Cryptanalysis of Candidate Obfuscators for Affine Determinant Programs

Li Yao[1] Yilei Chen[2,3] Yu Yu[1,3]

1. Shanghai Jiao Tong University
2. Tsinghua University
3. Shanghai Qizhi Institution

Overview

- [BIJ+20] proposed candidate obfuscators, whose computational model is affine determinant program.
- We show cryptanalytic attack on their candidate, as well as a plausible fix which may defend against our attack.

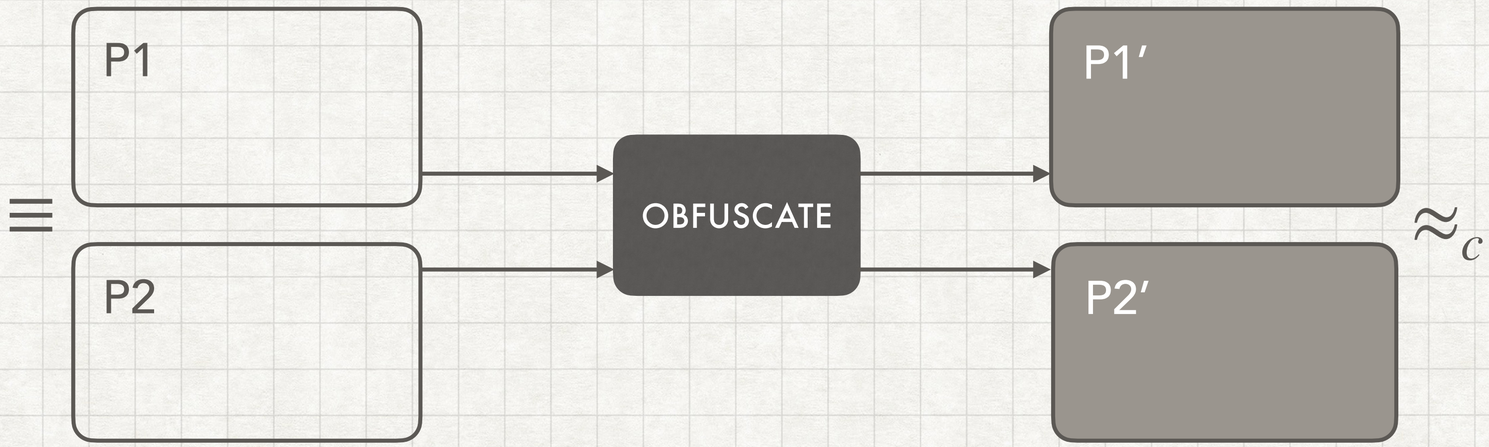
Indistinguishability Obfuscation ($i\mathcal{O}$)

- Correctness : Functionality-preserving

  P' = Obfuscate(P) $\implies$ P' $\equiv$ P
- Security : Indistinguishable

  P1 $\equiv$ P2 $\implies$ P1' $\approx_c$ P2'

$\equiv$ means the two programs compute the same function



Branching Program (BP)

Computes $f : \mathbf{x} \in \{0,1\}^n \to y \in \{0,1\}$

DAG : G

1 Source Node:

1 Sink Node:

Edges: labelled by $x_i, \bar{x}_i$ or 1

Evaluates on $\mathbf{x}$

$G_{\mathbf{x}}$: delete edges labelled by x_i (resp. $\bar{x}_i$)

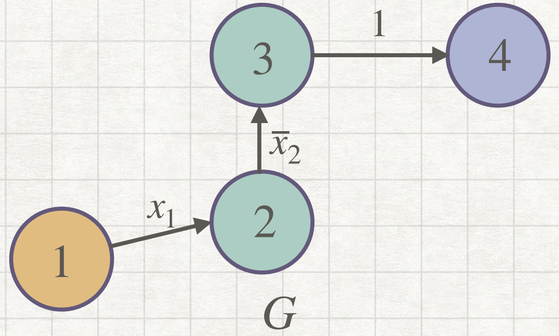
$\iff x_i = 0$ (resp. $\bar{x}_i = 0$)

