Static Analysis with Set-closure in Secrecy

[1] Woosuk Lee, [2] Hyunsook Hong
[1] Kwangkeun Yi, and [2] Jung Hee Cheon

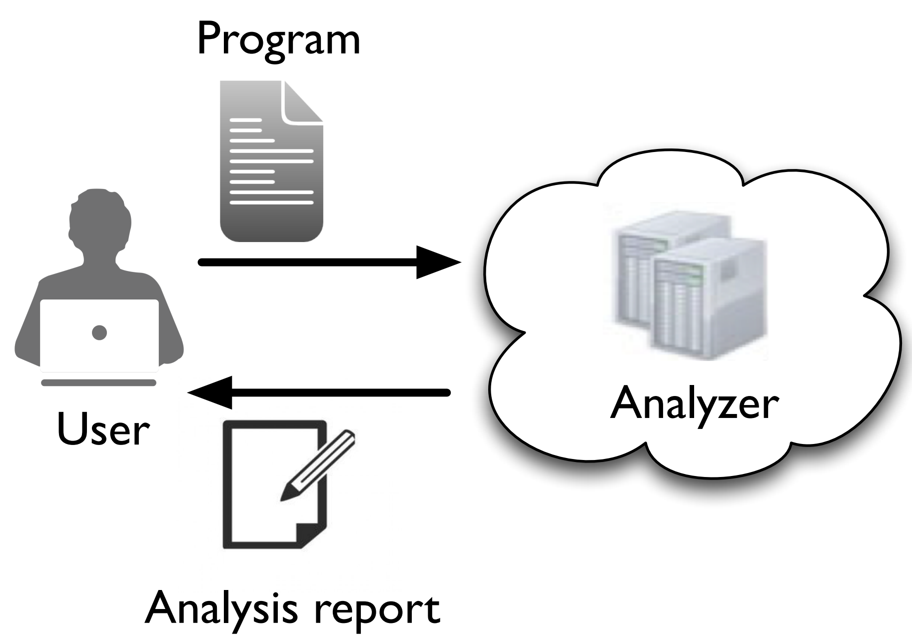
[1] Seoul Nat'l Univ. Computer Science Dept.
[2] Seoul Nat'l Univ. Mathematical Science Dept.

SAS'15 @ Saint-Malo

Problem

- Two ways to use static analyzer

(1)



(2)

runs in local

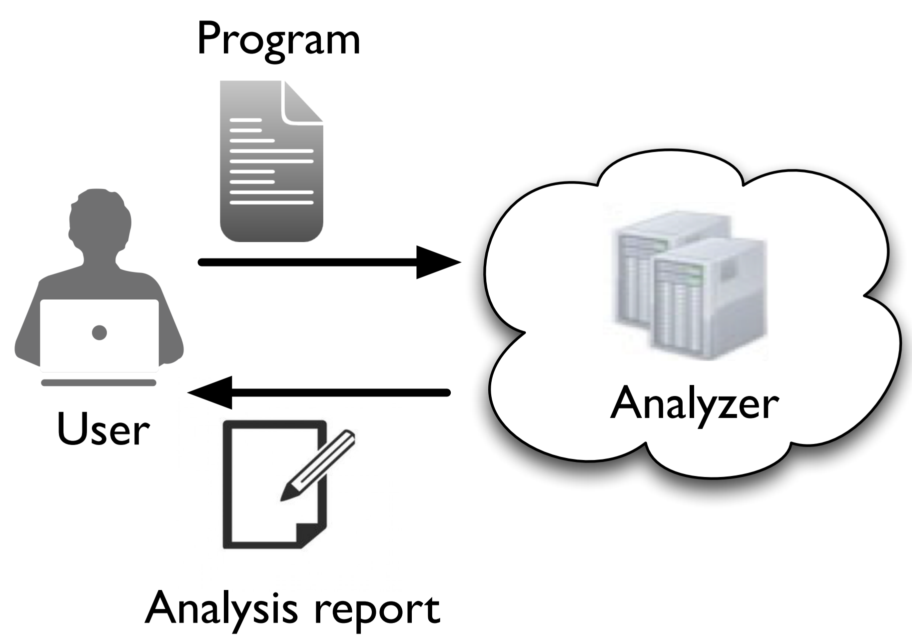
(1) - target program is revealed.

(2) - analyzer is revealed.

Problem

- Two ways to use static analyzer

(1)



(2)

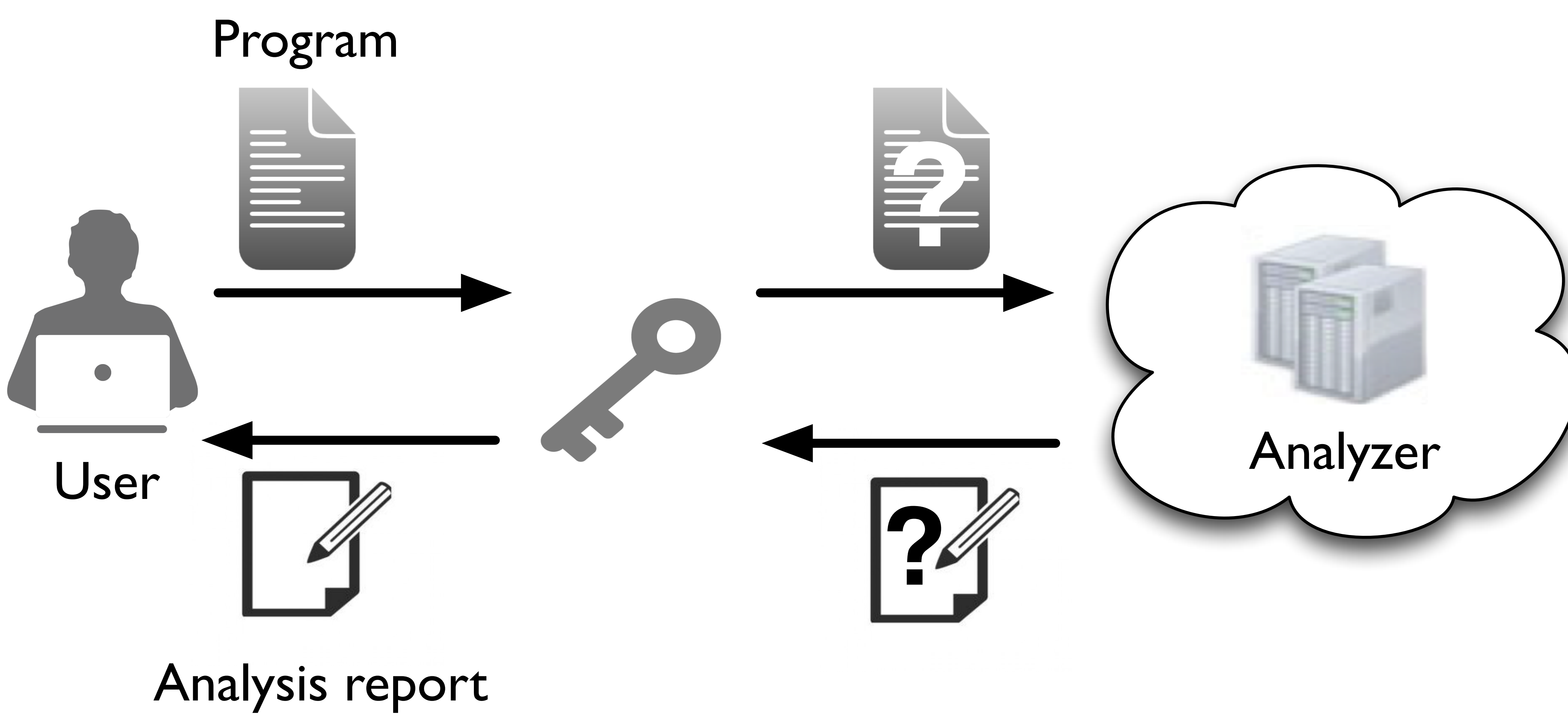
(1) - target program is revealed

(2) - analyzer is rev

Without information leaks of both sides!

Our Solution

- Static analysis on encrypted programs



Key : Homomorphic Encryption

- HE enables computation of arbitrary functions on encrypted data.

$$f' \ \overline{x} \equiv \overline{f \ x}$$

A Simple HE Scheme

- Based on the approximate common divisor problem
- p : integer as a secret key
- q : random integer
- $r(\ll |p|)$: random noise for security

$$\text{Enc}(\mu \in \{0,1\}) = pq + 2r + \mu$$

$$\text{Dec}_p(c) = (c \bmod p) \bmod 2$$

A Simple HE Scheme

- Based on the approximate common divisor problem
- p : integer as a secret key
- q : random integer
- $r(\ll |p|)$: random noise for security

$$\mathsf{Enc}(\mu \in \{0,1\}) = pq + 2r + \mu$$

$$\mathsf{Dec}_p(c) = \underline{(c \bmod p)} \bmod 2$$

$$\cancel{pq} + 2r + \mu$$

A Simple HE Scheme

- Based on the approximate common divisor problem
- p : integer as a secret key
- q : random integer
- $r(\ll |p|)$: random noise for security

$$\text{Enc}(\mu \in \{0,1\}) = pq + 2r + \mu$$

$$\text{Dec}_p(c) = \underline{(c \bmod p)}\ \underline{\bmod 2}$$

$$\cancel{pq} + \cancel{2r} + \mu$$

A Simple HE Scheme

- For ciphertexts $c_1 \leftarrow \mathsf{Enc}(\mu_1)$, $c_2 \leftarrow \mathsf{Enc}(\mu_2)$ the followings hold:

$$\mathsf{Dec}_p(c_1 + c_2) = \mu_1 + \mu_2$$

$$\mathsf{Dec}_p(c_1 \times c_2) = \mu_1 \times \mu_2$$

- The scheme can evaluate all boolean circuits as $+$ and $\times$ in $\mathbb{Z}_2 = \{0, 1\}$ equal to XOR and AND.

