11th International Conference on Reliable Software Technologies, Porto, Portugal, 5 - 9 June, 2006

Secure Execution of Computations in Untrusted Hosts

S. H. K. Narayanan[1], M.T. Kandemir[1], R.R. Brooks[2] and I. Kolcu[3]

[1] **Embedded Mobile Computing Center (EMC²)**
The Pennsylvania State University.

[2] **Department of Electrical and Computer Engineering,**
Clemson University.

[3] **The University of Manchester**

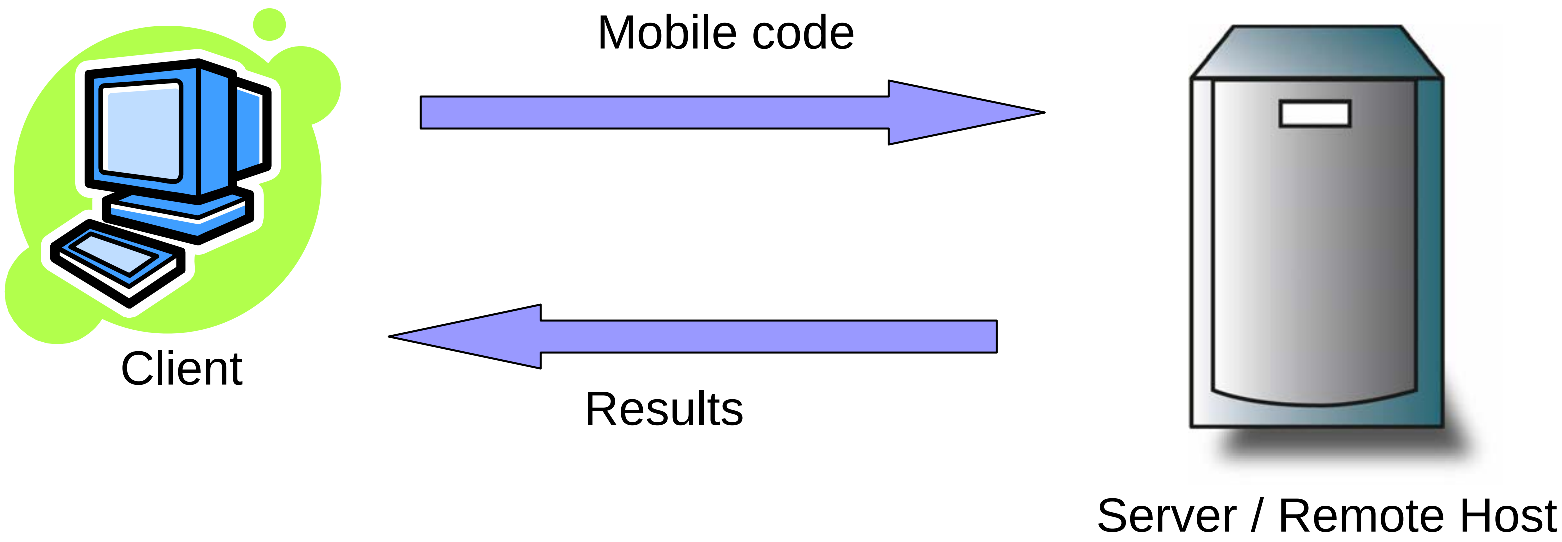
Outline

- Mobile Code
- Security Concerns with Mobile Code
- Some Related Work
- High Level Views
- Mathematical Details
- Example
- Experiments

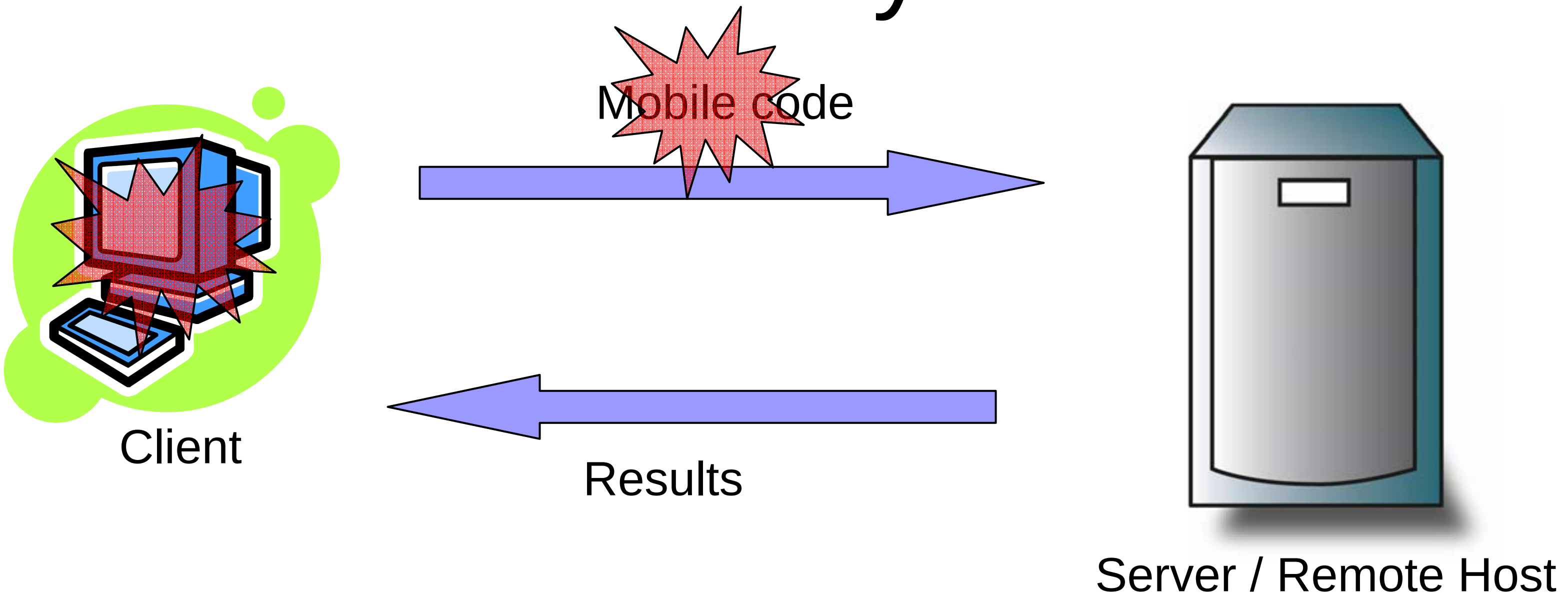
What is Mobile Code?