$y = \#PATH(G_{\mathbf{x}},\ ,\ )$

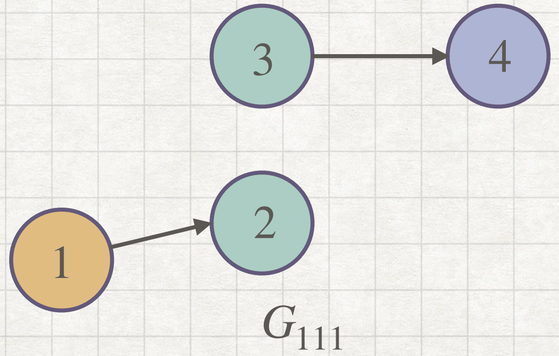
Count paths in $G_{\mathbf{x}}$ from to

Example:

$y = f(\mathbf{x} \in \{0,1\}^3) = x_1 \wedge \bar{x}_2$



G



G_{111}

Affine Determinant Program (ADP)

Computes $f : \mathbf{x} \in \{0,1\}^n \to y \in \{0,1\}$

$n+1$ matrices : $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n$

Evaluates on $\mathbf{x}$

Compute $L(\mathbf{x}) = \mathbf{A} + \sum_{i=1}^{n} x_i \cdot \mathbf{B}_i$

$y = \det(L(\mathbf{x}))$

Example:

$$\begin{bmatrix} 0 & x_1 & 0 & 0 \\ 0 & 0 & \bar{x}_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \to \begin{bmatrix} x_1 & 0 & 0 \\ -1 & \bar{x}_2 & 0 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} + x_1 \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + x_2 \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + x_3 \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$M(\mathbf{x})$ $\quad L(\mathbf{x})$ $\quad \mathbf{A}$ $\quad \mathbf{B}_1$ $\quad \mathbf{B}_2$ $\quad \mathbf{B}_3$

BP $\to$ ADP:

$M(\mathbf{x})$: Adjacency matrix of $G_{\mathbf{x}}$

$L(\mathbf{x})$: delete leftmost column and lowermost row of $(M(\mathbf{x}) - \mathbf{I})$

$\det(L(\mathbf{x})) = \#PATH(G_{\mathbf{x}}, \bigcirc, \bigcirc)$ [IK97]

$x_i = 0 + x_i \cdot 1 \qquad \bar{x}_i = 1 + x_i \cdot (-1)$

$\implies L(\mathbf{x}) = \mathbf{A} + \sum_{i=1}^{n} x_i \cdot \mathbf{B}_i$

The BIJMSZ scheme [BIJ+20]

Consists of 4 functionality-preserving transformations:

- Random Local Substitution (RLS)
- Add Small Even-valued Noise (AddNoise)
- Add Block-Diagonal Matrices (AddDiag)
  - Prevent dishonest evaluation
- Re-Randomization (ReRand)
  - Prevent leakage other than determinant and rank

The final obfuscation:

ReRand(AddDiag(ReRand(AddNoise(RLS(ADP)))))

Our attack only concerns RLS and AddNoise

Add Small Even-Valued Noise

Let l denote the dimension of matrices $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n$

Sample $(n+1)l^2$ small noise : $\mathbf{E}_0, \mathbf{E}_1, \ldots, \mathbf{E}_n$

$$(\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_n) \xrightarrow{\text{AddNoise}} (\mathbf{A} + 2\mathbf{E}_0, \mathbf{B}_1 + 2\mathbf{E}_1, \ldots, \mathbf{B}_n + 2\mathbf{E}_n)$$

$$y = \det(L(\mathbf{x})) \xrightarrow{\text{AddNoise}} y = \det(L(\mathbf{x})) \mod 2$$

"mod 4" attack [BIJ+20] :

$$\det(\mathbf{A} + 2\mathbf{E}_0 + \textstyle\sum_{i\in[n]} x_i(\mathbf{B}_i + 2\mathbf{E}_i))$$

$$\equiv \det(L(\mathbf{x})) + \textstyle\sum_{j\in[l],k\in[l]} (2e^{(0)}_{j,k} + \sum_i x_i 2e^{(i)}_{j,k}) \det(L(\mathbf{x})_{(j,k)}) \mod 4$$

