AFRL-RI-RS-TR-2021-002

TAMBA: TESTING AND MODELING BRANDEIS ARTIFACTS

GALOIS, INC.

JANUARY 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2021-002 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
CARL R. THOMAS
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisors
Computing & Communications Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JANUARY 2021	FINAL TECHNICAL REPORT	OCT 2015 – APR 2020

4. TITLE AND SUBTITLE

TAMBA: TESTING AND MODELING BRANDEIS ARTIFACTS

5a. CONTRACT NUMBER
FA8750-16-C-0022

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

Stephen McGill, Jose Cladera, Benjamin Davis

5d. PROJECT NUMBER
BRAN

5e. TASK NUMBER
GA

5f. WORK UNIT NUMBER
SL

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Galois, Inc.
421 SW 6th Ave, Suite 300
Portland, OR 97204-1622

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

AFRL/RITA
525 Brooks Road
Rome, NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2021-002

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. DARPA DISTAR CASE # 33505
Date Cleared: 10/14/20

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report summarizes the TAMBA teams' research and engineering contributions on the DARPA BRANDEIS program. The focus has been on techniques and technologies for reasoning about privacy. We developed technologies for measuring the privacy implications and enforcing the privacy guarantees of programs and testing the privacy properties of databases. Additionally, we developed the theory for modeling the trade-offs between privacy and utitily. We worked with BRANDEIS technology users to understand and measure end users concerns regarding privacy and privacy preserving technologies. The program results have been demonstrated through software artifacts and peer reviewed publications.

15. SUBJECT TERMS

Quantitative Information Flow, Privacy, Information Leakage, Differential Privacy, SQL, Abstract Interpretation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**CARL R. THOMAS**
U	U	U	UU	77	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

1.0 Summary 1

2.0 Introduction 1

3.0 Methods, Assumptions, and Procedures 3
3.1 Quantitative Information Flow 3
3.1.1 QIF Theory Background 3
3.1.2 Deterministic Channels 4
3.1.3 Indistinguishability Relation 4
3.1.4 The Password Checker Example 5
3.1.5 Adversary Games 5
3.1.6 Adversary Beliefs and Metrics 6
3.1.7 Prior Vulnerability 6
3.2 Abstract Interpretation for QIF 8
3.3 Model-Counting for QIF 11
3.3.1 Computing Static Leakage on Deterministic Programs 11
3.4 Bayesian Inference and Privacy 12
3.4.1 Prior Model Construction 13
3.4.2 Inference 14
3.5 SQL and Privacy 14
3.6 Information-Flow Control and Liquid Types 14
3.7 Oblivious Computation 15
3.8 IoT and Virtual Sensors 16
3.9 Sparse Vector Technique 17
3.10 Economic Background 19
3.11 Human Factors and Privacy Controls 19
3.12 Mental Models for Privacy 19

4.0 Results and Discussion 20
4.1 Utilizing prob for analysis of Disaster Response Scenario 20
4.2 Decomposition-Assisted Scalability 21
4.3 Improving the scalability of QIF analysis via Sampling and Symbolic Execution 21
4.3.1 Overview 22
4.3.2 Improving precision with sampling and concolic execution 24
4.3.3 Syntax and Semantics 26
4.3.4 Computing Vulnerability: Basic procedure 27
4.3.5 Improving precision with sampling 28
4.3.6 Improving precision with concolic execution 29
4.3.7 Implementation 31
4.3.8 Experiments 32
4.3.9 Conclusions of this work 36
4.4 QIF-Supported Workflow Adaptation 36
4.4.1 HADR Aid Delivery Problem 36
4.4.2 Scheduling Workflow 38
4.4.3 Modeling Workflows 39

4.4.4 Supporting Workflow Adaptation 40
4.4.5 Posterior Vulnerability for Steps A+B 42
4.4.6 Predictive Vulnerability for Step C 42
4.4.7 Posterior Vulnerability for Steps C 42
4.4.8 Privacy-aware Workflow Adaptation 43
4.5 Programming Language Enforcement of Privacy 45
4.5.1 LWeb 45
4.5.2 A Language for Probabilistically Oblivious Computation 45
4.6 Sparse Vector Technique for DP 45
4.7 By-hand Analysis of IoT CRT's Virtual Sensors 46
4.8 Privacy and SQL analysis 47
4.8.1 The Relational Scheme Abstract Domain 48
4.8.2 Per-block Metadata 49
4.8.3 Numeric Abstract Domains in db-priv 50
4.8.4 Input and Outputs 50
4.8.5 Interactive Weight Revision 51
4.8.6 Performance and Optimizations 51
4.9 Bayesian Evaluations of DP Systems 52
4.9.1 Deep Learning Approaches to Prior Construction 53
4.10 Equilibrium in Privacy Economics 57
4.11 Differential Privacy and Privacy Budgets 57
4.12 Modern Data Reconstruction and Other Attacks 58
4.13 User-centered privacy studies 60
4.13.1 Virginia Task Force 60
4.13.2 TIPPERS 60
4.14 Mental Models for End-to-End Encrypted Communications 61
4.15 Publications 62
5.0 Conclusions 65
6.0 References 66
7.0 List of Symbols, Abbreviations, and Acronyms 70

