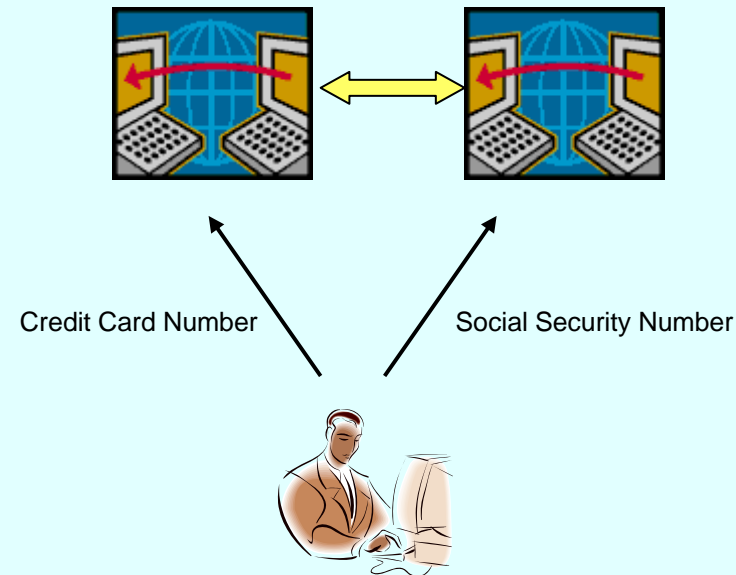


# Information-Flow Control for Location Based Services

Nishkam Ravi, Liviu Iftode  
DiscoLab, Computer Science  
Rutgers University

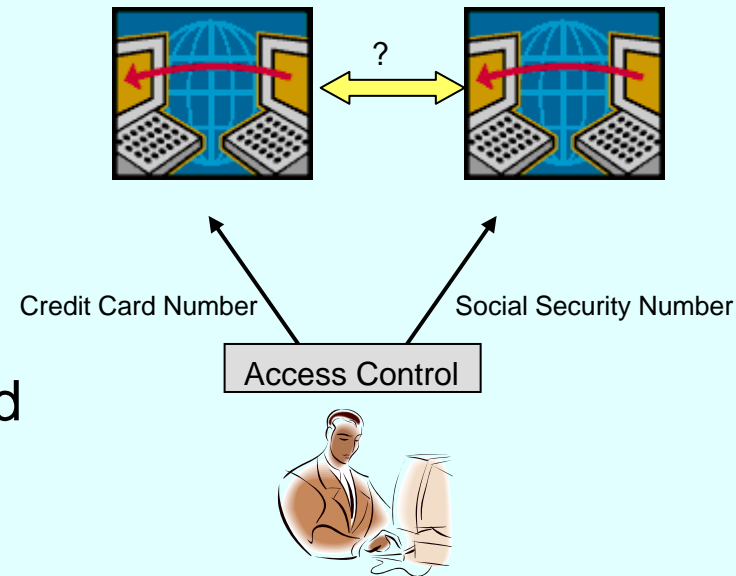
# Motivation

- Personal data commonly used in internet-based computing
  - Social security number
  - Credit card information
  - Contact information
- User concerns
  - Where is my data going?
  - How is it being used?
- Identity theft incidents prevalent
- Database community working on countering illegitimate use of private information



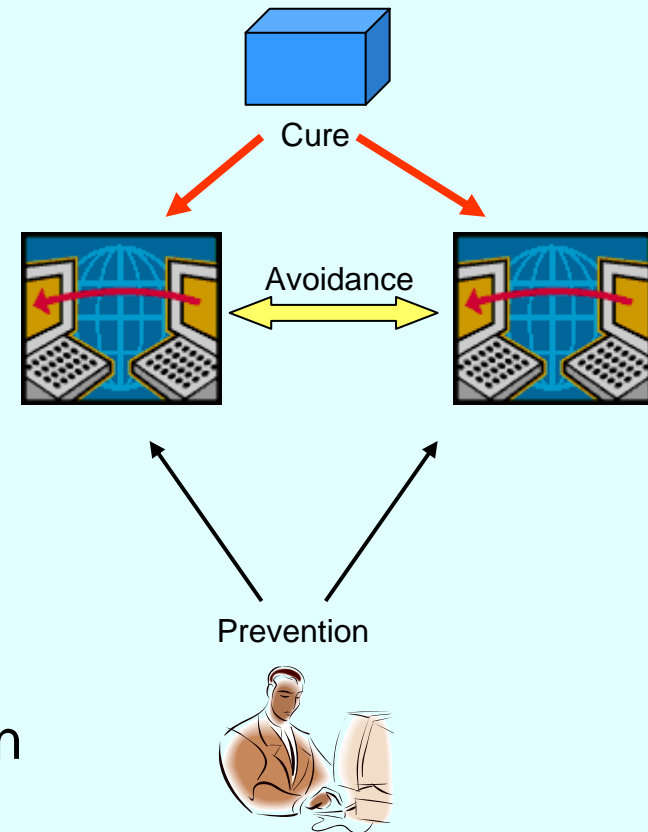
# Privacy

- Sharing sensitive information while preserving privacy is a challenging task
- Access Control is not sufficient
  - No control over data after it is read and shared
- Need to restrain flow of information