$e^{(i)}_{j,k}$ is the (j,k) element of $\mathbf{E}_i$

$\det(L(\mathbf{x})_{(j,k)})$ is the (j,k) minor of $L(\mathbf{x})$

represents unknown part

RLS tries to fix the problem by hiding part

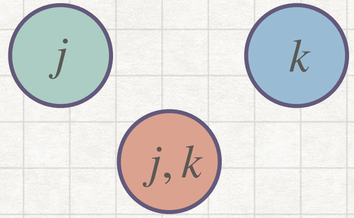
Random Local Substitution (RLS)

Subgraph of G

Submatrix of $L(\mathbf{x})$

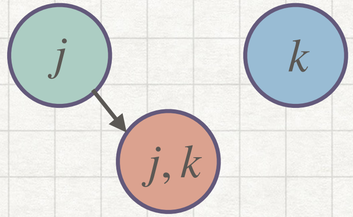
$$\begin{array}{cc} & v_k \\ v_j & [0] \end{array}$$

RLS



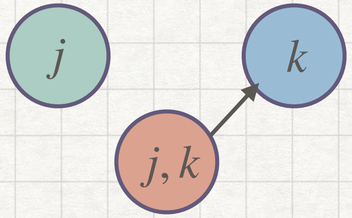
$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix} \\ v_{j,k} & \end{array}$$

Or



$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \\ v_{j,k} & \end{array}$$

Or



$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 0 & 0 \\ -1 & 1 \end{bmatrix} \\ v_{j,k} & \end{array}$$

Similarly,

$$\begin{array}{cc} & v_k \\ v_j & [1] \end{array}$$

RLS

$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \\ v_{j,k} & \end{array}$$

Or

$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix} \\ v_{j,k} & \end{array}$$