LIST OF FIGURES

Figure 1. The (over)approximation of a polyhedron 10
Figure 2. Numeric Sparse Vector Technique (NSVT) Algorithm 18
Figure 3. Prob Encoding of an SQL Query 20
Figure 4. Scalability Improvements with Decomposition 21
Figure 5. Data Model Used in the Evacuation Scenario 23
Figure 6. Algorithm to Solve the Evacuation Problem for a Single Island 24
Figure 7. Computing Vulnerability (Max Probability) Using Abstract Interpretation 25
Figure 8. Core Language Syntax 26
Figure 9. Concolic Semantics 30
Figure 10. Experimental Results 34
Figure 11. Constraints in the Scheduling Problem 37
Figure 12. Predictive Vulnerability of Steps A+B 41
Figure 13. Posterior Belief over the Position of Ship #9 in the HADR Scenario at Step D 41
Figure 14. Predictive Vulnerability of Step C 42
Figure 15. Comparison of Initial (Prior) and Post-Task (Posterior) Vulnerability Assessments . 44
Figure 16. Occupancy Algorithm 47
Figure 17. A Conceptual Depiction of Factored Particle Filtering (FPF) 52
Figure 18. Example of the "Guessing" Probability with Factored Particle Filtering 52
Figure 19. Application of the Recurrent Autoencoder Architecture 53
Figure 20. Bottlenecking Cross-Individual Correlation via Global Features 54
Figure 21. Time Series Plot of Total Presence in Building 54
Figure 22. Scaled Similarity Matrix 55
Figure 23. Transition Matrix with Log Probabilities 56

LIST OF TABLES

Table 1. Analyzing a 3-ship Resource Allocation Run 35
Table 2. Summary of Private Data by Data Owner 38

1.0 SUMMARY

In this report we summarize our research and engineering contributions on the Brandeis program. The focus of our work has been on techniques and technologies for *reasoning* about privacy. With this goal in mind, we developed technologies for *measuring* the privacy implications of programs, for *enforcing* the privacy guarantees of programs, for *testing* the privacy properties of databases. Additionally, we developed theory for *modeling* the trade-off between privacy and utility. We worked with users to gain a better understanding of the end-user concerns regarding privacy and privacy-preserving technologies. Our results have been demonstrated through a collection of software artifacts and peer-reviewed research articles.

Our key results include:

- A technique for improving bounds on the results of box-based Quantitative Information Flow (QIF) analysis [1]
- A system for analyzing and displaying the possible locations of ships taking part in the Coalitions Scenario
- A tool for performing leakage analysis on SCALE-MAMBA programs
- A framework for the analysis of a subset of SQL that supporting the computation of several privacy-preserving metrics
- Techniques for using *Liquid Types* for privacy policy enforcement
- An implementation of the Online Multiplicative Weights algorithm for Differential Privacy
- Organized user-studies for determining how users perceive the privacy/utility trade-off
- Studies of users' mental models for properties of secure messaging apps and how to improve users' understanding

We discuss each of these accomplishments in detail in this report.

2.0 INTRODUCTION

The TAMBA effort provided new results in both the theoretical and practical aspects of privacy measurement and reasoning. This range of results includes new scalability techniques for the analysis of Quantitative Information Flow in programs, as well as work on the human-factors of privacy. Because of the variety of work done under the TAMBA umbrella, we will describe each in turn.

Privacy Measurement and Policy Enforcement

In Section 3.1 we provide background on the theory of Quantitative Information Flow (QIF), a technology that allows for the measurement of privacy leakage. QIF is one of the key theories that informed a significant portion of TAMBA's work on Brandeis. The theory of Quantitative

Information Flow can be made into a tool via several implementation strategies. Section 3.2 describes the use of Abstract Interpretation as a method of measuring Quantitative Information Flow. Another strategy for measuring Quantitative Information Flow, described in Section 3.3, is to use Model Counting. Section 4.8 describes how the TAMBA team extended some of the concepts from the prob tool to aid in the analysis of Structured Query Language (SQL), which is a commonly-used query language in deployed systems and a frequent integration point for privacy controls.

It is also possible to relate privacy and information leakage to the principles of *Bayesian Inference*. This connection is described in Section 3.4 and our research results in this space are given in Section 4.9. Finally, type systems provide another mechanism for reasoning about and enforce privacy policies and Section 3.6 describes this research area.

Privacy-Preserving Technologies

We used the tools described above to reason about the privacy properties of systems build atop a number of privacy-preserving technologies, including differential privacy and secure multi-party computation. Section 3.7 describes an important aspect of privacy-preserving computation, Oblivious Computation, which enables computations to be free from standard side-channel leaks. Section 4.11 describes background on differential privacy. In Section 4.6 we describe a novel instantiation of the Sparse Vector Technique for differential privacy that incorporates technology we developed during TAMBA.

