Capabilities and Information Flow

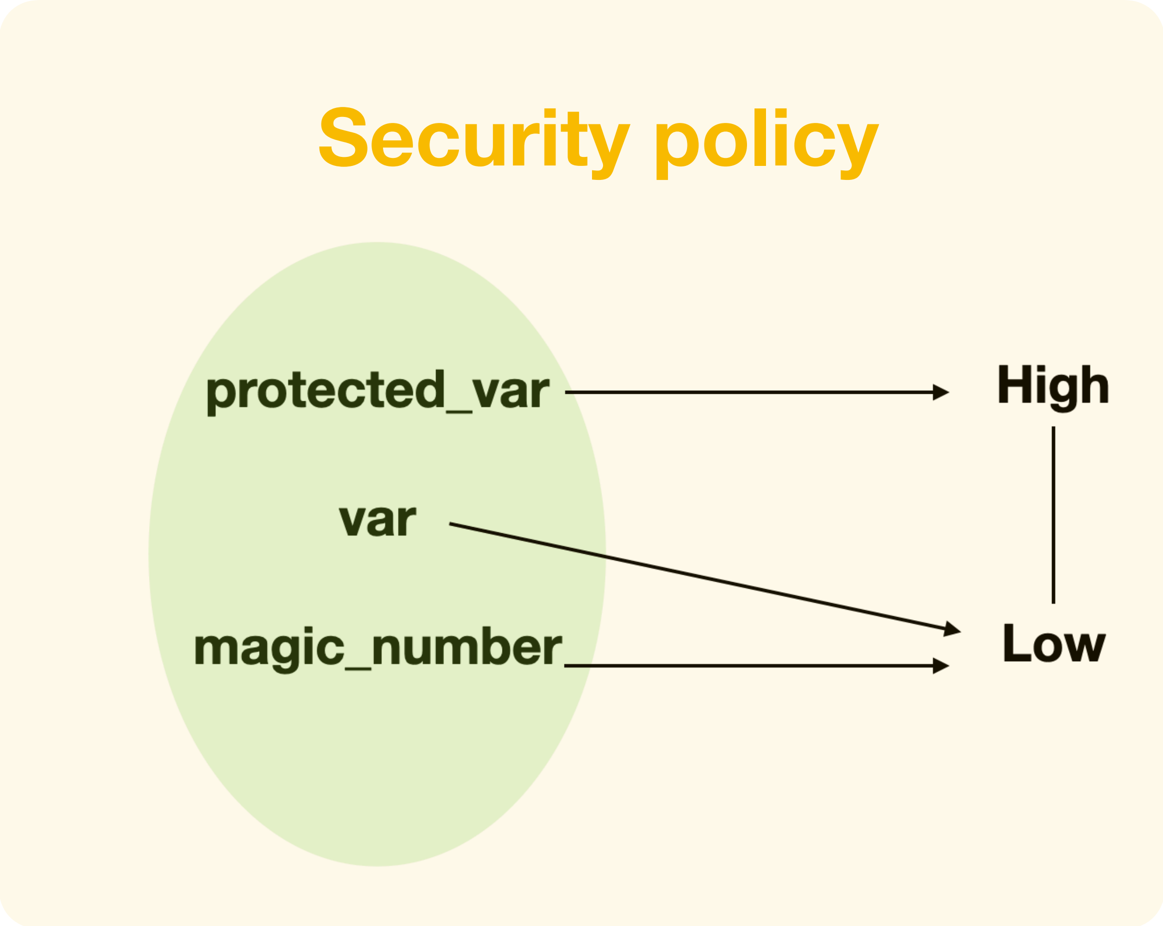
Leakage of secrets

Heartbleed bug
in
OpenSSL library

Buffer overflow
(unprotected memory)

Leakage of secrets

```
int leaking_secrets(int var){
    ...
    if(var > magic_number}{
        leak_info=2;
    }
    else if(var < protected_var){
        leak_info=1;
    }
    else
        leak_info=0;
    ...
    return leak_info; }
```