- Code belonging to a client that is executed on a remote host.
- Not just relegated to a mobile platform.
- Applicable where data is not movable but code is.
  - Due to large volume or concerns for privacy.

Mobile code is being widely used for a variety of applications

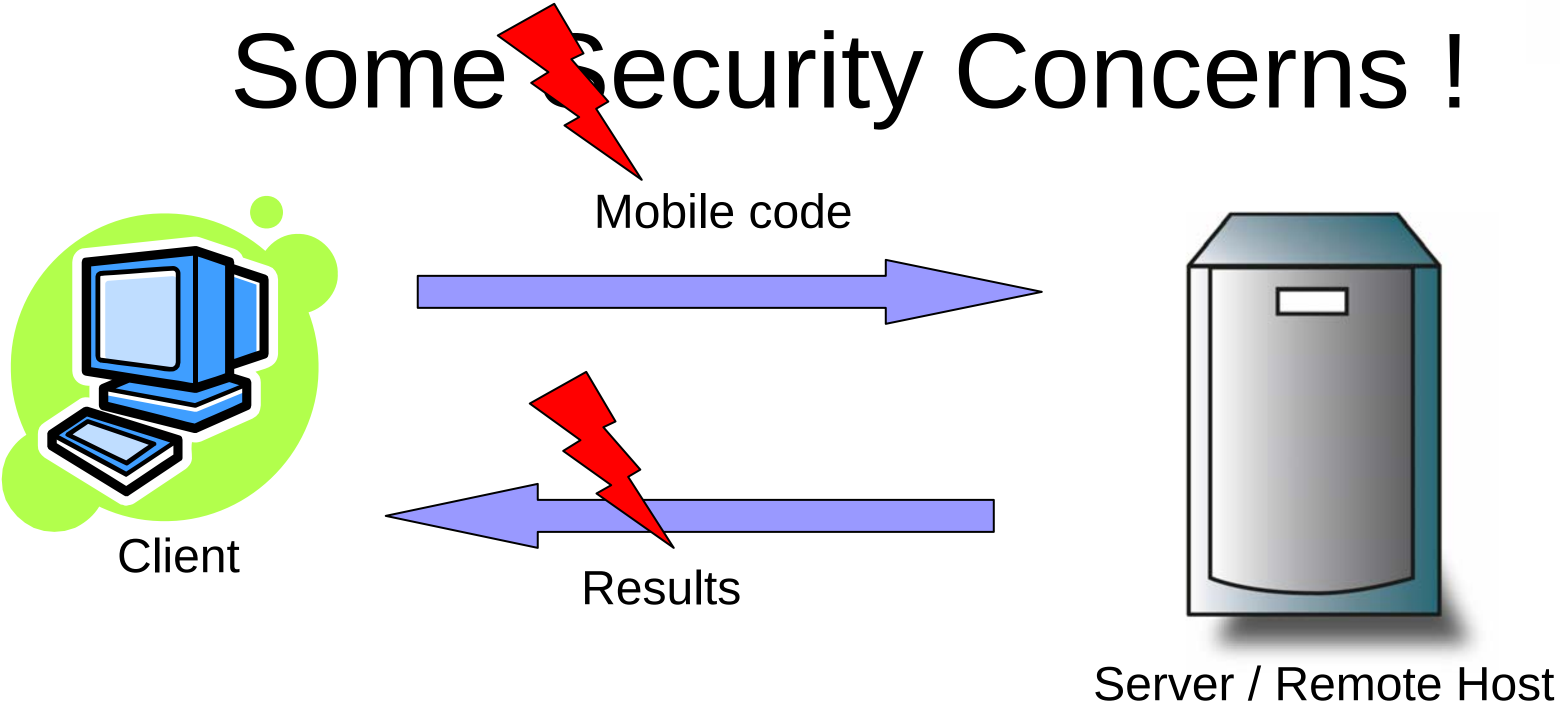


Some Security Concerns !



