cdot g}{\prod_{i \in S} z_i}$$

Garbled Branching Program

$$\widehat{\mathcal{RND}}_p(BP) = \left\{ \begin{array}{l} \mathsf{prms}, \quad \hat{\mathbf{s}} = \mathsf{Encode}_{\{1\}}(\tilde{\mathbf{s}}),\ \hat{\mathbf{t}} = \mathsf{Encode}_{\{n+2\}}(\tilde{\mathbf{t}}), \\ \left\{\hat{D}_{i,b} = \mathsf{Encode}_{\{i+1\}}(\tilde{D}_{i,b}) : i \in [n], b \in \{0,1\}\right\}, \\ \\ \hat{\mathbf{s}}' = \mathsf{Encode}_{\{1\}}(\tilde{\mathbf{s}}'),\ \hat{\mathbf{t}}' = \mathsf{Encode}_{\{n+2\}}(\tilde{\mathbf{t}}') \\ \left\{\hat{D}'_{i,b} = \mathsf{Encode}_{\{i+1\}}(\tilde{D}'_{i,b}) : i \in [n], b \in \{0,1\}\right\} \end{array} \right\}.$$

Program Evaluation

$$\mathcal{F}_{\chi}\big(\mathcal{RND}_p(BP)\big) \;=\; \tilde{\mathbf{s}}\Big(\prod_i \tilde{D}_{i,\chi_{\mathsf{inp}(i)}}\Big)\tilde{\mathbf{t}} \;-\; \tilde{\mathbf{s}}'\Big(\prod_i \tilde{D}'_{i,\chi_{\mathsf{inp}(i)}}\Big)\tilde{\mathbf{t}}' \bmod p.$$