1
1 → 0
1

2
1 → 1
1

Hypertest

Fix low inputs
Vary high inputs

but

1
2 → 2
1

2
2 → 2
1

Both poor memory handling
and poor information flow control
can cause leaks of confidential information

Software requirements

LBAC: Lattice Based Access Control

Security policy + software → Noninterference **property holds**

Definition 1 (Noninterference property). *A program P satisfies the noninterference property for the High-Low security policy if for every pair of initial states s_1, s_2, $s_1 \equiv_L s_2 \Rightarrow P(s_1) \equiv_L P(s_2)$.*

input-output semantics

Low labelled data containers have the same values

General: lower level users must be unaware of the activities
of higher level users — [Goguen and Meseguer, 1982]

I/O semantics based noninterference is a hyper property of program executions (in fact a 2-hyper property)

A hyper property is a property that can only be expressed as sets of sets of executions

Properties
Execution speed bounds
Traverse loops
Update a given variable

↓

Set of executions slower than bound
Set of executions that traverse a loop
etc.

Hyper properties
Noninterference
Nondeducibility on strategies

↓

Set of low equivalent input pairs that produce low equivalent output pairs
Set of low equivalent input strategies that produce low equivalent output sequences

Hypertest for the noninterference property is a low equivalent input pair

Suppose we discover a leak (using a hypertest)

How bad is the leak?

How do we quantify it?

Turns out that exactness is expensive (many many executions) to compute (#P - effectively NP)
Yasuoka and Tarauchi 2010

But the theory is simple

and

Estimates can be useful

Uncertainty and information

- information should be additive
- information in an event should measure "reduction in uncertainty" when the event occurs
- low probability $\Rightarrow$ high reduction in uncertainty
- highest when every possible event is equally likely

- uncertainty reduction when an event $a \in A$ occurs is $log_2 \frac{1}{p(a)}$
  - $\frac{1}{p(a)}$: low probability $\Rightarrow$ high reduction in uncertainty
  - log_2: information should be additive
  - 2: base 2 produces information "bits"
- get weighted average over all events: sum uncertainty reduction for each event weighted by the probability of each event

Entropy of a set of events

$$\mathcal{H}(A) = \sum_{a \in A} p(a) \log_2 \frac{1}{p(a)}$$

More formal

- A random variable (or discrete random element in this case) is a total function $X : D \to R$. D and R are finite sets, D has a probability distribution.
- joint random variable: (X, Y) defined as $\langle X, Y \rangle$
- Entropy of a random variable X:

$$\mathcal{H}(X) = \sum_{x \in R} p(x) \log \frac{1}{p(x)}$$

- Associate random variables with expressions , particularly program variables, at program points within a program.
- Of interest are observations of values of variables at ι (the *entry point*) and the special node ω (the *exit point*).

Conditional entropy

- $P((X \upharpoonright (Y = y)) = x) = P(X = x|Y = y)$, where

$$P(X = x|Y = y) = \frac{p(x,y)}{p(y)}$$

- $$\mathcal{H}(X|Y) = \sum_{y} p(y)\mathcal{H}(X \upharpoonright (Y = y))$$

- A key property of conditional information is that $\mathcal{H}(X|Y) \leq \mathcal{H}(X)$, with equality iff X and Y are independent.