# Privacy Solutions

- **Prevention**
  - Anonymization/Pseudonymization
  - Data suppression/cloaking
- **Avoidance**
  - Information-flow control
  - End-to-end policies
- **Cure**
  - Tracking illegitimate flow of information
  - Punishing adversary



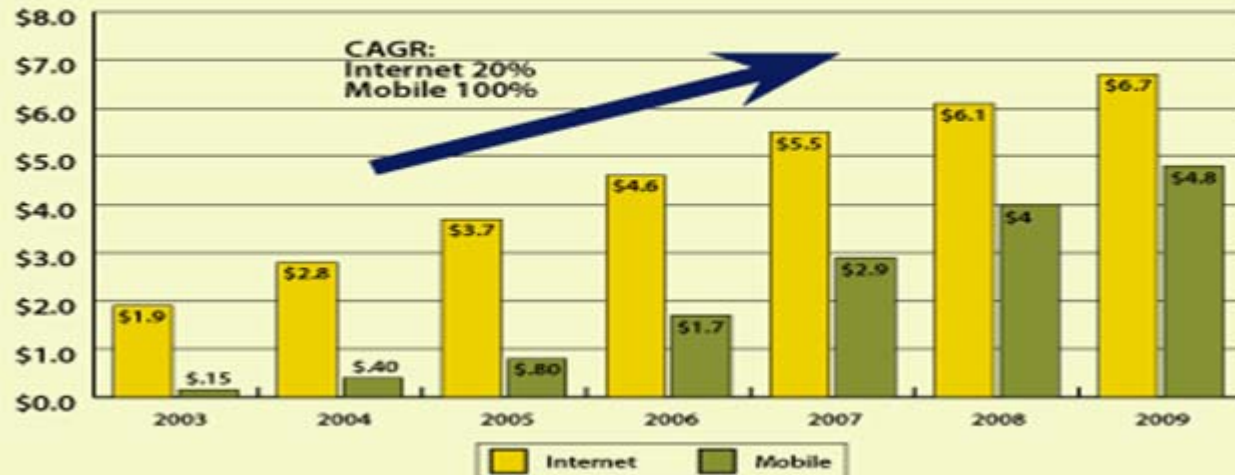
# Context-aware Computing

- Shift from “internet” to “ubiquitous” computing
- Ubiquitous computing heavily relies on user context
  - Location
  - Activity
  - Environment
- Context is dynamic in nature
  - Changes with time and space

# Market Trends

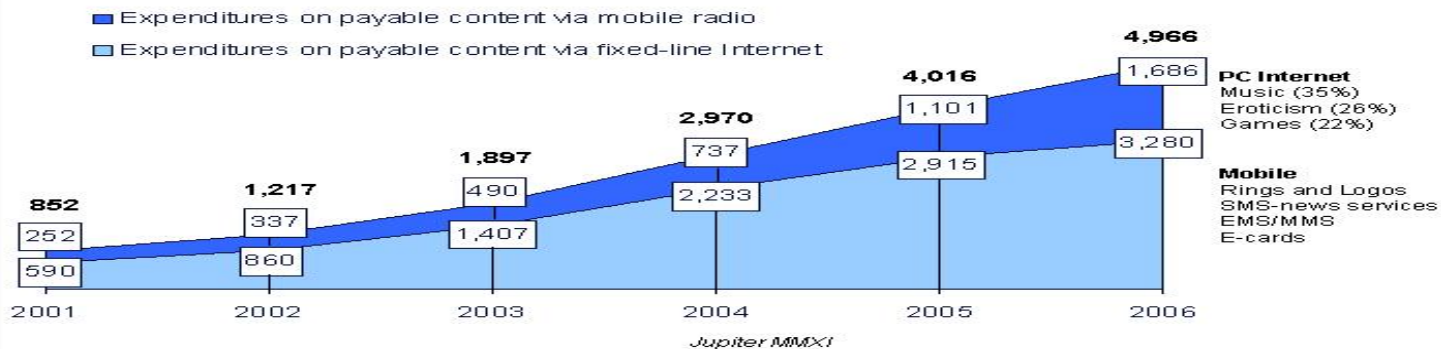
## Internet and Mobile "Micropayments" to Grow to Over \$11 Billion in US by 2009

Internet and mobile micropayments revenues, US market (US\$ in billions)