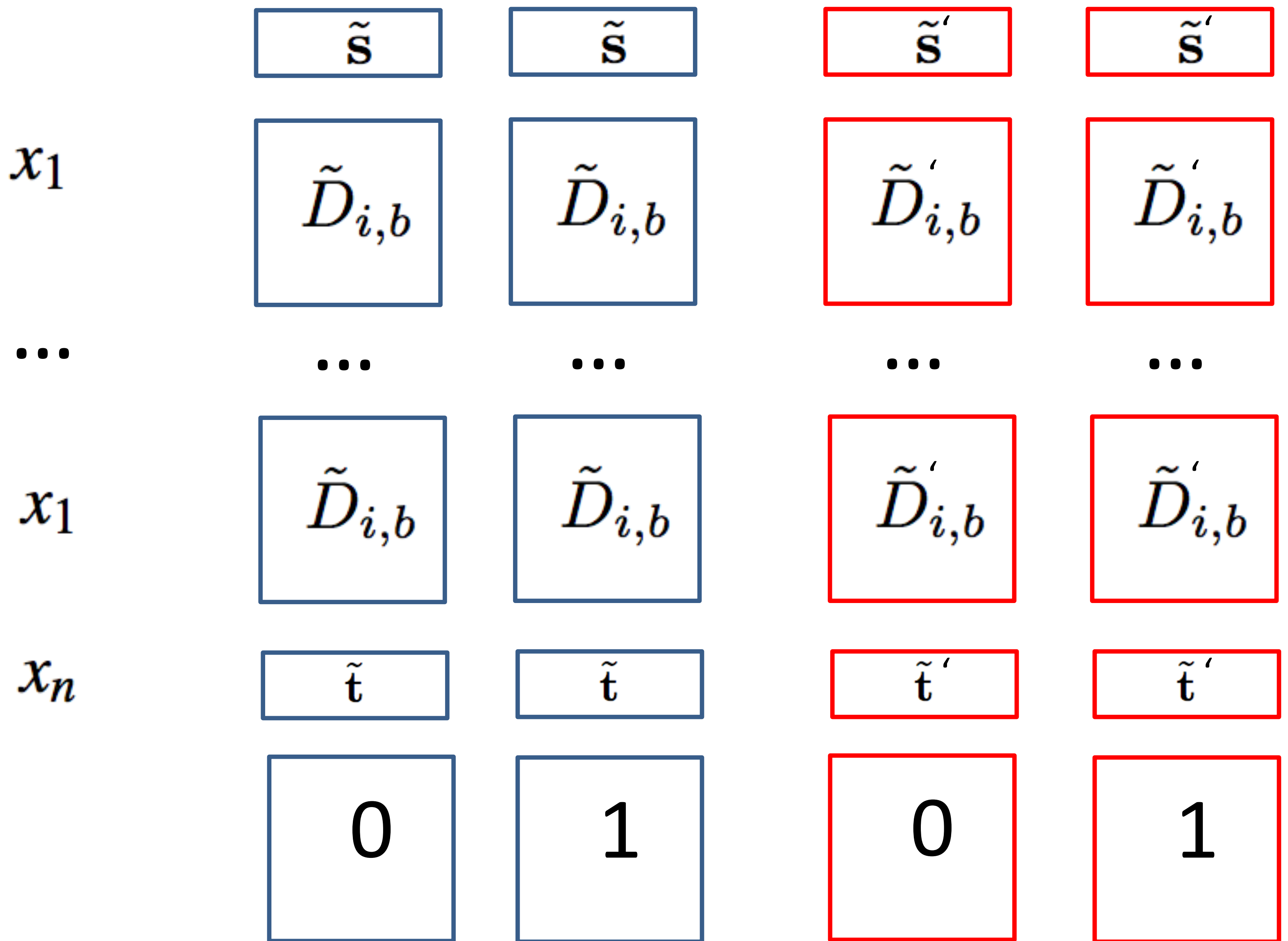
Program Evaluation

$$\mathcal{F}_\chi(\mathcal{RND}_p(BP)) \;=\; \tilde{\mathbf{s}}\Big(\prod_i \tilde{D}_{i,\chi_{\mathsf{inp}(i)}}\Big)\tilde{\mathbf{t}} \;-\; \tilde{\mathbf{s}}'\Big(\prod_i \tilde{D}'_{i,\chi_{\mathsf{inp}(i)}}\Big)\tilde{\mathbf{t}}' \bmod p.$$