Mutual Information

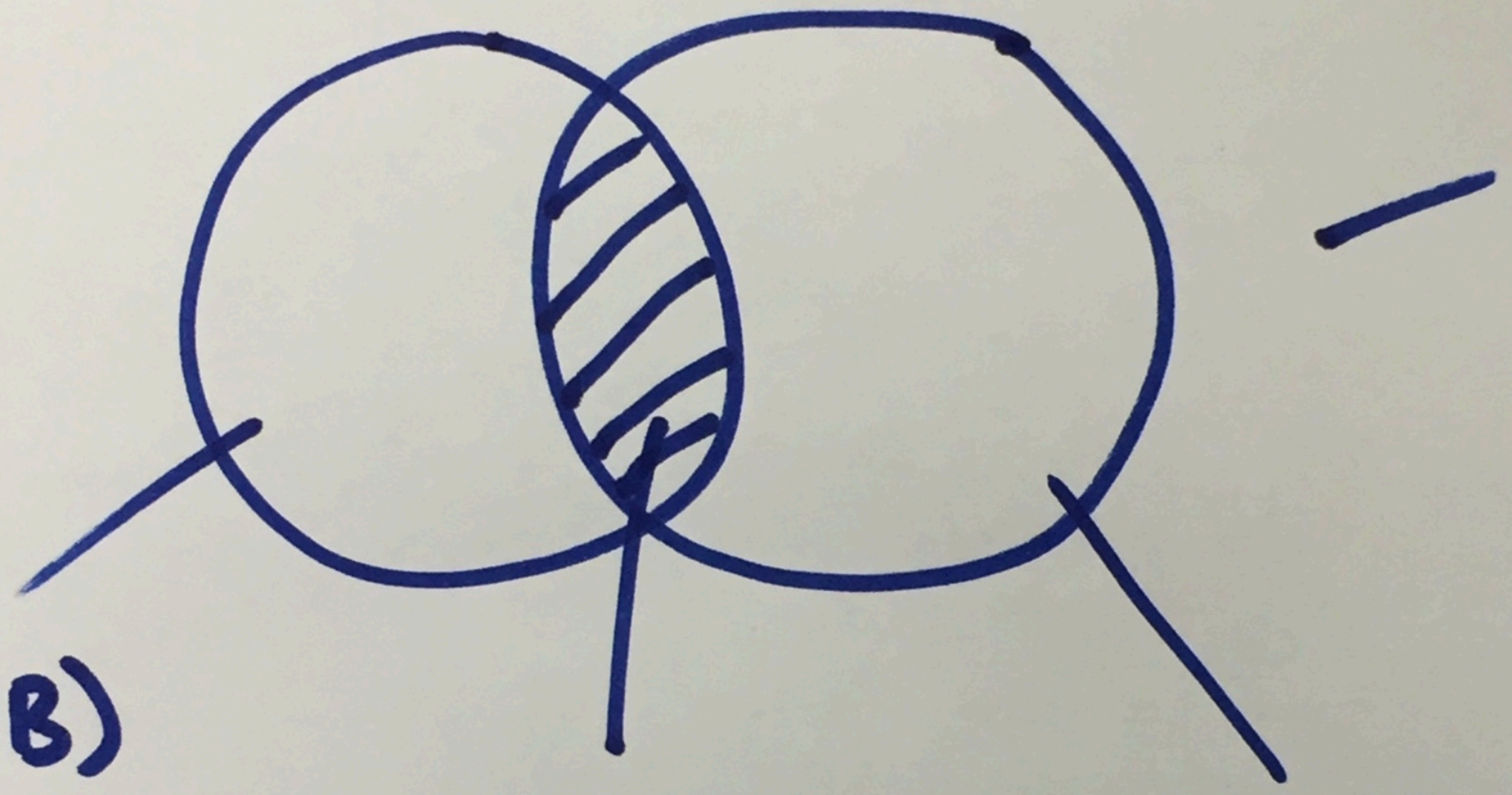
- Given two random variables X and Y, the mutual information between X and Y, written $\mathcal{I}(X;Y)$ or $\mathcal{M}(X;Y)$ is defined as follows:

$$\mathcal{I}(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

- routine manipulation of sums and logs yields:

$$\mathcal{I}(X;Y) = \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y)$$

This quantity is a direct measure of the amount of information carried by X which can be learned by observing Y (or vice versa).



$$I(A;B) = \mathcal{H}(A) - \mathcal{H}(A|B) = \mathcal{H}(B) - \mathcal{H}(B|A)$$
$$= \mathcal{H}(A) + \mathcal{H}(B) - \mathcal{H}(A,B)$$

CHAIN RULE: $\mathcal{H}(A,B) = \mathcal{H}(A) + \mathcal{H}(B|A) = \mathcal{H}(B) + \mathcal{H}(A|B)$

- As with entropy one can define conditional mutual information. The mutual information between X and Y given knowledge of Z, written $\mathcal{I}(X;Y|Z)$, may be defined

$$\mathcal{I}(X;Y|Z) = \mathcal{H}(X|Z) + \mathcal{H}(Y|Z) - \mathcal{H}(X,Y|Z)$$

- This expression is used in the most general definition of leakage.