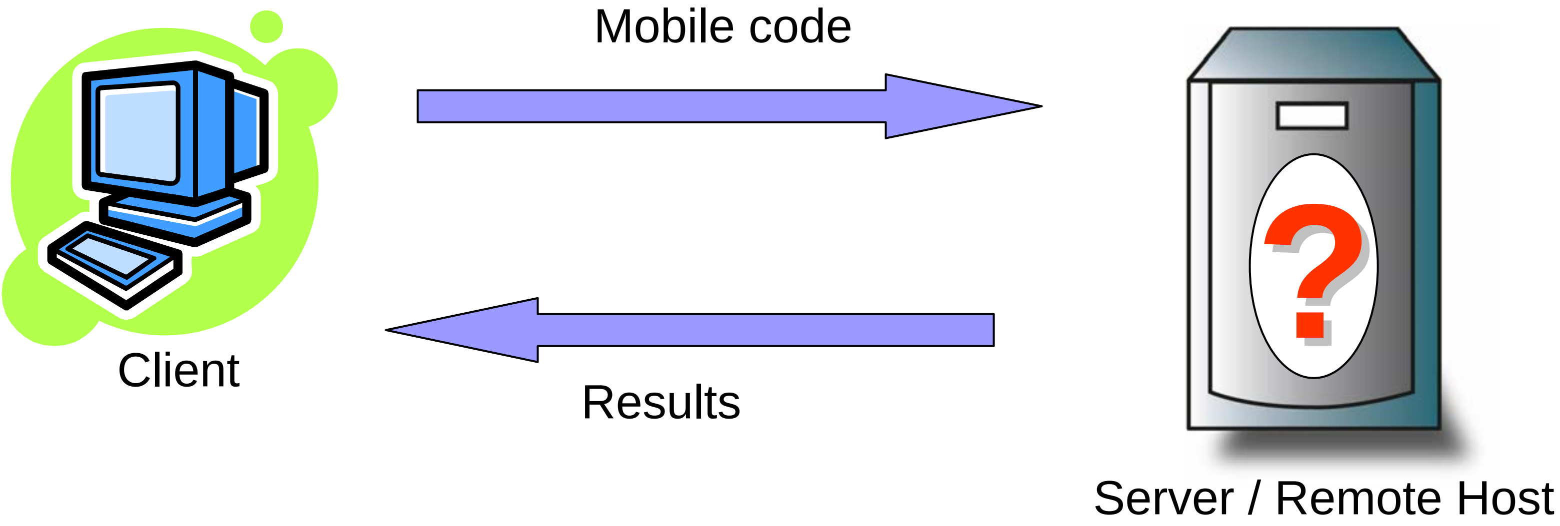
- Threat : To the host from malicious code/ malicious client
- Solution : Run the code in a Sandbox.

Some Security Concerns !



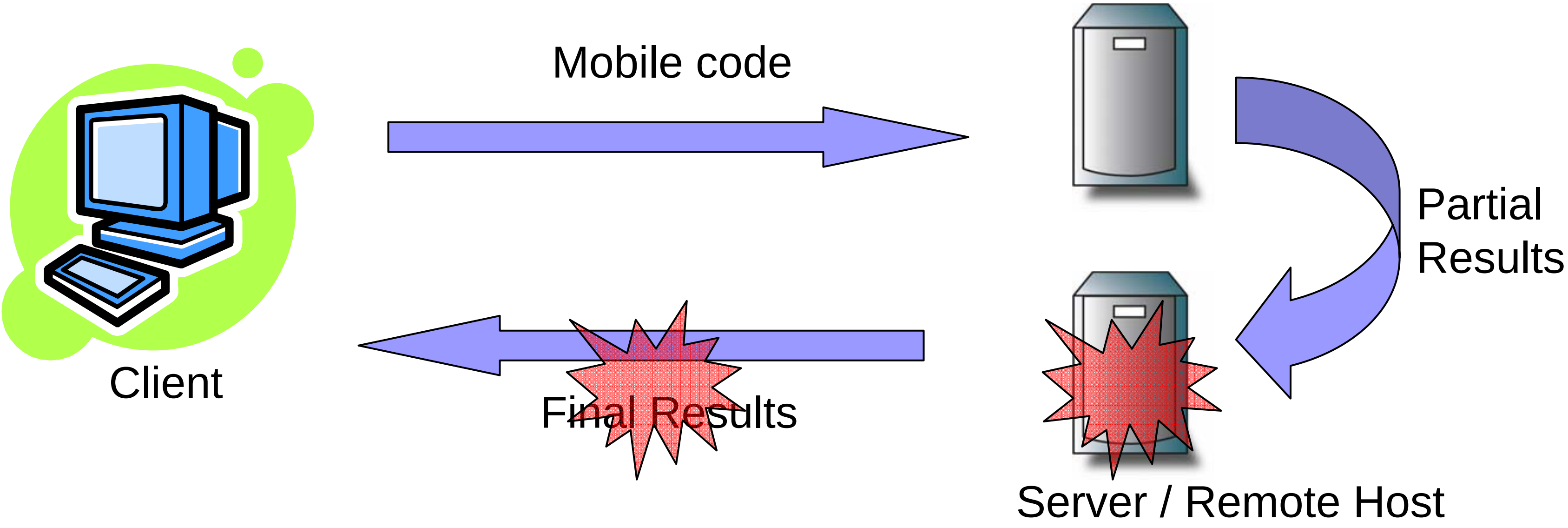
- Threat : To the code/results from intermediate attacks.
- Solution : Encryption and authentication techniques.