The privacy controls above are designed to protect against various *privacy attacks*. Section 4.12 provides a survey of these attack techniques, which involve various approaches to reconstructing or de-anonymizing data.

Application Areas

One of the application areas we studied was the Internet of Things (IoT). IoT technology is increasingly found in homes, buildings, and public spaces. Section 3.8 describes one such system and Section 4.7 describes the results of our analysis of that system. Section 4.1 describes another application area we examined: the Enterprise Collaborative Research Team (CRT) Coalitions Scenario. We describe in that section our process and results for using the prob tool to analyze Quantitative Information Flow in the Enterprise CRT Coalitions Scenario. This work informed algorithm changes that preserved more privacy while maintaining the ability to find solutions to the resource allocation problem. In attempting to scale the analysis along with the increase in complexity of the Coalitions Scenario, we determined that in some instances *Model Counting* performs more efficiently. The use of model counting for the analysis of the Coalitions Scenario is described in detail in Section 4.4.

Human Factors and Economics

In addition to privacy tools and algorithms, it is important to consider the role of the human in privacy-aware systems. Users will alter their behavior based on their understanding of privacy controls, risk, and tradeoffs. It is critical to ensure that the user understands these tradeoffs and to study what affect the user's actions have on long-term performance off a privacy-enabled system. Section 4.14 describes a set of user-studies which aimed to understand users'

understanding of end-to-end encryption, an important privacy control. Section 4.13 describes the outcome of interviews with subject matter experts from some key application domains targeted by the Brandeis CRTs. This understanding helped shape scenarios drawn from those domains and highlighted the core set of user concerns to focus on when doing privacy and utility analysis. Section 4.11 describes our work on economic systems for setting privacy levels, which leverages users' internal preferences regarding risk mitigation and value accrual together with markets to set fair prices and privacy levels. Section 4.10 describes the surprising outcomes in these models when we consider user actions and the effect those have on the equilibrium points in these economic systems.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Quantitative Information Flow

Consider the function $f(x) = 5$. Observing the output of f does not reveal anything about the input, x. Early work on measuring the amount of information flow focused on the qualitative approach, one form of which is known as *non-interference* [2].

In practical systems, it may be difficult, if not impossible, to provide total isolation. In other systems, you may wish to share some small part of your private data, but you wish to ensure that your private data as a whole remains private.

A multi-party computation revealing the average of a set of positive integer inputs from various parties, is an example of a computation that violates non-interference yet also does not fully reveal the inputs. The output of this averaging computation reveals *something* about the inputs (and in some cases quite a lot — if the average is 0 then we can infer that *all* inputs were 0). QIF aims to answer the question: how much information do computations such as these actually reveal about the private inputs?

3.1.1 QIF Theory Background

Information-Theoretic Channels form the basis of the modern formulation of QIF, for this reason it is worthwhile to formalize the notion of a channel. A channel consists of a triple, (X, Y, C). X is the set of inputs to the channel and Y is the set of outputs from the channel. We say that the input X is private, and the output Y is public.

C is a $|X| \times |Y|$ matrix. Let $X = \{x_1, \dots, x_m\}$. Let $Y = \{y_1, \dots, y_n\}$. Each row of C corresponds to one possible input of the channel. Each column corresponds to one possible output of the channel. C_{ij} denotes the probability that, if the input to the channel is x_i, the output will be y_j. To put it another way, if you imagine that Y is a random variable, $C_{ij} = \Pr[Y = y_j | X = x_i]$. As a result, in a valid channel matrix, each row must sum to 1, and each cell must be at least 0.

We will often just talk about channels in terms of the matrix itself, and not explicitly write the X and Y sets. They can be inferred from the matrix size. In addition, we occasionally refer to channels as programs.

In addition, we often pun, and use X and Y both as jointly distributed random variables and as the support of those random variables: $X \sim \pi$ and $Y \sim C_{X,\_}$ (the distribution over Y's emitted from the channel).

3.1.2 Deterministic Channels

A common example of a deterministic channel is based on the game 20 questions. Suppose that the 'secret' is taken from the following set: {mango, carrot, broccoli, rhubarb}. The channel representation of the question "is it a fruit?", is as follows:

Subject $= X$	Pr[Fruit$\|X = x$]	Pr[Vegetable$\|X = x$]
Mango	1	0
Carrot	0	1
Broccoli	0	1
Rhubarb	0	1

Note that the channel matrix C, itself is just $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$, without any of the labels on input and output.

Because the channel matrix consists only of 0's and 1's, we say that the channel is *deterministic*; the output is entirely determined based on the input.

With this deterministic channel, the amount of information revealed about the secret varies with the response to the question. If the answer is "yes", then the channel has revealed the entirety of the secret. We know that the secret must be 'mango' because it is the only fruit. However, if the answer is 'no' then we have revealed much less about the secret, as it could be any of the possible vegetables, all we have learned is that the secret is definitely not 'mango'.

3.1.3 Indistinguishability Relation

Let's formalize the intuition from above.