Leakage

The amount of leakage of confidential information into variable X due to execution of the program:

$$\mathcal{L}(X) = \mathcal{H}(X^\omega | L^\iota)$$

Alternatively: information shared between final value of X and and the initial value of H, given that the initial value of L was already known:

$$\mathcal{L}'(X) = \mathcal{I}(H^\iota; X^\omega | L^\iota)$$

Why care about measuring leakage when bounding it or measuring it exactly is #P complexity?

Because

testing

And

because leakage = 0 iff policy + program satisfies noninterference

[Clark et alia 2007]

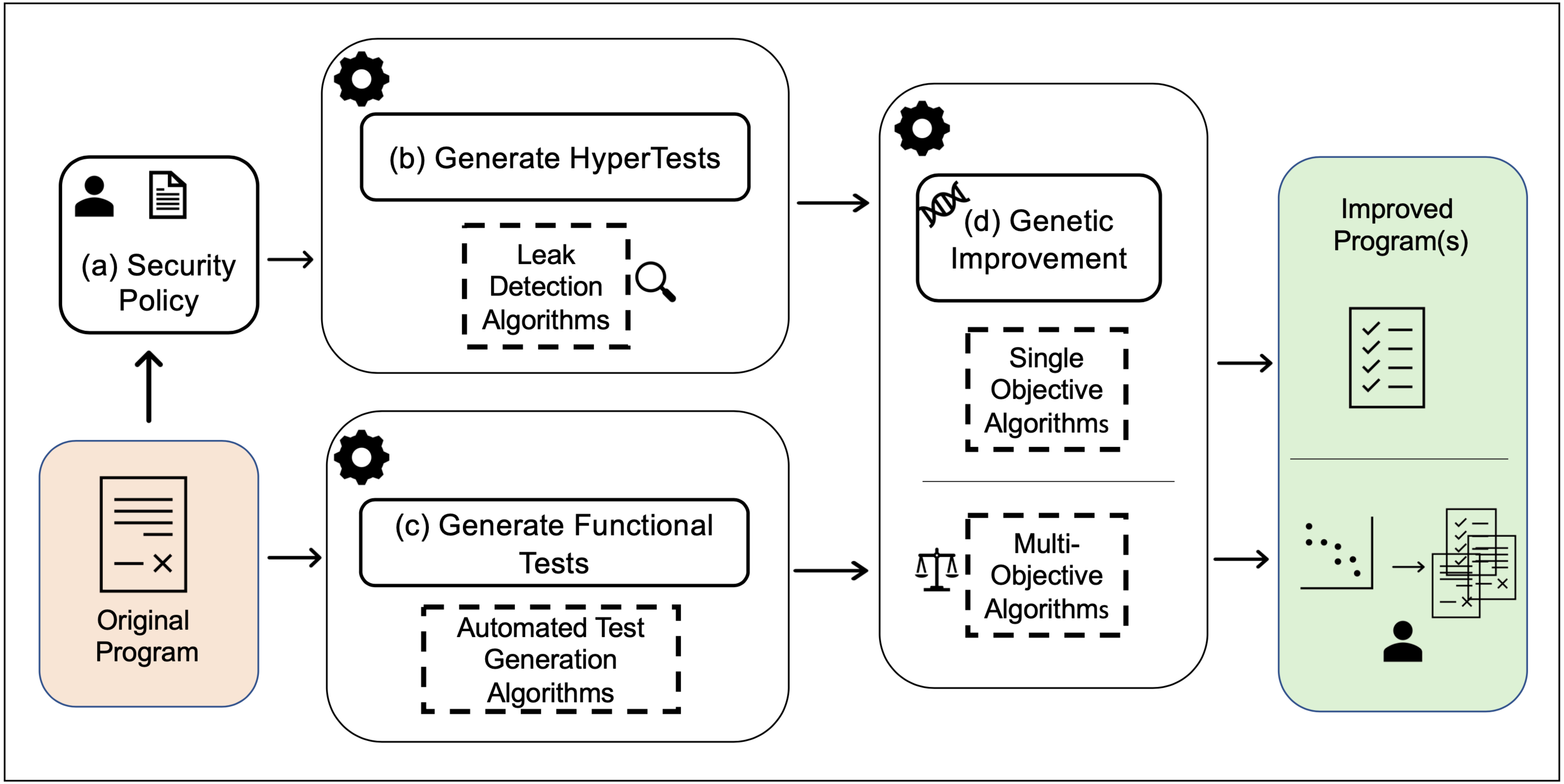
And

HyperGI

HyperGI: Automated Detection and Repair of Information Flow Leakage,
Ibrahim Mesecan, Daniel Blackwell, David Clark, Myra B Cohen, Justyna Petke
2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)

Keeping Secrets: Multi-objective Genetic Improvement for Detecting and Reducing Information Leakage
Ibrahim Mesecan, Daniel Blackwell, David Clark, Myra B Cohen, Justyna Petke
2022 37th IEEE/ACM International Conference on Automated Software Engineering (ASE)

HyperGI (leakage)



Subject	Ref	LoC	CVE-#	Security Policy High input	 Low input	 Low Output
Triangle	[32]	14	–	secret	side2 & side3	*function return value*
Atalk	[23]	33	CVE-2009-3002	*internal memory*	sock & peer	*function return value* & uaddr
Underflow	[23]	18	CVE-2007-2875	h	ppos	*function return value*
Classify	authors	18	–	High	Low	*function return value*
Heartbleed	[41]	1,082	CVE-2014-0160	*internal memory*	payload_sent & payload_length	payload_received
Bignum	[42]	778	–	*internal memory*	s, len & ret	s & *function return value*

Hypertests — can find both types of leaks manifested in these programs
abstract away from the causes of the leaks

Memory errors (e.g. Heartbleed, Bignum)
Information control flow errors (e.g. Triangle, Classify)

Currently:
Collecting CVE entries: to create a test bench that illustrates different leak causes