Some Security Concerns !



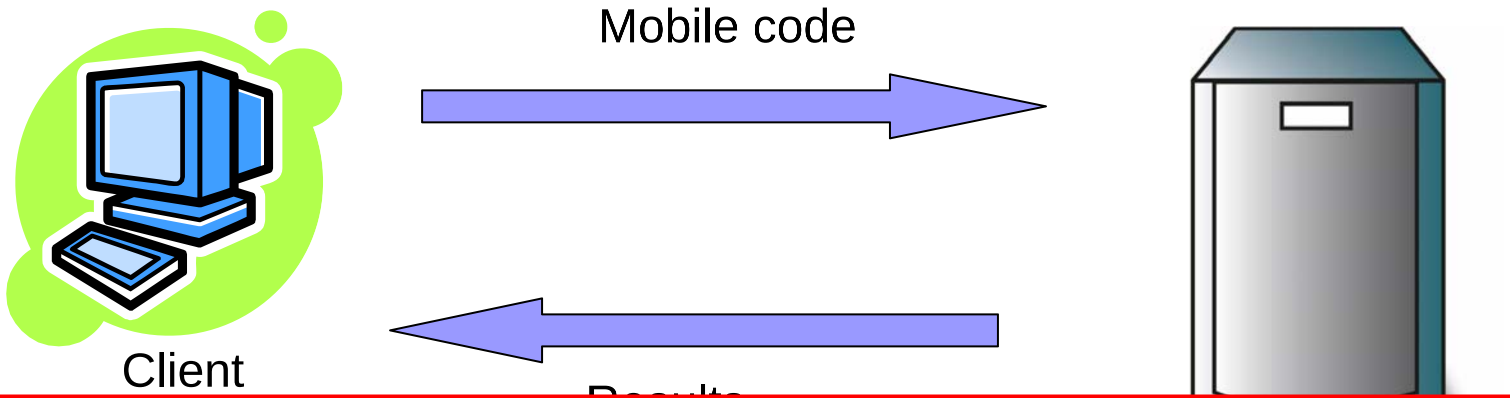
- Threat : Will the right code be executed at all?
- Solution : Make the remote host include a proof of correct execution.

Some Security Concerns !



- Threat : One server changing the intermediate result generated by another?
- Solution : Encryption Techniques.

Some Security Concerns !



This paper presents a method to protect the semantics of the mobile code that is to be executed at a remote host. Thus, a client's intellectual capital is preserved.