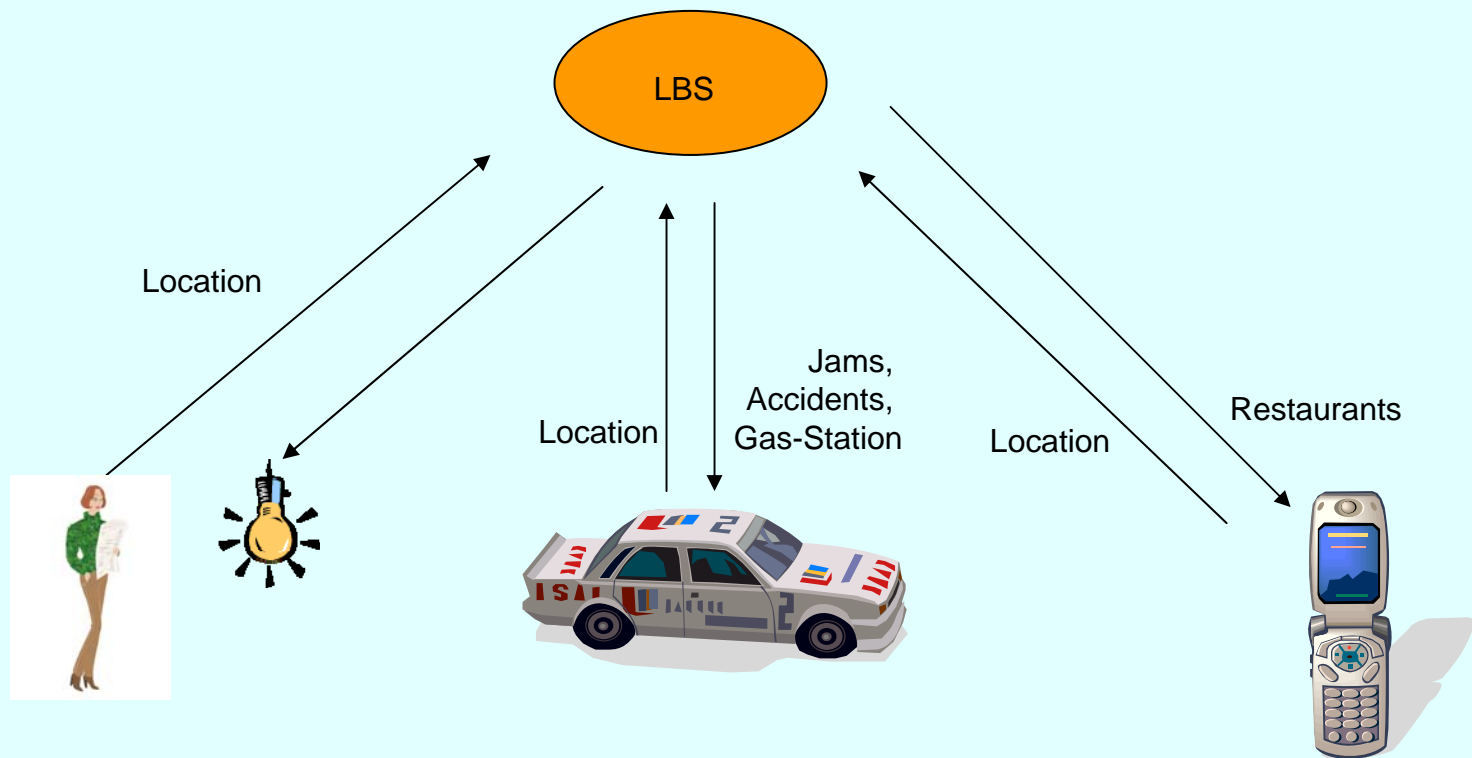
## More money is spent on mobile content than on main-line Internet content

Europe: Consumer expenses for payable content in millions of EUR, 2001-2006



# Location Based Services

- Location deemed most important context info
- Immense interest in Location-Based services (LBS)