-Publish 0 encoded at $g^{abc\ldots z}$

-Compare result of program to 0

Fix Inputs in Program

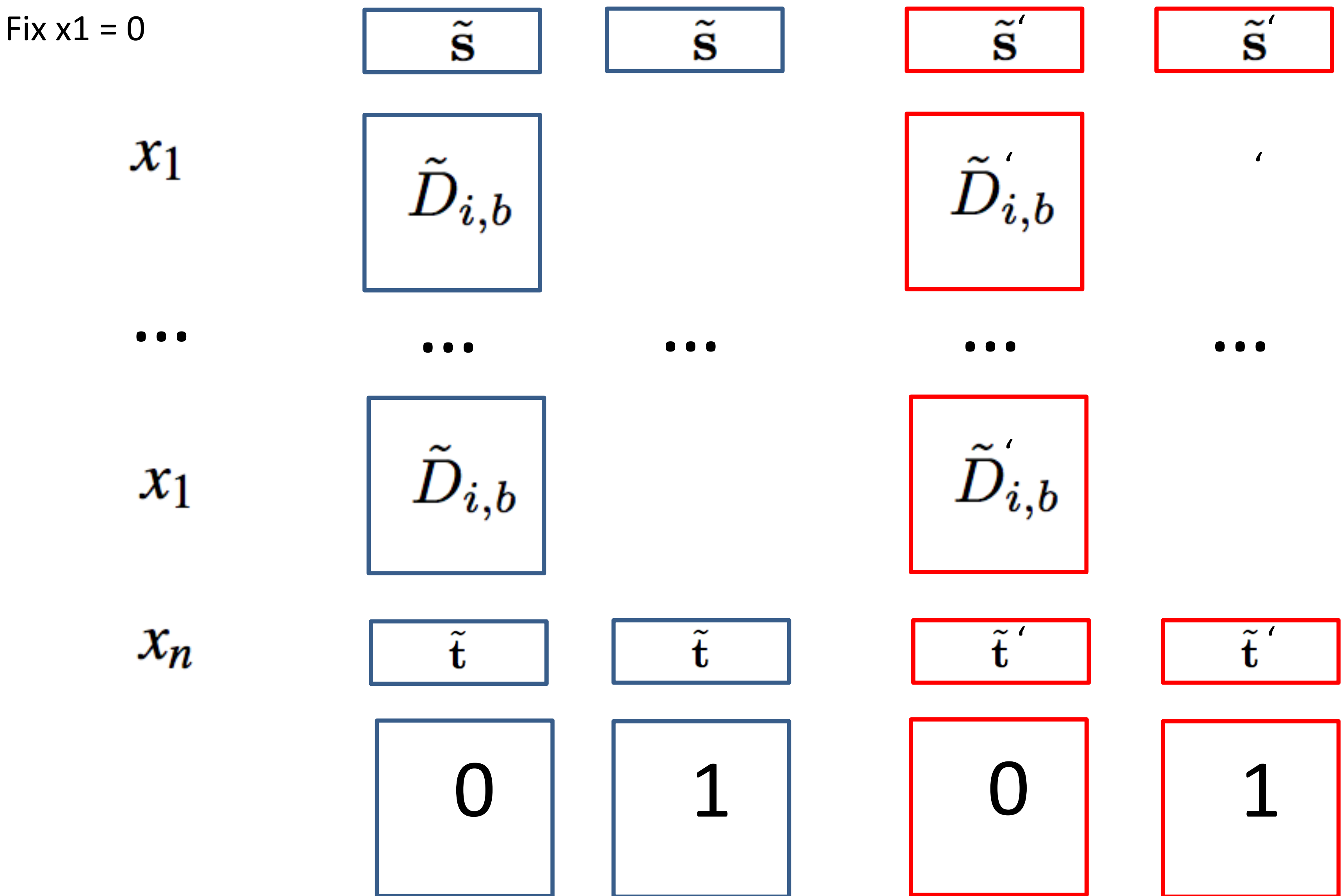


Fix Inputs in Program

Fix x1 = 0

	$\tilde{\mathbf{s}}$	$\tilde{\mathbf{s}}$	$\tilde{\mathbf{s}}'$	$\tilde{\mathbf{s}}'$
x_1	$\tilde{D}_{i,b}$	$\tilde{D}_{i,b}$	$\tilde{D}'_{i,b}$	$\tilde{D}'_{i,b}$
…	…	…	…	…
x_1	$\tilde{D}_{i,b}$	$\tilde{D}_{i,b}$	$\tilde{D}'_{i,b}$	$\tilde{D}'_{i,b}$
x_n	$\tilde{\mathbf{t}}$	$\tilde{\mathbf{t}}$	$\tilde{\mathbf{t}}'$	$\tilde{\mathbf{t}}'$
	0	1	0	1