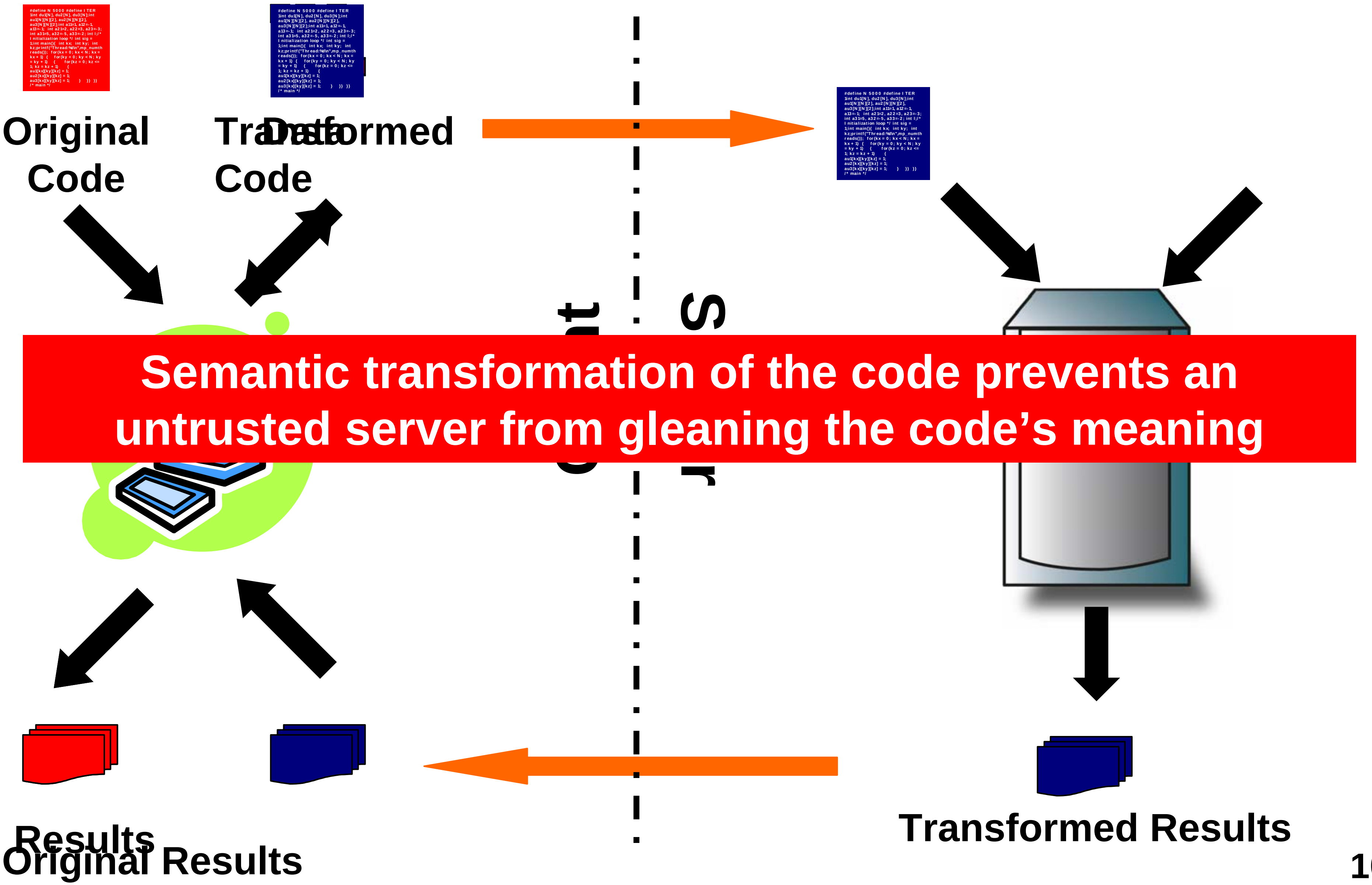
particularly important when the algorithm used is a proprietary one.

- Solution :

Some Related Work in Code Privacy

- Code Obfuscation
  - Collberg et al. 1997, Hohl 1997, Jansen et al.
  - Makes the code hard to read
- Function hiding scheme
  - Sander and Tschudin
  - Encrypting transformation applied to the function.
- Encrypted functions
  - Loureiro et al.
  - Host runs code encrypted with error codes
  - Requires tamper proof hardware support

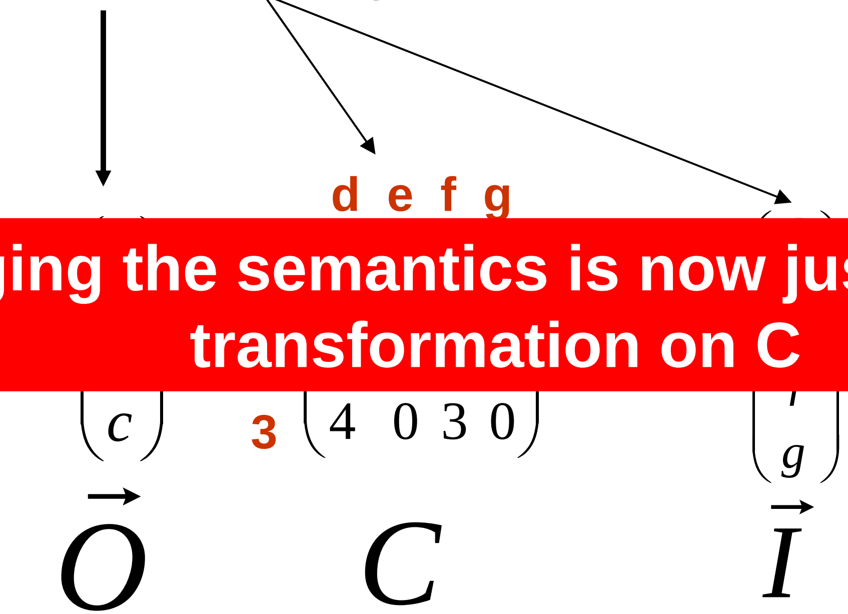
Scalar Codes - High level view



Semantic transformation of the code prevents an untrusted server from gleaning the code’s meaning

Transformation – Scalar Codes

1 a := d + e + f ;
2 b := g -2e ;
3 c := 3f + 4d;



Changing the semantics is now just an matrix transformation on C

- **Obtain Computation matrix, C.**
  - **Rows correspond to statements**
  - **Columns correspond to variables**
- **By multiplying C and I, the output vector O is obtained.**
- **Using a different C means that different code is executed.**

Transformation – Scalar Codes

$$C * T = C'$$

- Client uses a transformation matrix T to transform C into C'.
- C' is sent to the untrusted server.
- The server then executes C' to produce O' and sends it to the client.

$$\vec{O}' * M = \vec{O}$$

- Client uses an inverse transformation matrix M to obtain O.
- O is the same vector that would have been obtained had C been executed locally at the client.