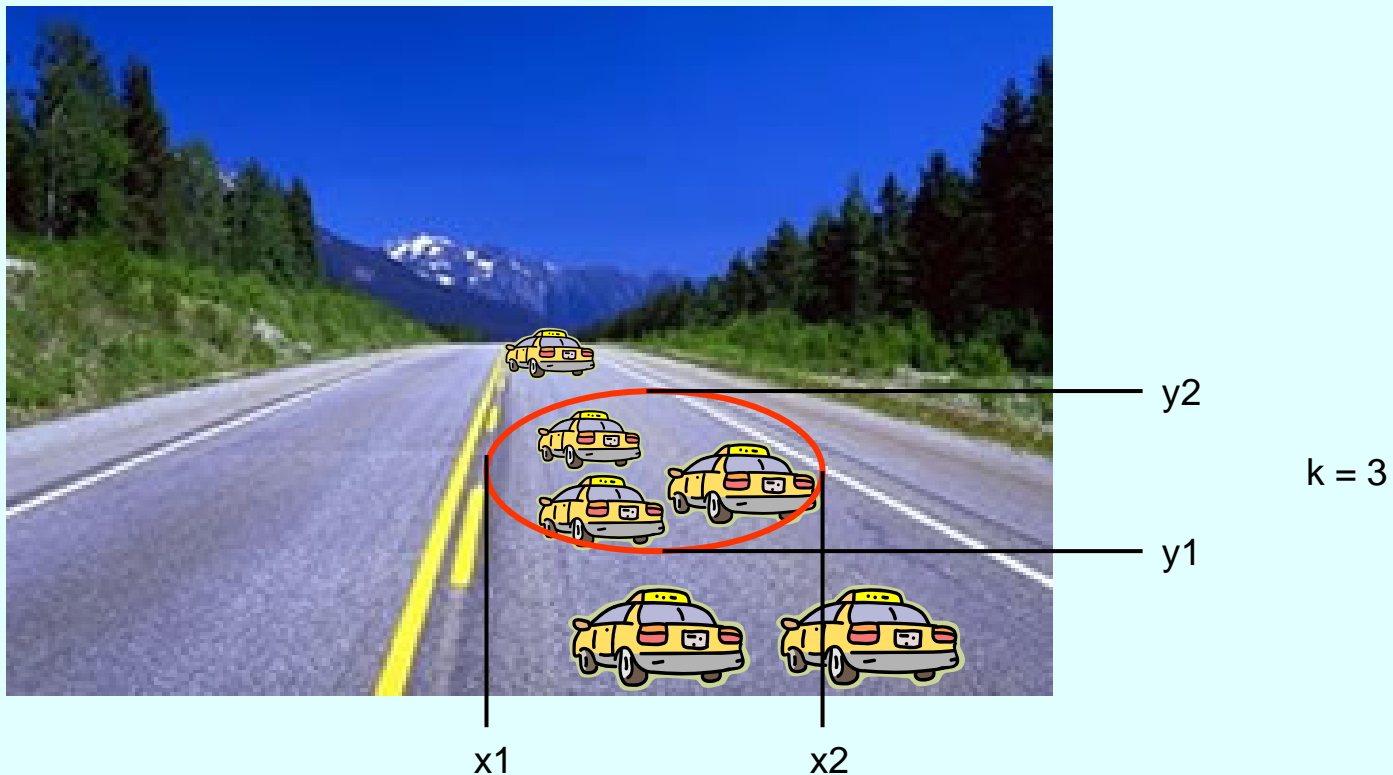


# Location Privacy

- Potential for privacy abuse
  - They know where I am!
- More serious consequences
  - Location information could aid in criminal investigations
- Recognized by US government
  - *“Location Privacy Protection Act, 2001”*
  - *“Wireless Privacy Protection Act, 2003”*

# Solutions for Location Privacy

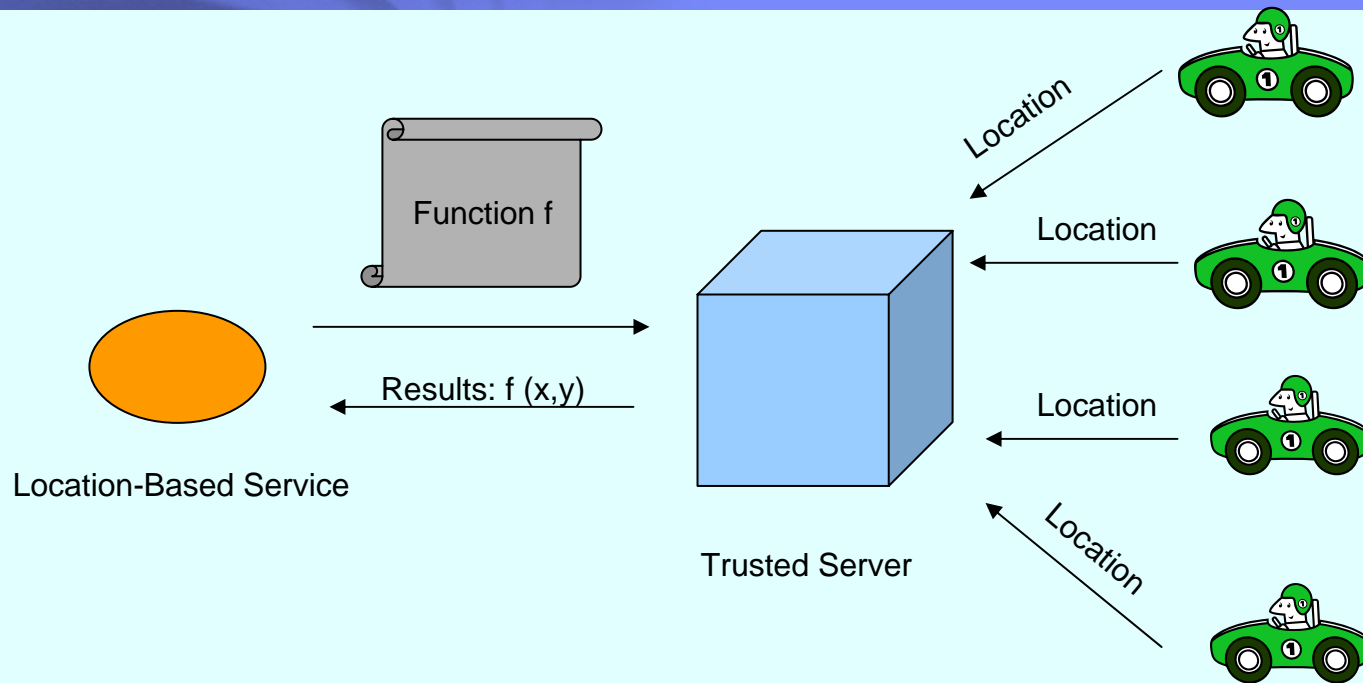
- State of the art:  $k$ -anonymization using spatial cloaking
- Instead of disclosing location, disclose an interval  
 $(x, y) \rightarrow ([x_1, x_2], [y_1, y_2])$  ( $x_1 < x < x_2, y_1 < y < y_2$ )



# How good is location cloaking?

- Cannot support applications which need precise location information
- Value of  $k$  not tailored for services
- Quality of service suffers
  - Inferior accuracy of results
- Can we have a framework + information-flow control model that preserves both location privacy and quality of service?