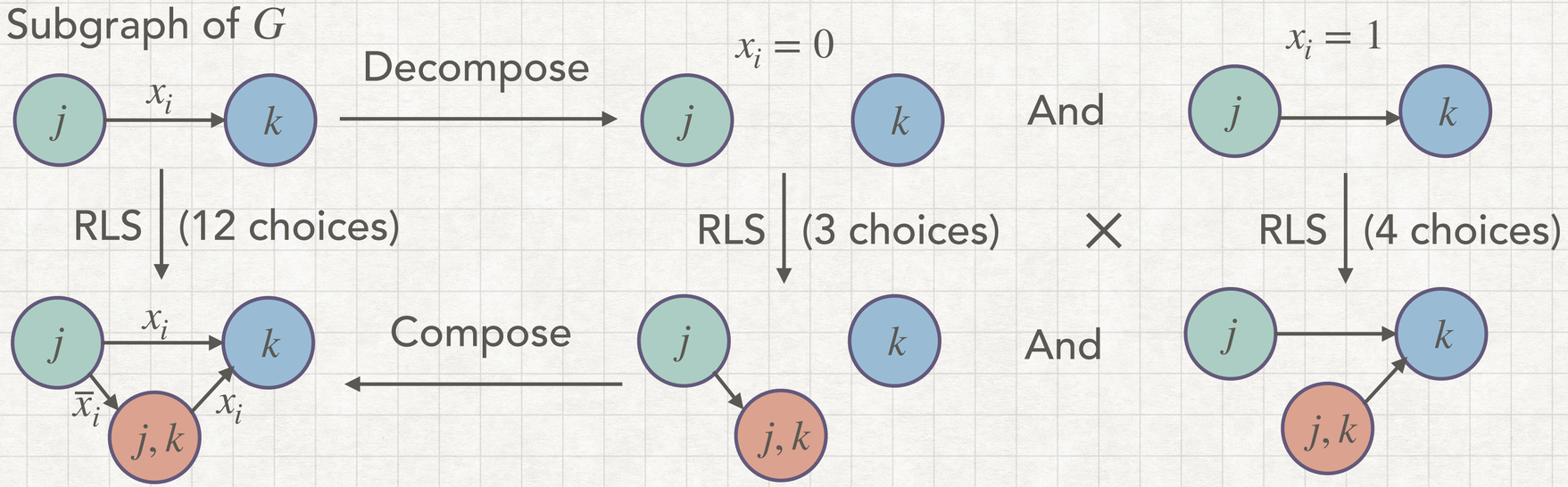
Or

$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\ v_{j,k} & \end{array}$$

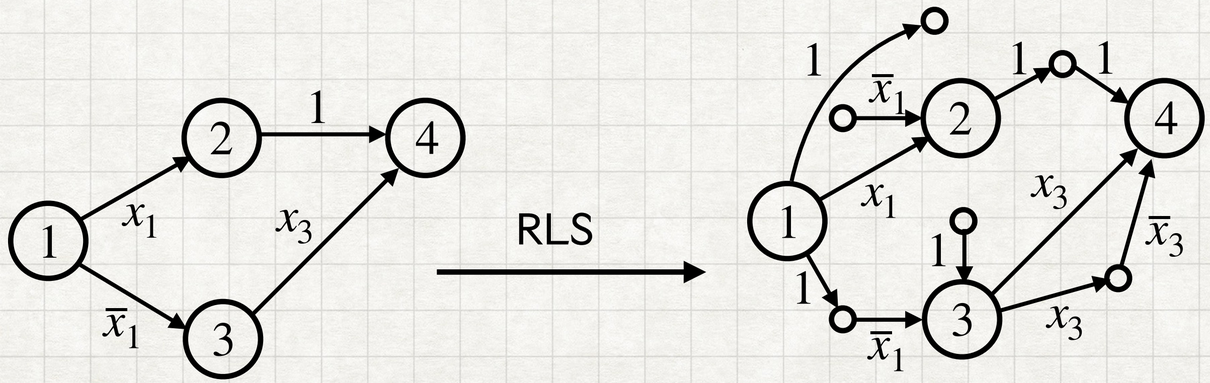
Or

$$\begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \\ v_{j,k} & \end{array}$$

Random Local Substitution (RLS)



Example:



Our Attack (Base ver.)

Notation:

$$L(\mathbf{x}) = \mathbf{A} + \sum x_i \mathbf{B}_i \xrightarrow{\text{RLS}} L'(\mathbf{x}) = \mathbf{A}' + \sum x_i \mathbf{B}'_i$$

$$\xrightarrow{\text{AddNoise}} L''(\mathbf{x}) = \mathbf{A}' + 2\mathbf{E}_0 + \sum x_i(\mathbf{B}'_i + 2\mathbf{E}_i)$$

Observation 1 :

$$\mathbf{A}, \mathbf{B}_1, \ldots, \boxed{\mathbf{B}_i = \mathbf{0},} \ldots \mathbf{B}_n \xrightarrow{\text{RLS}} \mathbf{A}', \mathbf{B}'_1, \ldots, \boxed{\mathbf{B}'_i = \mathbf{0},} \ldots \mathbf{B}'_n$$

independent of x_i independent of x_i

Toy Example :

$$\mathbf{A}, \mathbf{B}_1 = \mathbf{0}, \mathbf{B}_2 = \mathbf{0} \xrightarrow{\text{RLS}} \mathbf{A}', \mathbf{B}'_1 = \mathbf{0}, \mathbf{B}'_2 = \mathbf{0}$$

$$\xrightarrow{\text{AddNoise}} \mathbf{A}' + 2\mathbf{E}_0, 2\mathbf{E}_1, 2\mathbf{E}_2$$

$$L'(00) = L'(01) = L'(10) = L'(11) = \mathbf{A}'$$

Our Attack (Base ver.)

Use "mod 4" attack, we have $L'(00)_{j,k} = L'(01)_{j,k} = L'(10)_{j,k} = L'(11)_{j,k} = \mathbf{A}'_{j,k}$