Developing hypertesting for fuzzers

Testing — control of all inputs
vs
Experimentation — partial control of inputs

Test set considerations

Syntax coverage criteria inadequate

(A)
Intended program:
x = (x + 2) % 4;

(B)
Actual program:
x = 3 * x;
x = x % 4;

100% path coverage

Random input: x == 3 -> (A): x == 1, (B) x == 1
Symbolic execution: x == -1999 -> (A): x == -1, (B) x == -1

Test inputs, however generated, may suffer from coincidental correctness

High information test sets

Sampling from uniform distributions, maximally dissimilar tests

Near uniform : L2 test for discrete types, Kolmogorov-Smirnov for continuous types
— Uniform distributions have maximum entropy for a given distribution support

Dissimilarity: Normalised Information Distance for strings
— based on algorithmic information

Diversifying focused testing for unit testing
HD Menéndez, G Jahangirova, F Sarro, P Tonella, D Clark
ACM Transactions on Software Engineering and Methodology, 2021

Output Sampling for Output Diversity in Automatic Unit Test Generation
H Menéndez, M Boreale, D Gorla, D Clark
IEEE Transactions on Software Engineering, 2020

Test set diameter: Quantifying the diversity of sets of test cases
R Feldt, S Poulding, D Clark, S Yoo
2016 IEEE international conference on software testing, verification and validation

Augmenting test suites effectiveness by increasing output diversity
N Alshahwan, M Harman
2012 34th International Conference on Software Engineering (ICSE)

Optimising Channel Capacity

Not possible to achieve in a testing scenario (usually)

But can optimise output diversity

$$\bigsqcup_{\sigma_I} \mathcal{I}(I; O)$$

Channel capacity

$$\bigsqcup_{\sigma_I} \mathcal{I}(H^{\iota}; L^{\omega} | L^{\iota})$$

Capacity of the leakage channel

$$\bigsqcup_{\sigma_I} \mathcal{H}(L^{\omega})$$

Worst case scenario for Deterministic code

$$\bigsqcup_{\sigma_I} \mathcal{H}(L^{\omega}) \quad \longrightarrow \quad log_2 |L^{\omega}|$$