Fix Inputs in Program



Input Fixed Program

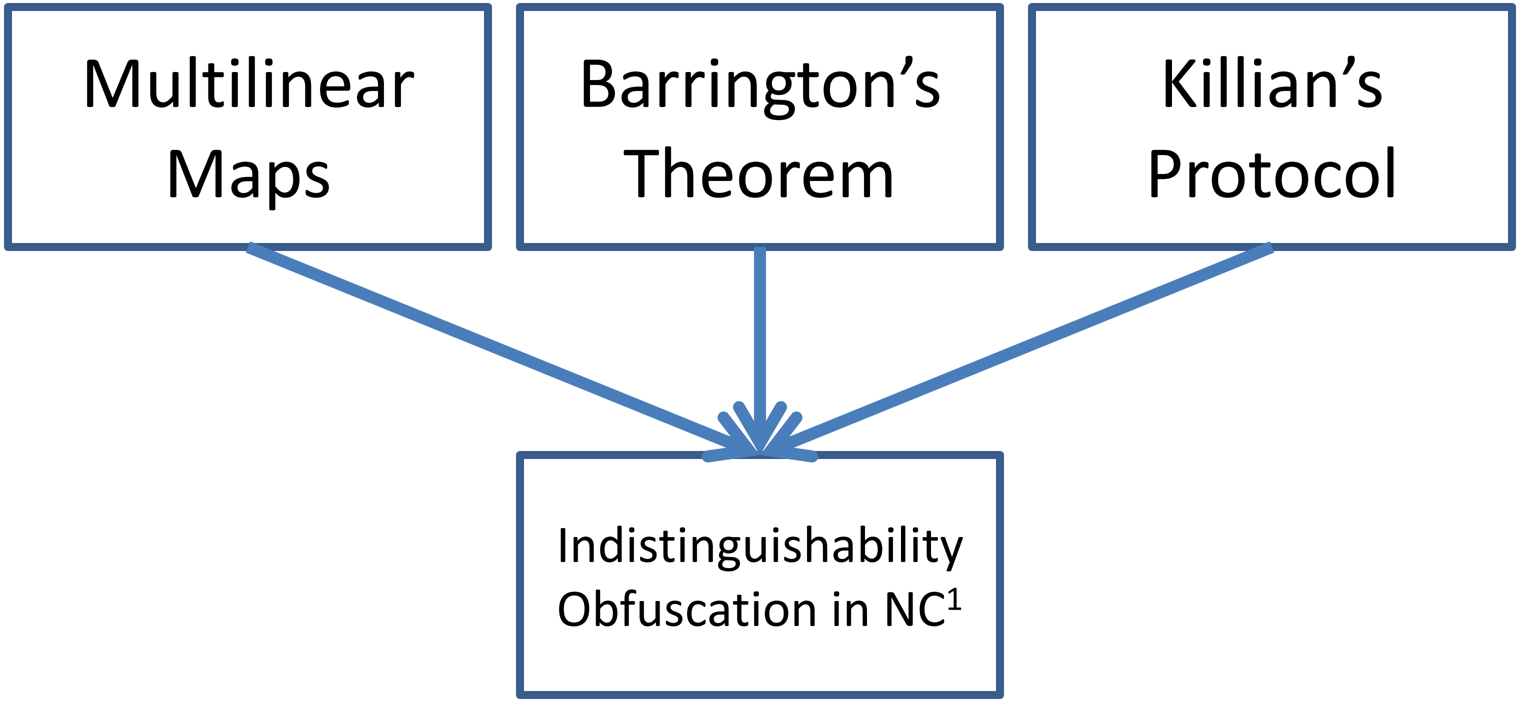
Input-fixing. Given $\widehat{\mathcal{RND}}_p(BP)$ as above and a partial assignment for the input bits, $\sigma : J \to \{0,1\}$ (for $J \subset [\ell]$), the Parameter-fixing procedure just removes all the matrices $\tilde{D}_{i,b}$, $\tilde{D}'_{i,b}$ that are not consistent with that partial assignment σ (i.e., where $i \in I_j$ and $b \neq \sigma(\mathsf{inp}(i))$). Thus we have

$$\mathsf{GARBLE}\big(\widehat{\mathcal{RND}}_p(BP), (J,\sigma)\big) \quad = \quad \left\{ \begin{array}{ll} \mathsf{prms},\ \ \hat{\mathbf{s}},\ \hat{\mathbf{t}}, & \hat{\mathbf{s}}',\ \hat{\mathbf{t}}' \\ \big\{\hat{D}_{i,b} : i \in I_J, b = \sigma(\mathsf{inp}(i))\big\}, & \big\{\hat{D}'_{i,b} : i \in I_J, b = \sigma(\mathsf{inp}(i))\big\} \\ \big\{\hat{D}_{i,b} : i \notin I_J, b \in \{0,1\}\big\}, & \big\{\hat{D}'_{i,b} : i \notin I_J, b \in \{0,1\}\big\} \end{array} \right\}.$$