# Framework for service-specific location privacy



- Location of subjects maintained on a trusted server
- When an LBS needs location information, it migrates a piece of code to the trusted server
- The code executes, reads location information and returns a result
  - Distance
  - Density, Average Speed

# Example Applications

- Application of distance function
  - Geographical Routing Service
  
- Application of density, average speed
  - Traffic information service

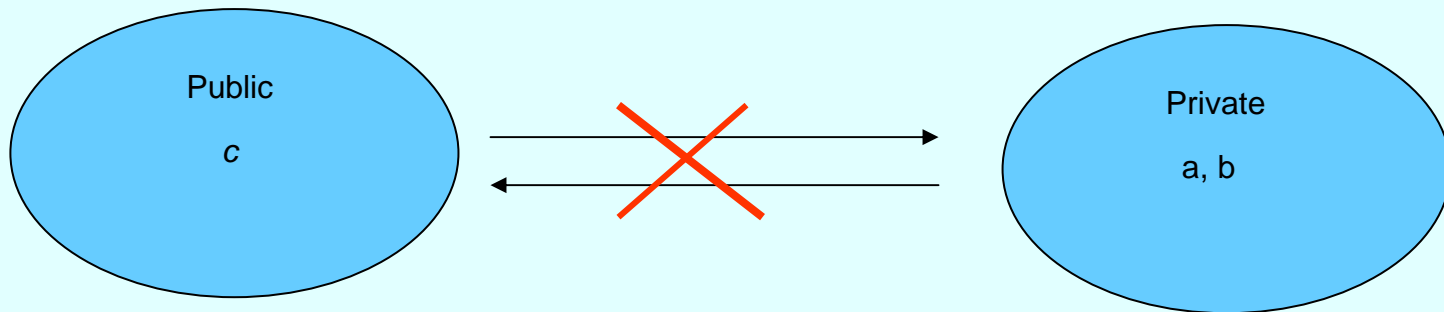
# Main Problem

- The trusted server needs to ensure that the code is location safe
  - Should not leak location information

# Information-flow Control

- Information-flow control models restrict flow of sensitive information in a program/system
- State of the art: Non-interference
  - Isolates public data from private data

```
int f(int a, int b){  
    int c = (a + b)/2; ← Isolation Broken  
    output (c);  
}
```



# Non-inference

- In many real systems data isolation is not possible, including LBS
- We propose a new model of information-flow control that
  - allows public data to be derived from private data
  - requires that the adversary does not *infer* private data from public data from a single execution of the program
- Example:

```
int f(int a, int b){  
    int c = (a + b)/2;  
    output (c);  
}
```

Value of either  $a$  or  $b$  cannot be inferred from  $c$

Non-inference satisfied

# Independent Executions

## Example:

```
int f(int a, int b, int i){  
    int c;  
    if (i > 1)  
        c = (a + b)/2;  
    else  
        c = (a * b);  
    output (c);  
}
```

Private:  $a, b$  Public:  $i, c$

$a$  and  $b$  can be derived from  $(a + b)/2$  and  $(a * b)$

$(a_1 + b_1)/2, (a_2 * b_2)$

If  $a, b$  are x-coordinates of two cars, their values would be different for the two executions

# Theoretically...

- Non-inference is undecidable in general
- Decidable for independent executions/unidirectional information flow

# Deciding Non-inference: Overview

- Derive information-flow relations for a program
- Rewrite information-flow relations as linear equations, and apply theory of solvability of linear equations
- We assume all input and output variables are scalars

# Information-flow relations: R1

```
int f(int a, int b){  
    int c = (a + b)/2;  
    output (c);  
}
```

$V = \{a, b, c\}$ ,  $E = \{(a+b)/2\}$ ,  $P = \{a, b\}$ ,  $O = \{c\}$

$R1(v, e)$ : “the value of variable  $v$  may be used in evaluating  $e$ ”

$R1(a, a+b/2) = 1$ ,  $R1(b, a+b/2) = 1$

# Information-flow relations: R2

```
int f(int a, int b){  
    int c = (a + b)/2;  
    output (c);  
}
```

$V = \{a, b, c\}$ ,  $E = \{(a+b)/2\}$ ,  $P = \{a, b\}$ ,  $O = \{c\}$

$R2(e, v)$ : “value of expression  $e$  may be used in evaluating variable  $v$ ”

$R2(a+b/2, c) = 1$ ,  $R2(a+b/2, c) = 1$

# Information-flow relations: R3

```
int f(int a, int b){  
    int c = (a + b)/2;  
    output (c);  
}
```

$V = \{a, b, c\}$ ,  $E = \{(a+b)/2\}$ ,  $P = \{a, b\}$ ,  $O = \{c\}$

$R3(v1, v2)$ : “value of variable  $v1$  may be used in evaluating variable  $v2$ ”

$R3 = R1R2 \cup A$ , where  $A$  is the set of assignments

$R3(a, c) = 1$ ,  $R3(b, c) = 1$

$M = R3(P, O) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

# Linear Equations

- A set of linear equations can be written as:

$$Ax = B$$

- Solvable if  $\text{Rank}(A) = \text{Rank}([A|B]) = N$ 
  - Where  $A$  is an  $M \times N$  matrix

- We can show that:

*A program satisfies non-inference if*

*$M_{\tau}P = 0$  and all its subsystems are not solvable*

# Linear Equations for the Example

$$M_{\tau}P = 0:$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} c \end{bmatrix}$$

$$\text{Rank}(M_{\tau}) = 1 < (|P| = 2)$$

Not solvable  $\rightarrow$  satisfies non-inference

# Example

```
int f(int x1, int y1, int x2, int y2, int k){
    int x, y, dist, avg_x, avg_y;
    int x = (x2 - x1)^2;
    int y = (y2 - y1)^2;
    dist = sqrt(x + y);
    output(dist);
    if (k > 100){
        avg_x = (x1 + x2)/2;
        avg_y = (y1 + y2)/2;
        output(avg_x);
        output(avg_y);
    }
}
```

$V = \{x1, x2, y1, y2, k, x, y, dist, avg\_x, avg\_y\}$ ,  $P = \{x1, x2, y1, y2\}$

$E = \{(x2 - x1)^2, (y2 - y1)^2, sqrt(x + y), k > 100, (x1 + x2)/2, (y1 + y2)/2\}$

# Information-flow relations

$$R_1 = V \times E = \begin{bmatrix} 101010 \\ 011001 \\ 101010 \\ 011001 \\ 000111 \\ 001000 \\ 001000 \\ 000000 \\ 000000 \\ 000000 \end{bmatrix}$$

$$R_2 = E \times V = \begin{bmatrix} 0000010100 \\ 0000001100 \\ 0000000100 \\ 0000000011 \\ 0000000010 \\ 0000000001 \end{bmatrix}$$

$$R_3 = V \times V = \begin{bmatrix} 1000010110 \\ 0100001101 \\ 0010010110 \\ 0001001100 \\ 0000100011 \\ 0000010100 \\ 0000001100 \\ 0000000100 \\ 0000000010 \\ 0000000001 \end{bmatrix}$$

$$M = P \times O = \begin{bmatrix} 110 \\ 101 \\ 110 \\ 101 \end{bmatrix}$$

# Linear Equations for the Example

$$\begin{bmatrix} 1111 \\ 1010 \\ 0101 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ x2 \\ y2 \end{bmatrix} = \begin{bmatrix} dist \\ avg\_x \\ avg\_y \end{bmatrix}$$

$$Rank(M_T) = 3 < (|P| = 4)$$

$$\begin{bmatrix} 1111 \\ 1010 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ x2 \\ y2 \end{bmatrix} = \begin{bmatrix} dist \\ avg\_x \end{bmatrix}$$

$$Rank(M_T^1) = 2 < (|P| = 4)$$

$$\begin{bmatrix} 1010 \\ 0101 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ x2 \\ y2 \end{bmatrix} = \begin{bmatrix} avg\_x \\ avg\_y \end{bmatrix}$$

$$Rank(M_T^2) = 2 < (|P| = 4)$$

$$\begin{bmatrix} 1111 \\ 0101 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ x2 \\ y2 \end{bmatrix} = \begin{bmatrix} dist \\ avg\_y \end{bmatrix}$$

$$Rank(M_T^3) = 2 < (|P| = 4)$$

$$\begin{bmatrix} 1111 \\ 1111 \\ 1111 \\ 1111 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ x2 \\ y2 \end{bmatrix} = [dist]$$

$$Rank(M_T^4) = 1 < (|P| = 4)$$

$$\begin{bmatrix} 11 \\ 11 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = [avg\_x]$$

$$Rank(M_T^5) = 1 < (|P_1| = 2)$$

$$\begin{bmatrix} 11 \\ 11 \end{bmatrix} \begin{bmatrix} y1 \\ y2 \end{bmatrix} = [avg\_y]$$

$$Rank(M_T^6) = 1 < (|P_2| = 2)$$

None of the subsystems is solvable → Satisfies non-inference

# Approach Overview

- Perform *use-def* analysis, *def-use* analysis
- Take transitive closures of *def-use* and *use-def* analysis to obtain  $R1$  and  $R2$
- $R3 = R1R2 \cup A$
- Store  $R3(P, O)$  in matrix  $M$
- Inspect solvability of  $M_T P = O$

# Implementation and Evaluation

- Implemented a static analyzer that decides non-inference for Java programs
  - Doesn't handle inter-procedural data analysis yet
  - Used Soot (API for Java bytecode analysis)
  - Used Indus (API for dataflow analysis)
- Evaluated by testing it on a benchmark
  - *Distance* (calculates distance between 2 cars)
  - *Speed* (calculates speed in a region)
  - *Density* (calculates density of cars in a region)
  - Attacks such as *Wallet*, *WalletAttack*, *PasswordChecker*, *AverageAttack*

# Case Study 1: AverageAttack

## Can there be false positives?

```
int average(int x1, int x2, ..int xn){  
    average = (x1 + x2 ..+ xn)/n;  
    output(average);  
}
```

```
int average-attack(int x1, int x2, ..int xn){  
    x1 = x3; x2 = x3; x4 = x3.....; xn = x3;  
    average = (x1 + x2 ..+ xn)/n;  
    output(average);  
}
```

$M_{TP} = 0:$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ \dots \\ xn \end{bmatrix} = \text{Average}$$