Performance Hurdle : Growing Noise

- noise increases during operations.
  For $c_i = pq_i + 2r_i + \mu_i$,

$$c_1 + c_2 = p(q_1 + q_2) + \underbrace{2(r_1 + r_2) + \mu_1 + \mu_2}_{\text{noise}},$$

$$c_1 \times c_2 = p(pq_1q_2 + \cdots) + \underbrace{2(2r_1r_2 + r_1\mu_2 + r_2\mu_1) + \mu_1 \cdot \mu_2}_{\text{noise}}$$

- **noise > p → incorrect results after decryption**
- noise increase: doubly by add, **quadratically by mult**.

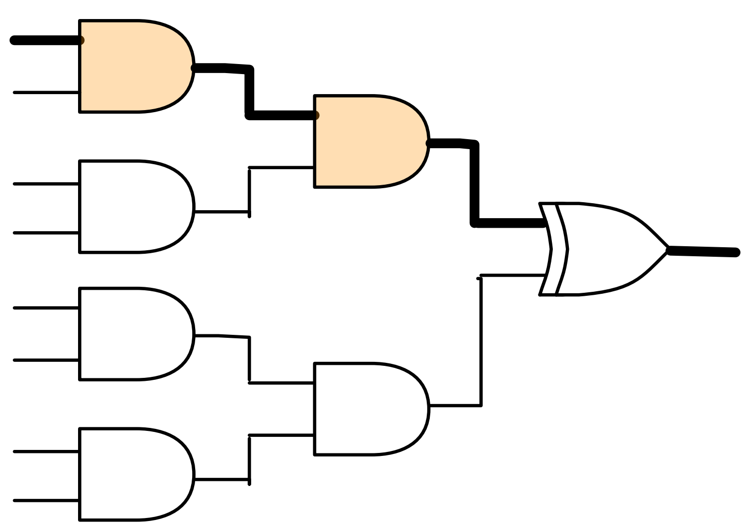
Performance Hurdle : Growing Noise

- Then, making p larger ?

  cipher text size ↑ , thus computational cost ↑

- only small circuits (multi. depth < 100) are allowed for now.

  Multiplicative depth : log (max # of mult. in in-out paths)



Depth : 2
(additions are ignored)

- *application-specific* techniques are necessary.

Our Contributions

- We propose an inclusion-based pointer analysis in secrecy
- We encode the pointer analysis into homomorphic matrix operations with *application-specific* optimizations:
    - reducing depth on the fact the maximal pointer level is usually small
    - reducing cost and ciphertext sizes by using ciphertext packing

Pointer Analysis

- Program P : a finite set of assignments

$$\mathtt{A} \;\rightarrow\; \mathtt{x} := \&\mathtt{y} \mid \mathtt{x} := \mathtt{y} \mid *\mathtt{x} := \mathtt{y} \mid \mathtt{x} := *\mathtt{y}$$

- Resolution rules :

$$\frac{}{\mathtt{x} \longrightarrow \&\mathtt{y}} \;(\text{if } \mathtt{x} = \&\mathtt{y} \text{ in } P) \qquad \text{(New)}$$

$$\frac{}{\mathtt{x} \longrightarrow \mathtt{y}} \;(\text{if } \mathtt{x} = \mathtt{y} \text{ in } P) \qquad \text{(Copy)}$$

$$\frac{\mathtt{x} \longrightarrow \&\mathtt{z}}{\mathtt{y} \longrightarrow \mathtt{z}} \;(\text{if } \mathtt{y} = *\mathtt{x} \text{ in } P) \qquad \text{(Load)}$$

$$\frac{\mathtt{x} \longrightarrow \&\mathtt{z}}{\mathtt{z} \longrightarrow \mathtt{y}} \;(\text{if } *\mathtt{x} = \mathtt{y} \text{ in } P) \qquad \text{(Store)}$$

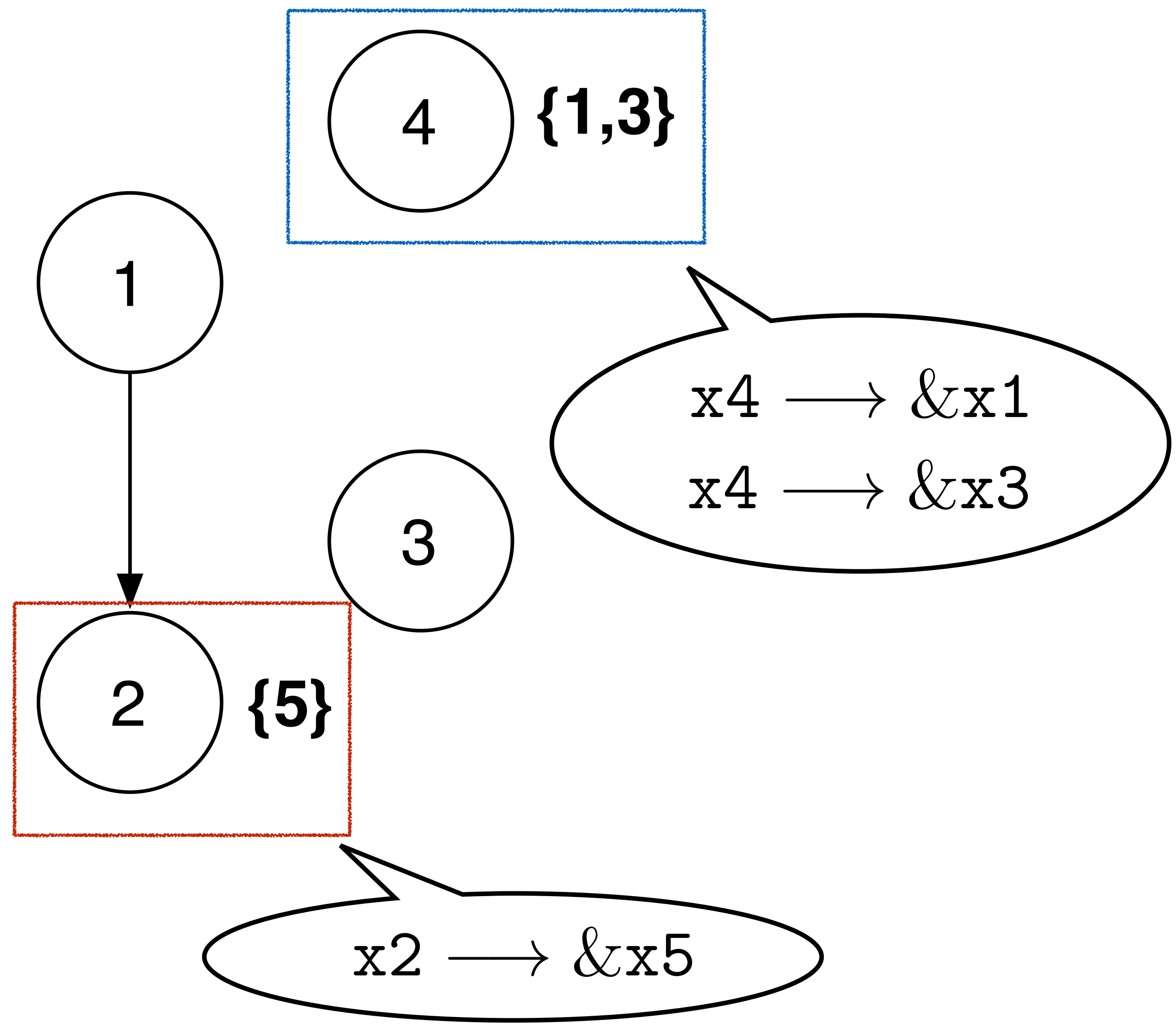
$$\frac{\mathtt{x} \longrightarrow \mathtt{z} \quad \mathtt{z} \longrightarrow \&\mathtt{y}}{\mathtt{x} \longrightarrow \&\mathtt{y}} \qquad \text{(Trans)}$$

Pointer Analysis

- Example

```
int *x1, *x2, *x3
int **x4
int x5
x2 = &x5
x1 = x2
x4 = &x1
x4 = &x3
*x4 = x2
```