Let C be a deterministic channel, and let $\approx_C$ denote a relation on X such that $x_i \approx_C x_j \Leftrightarrow \forall y \in Y, C_{i,y} = C_{j,y}$. That is, $x_i \approx_C x_j$ if they go to the same output when run through C. Note that the equivalence classes of $\approx_C$ are in one-to-one correspondence with the (non-empty) outputs of C. For example, the equivalence classes of $\approx_{20\text{ questions}}$ are {{Mango}, {Carrot,Broccoli,Rhubarb}}.

We call $\approx_C$ the *indistinguishability relation*, since $X_i \approx_C X_j$ means that the adversary cannot distinguish X_i from X_j in the output of the channel.

Intuitively, a program will be more secure if more inputs are indistinguishable (the equivalence classes are larger). For example, a channel which maps all outputs to the same value reveals no information in its output, since all the inputs appear identical when viewed through the channel.

3.1.4 The Password Checker Example

Another common deterministic channel is the password checker. The password checker is a model of a login page. The adversary attempts to guess the password of a single user, and then gets told whether their guess is correct or incorrect.

The channel is shown below.

Password $= X$	$\Pr[X = g]$	$\Pr[X \neq g]$
1	0	1
2	0	1
⋮	⋮	⋮
g	1	0
⋮	⋮	⋮
N	0	1

In other words, given a password guess g, we can make a channel that reveals whether the secret password matches the guessed password.

Note that our experimental model does not consider the generation of an adversary's public input. Because of this, models like the above only hold when the possible private values are drawn from a uniform distribution.

3.1.5 Adversary Games

In the study of QIF it is common to use *adversarial games* as a method for reasoning about a given scenario. These provide a way for us to be explicit about how an adversary can behave, what information it is told, and what constitutes a 'win' for that adversary. These games are played by two players: the defender who is trying to defend the secret data that they input, and the attacker/adversary who is trying to attack/guess the secret data.

The Prior Game

1. The defender, picks an input, $x \in X$

2. The adversary then attempts to guess x

3. The adversary wins if they correctly guess the secret x (otherwise, the adversary loses)

Let us consider the twenty questions example once again. One possible way to choose the secret is *uniformly*: In this case, we'd say $\pi = \text{Uniform}(x) = [.25 \quad .25 \quad .25 \quad .25]$. Another possibility is that, for whatever reason, we are less likely to choose Rhubarb. If an adversary knows this, they could then assign a lower probability to Rhubarb, and a higher probability to the other possible subjects. For example: $\pi = [.3 \quad .3 \quad .3 \quad .1]$.

Formally the prior game is described as follows [3]:

(1) $x \xleftarrow{\$} \pi$ — *The defender picks the secret x from the prior.*

(2) $w \xleftarrow{\$} \text{Adversary}(\pi)$ — *The defender gives the adversary the prior distribution π, and then the adversary (randomly) computes a guess.*

(3) Adversary wins if $w = x$

(The notation $v \xleftarrow{\$} D$ means that, at this step in the game, the variable v is randomly sampled from the distribution D.)

Note that it is incorrect to think of the prior, π as what the adversary knows about X. π is how the defender chooses $x \in X$. In this particular game, we are revealing π to the adversary. In later games, the adversary will not be told π. In these games, the adversary belief is distinct from the prior.

Alternatively, you could view π as the adversary's belief about the distribution of X, under assumption that the adversary's belief is correct.

3.1.6 Adversary Beliefs and Metrics

We are now able to look at various privacy metrics, which are described in terms of adversarial games. An important concept in these metrics is that of *Vulnerability*. Vulnerability measures the probability that the adversary will guess the secret $x \in X$ in one guess.

3.1.7 Prior Vulnerability

The prior vulnerability, which is written as $V(\pi)$, is defined as the the probability that an optimal adversary has of winning the prior game.

Two optimal strategies are:

1. Guess W sampled from $\text{Uniform}(\underset{x \in X}{\text{argmax}} \pi(x))$, or
2. Deterministically always guess some element $w \in \underset{x \in X}{\text{argmax}} \pi(x)$

Posterior Game

If prior vulnerability quantifies an adversary's knowledge prior to seeing the output of a channel, then the posterior vulnerability quantifies how much an adversary knows *after* seeing the output of the channel.

This is modeled by the posterior game [3]:

(1) $x \xleftarrow{\$} \pi$ — *The defender picks the secret x from the prior.*

(2) $y \xleftarrow{\$} C_{x,\cdot}$ — *Given the secret value, the defender runs the channel on the secret value to produce y*

(3) $w \xleftarrow{\$} \text{Adversary}(\pi, C, y)$ — *The defender gives the adversary the prior distribution π, the channel, and the public output, and then the adversary (randomly) computes a guess.*

(4) Adversary wins if $w = x$

The defender, picks an input, $x \in X$. Next, the defender then reveals to the adversary the result $y \in Y$ of running the input through the channel. Finally, the adversary then attempts to guess x. They win if they correctly guess our secret x, and they lose if they do not.

There are two scenarios that we're going to consider with the posterior game:

Static Scenario

We are attempting to predict, before the channel is run on the concrete secret, how much will be leaked about the secret. This is called *static leakage*, or *predictive leakage*, and we'll be introducing it later.