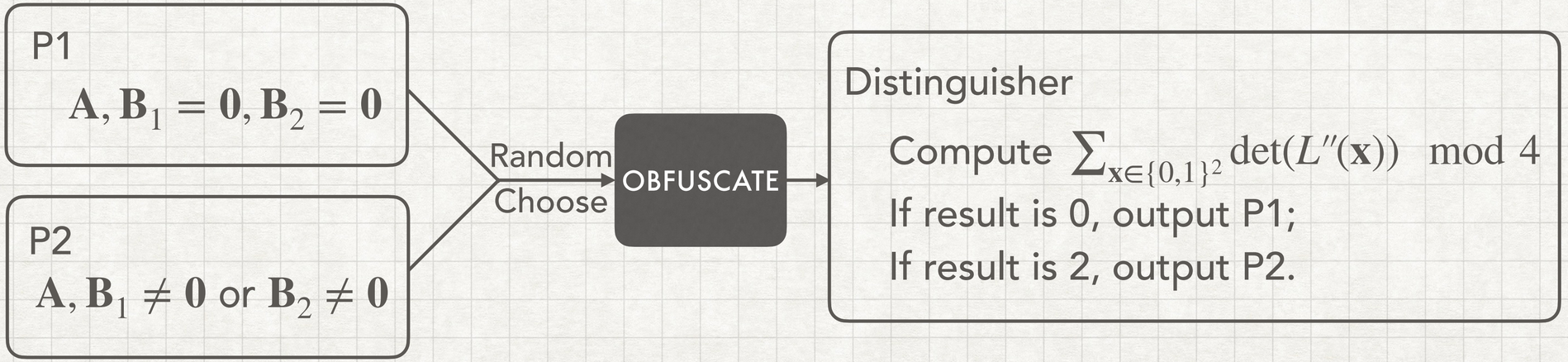
$$\det(L''(00)) = \det(\mathbf{A}' + \mathbf{E}_0) \equiv \det(\mathbf{A}') + \sum_{j,k}^{l',l'} \det(\mathbf{A}'_{j,k})\,(2e^0_{j,k}) \mod 4$$

$$\det(L''(01)) = \det(\mathbf{A}' + \mathbf{E}_0 + \mathbf{E}_1) \equiv \det(\mathbf{A}') + \sum_{j,k}^{l',l'} \det(\mathbf{A}'_{j,k})\,(2e^0_{j,k} + 2e^1_{j,k}) \mod 4$$

$$\det(L''(10)) = \det(\mathbf{A}' + \mathbf{E}_0 + \mathbf{E}_2) \equiv \det(\mathbf{A}') + \sum_{j,k}^{l',l'} \det(\mathbf{A}'_{j,k})\,(2e^0_{j,k} + 2e^2_{j,k}) \mod 4$$

$$\det(L''(11)) = \det(\mathbf{A}' + \mathbf{E}_0 + \mathbf{E}_1 + \mathbf{E}_2) \equiv \det(\mathbf{A}') + \sum_{j,k}^{l',l'} \det(\mathbf{A}'_{j,k})\,(2e^0_{j,k} + 2e^1_{j,k} + 2e^2_{j,k}) \mod 4$$

Add them all, $\sum_{\mathbf{x}\in\{0,1\}^2} \det(L''(\mathbf{x}) = \mathbf{A}' + \mathbf{E}_0 + x_1\mathbf{E}_1 + x_2\mathbf{E}_2) \equiv 0 \mod 4$



Our Attack (Advanced ver.)

Base ver. attack is restricted :

only works for functionalities that ignore two bits of its input

A nature question:

Can we generalize the attack s.t. it can work on more functionalities?

What we need :

4 inputs $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ which differ on 2 bits s.t

$$\forall j,k \in \{0,1\}^{l'}, \det(L'_{j,k}(\mathbf{x}_1)) = \det(L'_{j,k}(\mathbf{x}_2)) = \det(L'_{j,k}(\mathbf{x}_3)) = \det(L'_{j,k}(\mathbf{x}_4)) \quad ✓$$

$$L'(\mathbf{x}_1) = L'(\mathbf{x}_2) = L'(\mathbf{x}_3) = L'(\mathbf{x}_4) \quad ✗$$

Our Attack (Advanced ver.)

Observation 2 :

We can quasi-cancel RLS when computing the minor of $L'(\mathbf{x})$

i.e. RLS may not bring much uncertainty to the minor of $L'(\mathbf{x})$

Why?

Rows/Columns of intermediate nodes are sparse (at most 2 non-zero elements).

Classification :

Node:

1. Original Node (e.g. v_j)
2. Intermediate Node (e.g. $v_{j,k}$)

Minor:

1. Two Original Nodes (e.g. $L'(\mathbf{x})_{v_j,v_k}$)
2. Two repeated Intermediate Nodes (e.g. $L'(\mathbf{x})_{v_{j,k},v_{j,k}}$)
3. Other cases (e.g. $L'(\mathbf{x})_{v_s,v_{j,k}}$)

Our Attack (Advanced ver.)