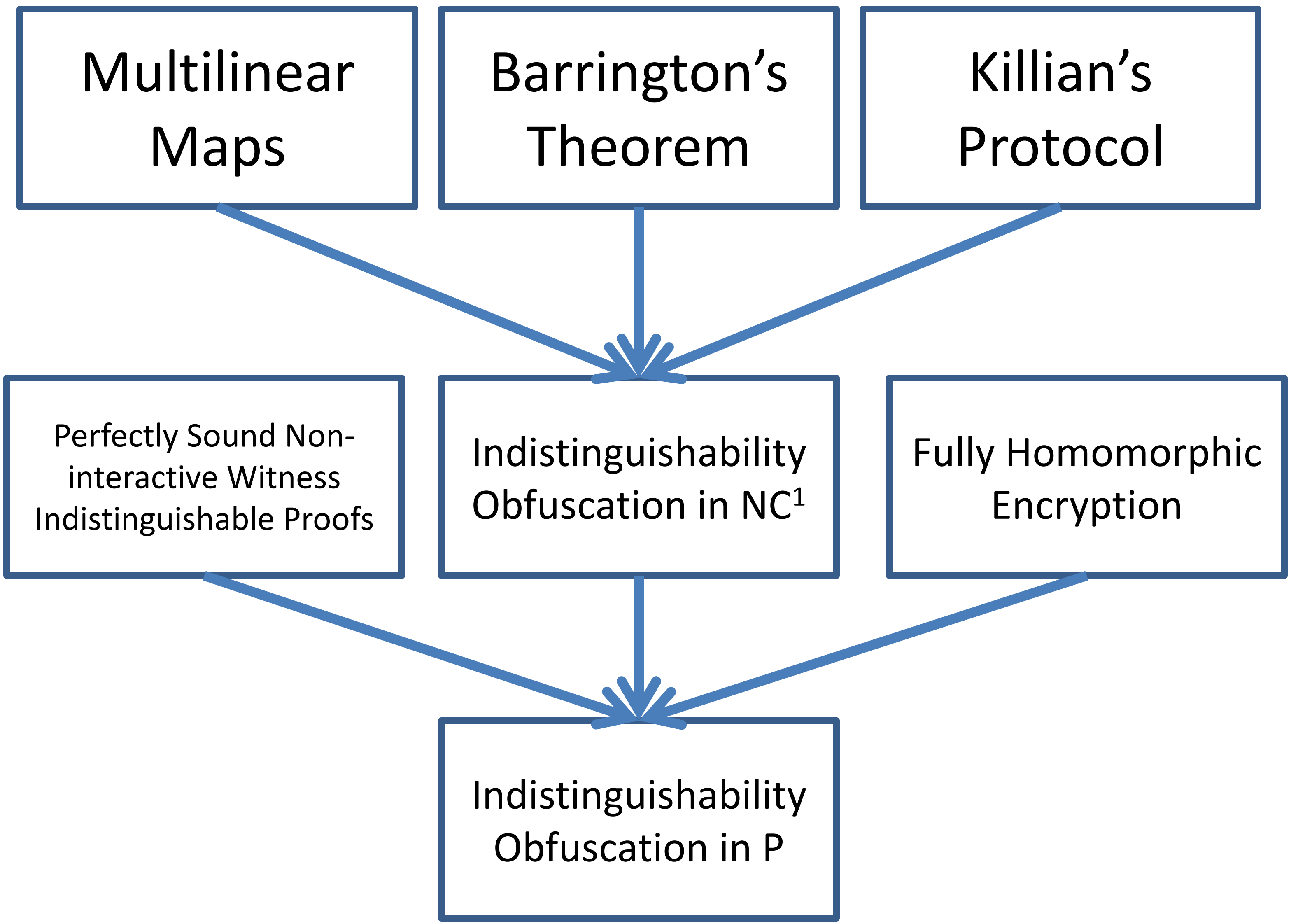
Indistinguishability

$$\mathsf{GARBLE}\big(\ \widehat{\mathcal{RND}}(BP), (J, \sigma_0)\ \big) \ \overset{(c)}{\approx}\ \mathsf{GARBLE}\big(\ \widehat{\mathcal{RND}}(BP), (J, \sigma_1)\ \big).$$

Obfuscation Results



Obfuscation Results



Papers

- On the (Im)possibility of Obfuscating Programs [http://eprint.iacr.org/2001/069]
- Candidate Multilinear Maps from Ideal Lattices [http://eprint.iacr.org/2012/610]
- Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits [http://eprint.iacr.org/2013/451]
- Virtual Black-Box Obfuscation for All Circuits via Generic Graded Encoding [http://eprint.iacr.org/2013/563]
- Protecting Obfuscation Against Algebraic Attacks [http://eprint.iacr.org/2013/631]
- Implementing Cryptographic Program Obfuscation [http://eprint.iacr.org/2014/779]

SafeWare Broad Agency Annoucement

- DARPA is spending millions $$$

"The goal of the SafeWare research effort is to drive fundamental advances in the theory of program obfuscation and to develop highly efficient and widely applicable program obfuscation methods with mathematically proven security properties"

Obfuscation Crackme

Implementing Cryptographic Program Obfuscation [http://eprint.iacr.org/2014/779]

- https://www.dropbox.com/s/85d03o0ny3b1c0c/point-14.circ.obf.60.zip
- 23.96 GB