Dynamic Scenario

We have a concrete channel output that is the result of running the channel on some concrete input. We want to assess how much was leaked by a specific, concrete run of a channel. This scenario is called *dynamic leakage*.

In other words, in the static scenario, we want to determine the probability that the adversary will win the posterior game before step one. In the dynamic scenario, we want to estimate the probability that the adversary will win after step two (and having seen y, but not x).

The two most common vulnerability metrics in the literature are the *expected* and the *worst-case* vulnerabilities.

Expected/Average Posterior Vulnerability

Expected posterior vulnerability is defined as:

$\widehat{V}^{\text{avg}}([\pi, C])$ is the probability that an optimal adversary has of winning the posterior game.

Note that, for the above definition, we only know π and C. Because we are looking at the static scenario, we do not have a concrete output value y.

Worst-Case Posterior Vulnerability

The *worst-case* posterior vulnerability, is $\hat{V}^{\text{worst}}([\pi, C])$ is the worst conditional vulnerability which might be achieved by any output.

$$\begin{aligned}\hat{V}^{\text{worst}}([\pi, C]) &= \max\lceil \mathbb{V}_{X|y} \rceil \\ &= \max_{y \in Y} V(p_{X|y}) \\ &= \max_{y \in Y, x \in X} \Pr[X = x | Y = y]\end{aligned}$$

Worst-Case vs Expected Posterior Vulnerability

The key difference between worst-case and expected posterior vulnerability is that worst-case posterior vulnerability does not take into account how *unlikely* a particular input may be, only how much it might reveal about the private data.

This is most evident in the password checker example, suppose there are 2^{256} possible passwords. Then, with high probability, the channel will not reveal the secret, and leakage will be small. There is a very small probability that the channel will reveal to the adversary the entire password. Therefore, the worst-case vulnerability is 1 for the password checker, despite it being very unlikely that it would reveal the password.

The expected posterior vulnerability will weigh each outcome according to how likely it is, and will therefore report a lower vulnerability than the worst-case metric.

3.2 Abstract Interpretation for QIF

Abstract interpretation is a technique for making tractable the verification of otherwise intractable program properties [4]. As the term implies, abstraction is its main principle: instead of reasoning about potentially large sets of program states and behaviors, we abstract them and reason in terms of their abstract properties. We begin by describing the two principal aspects of abstract interpretation in general: an *abstract domain* and *abstract semantics* over that domain.

Abstract Domain:

Given a set of concrete objects C an *abstract domain* $\mathbb{A}$ is a set of corresponding abstract elements as defined by two functions:

- an *abstraction* function $\alpha: \mathbf{2}^C \to \mathbb{A}$, mapping sets of concrete elements to abstract elements, and
- a *concretization* function $\gamma: \mathbb{A} \to \mathbf{2}^C$, mapping abstract elements to sets of concrete elements.

C will be instantiated to either program states Σ or distributions $\mathbb{D}\Sigma$ over program states. In either case, we assume the standard concrete semantics of a simple imperative programming language, Imp. Because you can view a program's state as a point in a multidimensional space, we often refer specific sets or distributions of program states as *regions*.

For convenience we will consider abstract domains that can be defined as predicates over concrete states. I.e., $\gamma: a \mapsto \{c \in C: \varphi_a(c)\}$ where φ_a is a predicate parameterized by the abstract element a.

The second aspect of abstract interpretation is the *interpretation* part: an abstract semantics, written $\langle\langle stmt \rangle\rangle: \mathbb{A} \to \mathbb{A}$. We require that the abstract semantics be *sound* in that it over-approximates the concrete semantics.

Definition 1 (Sound Abstraction): Given an abstract domain $\mathbb{A}$ and its abstract semantics, the abstraction is *sound* if whenever $c \in \gamma(a)$ then $[[stmt]]c \in \gamma(\langle\langle stmt \rangle\rangle\, a)$.

Abstractions generally sacrifice some precision: the abstraction of a set of elements C can be imprecise in that $\gamma(\alpha(C))$ contains strictly more than just C and likewise that $\gamma(\langle\langle stmt \rangle\rangle\, a)$ contains strictly more elements than $\{[[stmt]]c: c \in \gamma(a)\}$. For this reason, an analysis satisfying Definition 1 is called a *may* analysis in that it contains the set of all states that may arise during program execution.

Numeric Abstractions

A large class of abstractions are designed specifically to model numeric values; in this chapter we restrict ourselves to integer-valued variables. The *interval domain* $\mathbb{I}$ represents "boxes" or non-relational bounded ranges of values for each variable X_i in a state [5]:

$$\gamma: \{(l_i, u_i)\}_i \mapsto \{\sigma \in \Sigma: l_i \leq \sigma(X_i) \leq u_i \text{ for every } i\}$$

Abstract elements here are sets of bound pairs, l_i and u_i, forming the lower and upper bound, respectively, for every variable X_i. Intervals are efficient to compute, but imprecise, in that they cannot characterize invariants among variables. More precise, but less efficient numeric domains can be used.