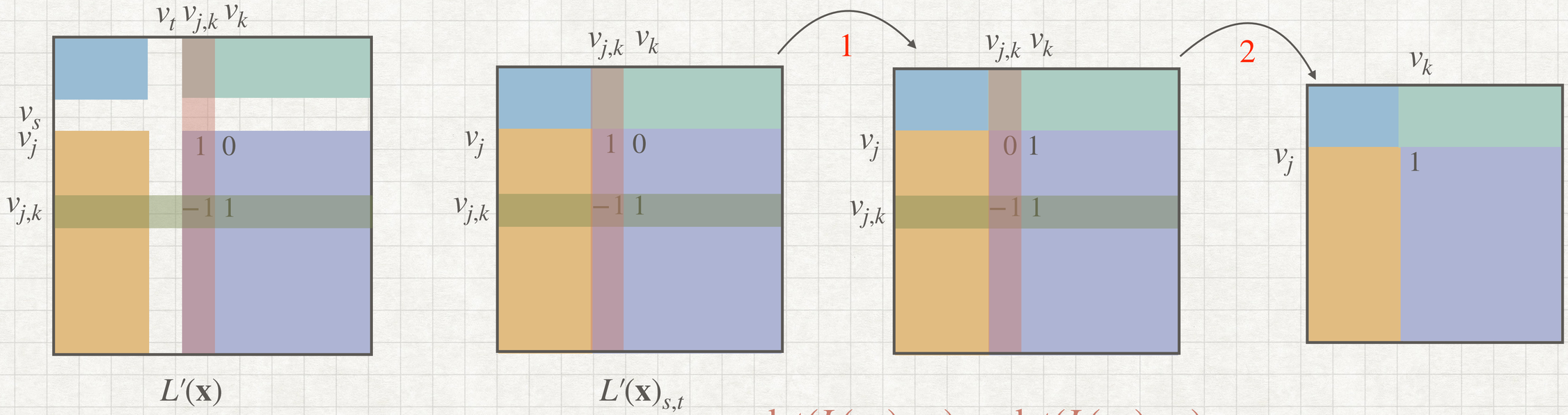
Case 1 :

$$\begin{array}{c} \\ v_j \\ v_{j,k} \end{array}\begin{array}{c} v_k \\ [1] \\ \\ \end{array} \xrightarrow{\text{RLS}} \begin{array}{c} \\ v_j \\ v_{j,k} \end{array}\begin{array}{c} v_{j,k}\ v_k \\ \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \end{array} \xrightarrow{\text{quasi-cancel}} \begin{array}{c} \\ v_j \\ \\ \end{array}\begin{array}{c} v_k \\ [1] \\ \\ \end{array}$$

1 : row $v_j :=$ row $v_j +$ row $v_{j,k}$

2 : Laplace expansion by column $v_{j,k}$

$$2e \equiv -2e \mod 4$$



$$\det(L'(\mathbf{x})_{v_s,v_t}) = \det(L(\mathbf{x})_{v_s,v_t})$$

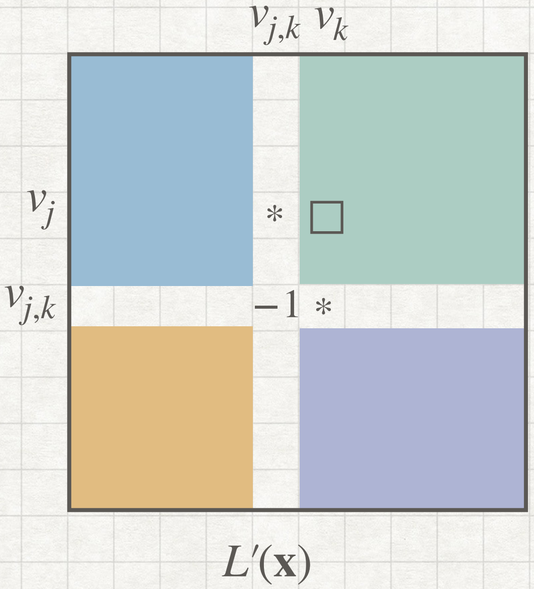
$$\det(L(\mathbf{x_1})_{v_s,v_t}) = \det(L(\mathbf{x_2})_{v_s,v_t})$$

$$\Longrightarrow \det(L'(\mathbf{x_1})_{v_s,v_t}) = \det(L'(\mathbf{x_2})_{v_s,v_t})$$