Max possible value of H(A)

Channel capacity does not have a general, closed form expression
As a black box method it provides diversity with semantic content

Search algorithms may be employed to partially reverse the program semantics
(Currently under investigation)

Search for test set with max output diversity

Testing against decentralised policies

HyperGI and recent work on side channel leakage use global (monolithic) policies
e.g. LBAC

An alternative, and closer to the spirit of CHERI (possibly)
Is the Myers/Liskov decentralised label model (DLM)

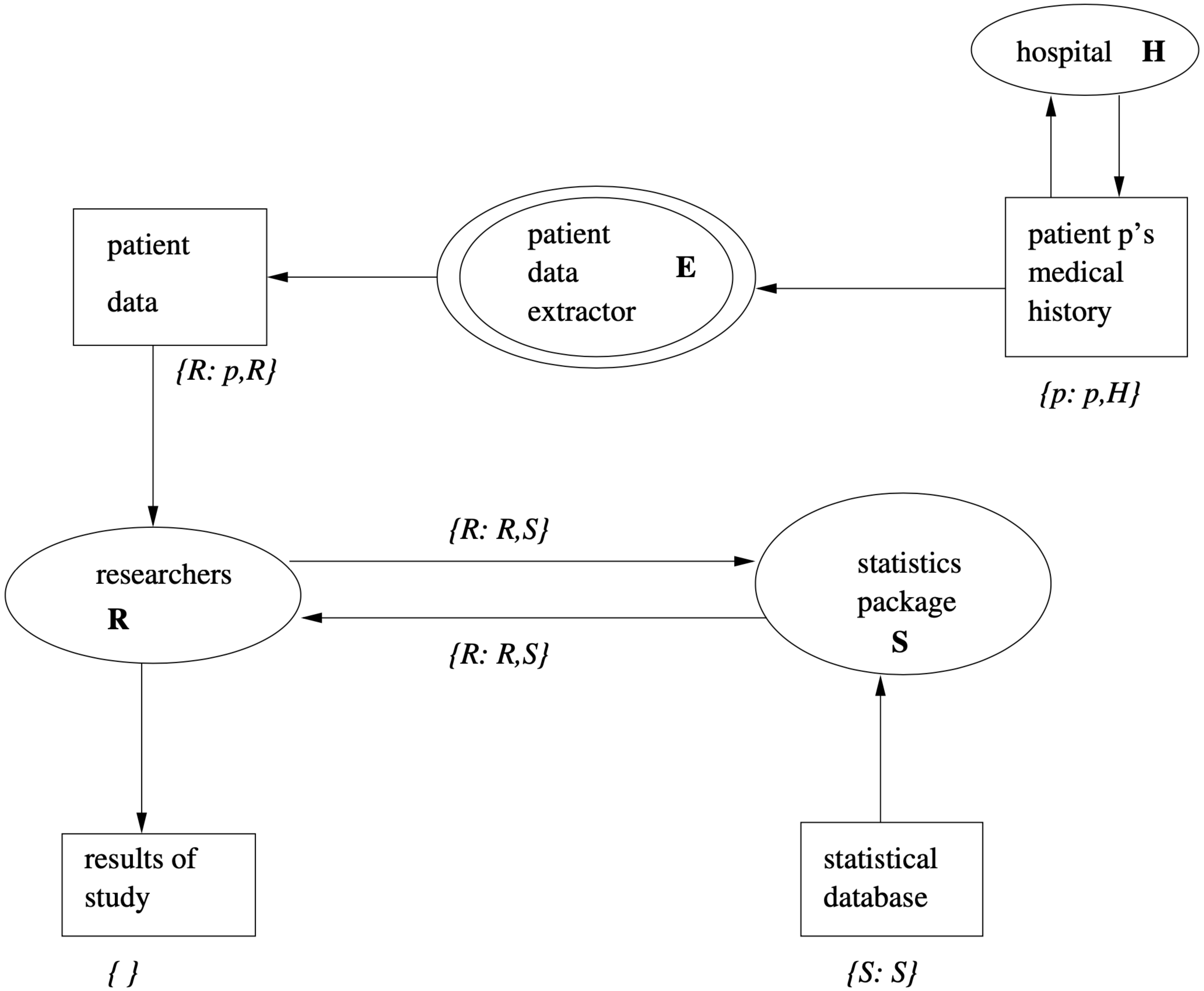
Provides security guarantees to users and groups

