
**Purdue University**

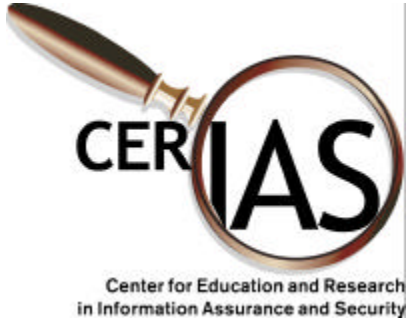
**Center for Education and Research in Information Assurance and Security**


# SECURE OUTSOURCING

*M.J. Atallah, K.N. Pantazopoulos, J.R. Rice, and E.H. Spafford*

Purdue University, W. Lafayette, IN

CERIAS Symposium, April 21, 2000




# DEFINITIONS

*C* has data and a task (a problem).

*A* has hardware and appropriate software (a server).

**Outsourcing**: *C* sends problem to *A*.
*A* computes a solution and returns answer to *C*.

**Secure Outsourcing**: *A* never knows the problem or its solution.

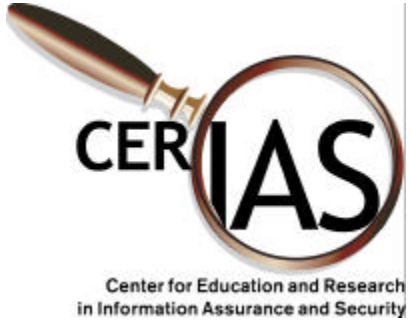
**Mutually Secure Outsourcing**: $C_1$ and $C_2$ send parts of the problem to *A* who returns the solution to both. $C_1$ and $C_2$ do not learn each others problem data, *A* still knows nothing.


# SIMPLE OUTSOURCING EXAMPLE

**Problem**: Compute $\int_1^a f(x)dx.$

- Disguise problem by choosing a random function $s(x)$ of smooth piecewise cubics.
- Send *A* the problem: Compute $\int_1^9 [f(x)+s(x)]dx$ and get *I* back.

3. Find answer $I - \int_1^9 s(x)dx.$ It is trivial to integrate $s(x)$.




# SECURE OUTSOURCING REQUIRES

- A cheap way to disguise the problem.
- A cheap way to unveil the answer.
- A disguise that is completely secure.


# A REAL EXAMPLE: Matrix Multiplication $M_1M_2$

**Step 1**: Create 3 random permutations of $1 \rightarrow N$ and 3 sets of random, non-zero numbers.

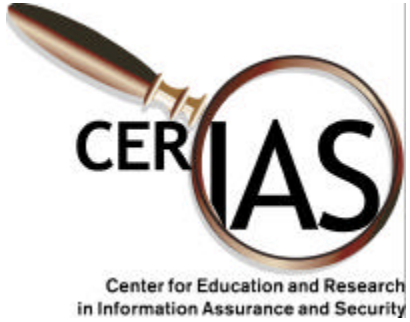
**Step 2**: Put the numbers into the non-zero places of the permutation matrices $P_1$, $P_2$, and $P_3$.

**Step 3**: Compute $X = P_1M_1P_2^{-1}$, $Y = P_2M_2P_3^{-1}$.

**Step 4**: Outsource problem $Z = XY$.

**Step 5**: Compute $P_1^{-1}\,ZP_3$ which is $M_1M_2$.

Work of Steps 1, 2, 3 and 5 is order $N^2$. Work of Step 4 is order $N^3$. $X$, $Y$ and $Z$ reveal nothing about $M_1$ and $M_2$.




# Disguise vs. Encryption

Disguise makes a functional or mathematical transformation of the entire object. It preserves distance and is really a “floating point” methodology. Encryption processes the symbols of an object and destroys distance.

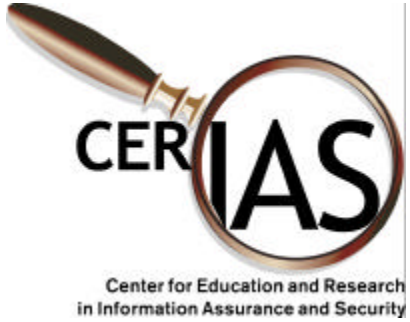
Secure Outsourcing is a form of Secure Multi-party Computing which uses disguise instead of the usual encryption.


# KEY QUESTION ONE

How general is outsourcing disguise? One can do:

- **Linear algebra**: Multiplication, Inversion, Solve $Ax = b$, Convolution, etc.
- **Sorting**.
- **Calculus**: integration, differential equations, transforms.
- **Matching**: Strings, approximate templates, images.

The work to do the disguise and unveil the answer is small compared to the problem solving. The server's work is comparable to the work without disguise.




# KEY QUESTION TWO

How secure is the outsourcing disguise? Our studies suggest the disguises can be made unbreakable to:

- Statistical attacks (try to find the random numbers, functions, mappings,..)
- Approximation theory attacks (try to discover the function behaviors).
- Symbolic code attacks (try to decipher the problem text).

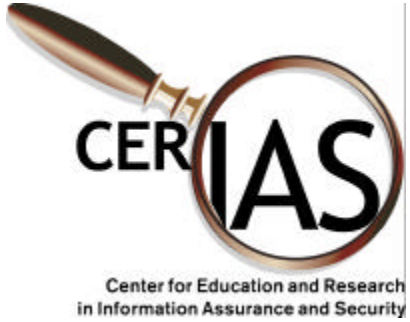

# KEY QUESTION THREE

How broad are the applications of disguise? Is it only applicable to scientific computations?

***Answer 1***: There are some substantial direct applications.

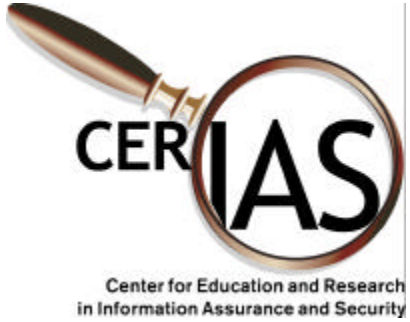
- Schlumberger seismic data analysis service.
- Pattern matching.
  - Was your son caught on the video tapes of the Purdue riots?
  - Is there an oil well being drilled in Benton county?
  - Is a variation of my program being used in MS word?
- Device design.
  - Will your new windmill really be 150% more efficient?

***Answer 2***: Mutually secure disguises have much broader applications.




# MUTUALLY SECURE OUTSOURCING APPLICATIONS

- Joint seismic exploration.
- Joint device design.
  - These are impractical without such a technology.
- Medical diagnosis (records compared to symptoms).
  - Does anyone here have AIDS? Leprosy?
  - What's wrong with my daughter?
- Biometric identification (characteristics compared to records).
  - Do your fingerprints match those on the gun?
  - Are your Mike Atallah?




# KEY QUESTION FOUR

How would this work in practice?

**Sophisticated Users**: A problem solving environment would allow the "programming" of disguises using high level, built-in operators (random functions, key management, symbolic transformations, domain transformations, one time random sequences,…)

**Ordinary Applications**: A fixed disguise program handles everything for a particular application. The only input is a key and the information. The program may be embedded in a device.