Initialization

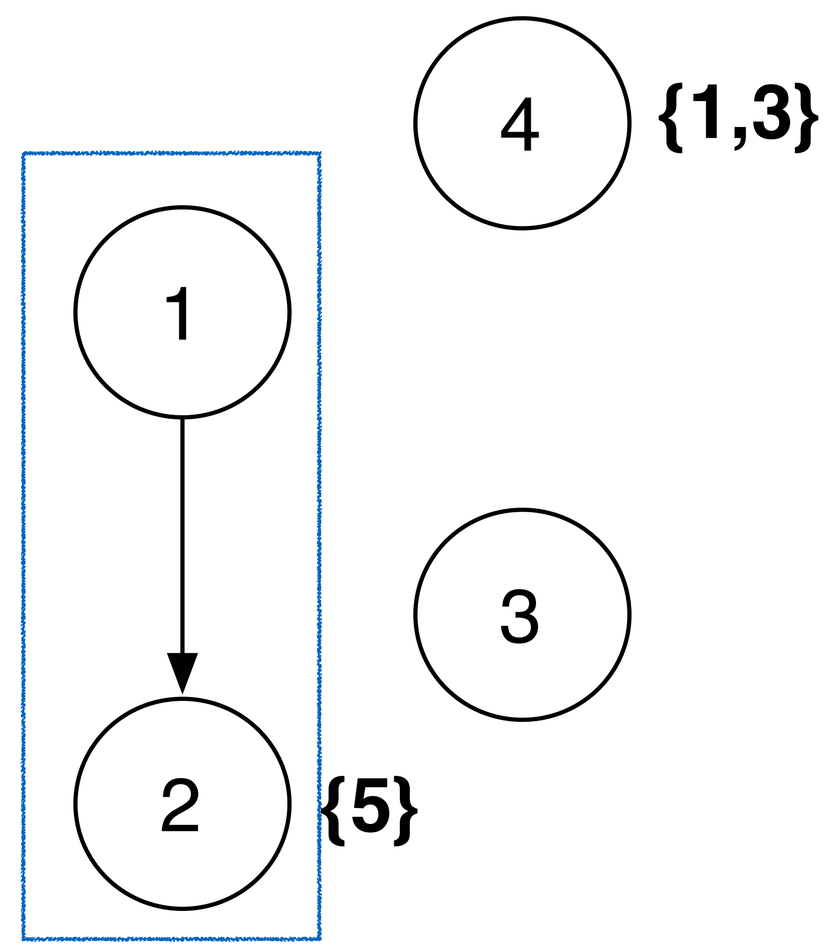


Pointer Analysis

- Example

```
int *x1, *x2, *x3
int **x4
int x5
x2 = &x5
x1 = x2
x4 = &x1
x4 = &x3
*x4 = x2
```

Initialization

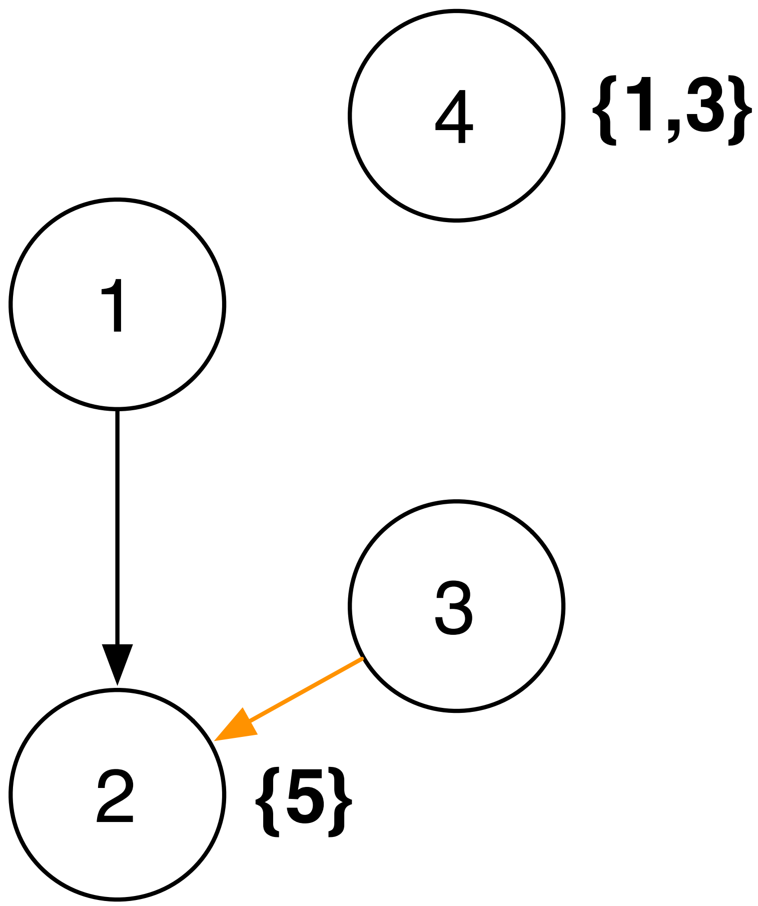


Pointer Analysis

Edge addition

- Example

```
int *x1, *x2, *x3
int **x4
int x5
x2 = &x5
x1 = x2
x4 = &x1
x4 = &x3
*x4 = x2
```

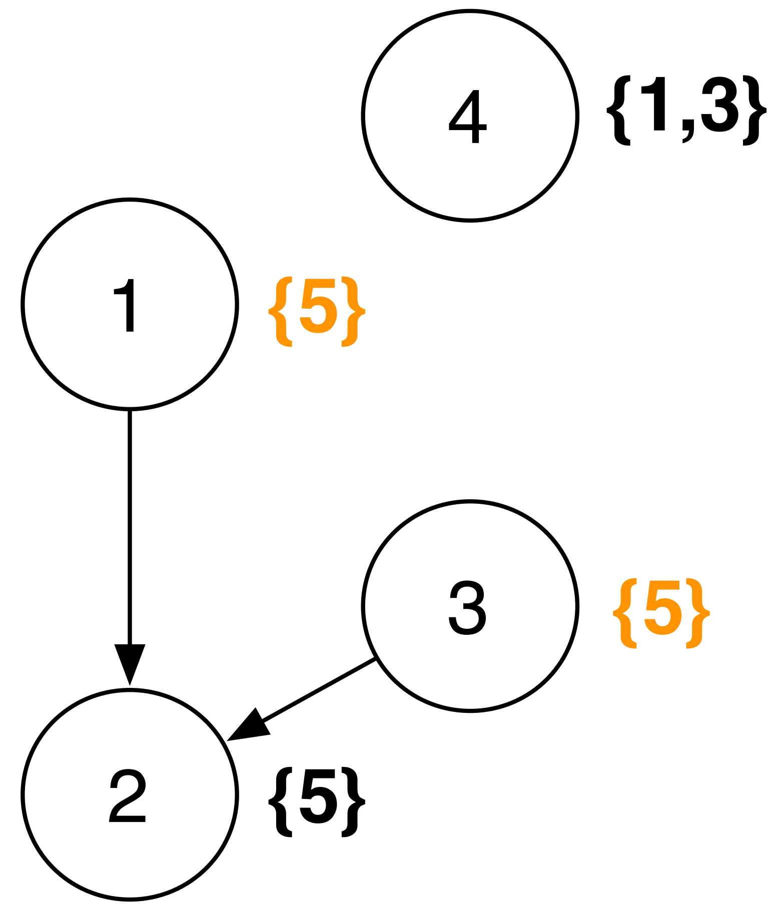


Pointer Analysis

- Example

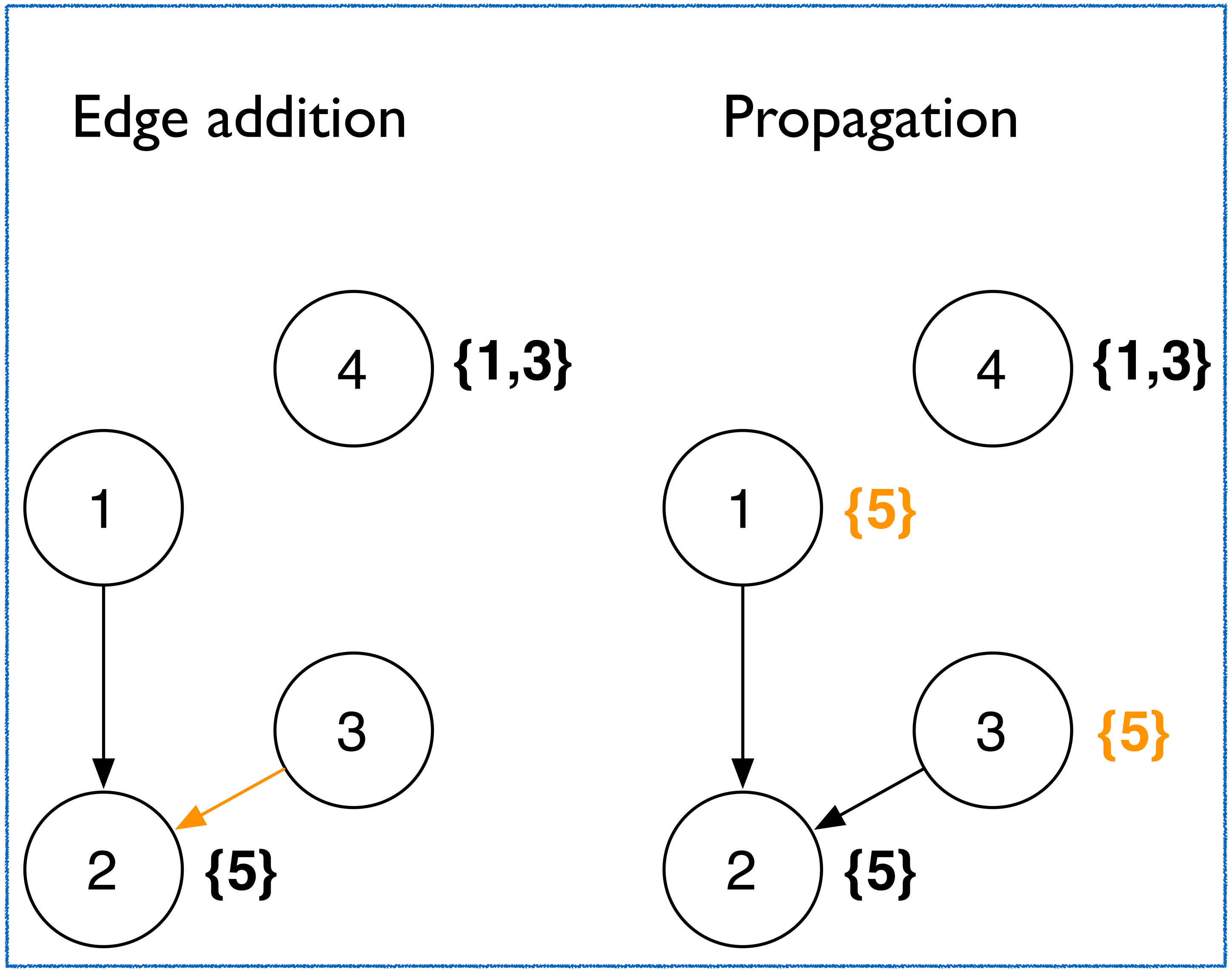
```
int *x1, *x2, *x3
int **x4
int x5
x2 = &x5
x1 = x2
x4 = &x1
x4 = &x3
*x4 = x2
```