Uses labels that describe the allowed flow of information in the program

Allows users to declassify their own data

Jif: Java information flow
AC Myers, L Zheng, S Zdancewic, S Chong, N Nystrom
Software release. Located at http://www.cs.cornell.edu/jif 2001

A decentralized model for information flow control
AC Myers, B Liskov
ACM Symposium on Operating Systems Principles (SOSP) 1997


hospital H
patient p's medical history
{p: p,H}
patient data extractor E
patient data
{R: p,R}
researchers R
{R: R,S}
{R: R,S}
statistics package S
results of study
{ }
statistical database
{S: S}

Labels

DLM achieves global IFC through local enforcement
Chief instrument is an ordering on labels

Definition of $L_1 \sqsubseteq L_2$:

$$\begin{aligned} \mathit{owners}(L_1) &\subseteq \mathit{owners}(L_2) \\ \forall O \in \mathit{owners}(L_1), \mathit{readers}(L_1, O) &\supseteq \mathit{readers}(L_2, O) \end{aligned}$$

Can extend labels to writers

Labels for derived values

When a program combines values with different labels

Definition of $L_1 \sqcup L_2$:

$$owners(L_1 \sqcup L_2) \quad = \quad owners(L_1) \cup owners(L_2)$$

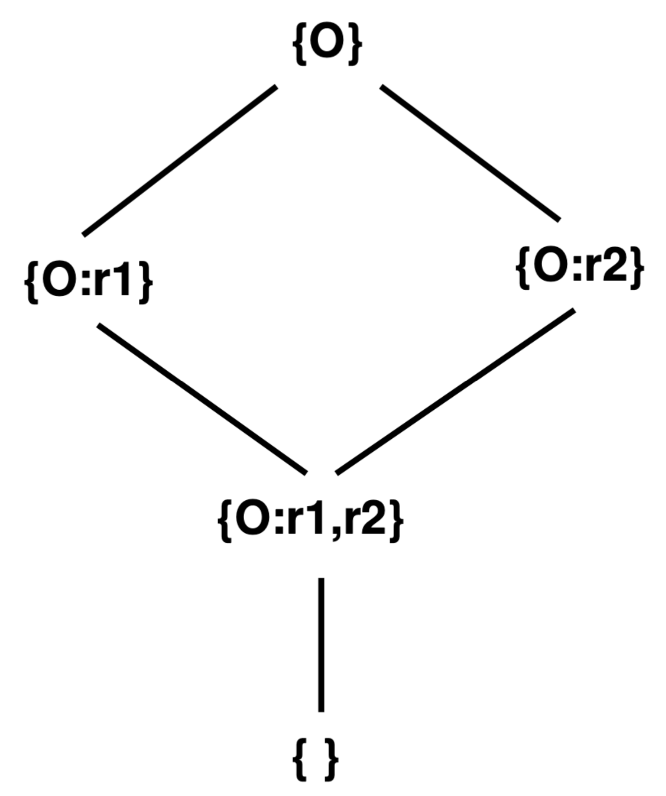
$$\begin{aligned} &\forall O \in owners(L_1 \sqcup L_2). \\ &\quad readers(L_1 \sqcup L_2, O) \quad = \quad readers(L_1, O) \cap readers(L_2, O) \end{aligned}$$