More generally, an abstract domain can be defined in terms of a set of predicates over states, interpreted conjunctively:

$$\gamma: \{\phi_j\}_j \mapsto \{\sigma \in \Sigma: \phi_j(\sigma) \text{ for every } j\}$$

Restrictions on the types of predicates allowed define a family of abstractions. Examples include intervals $\mathbb{I}$ already mentioned, *polyhedra* $\mathbb{P}$ where ϕ_j are restricted to linear inequalities, and *octagons* [6] where the linear inequality coefficients are further restricted to the set $\{-1,0,1\}$. Polyhedra and octagons are relational in that they allow precise representations of states that constrain variables in terms of other variables (note this is not the case for intervals). In terms of tractability, intervals are faster to compute with than octagons which are faster than polyhedra. Precision follows the reverse ordering: polyhedra are more precise than octagons which are more precise than intervals. In other words, intervals can over-approximate the set of points represented by octagons which themselves can over-approximate the set of points represented by

polyhedra. This relationship is visualized in Figure 1, which shows the (over)approximation of a polyhedron (black) using an octagon (shaded, left) and an interval (shaded, right).

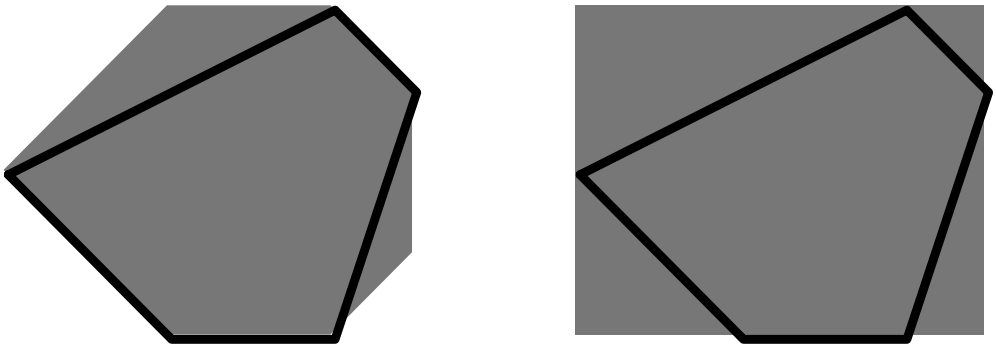


Figure 1. The (over)approximation of a polyhedron

Abstract domains implement a set of standard operations including:

- *Meet*, $a \sqcap b$ is the smallest region containing the set of states in the intersection of $\gamma(a), \gamma(b)$. For convex linear domains this operation is least expensive and is exact.
- *Join*, $a \sqcup b$ is the smallest region containing both $\gamma(a)$ and $\gamma(b)$. For linear convex domains, this is supported by the convex hull operation.
- *Transform*, $a[x \to exp]$, computes an over-approximation of the state-wise assignment $x \mapsto exp$. In the case of invertible assignments, this operation is supported by linear domains via affine transformation. Non-invertible assignments require special consideration [7].

Abstraction combinators

Abstractions can also be extended disjunctively as in the *powerset* construction [8]. For a base domain $\mathbb{A}$, the *powerset* $\mathbf{2}^{\mathbb{A}}$ domain has concretization:

$$\gamma\colon \{a_j\}_j \mapsto \{\sigma \in \Sigma\colon \sigma \in \gamma(a_j) \text{ for some } j\} = \bigcup_j \gamma(a_j)$$

That is, an abstract element in $\mathbf{2}^{\mathbb{A}}$ is itself a set of base elements from $\mathbb{A}$ and represents the set of states represented by at least one of its constituents' base elements.

Abstraction in the manner outlined can also be applied to probability distributions which serve as the concrete elements. Earlier techniques [9] attached probability constraints to standard state domains. Given a state domain $\mathbb{A}$ we form the probabilistic (upper bound) domain $\overline{\mathbb{D}}(\mathbb{A})$ that adds a probability bound on all states represented by the base domain elements:

$$\gamma\colon (a, p) \mapsto \{\mathbb{D}X \in \mathbb{D}\Sigma\colon \mathbb{D}X(\sigma) \leq p \text{ for all } \sigma \in \gamma(a)\}$$

We can combine the probabilistic upper bound construction the powerset construction to define a domain for representing more complex distributions. A more expressive variant of powerset for probabilistic abstractions imposes a sum bound (as opposed to disjunction of bounds):

$$\gamma: \{(a_j, p_j)\}_j \mapsto \left\{ \mathbb{D}X \in \mathbb{D}\Sigma : \mathbb{D}X(\sigma) \leq \sum_{j:\sigma \in \gamma(a_j)} p_j \text{ for every } \sigma \right\}$$

We emphasize in these abstractions the focus on the upper bounds of probability; such abstractions do not explicitly track lower bounds (beyond the assumed trivial 0). That is, for any probabilistic abstraction a and any state σ, there exists $\mathbb{D}X \in \gamma(a)$ such that $\mathbb{D}X(\sigma) = 0$. Because of this, these upper bound abstractions lack sound definitions of conditioning.