Propagation



Pointer Analysis

```
int *x1, *x2, *x3
int **x4
int x5
x2 = &x5
x1 = x2
x4 = &x1
x4 = &x3
*x4 = x2
```

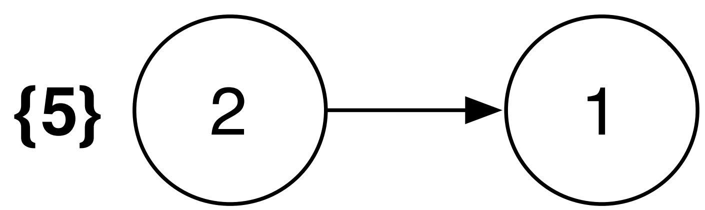


- repeating two steps until reaching a fix point

Notations

- $$\delta_{i,j} \neq 0 \quad \Longleftrightarrow \quad \mathtt{x_i} \longrightarrow \&\mathtt{x_j}$$
  $$\eta_{i,j} \neq 0 \quad \Longleftrightarrow \quad \mathtt{x_i} \longrightarrow \mathtt{x_j}$$

ex)



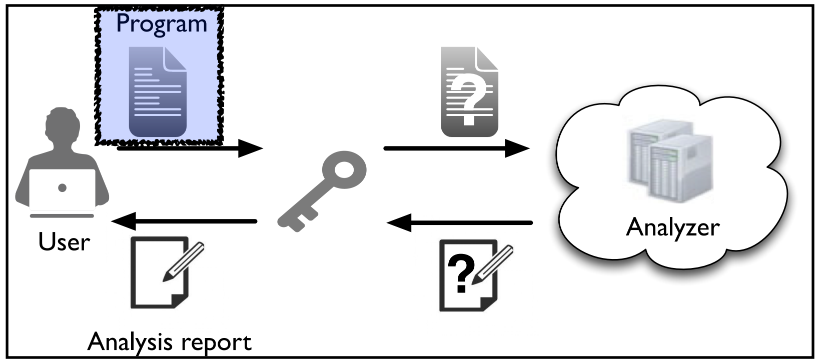
$$\delta_{2,5} \neq 0$$
$$\eta_{2,1} \neq 0$$

- $\mathbb{Z}_t = \{0, 1, 2, \cdots, t-1\}$ (our message space)

$$\text{Enc}(\mu \in \mathbb{Z}_t) = pq + tr + \mu$$

$$\text{Dec}_p(c) = (c \bmod p) \bmod t$$

Inputs from Client



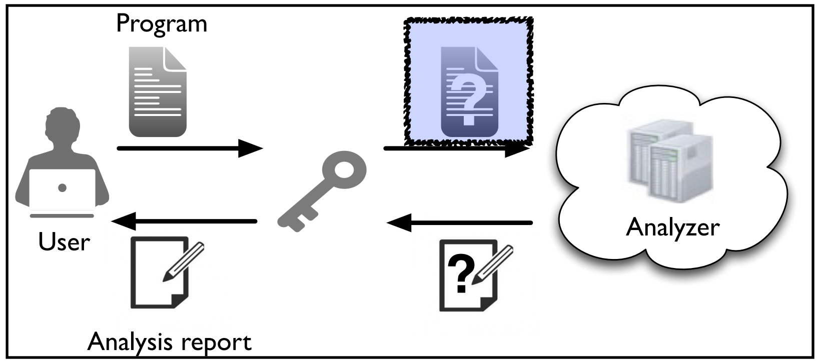
- A client derives the following numbers (m : the number of variables)

$$\{(\delta_{i,j}, \eta_{i,j}, u_{i,j}, v_{i,j}) \in \mathbb{Z} \times \mathbb{Z} \times \{0,1\} \times \{0,1\} \mid 1 \le i, j \le m\}$$

$$\delta_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists \mathtt{x_i} = \&\mathtt{x_j} \\ 0 & \text{otherwise} \end{cases} \qquad \eta_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists \mathtt{x_i} = \mathtt{x_j} \text{ or } i = j \\ 0 & \text{otherwise} \end{cases}$$

$$u_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists \mathtt{x_j} = *\mathtt{x_i} \\ 0 & \text{otherwise} \end{cases} \qquad v_{i,j} \leftarrow \begin{cases} 1 & \text{if } \exists *\mathtt{x_j} = \mathtt{x_i} \\ 0 & \text{otherwise} \end{cases}$$

Inputs from Client

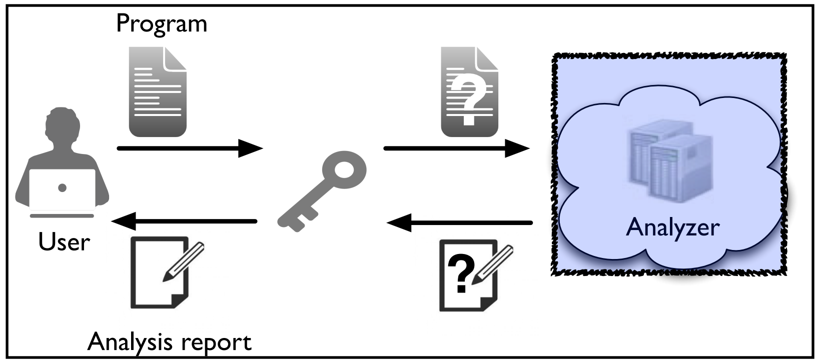


- The client $\overline{\text{encrypts}}$ the derived numbers using a HE scheme and provides the following set to server

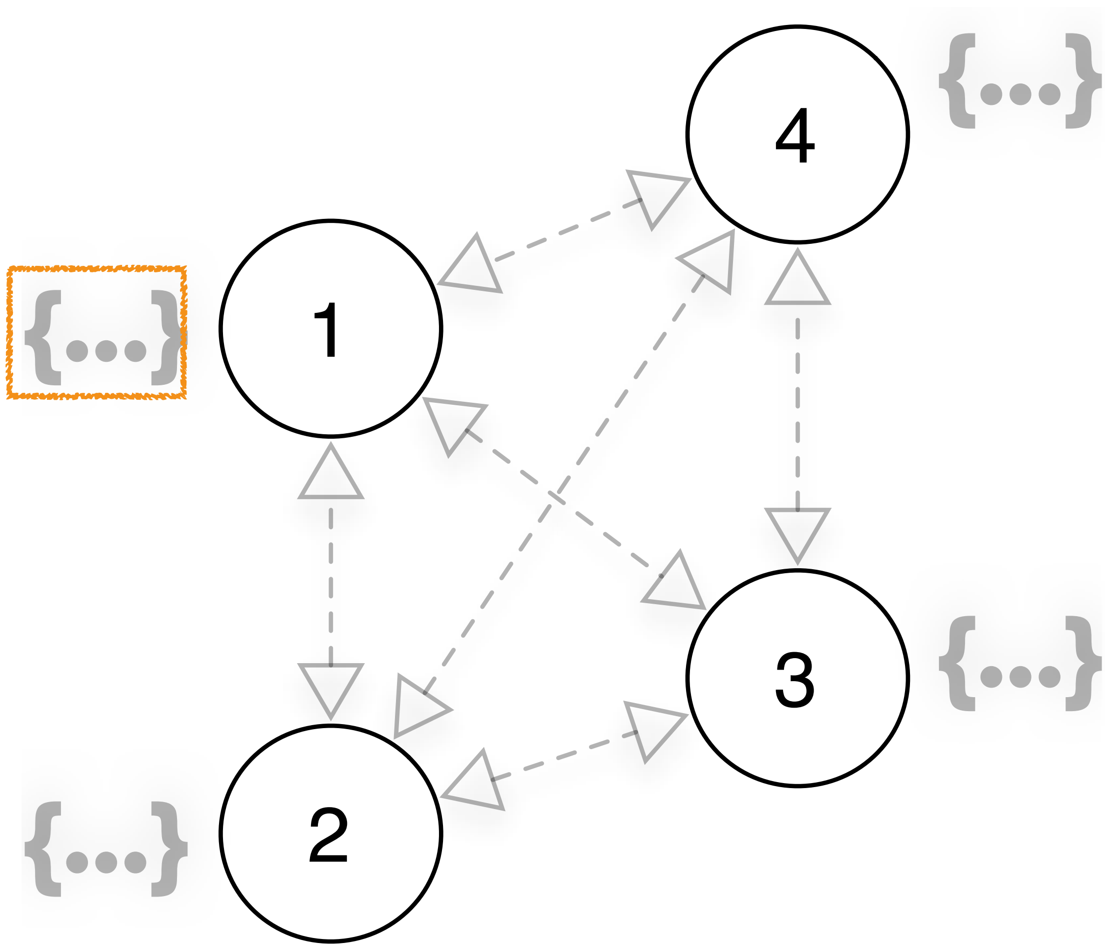
$$\{(\bar{\delta}_{i,j}, \bar{\eta}_{i,j}, \bar{u}_{i,j}, \bar{v}_{i,j}) \mid 1 \leq i, j \leq m\}$$