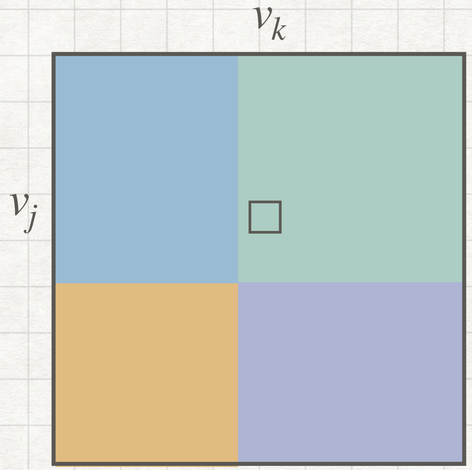
Our Attack (Advanced ver.)

Case 2 :

$$\begin{array}{cc} & v_k \\ v_j & [\blacksquare] \end{array} \xrightarrow{\text{RLS}} \begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \\ v_{j,k} & \end{array} \begin{bmatrix} * & \square \\ -1 & * \end{bmatrix}$$



$L'(\mathbf{x})$



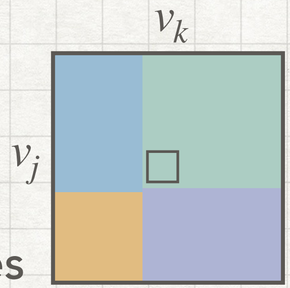
$L'(\mathbf{x})_{v_{j,k},v_{j,k}}$

Note : ■ may be different from □

i.e.

$$\begin{array}{cc} & v_k \\ v_j & [1] \end{array} \xrightarrow{\text{RLS}} \begin{array}{cc} & v_{j,k} \quad v_k \\ v_j & \\ v_{j,k} & \end{array} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

delete all intermediate nodes



$L(\mathbf{x})$ but the (v_j, v_k) entry is □

$$\det(L'(\mathbf{x})_{v_{j,k},v_{j,k}}) = \underline{\det(L(\mathbf{x})_{v_j,v_k=0})} \text{ or } \det(L(\mathbf{x}))$$

modify the (v_j, v_k) entry to 0

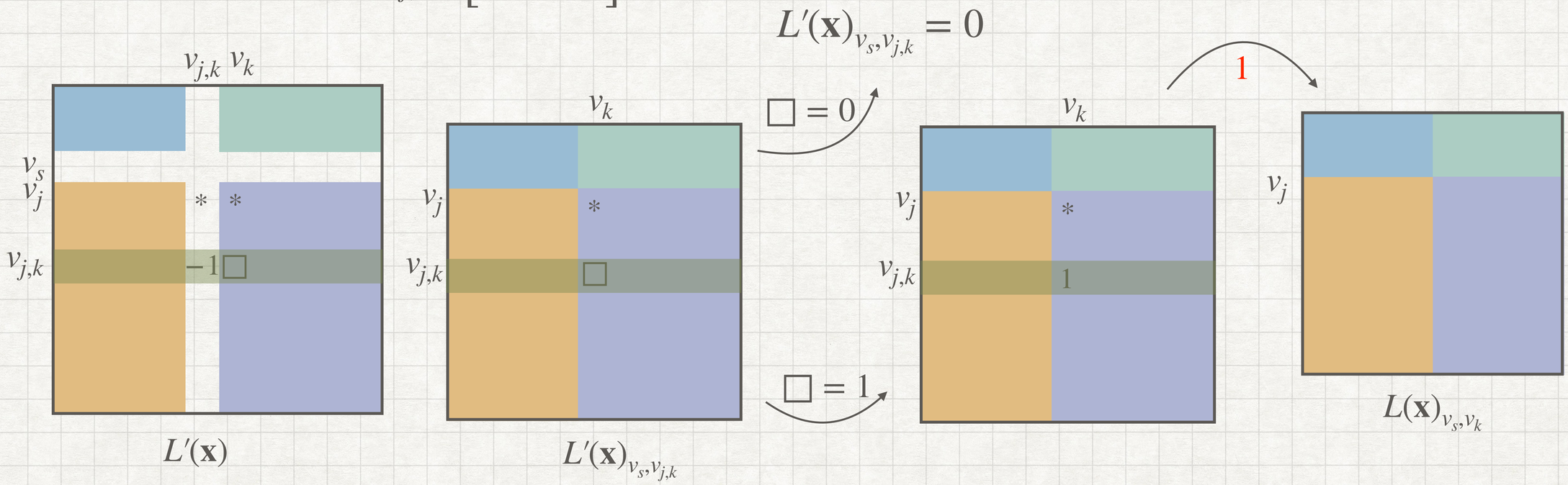
We want $\det(L(\mathbf{x})_{v_j,v_k=0}) = \det(L(\mathbf{x}))$