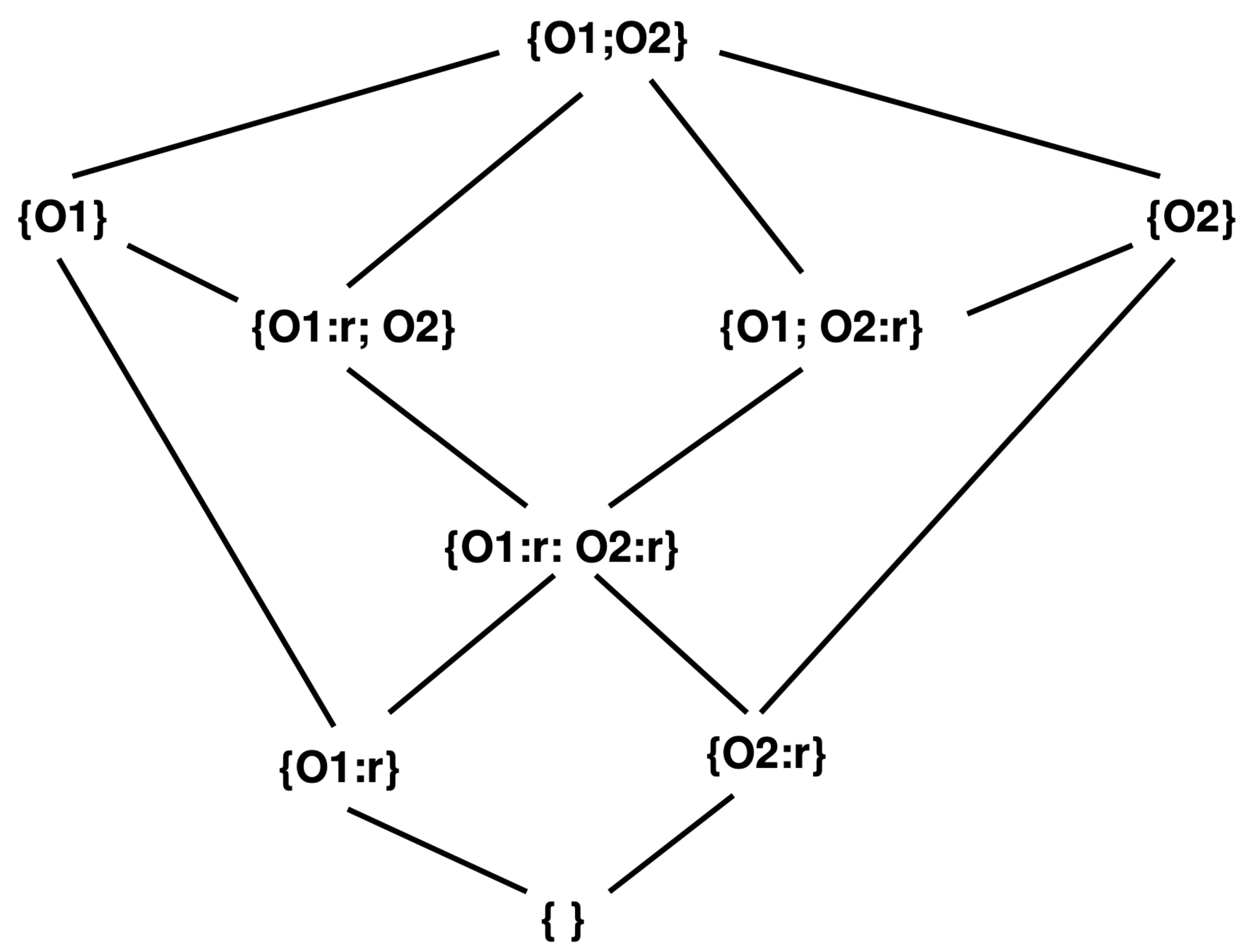
Data in a program is input/output via user labeled channels

Data containers are called slots

Values can only be written to a slot if the resulting label is a restriction
i.e. label on data lower than label on slot

Can create LBAC lattices from the user labels





Some questions

Does CHERI need IFC?

Does DLM suggest a basis for testing and repair of IFC in a CHERI context?

Can DLM provide a basis for a formal method for reasoning about correct IFC in a CHERI context?