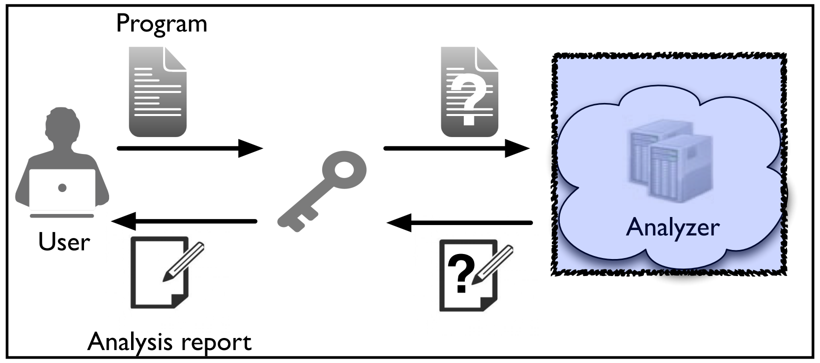
- Total # of cipher texts = $4m^2$

Server's Analysis



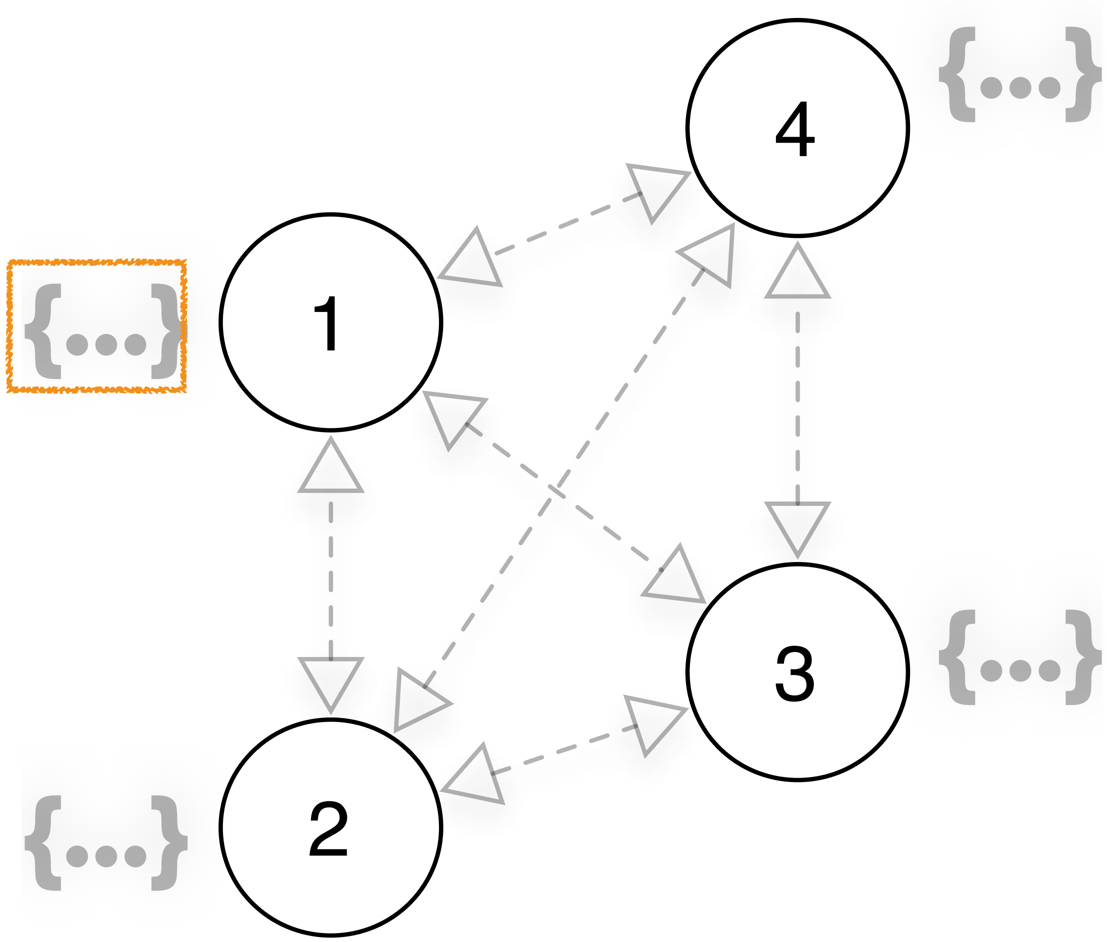
- Ex) deriving $\mathtt{x}_1 \longrightarrow \&\mathtt{x}_5$ in cipher-world





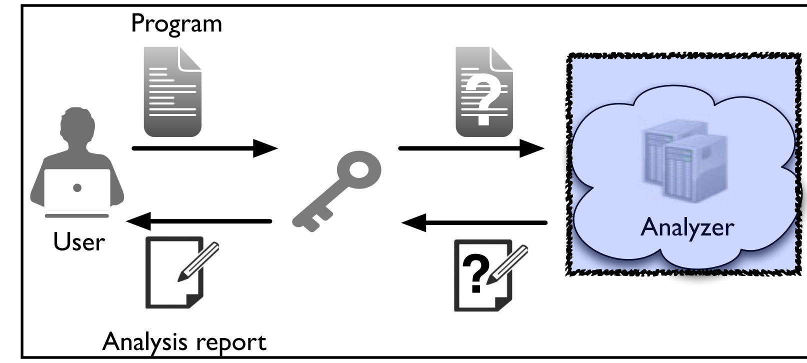
Server's Analysis

- Ex) deriving $\mathtt{x}_1 \longrightarrow \&\mathtt{x}_5$ in cipher-world

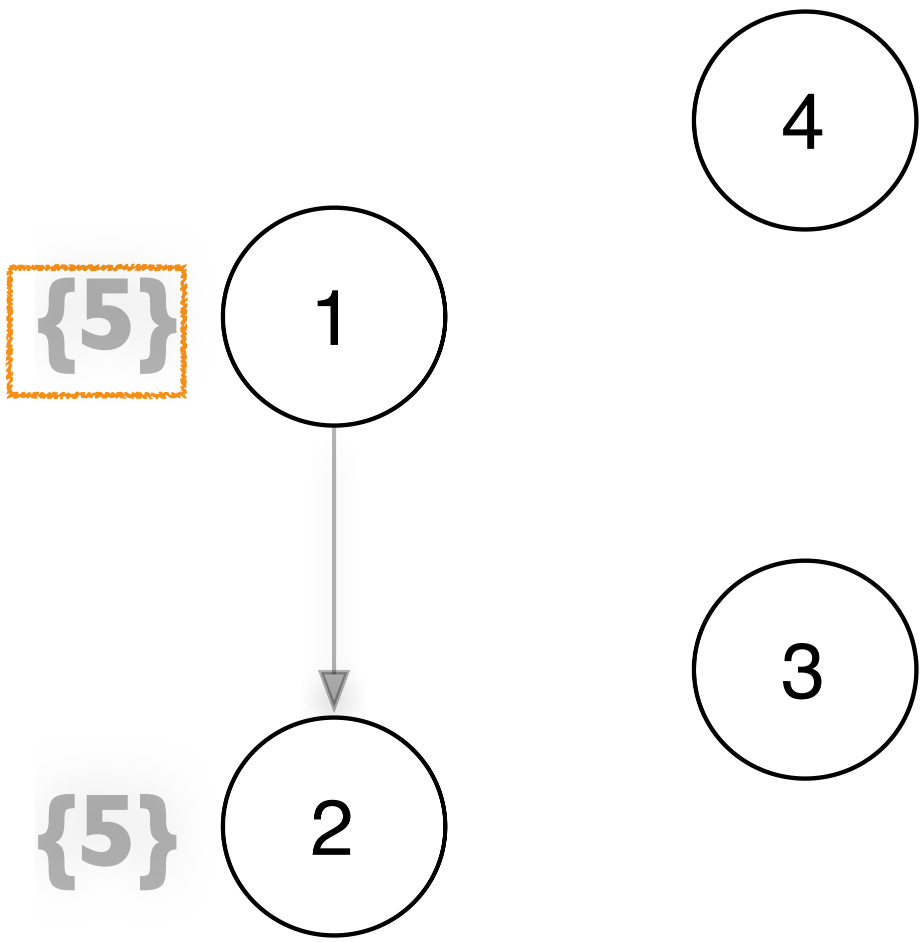


$$\begin{aligned}\bar{\delta}_{1,5} \leftarrow \bar{\delta}_{1,5} \quad &+ \quad \bar{\eta}_{1,2}\cdot\bar{\delta}_{2,5}\\ &+ \quad \bar{\eta}_{1,3}\cdot\bar{\delta}_{3,5}\\ &+ \quad \bar{\eta}_{1,4}\cdot\bar{\delta}_{4,5}\end{aligned}$$