Selection of T and M

$$\vec{O} = M\vec{O}'$$

$$\vec{O} = MC'\vec{I}$$

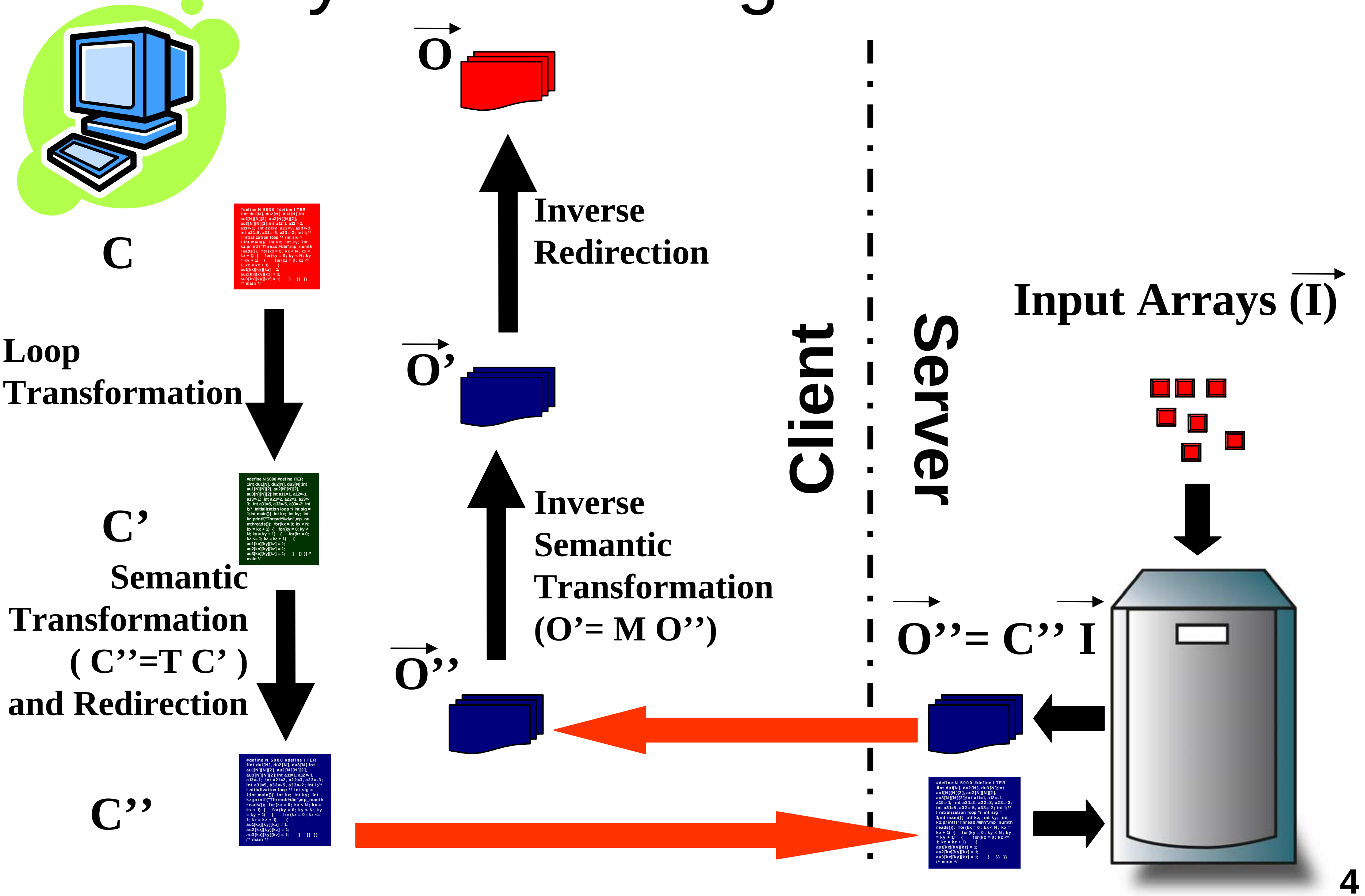
$$\vec{O} = MTC\vec{I}$$

$$C\vec{I} = MTC\vec{I}$$

$$\therefore C = MTC$$

- T and M should be the inverse of each other.
- Dimensionalities
  - If C is an m * n matrix, then M is m * k and T is k * m.
  - This means that we can introduce extra statements into C’ that did not exist in C.

Array Codes - High level view



Transformation –Array Codes

- Array based codes give more opportunities for transformation
  - Loop Transformation on the loop bounds
    - Does not change the semantics, simply the order in which the elements are accessed.
    - **C → C'**

$$Li + o$$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$$LT_L^{-1} i' + o$$

1	5	8	10
11	2	6	9
14	12	3	7
16	15	13	4

Transformation –Array Codes

- Semantic Transformation on the body
  - Does not change the loop bounds
  - Client uses a transformation vector T to transform C’ into C’’.