This system is solvable, rejected by our analyzer

# Case Study 2: Wallet

## Can there be false negatives?

```
int wallet(int p, int q, int c){  
  if (p > c){  
    p = p - c;           p is private : amount of money in the wallet  
    q = q + c;           q is public : amount of money spent  
  }                       c is public : cost of an item  
  output(q);  
}
```

$$M_T P = O: [1][p] = [q]$$

System is solvable, rejected by our analyzer:  
**False Negative!** (Implicit information flows)

# Case Study 3: Wallet Attack

## How bad are false negatives?

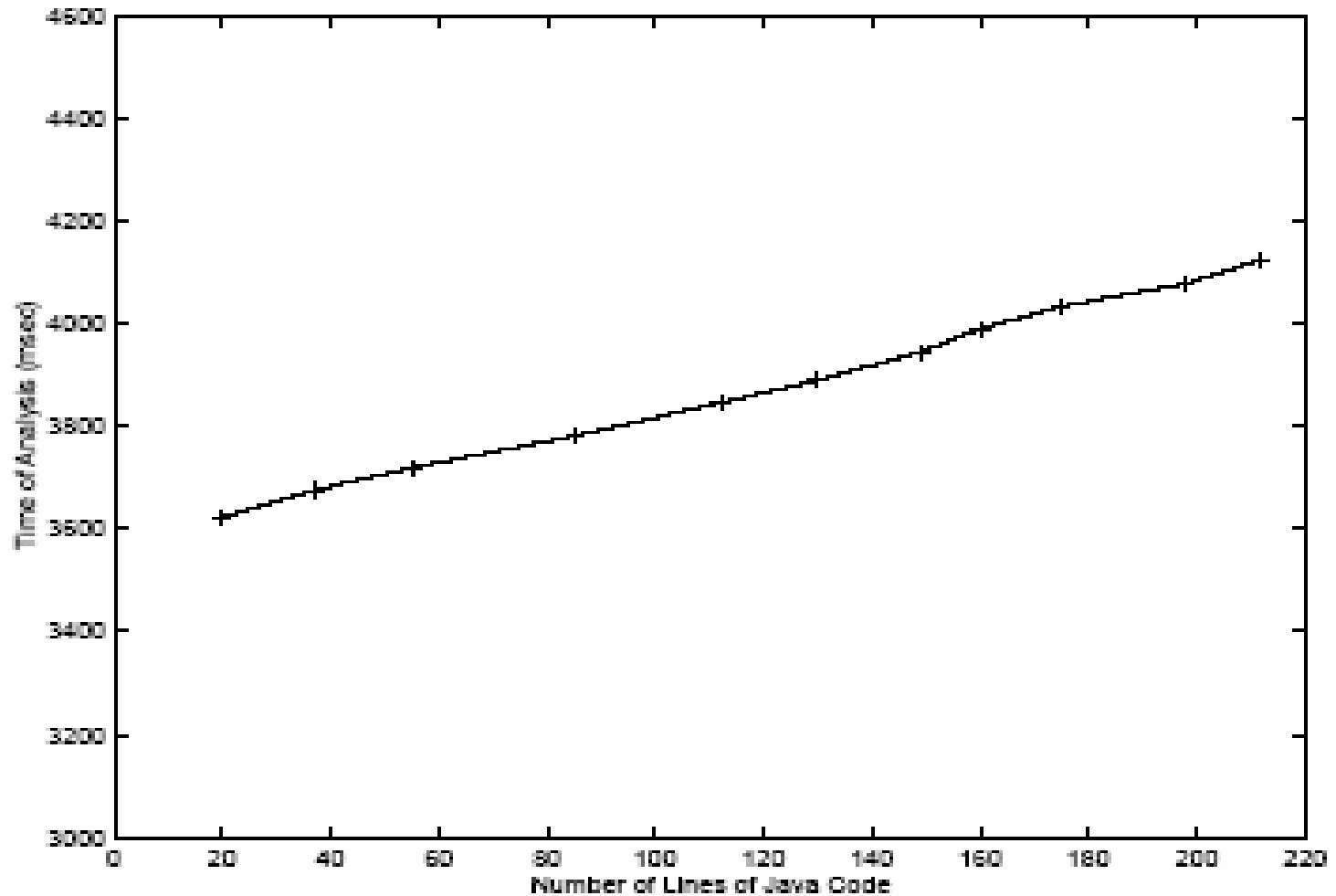
```
int wallet-attack(int p, int q, int c){
  n = length(p);
  while(n >= 0){
    c = 2^(n-1);
    if (p > c){
      p = p - c;
      q = q + c;
      n = n - 1;
    }
  }
  output(q);
}
```