Server's Analysis



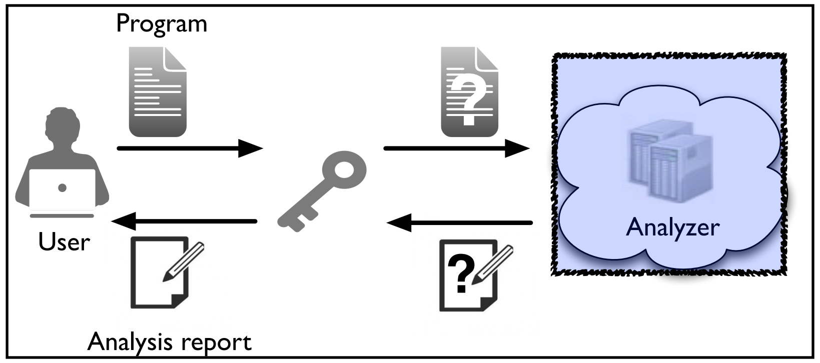
- Ex) deriving $\mathtt{x}_1 \longrightarrow \&\mathtt{x}_5$ in cipher-world



non-zero

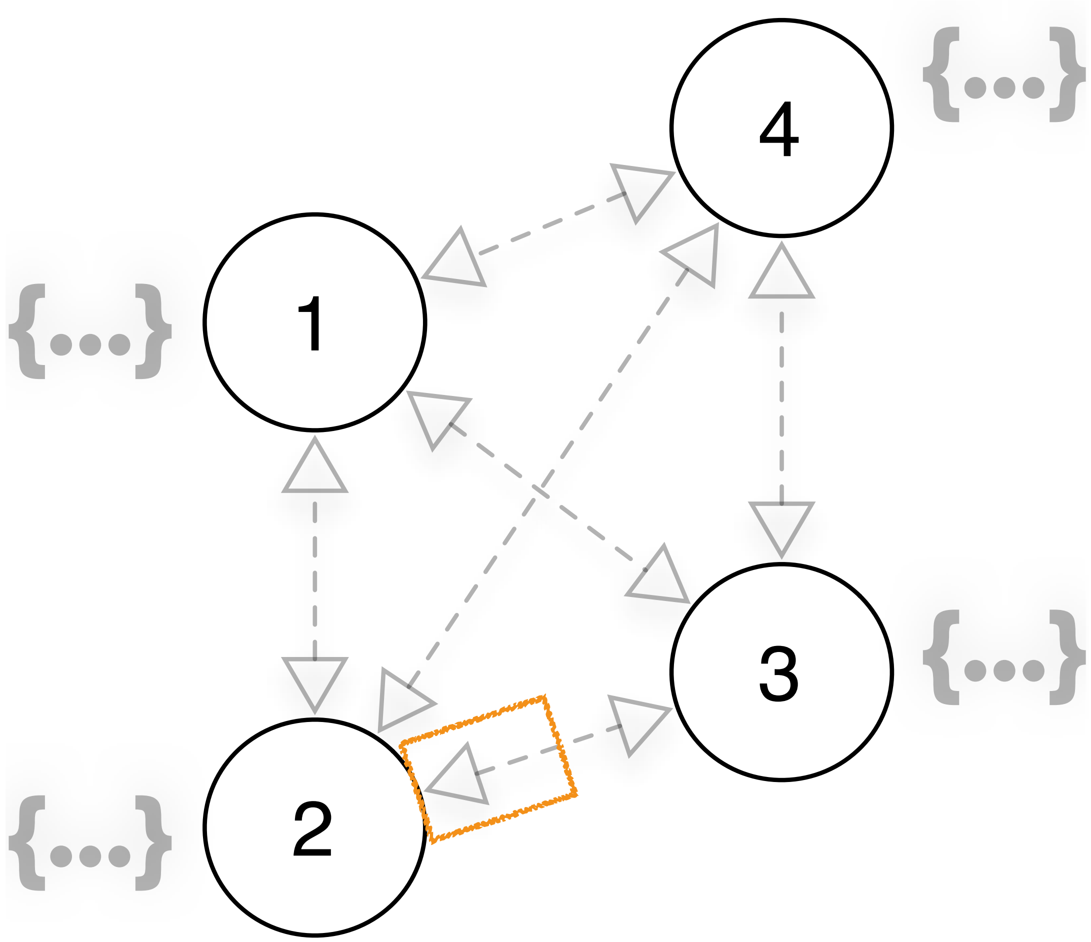
$$\boxed{\bar{\delta}_{1,5}} \leftarrow \bar{\delta}_{1,5} + \boxed{\bar{\eta}_{1,2} \cdot \bar{\delta}_{2,5}} + \bar{\eta}_{1,3} \cdot \bar{\delta}_{3,5} + \bar{\eta}_{1,4} \cdot \bar{\delta}_{4,5}$$

non-zero

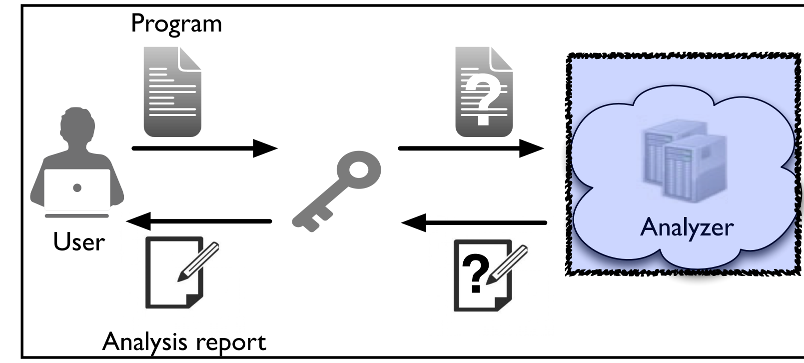


Server's Analysis

- Ex) deriving $\mathtt{x}_3 \longrightarrow \mathtt{x}_2$ in cipher-world

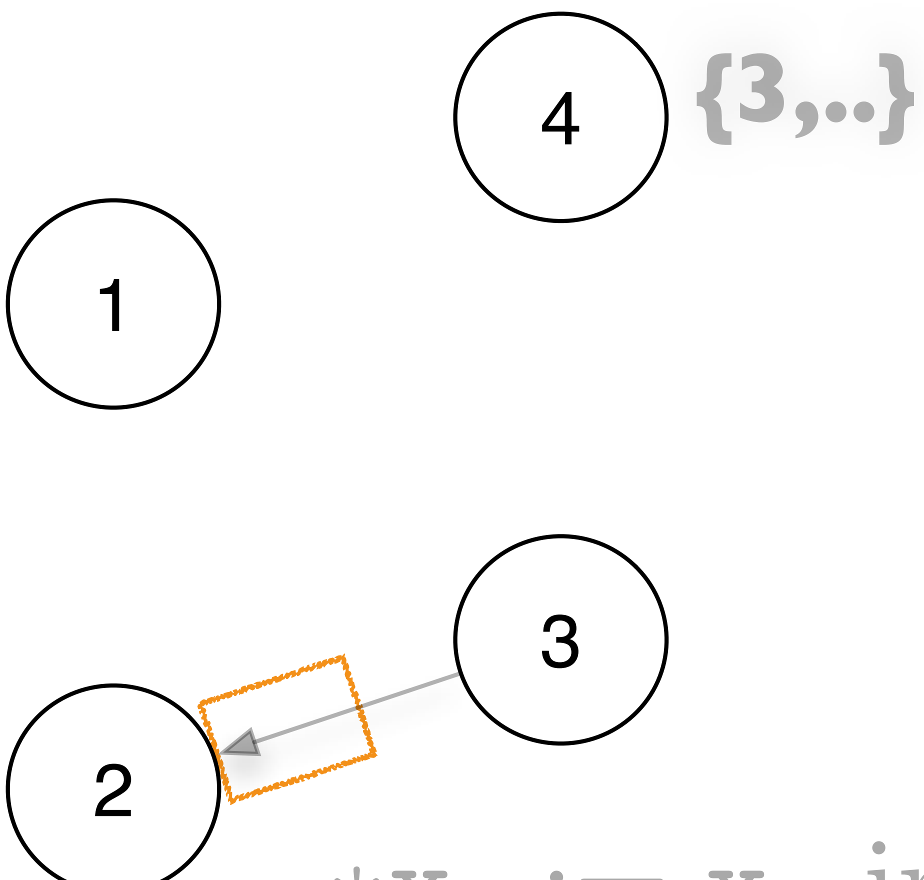


$$\begin{aligned}\bar{\eta}_{3,2} \leftarrow \bar{\eta}_{3,2} \quad &+ \quad \bar{v}_{2,1} \cdot \bar{\delta}_{1,3} \\ &+ \quad \bar{v}_{2,3} \cdot \bar{\delta}_{3,3} \\ &+ \quad \bar{v}_{2,4} \cdot \bar{\delta}_{4,3}\end{aligned}$$