$D_i = A_j + B_k$ $\qquad$ $D_i = B_k - A_j$

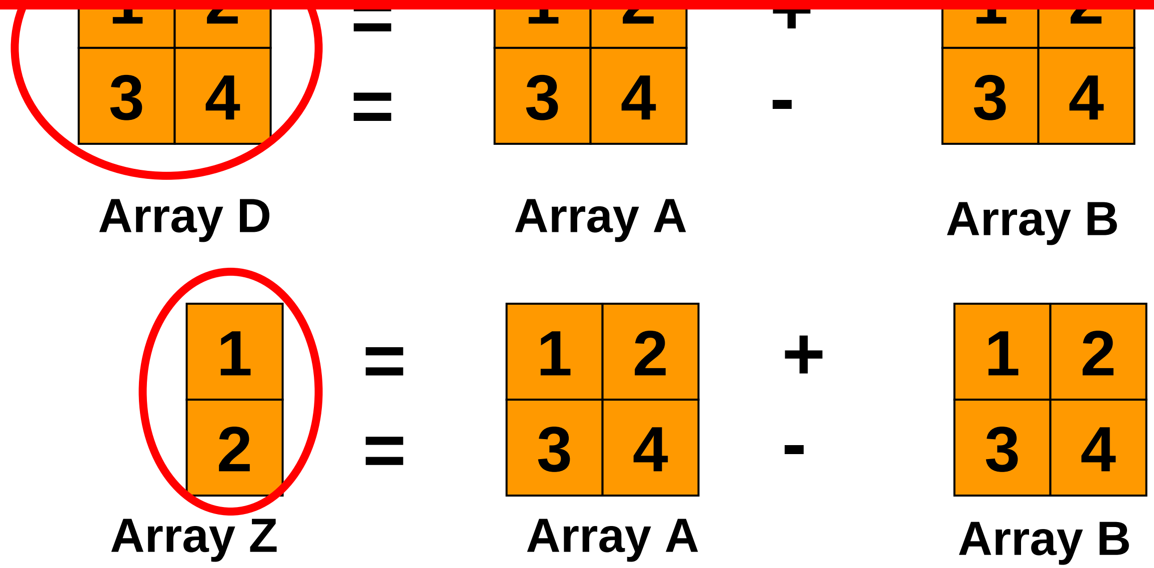
Transformation –Array Codes

- Redirection
  - Data transformation that changes the locations to which the assignments are performed.
  - The references in Array D, *Li+o,* are transformed

The untrusted server now executes a code that is semantically different, accesses data in a different pattern and whose stores take place to different locations.

$Li + o$

$SLi + So + s$



Transformation –Array Codes

- The untrusted server executes O'' = C'' * I.
- Client uses the inverse semantic transformation matrix M to transform O'' into O'.
- Inverse redirection using an inverse data transformation, *{Y,y}*, is then performed.