Leaks value of  $p$  bit by bit

$$M_{TP} = O: [1][p] = [q]$$

System is solvable, rejected by our analyzer

# Running Time of Analysis

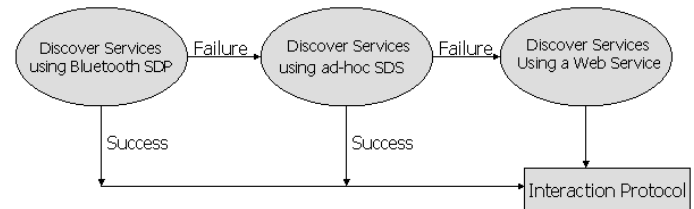
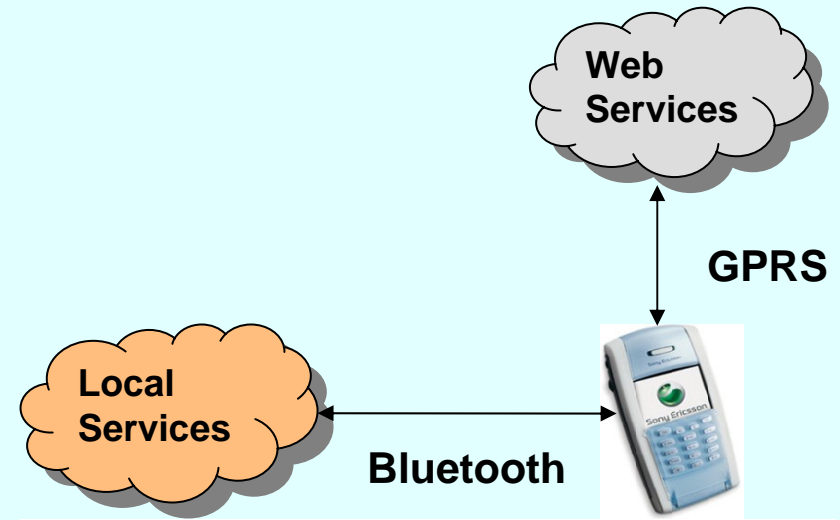


# Conclusions

- Non-inference : a novel information-flow control model
- Allows information to flow from private to public but not vice-versa
- Enforceable using static analysis for uni-directional information flow
- Applicable to location based services

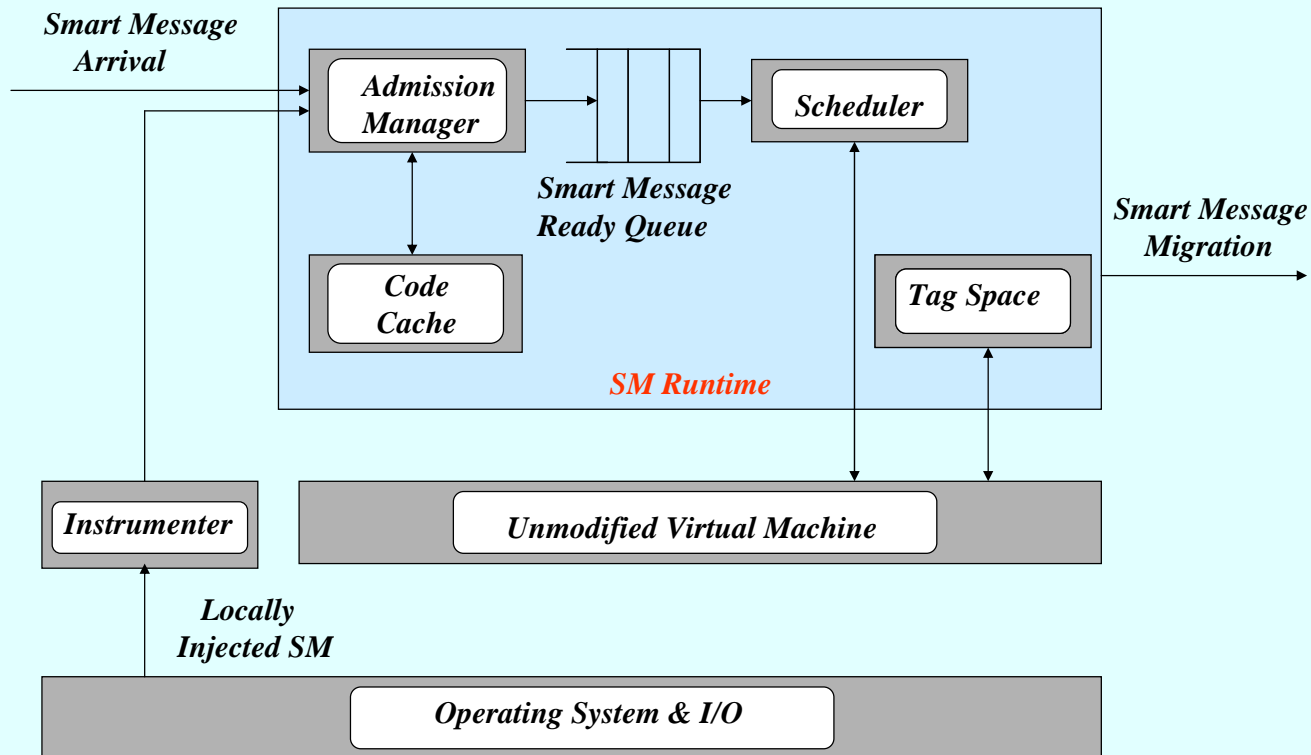
# Overview of Prior Work

- Dual connectivity on Smart Phones
- Service provisioning protocol
  - Heirarchical
  - Provides interaction capability
  - Provides payment capability
- Applications
  - Door
  - Profile Exchange
  - Bill Payment
- Indoor localization



# Portable Smart Messages

- Use execution migration, and content-based routing for programming distributed applications
- Provide store-and-forward mechanism for reliability



Thank You!

Questions?