Server's Analysis

- Ex) deriving $\mathtt{x}_3 \longrightarrow \mathtt{x}_2$ in cipher-world



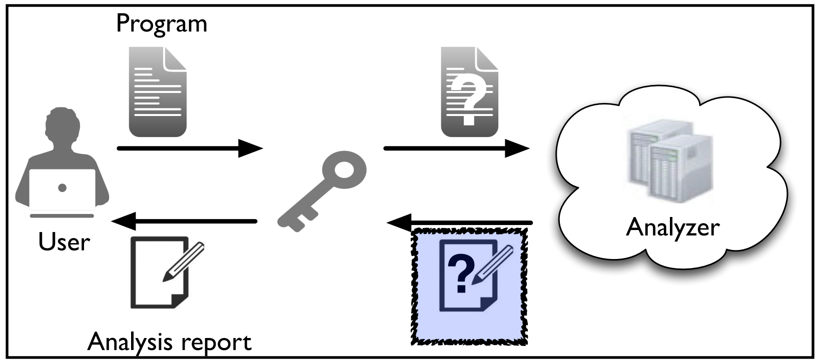
$$\boxed{\bar{\eta}_{3,2}} \leftarrow \bar{\eta}_{3,2} + \bar{v}_{2,1} \cdot \bar{\delta}_{1,3} + \bar{v}_{2,3} \cdot \bar{\delta}_{3,3} + \boxed{\bar{v}_{2,4} \cdot \bar{\delta}_{4,3}}$$

non-zero

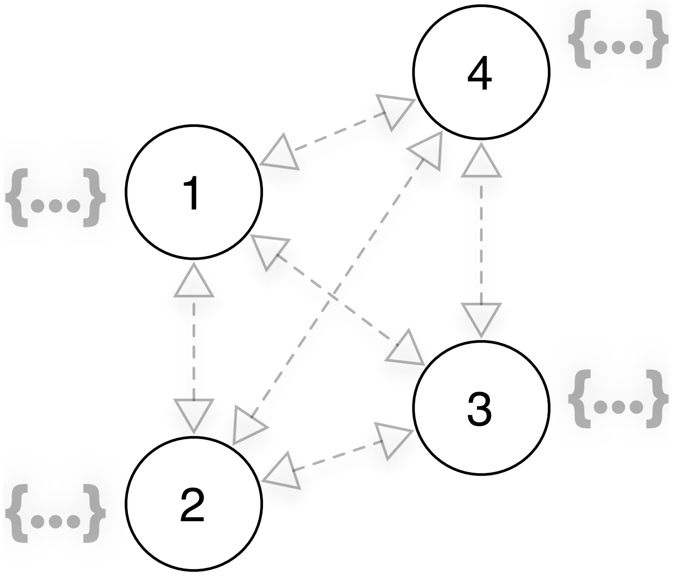
non-zero

$*\mathtt{x}_4 := \mathtt{x}_2 \text{ in } P$

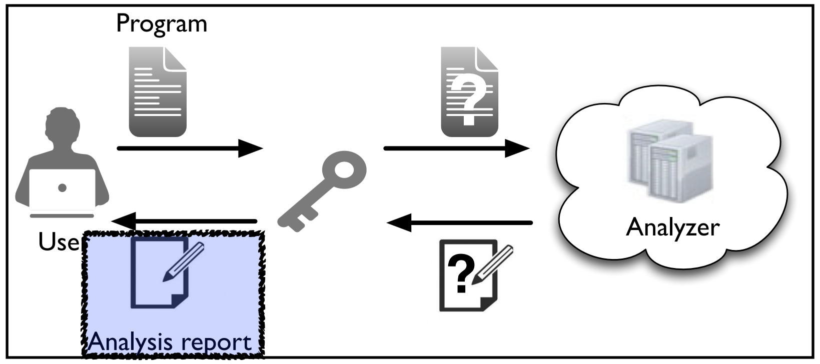
Server's Analysis



- Repeat the following 2 steps m^2 times
  - repeat propagation $m - 1$ times
  - edge addition



- NOTE : we must repeat <u>as if in the worst case</u> since we do not know whether a fixpoint reached.



Output Determination

- The client receives the updated points-to set info. from the server

$$\{\bar{\delta}_{i,j} \mid 1 \leq i, j \leq m\}$$

- derives points-to relations involving variables of interest, namely x_i

$$\{\mathtt{x_i} \longrightarrow \&\mathtt{x_j} \mid \mathsf{HE.Dec_{sk}}(\bar{\delta}_{i,j}) \neq 0 \text{ and } 1 \leq i, j \leq m\}$$

Problems of the Approach

- Huge multiplicative depth : $O(m^2 \log m)$

  ($\because$ [($\delta_{i,j}$ update) x m times $\Rightarrow$ ($\eta_{i,j}$ update)] x m^2 times

  ex) m = 10 $\Rightarrow$ depth > 300

- Huge # of ciphertexts : $4m^2$

  ex) m = 1000 $\Rightarrow$ 4M ciphertexts take over 8TB

- Decryption error may happen : during operations, non-zero values can be zero by accident.

  (msg space is $\mathbb{Z}_t$, and values may be the modulus. (e.g. $t \text{ in } \mathbb{Z}_t$))

Our Optimized Solution

- Huge multiplicative depth ⇛ **Level-by-level analysis**

  Depth: **O(m² log m)** → **O(n log m)** (n : maximal pointer level (< 5))

  (e.g. ptl(x : int**) = 2, ptl(y : int***) = 3)

- Huge # of ciphertexts ⇛ **Ciphertext packing**

$$\bar{u}_{i,1}, \bar{u}_{i,2}, \cdots, \bar{u}_{i,m} \longrightarrow \overline{\langle u_{i,1}, u_{i,2}, \cdots u_{i,m} \rangle}$$

  # necessary cipher texts : **$4m^2$ → (2n+2)m**

- Decryption error may happen ⇛ **Randomize the messages**

  balancing between ciphertext size and the prob. of incorrectness

  e.g. the success prob. is about 95% when n=2, m=1000, t=503

Homomorphic Matrix Multiplication

- The pointer analysis can be represented in matrix form.

Step	Integer form	Matrix form
Propagation	$\delta_{i,j} \leftarrow \sum_{k=1}^{m} \eta_{i,k} \cdot \delta_{k,j}$	$\Delta \leftarrow H \cdot \Delta$
Edge addition (Load)	$\eta_{i,j} \leftarrow \eta_{i,j} + \sum_{k=1}^{m} u_{i,k} \cdot \delta_{k,j}$	$H \leftarrow H + U \cdot \Delta$
Edge addition (Store)	$\eta_{i,j} \leftarrow \eta_{i,j} + \sum_{k=1}^{m} v_{j,k} \cdot \delta_{k,i}$	$H \leftarrow H + (V \cdot \Delta)^T$

- We encrypt a matrix in row-order
  - e.g. $\bar{\boldsymbol{\delta}}_i \leftarrow \text{BGV.Enc}(\delta_{i,1}, \cdots, \delta_{i,m})$

$$\bar{\Delta} \leftarrow \langle \bar{\boldsymbol{\delta}}_i, \cdots, \bar{\boldsymbol{\delta}}_m \rangle$$

- We can perform homomorphic matrix addition, multiplication, and transposition.

Experimental Result

- HW: Parallelized on 24 cores of Intel Xeon 2.6 GHz
  SW: HElib 1.3 - a library that implements BGV scheme
- Security : 72 (2^{72} brute force needed to break)

Program	**LOC**	**# Var**	**Enc**	**Propagation**	**Edge addition**	**Total**	**Depth**
toy	10	9	17s	28m 49s	5m 58s	35m 4s	37
buthead-1.0	46	17	48s	5h 41m 36s	56m 19s	6h 38m 43s	43
wysihtml-0.13	202	32	1m 39s	10h 41m 52s	1h 48m 39s	12h 30m 31s	49
cd-discid-1.1	259	41	2m 12s	12h 5m 20s	2h 3m 21s	14h 10m 53s	49

Applications

- Privacy preserving static-analysis-as-a-service
  - e.g. http://rosaec.snu.ac.kr/clinic (our own)
    https://scan.coverity.com
- Privacy preserving app reviewing
  - e.g. Apple app review system (currently on executables)
  - reviewing on encrypted app sources

Future Direction

- Adapting other kinds of analysis operations (arbitrary $\sqcup$, $\sqsubseteq$, semantic operations) into HE schemes.
- Allowing users to encrypt only sensitive sub-parts of programs

Backup

Level-by-level Analysis

- Analyzing the same pointer level together from the highest to lowest
- Lower levels cannot affect higher levels.
  ex) value of x may change by

Assignment	Levels
$\mathtt{x} = \mathtt{y}$	$\mathsf{ptl}(\mathtt{x}) = \mathsf{ptl}(\mathtt{y})$
$\mathtt{x} = *\mathtt{y}$	$\mathsf{ptl}(\mathtt{y}) = \mathsf{ptl}(\mathtt{x}) + 1$
$*\mathtt{p} = \mathtt{y}$	$\mathsf{ptl}(\mathtt{p}) = \mathsf{ptl}(\mathtt{x}) + 1 \wedge \mathsf{ptl}(\mathtt{y}) = \mathsf{ptl}(\mathtt{x})$

p and y have higher or equal level compared to x.