Our Attack (Advanced ver.)

Case 3 :

$$\begin{array}{c} \\ v_j \\ \\ \end{array}\begin{array}{c} v_k \\ [\,*\,] \\ \\ \end{array} \xrightarrow{\text{RLS}} \begin{array}{c} \\ v_j \\ v_{j,k} \end{array}\begin{array}{c} v_{j,k} \quad v_k \\ \begin{bmatrix} * & * \\ -1 & \square \end{bmatrix} \end{array}$$

1 : Laplace expansion by row $v_{j,k}$



$\det(L'(\mathbf{x})_{v_s,v_{j,k}}) = \det(L(\mathbf{x})_{v_s,v_k})$ or 0

We want $\det(L(\mathbf{x})_{v_s,v_k}) = 0$

Our Attack (Advanced ver.)

By further analyzing the 3 cases, we can conclude :

For $L(\mathbf{x}_1)$ and $L(\mathbf{x}_2)$ and all possible j, k satisfying following conditions :

1. $L(\mathbf{x}_1)_{v_j,v_k} = L(\mathbf{x}_2)_{v_j,v_k}$
2. $\det(L(\mathbf{x_1})_{(v_i,v_j)=0}) = \det(L(\mathbf{x_1})) = \det(L(\mathbf{x_2})) = \det(L(\mathbf{x_2})_{(v_i,v_j)=0})$
3. $L(\mathbf{x_1})[v_i, v_j] \neq L(\mathbf{x_2})[v_i, v_j] \implies$ row v_i and column v_j are all zero elements

$L'(\mathbf{x}_1)_{v_j,v_k} = L'(\mathbf{x}_2)_{v_j,v_k}$ always hold.

Example:

$$L(\mathbf{x}) = \begin{bmatrix} 0 & \bar{x}_3 & 0 & x_3 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & \bar{x}_1 & 0 & 0 \\ 0 & 0 & -1 & x_2 & 0 \\ 0 & 0 & 0 & -1 & \bar{x}_3 \end{bmatrix}$$

Computes $\bar{x}_1 \wedge x_2 \wedge \bar{x}_3$ (i.e. rely on x_1, x_2)

The j, k minors of $L'(001), L'(011),$
$L'(101), L'(111)$ are always equal

Our Fix

Break Observation 1 :

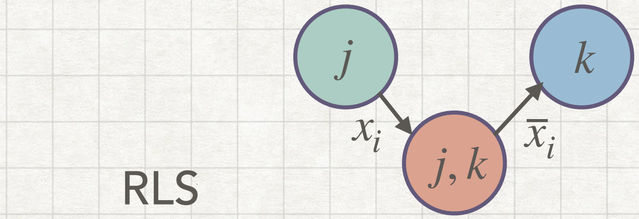
$$\mathbf{A}, \mathbf{B}_1, \ldots, \boxed{\mathbf{B}_i = \mathbf{0},} \ldots \mathbf{B}_n \xrightarrow{\text{RLS}} \mathbf{A}', \mathbf{B}'_1, \ldots, \boxed{\mathbf{B}'_i = \mathbf{0},} \ldots \mathbf{B}'_n$$

independent of x_i independent of x_i

Example :

Subgraph of G

RLS



Submatrix of $L(\mathbf{x})$

	v_k
v_j	$[0]$

	$v_{j,k}$	v_k
v_j	x_i	0
$v_{j,k}$	-1	$\bar{x}_i$

Future Work

- Other RLS candidates/revisions ?
- Provable security in restricted model ?
  e.g. assume that the adversary only applies “mod 4” attack