3.3 Model-Counting for QIF

Boolean SATisfiability problem (SAT) and Satisfiability Modulo Theories (SMT) solvers provide a method for users to solve a decision problem and determine whether or not there exists a solution which satisfies some constraints. A model counter is able to take the same set of constraints and determine the number of possible solutions which satisfy them. For the SAT decision problem, the analogue model counting problem is #SAT (pronounced "sharp-SAT"), and for SMT it is #SMT.

3.3.1 Computing Static Leakage on Deterministic Programs

Recall that, the min-capacity of a deterministic channel is the base-2 log of its number of (feasible) outputs [3]. This result allows us to use model counters as a method to computer the min-capacity. Importantly, because many existing program verification techniques require the conversion of source code to a formal model (that can be passed to an SMT solver or similar), #SAT approaches to computing min-capacity can leverage this existing technology to operate on actual programs.

Biondi et. al. [10] describe the general approach:

1. Convert a program to a SAT formula
2. (Approximately) count the number of satisfying assignments of the output variables[1]
3. Return the base-2 log of this count.

Biondi et. al. [10] use the CBMC model checker to convert C code to a formula, and then they use ApproxMC2 model counter to count the number of outputs.

Backes et. al. [11] use a different approach. They iteratively refine an indistinguishability relation until it matches the program they want to analyze. Starting with a partition which contains all outputs in a single bin, they use a SAT solver to prove or disprove whether it is possible to distinguish between two inputs in the same bin. If it is disproven, then the algorithm terminates. Otherwise, they split a bin in the partition and then step again.

[1] Because both input and output variables will be constrained by the formula, we need to be careful to only count the number of outputs.

3.4 Bayesian Inference and Privacy

In the course of the latter parts of the TAMBA effort, the Charles River Analytics (CRA) team has focused on developing empirical evaluations of privacy systems based on Bayesian probability theory. We've focused in particular on methods to attack and utilize differential privacy (DP) systems via Bayesian inference. Differential privacy systems provide a precise mathematical guarantee – the practical significance of which is unclear (and sometimes overstated):

$$\Pr(A(D_1) \vee D_1) \leq e^{\epsilon} \Pr(A(D_2) \vee D_2) \forall D_2 \in \mathrm{Adj}(D_1)$$

That is, the conditional probability of some ϵ-differentially private algorithm A output given an input dataset D_1 is bounded by a factor of e^{ϵ} from all other datasets $\mathrm{Adj}(D_1)$ adjacent to D_1. Adjacency can be freely defined by the system developer. Often it is defined as single perturbations to a dataset – the addition or removal of a single individual from a database, for instance.

This definition alone does not define pragmatic privacy guarantees, however, as one is really interested in the probability of the underlying data *given the output of the DP algorithm*. Applying Bayes rule:

$$\Pr\big(D_1 \vee A(D_1)\big) \leq e^{\epsilon} \Pr\big(D_2 \vee A(D_2)\big) \frac{\Pr(D_1)}{\Pr(D_2)} \forall D_2 \in \mathrm{Adj}(D_1)$$

The posterior probability of a given dataset is thus additionally related to the relative prior probabilities of adjacent datasets. When this ratio of prior probabilities is far from 1 there can be significant venue for adversarial reconstruction – potentially arbitrarily far from the promises of differential privacy. Common circumstances can induce this – for instance, we spent significant analysis time on a person localization dataset constructed by the University of California, Irvine (UCI) IoT CRT team. If one defined adjacency as the addition or removal of an individual from a single detection in a building on 5-minute increments, this would contradict most prior models. A dataset in which an individual sits in their office for 30 minutes, disappears for 5 minutes, and then reappears in their office for 30 minutes would be highly unlikely according to reasonable prior models. Thus $\frac{\Pr(D_1)}{\Pr(D_2)}$ could be very large – potentially as high as 100 or 1000 (or 1e-2 / 1e-3, equivalently).

One of our major goals in the Brandeis program was also to provide an effective utility-based analysis of DP systems. This, likewise, benefits greatly from strong inference capabilities. In general, a user might be interested in the posterior distribution of some function – for instance, one might want to know the distribution of possible individuals in a room (as a direct application of the DP system). Likewise, for evaluating a DP system, one might want to evaluate the accuracy of localizing individuals and demonstrate that it is not possible with high accuracy. Both of these cases are answered correctly and completely by posterior evaluation of the distribution of interest given DP output – that is:

$$\Pr\big(M|A(D_1)\big) = \int dD_1 \Pr(M \vee D_1) \Pr\big(D_1|A(D_1)\big)$$

One might also be interested in amortizing over the actual execution of the DP system – to understand, for instance, the typical range of various metrics. This is also naturally stated as Bayesian inference:

$$\Pr(M|D_1) = \int dD_1' \Pr(M|D_1') \int dA(D_1) \Pr\big(D_1'|A(D_1)\big) \Pr(A(D_1) \vee D_1)$$

In this case the formula describes typical inferred values for metric M given a ground-truth dataset D_1.