Level-by-level Analysis

- User provides

$$\{(\delta^{(\ell)}_{i,j}, \eta^{(\ell)}_{ij}) \mid 1 \le i,j \le m, 1 \le \ell \le n\} \cup \{(u_{i,j}, v_{i,j}) \mid 1 \le i,j \le m\}$$

$$\delta^{(\ell)}_{i,j} = \begin{cases} 1 & \text{if } \exists \mathtt{x_i} = \&\mathtt{x_j}, \boxed{\mathsf{ptl}(\mathtt{x_i}) = \ell} \\ 0 & \text{o.w.} \end{cases} \qquad \eta^{(\ell)}_{i,j} = \begin{cases} 1 & \text{if } (\exists \mathtt{x_i} = \mathtt{x_j} \text{ or } i = j), \boxed{\mathsf{ptl}(\mathtt{x_i}) = \ell} \\ 0 & \text{o.w.} \end{cases}$$

- ($\delta^{(n)}_{i,j}$ update) x m times
  $\Rightarrow$ ($\eta^{(n-1)}_{i,j}$ update) $\Rightarrow$ ($\delta^{(n-1)}_{i,j}$ update) x m times
  $\Rightarrow$ …
  $\Rightarrow$ ($\eta^{(1)}_{i,j}$ update) $\Rightarrow$ ($\delta^{(1)}_{i,j}$ update) x m times

- The multiplicative depth = O(n log m)

Ciphertext Packing

- We should use the BGV scheme.
- A vector of plaintext can be encrypted into a single ciphertext

$$\bar{\mathbf{c}}_1 = \mathsf{BGV.Enc}(\mu_{1,1}, \cdots, \mu_{1,m})$$

$$\bar{\mathbf{c}}_2 = \mathsf{BGV.Enc}(\mu_{2,1}, \cdots, \mu_{2,m})$$

$$\mathsf{BGV.Add}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2) \equiv \mathsf{BGV.Enc}(\mu_{1,1} + \mu_{2,1}, \cdots, \mu_{1,m} + \mu_{2,m})$$

$$\mathsf{BGV.Mult}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2) \equiv \mathsf{BGV.Enc}(\mu_{1,1} \cdot \mu_{2,1}, \cdots, \mu_{1,m} \cdot \mu_{2,m})$$

Homomorphic Matrix Multiplication

- The pointer analysis can be represented in matrix form.

Step	Integer form	Matrix form
Propagation	$\delta_{i,j}^{(\ell)} \leftarrow \sum_{k=1}^{m} \eta_{i,k}^{(\ell)} \cdot \delta_{k,j}^{(\ell)}$	$\Delta_\ell \leftarrow H_\ell \cdot \Delta_\ell$
Edge addition (Load)	$\eta_{i,j}^{(\ell)} \leftarrow \eta_{i,j}^{(\ell)} + \sum_{k=1}^{m} u_{i,k} \cdot \delta_{k,j}^{(\ell+1)}$	$H_\ell \leftarrow H_\ell + U \cdot \Delta_{\ell+1}$
Edge addition (Store)	$\eta_{i,j}^{(\ell)} \leftarrow \eta_{i,j}^{(\ell)} + \sum_{k=1}^{m} v_{j,k} \cdot \delta_{k,i}^{(\ell+1)}$	$H_\ell \leftarrow H_\ell + (V \cdot \Delta_{\ell+1})^T$

- We encrypt a matrix in row-order

$$\bar{\boldsymbol{\delta}}_i^{(\ell)} \leftarrow \mathsf{BGV.Enc}(\delta_{i,1}^{(\ell)}, \cdots, \delta_{i,m}^{(\ell)})$$

$$\bar{\Delta}_\ell \leftarrow \langle \bar{\boldsymbol{\delta}}_i^{(\ell)}, \cdots, \bar{\boldsymbol{\delta}}_m^{(\ell)} \rangle$$

similarly, $H_\ell = [\eta_{i,j}^{(\ell)}], U = [u_{i,j}], \text{ and } V = [v_{i,j}]$

Homomorphic Matrix Multiplication

- The i-th row of $H \cdot \Delta$ is

$$(\sum_j \eta_{i,j}\delta_{j,1}, \cdots, \sum_j \eta_{i,j}\delta_{j,m}) = \sum_j \eta_{i,j} \cdot (\delta_{j,1}, \cdots, \delta_{j,m})$$

- Using the Replication operator supported by the BGV scheme

$$\mathsf{Replicate}(\bar{c}, i) \equiv \mathsf{BGV.Enc}(\mu_i, \cdots, \mu_i)$$

where $\bar{c} = \mathsf{BGV.Enc}(\mu_1, \cdots, \mu_n)$

- we compute the encrypted i-th row of $H \cdot \Delta$

$$\mathsf{BGV.Mult}\left(\mathsf{Replicate}(\bar{\boldsymbol{\eta}}_i, 1), \bar{\boldsymbol{\delta}}_1\right) + \cdots + \mathsf{BGV.Mult}\left(\mathsf{Replicate}(\bar{\boldsymbol{\eta}}_i, m), \bar{\boldsymbol{\delta}}_m\right).$$

Randomizing the Messages

- Let $\mathbb{Z}_t$ be message space for a prime t
- During operations $(H\mathit{\Delta})_{i,j} = \eta_{i,1} \cdot \delta_{1,j} + \cdots + \eta_{i,m} \cdot \delta_{m,j}$ $(H\Delta)_{i,j}$ may accidentally become $kt (= 0 \text{ in } \mathbb{Z}_t)$ when it overflows
- In the following computation
$$(H\mathit{\Delta})'_{i,j} = r_1 \cdot \eta_{i,1} \cdot \delta_{1,j} + \cdots + r_m \cdot \eta_{i,m} \cdot \delta_{m,j}$$
the prob. that $(H\Delta)'_{i,j} \equiv 0 \bmod p$ is less than $\frac{1}{p-1}$ where $r_1, \cdots, r_m$ are randomly chosen.

Randomizing the Messages

- Since we need $n(\lceil \log m \rceil + 3) - 2$ matrix multiplications
- the probability of correct results is greater than

$$(1 - \frac{1}{t-1})^{n(\lceil \log m \rceil + 3)}$$

- e.g. the success prob. is about 95% when n=2, m=1000, t=503

The Pointer Analysis in Secrecy

User's setting: For $i, j = 1, \ldots, m$

$$
\begin{aligned}
\delta_{i,j}^{(\ell)} &\leftarrow 1 && \text{if } (\exists \mathtt{x_i} = \&\mathtt{x_j}) \wedge (ptl(x_i) = \ell) \\
\eta_{i,j}^{(\ell)} &\leftarrow 1 && \text{if } (\exists \mathtt{x_i} = \mathtt{x_j}) \wedge (ptl(x_i) = \ell) \\
u_{i,j} &\leftarrow 1 && \text{if } \exists \mathtt{x_i} = *\mathtt{x_j} \\
v_{i,j} &\leftarrow 1 && \text{if } \exists *\mathtt{x_i} = \mathtt{x_j}
\end{aligned}
$$

Inputs from user: $\{\bar{\Delta}_\ell = \mathrm{Enc}[\delta_{i,j}^{(\ell)}], \bar{H}_\ell = \mathrm{Enc}[\eta_{i,j}^{(\ell)}] \mid 1 \le \ell \le n\}$,
$\bar{U} = \mathrm{Enc}[u_{i,j}]$, $\bar{V} = \mathrm{Enc}[v_{i,j}]$

Server computation:

- Propagation $\bar{\Delta}_n \leftarrow \bar{\Delta}_n \cdot \bar{H}_n^m$
- For $\ell = n - 1$ down to 1
  - Edge addition $\bar{H}_\ell \leftarrow \bar{H}_\ell + \bar{U} \cdot \bar{\Delta}_{\ell+1} + \bar{\Delta}_{\ell+1}^T \cdot V$
  - Propagation $\bar{\Delta}_\ell \leftarrow \bar{\Delta}_\ell \cdot \bar{H}_\ell^m$

Output determination: The user receives $\{\delta_{i,j}^{(\ell)} \mid 1 \le i, j \le m, 1 \le \ell \le n\}$ and derives points-to sets of a variable of interest, namely x_j as follows:

$pt(\mathtt{x_i}) = \{\mathtt{x_j} \mid \delta_{i,j}^{(\ell)} \neq 0 \wedge ptl(\mathtt{x_i}) = \ell\}$