Each location in O' is referred to by $SLi + So + s$

Using the inverse data transformation we get,

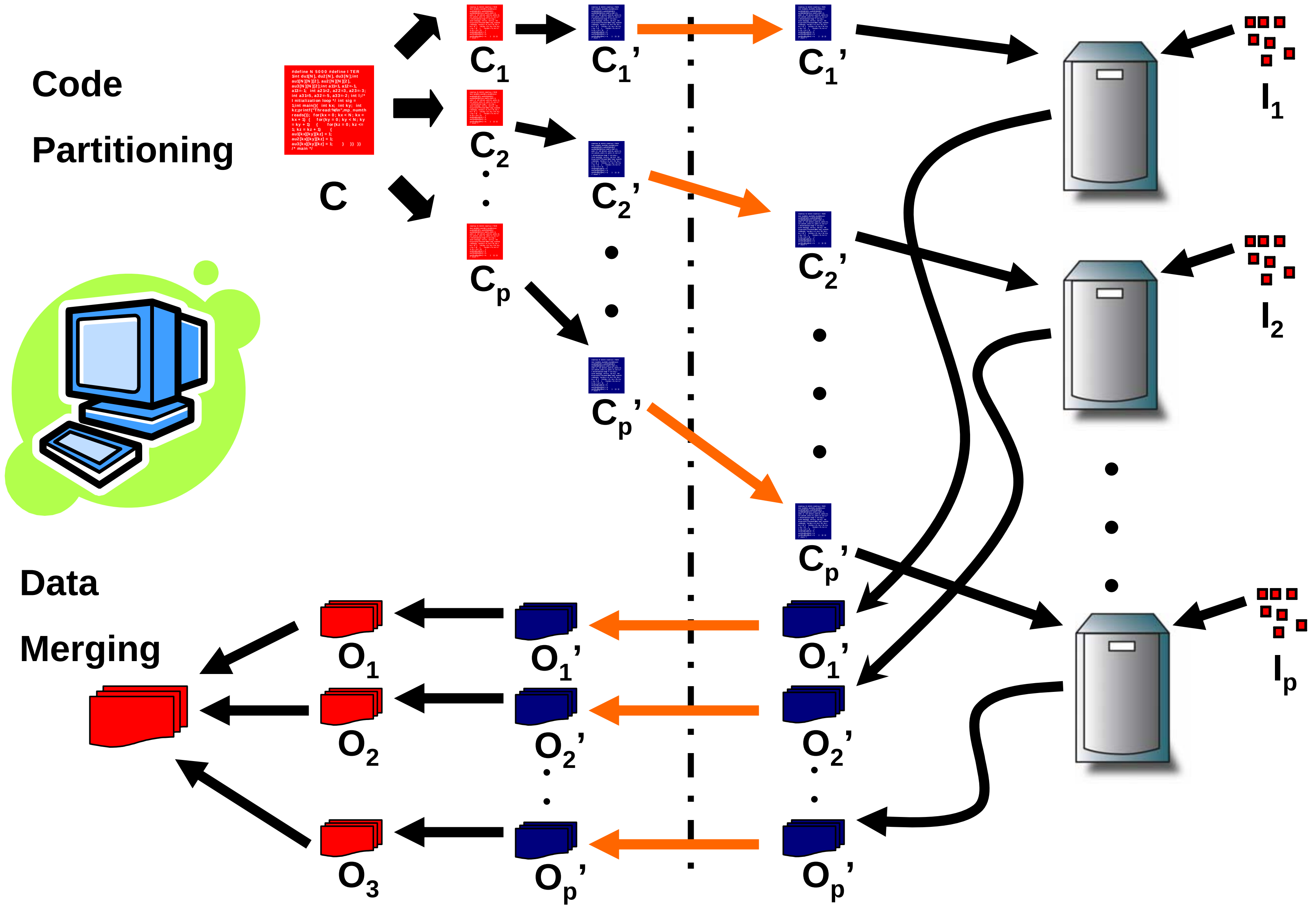
$$Y\{SLi + So + s\} + y$$

$$= YSLi + LSo + Ls + y$$

$$= Li + o$$

$$\therefore Y = S^{-1} \text{ and } y = -S^{-1}s$$

Multiple Hosts- High level view



Example – Scalar Code (1/4)

- Snippet of code from Mediabench benchmark.
- How would the code run locally on the client?

```
dx0 = x0 – x1 – x12
dy0 = y0 – y1 – y12
dx1 = x12 – x2 + x3
dy1 = y12 – y2 – y3
```

Code

$$C = \begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Computation Matrix

$$\vec{I} = \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x12 \\ y0 \\ y1 \\ y2 \\ y3 \\ y12 \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{pmatrix}$$

Input Vector

$$\vec{O} = \begin{pmatrix} dx0 \\ dy0 \\ dx1 \\ dy1 \end{pmatrix} = C * \vec{I} = \begin{pmatrix} -10 \\ -10 \\ 10 \\ 10 \end{pmatrix}$$

Computed Output Vector

Example – Scalar Code (2/4)

- Calculating C' using the transformation matrix T.

$$T = \begin{pmatrix} 1 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

Transformation matrix

$$C = \begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Computation matrix

$$C' = T * C = \begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 1 & -1 & 1 & -1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ -1 & 1 & -1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 2 \end{pmatrix}$$

Computation matrix of the code sent to the untrusted server

Example – Scalar Code (3/4)

- C' is run on the untrusted host to obtain the output vector O' and returned to the client.

$$O' = C' * I = \begin{pmatrix} -30 \\ -10 \\ 20 \\ 20 \end{pmatrix}$$

- The client calculates the inverse transformation matrix.

$$M = T^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Example – Scalar Code (4/4)

- The client applies the inverse transformation matrix to obtain the same results that would have been obtained had the code been run locally

$$O = M * O' = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} * \begin{pmatrix} -30 \\ -10 \\ 20 \\ 20 \end{pmatrix} = \begin{pmatrix} -10 \\ -10 \\ 10 \\ 10 \end{pmatrix} = C * I$$

Experiments

- Experiments were conducted to analyze the performance overhead involved.
- Benchmarks
  - C++ programs between 1,072 and 3,582 lines
    - TRACK_SEL 2.0
    - SMART_PLANNER
    - CLUSTER
- Setup
  - The default program was transferred from one workstation to another, executed and the results sent back and the time for the entire process was measured.
  - Similarly for the transformed program the total time was measured but the measured time included the time taken for transformation.

Experiments

- The overhead is the ratio:

$$\frac{(\text{Loop restructuring time} + \text{Data transformation time})}{\text{Total execution time}}$$

Benchmark	Overhead
TRACK_SEL 2.0	4.21%
SMART_PLANNER	3.88%
CLUSTER	3.93%

Conclusions

- This paper presents a method to protect certain classes of mobile applications from untrusted hosts.
- Reverse engineering is prevented through transformation of the source code.
- Measured performance overhead due to loop restructuring and data transformation were low.

Thank you!

This work is supported in part by NSF Career Award #0093082 and by a grant from the GSRC .

Embedded and Mobile Computing Center: www.cse.psu.edu/~mdl

My webpage : www.cse.psu.edu/~snarayan

PENNSTATE

1855