3.4.1 Prior Model Construction

There are two challenges associated with casting DP evaluation as a Bayesian inference problem. First, one must construct a prior model $p(D)$ describing what datasets are "reasonable". There are a number of approaches to this, a number of which were explored under this effort. A popular technique would be to employ one of a number of classical approaches to distribution construction that have nice mathematical properties. Our first explorations involved the construction of Markov models describing person movement. These models made two assumptions that are highly dubious, however, and inhibit realistic analysis. First, in the UCI dataset context, we assumed individuals moved independently. Second, we assumed that the most recent location of an individual fully described the system "state" – that is, the prediction of future movement could not be made more accurate by also accounting for older location information. These models could be refined from data, fortunately, and the single room-to-room transition matrix that defines that Markov model was estimated from the UCI dataset.

A second approach we explored was the application of deep neural networks (DNNs) to learn hidden Markov models describing the datasets. These learned HMMs could capture complex temporal and cross-individual information from available data, though they presented some challenges. Training these models is computationally onerous and practically difficult due to strong biases in the data. Furthermore, evaluating these models can be highly ambiguous – serious thought about cross-validation must be employed to determine if metric inference is due to a sophisticated "general" prior regarding person movement or if instead one has learned a model that memorized certain attributes. It can be difficult to ascertain if the model has memorized a class schedule or a given individual's office location, for instance. This is a critical point because one would like to employ these priors to evaluate DP algorithms under reasonable adversary models – priors leveraging strong machine learning can capture some of the optimal structure a domain expert might be capable of articulating, but they also might be unreasonably strong due to their application of data an adversary would not have access to.

The third, most general approach we explored was the construction of prior distributions based on probabilistic programs. Probabilistic programming provides a flexible architecture for constructing probability distributions using programming language constructs – for instance, one can use object-oriented approaches to describe entities such as *Person*, *Meeting*, or *Office* that capture the semantics of the domain of interest. These entities can then be related via a variety of probabilistic models, varying from simple models like the Markov models described above to complex models like those described by DNNs. By embedding the domain semantics in the model, it is possible to both capture the domain expertise in a manner an adversary might be

capable of as well as learn parameters related to the semantics of the domain rather than a black-box relation – this greatly reduces training data requirements and enhances generalization.

3.4.2 Inference

Given a prior model, the next challenge is performing probabilistic inference – that is, calculating Bayes rule variable posteriors given observations from a DP system. General inference is known to be intractable – in essence, one must calculate a number of integrals over a high dimensional space. There exist a number of feasible approaches for special cases, but only approaches under development now are capable of providing useful results for the more general problem settings we are interested in, e.g. for probabilistic program models. In the later sections we will describe our approaches to inference. Further advancement in the field of tractable inference on more complex models will make these analyses more feasible in the future.

3.5 SQL and Privacy

There is a significant amount of work on SQL implementations and tools that enable privacy-preserving database usage. Differential Privacy provides a theoretical framework for the study and development of privacy preserving mechanisms in query-based data analysis (such as SQL). Work such as McSherry's Privacy Integrated Queries (PINQ) [12] provides programmers with the advantages of differential privacy in familiar contexts. However, differential privacy does not provide a framework for discussing the *amount* of information that is revealed in a set of queries. This sort of reasoning requires an *analysis* of the SQL queries in question, and would be useful even in the case where no differentially private mechanism is used or desired in a system.

Work on the analysis of SQL queries has focused on more conventional concerns of database users, analysts, and administrators: performance, change impact prediction, type errors [13]. Many researchers have also investigated the use of static analysis of SQL for detecting security issues [14], [15].

3.6 Information-Flow Control and Liquid Types

There is a significant amount of work in Information-Flow Control (IFC) and type systems that enforce information-flow policies [16]. This work differs from QIF in that IFC analyses do not quantify *how much* information regarding the secret data has flowed (or leaked) to the output, but whether *any* information flow has occurred. This reduction in ambiguity allows for more tractable analysis and enforcement, at the cost of being able to quantify the risk of running a particular program.

Information-flow policies often deal with various forms of *declassification* along four axes: who, what, where, and when. *Who* determines which actors are able to receive declassified information. *What* determines what data is able to be declassified. *Where* determines what subsets of data can be declassified. *When* is used for when secret data is only sensitive for a certain period of time.

Static type systems are an attractive avenue for enforcing IFC policies for a few reasons:

- Runtime/dynamic analyses cannot observe all possible paths of the computation, allowing for the program to correctly enforce the policy in some instances and not in other. This means that any failure at runtime due to policy enforcement reveals the fact that secret information was going to flow to the output.

- External static analyses require the programmer to be judicious in the use and application of the static analysis tool itself. There is no guarantee that the results of any static analysis will be incorporated into the program in question.

- Type systems enforce the policies intrinsically: the program cannot be run or compiled if the program does not pass the static type-checker. This also gives confidence to the developer by guaranteeing that a compiled program is safe with regards to the information-flow policies.

One solution to these problems is embodied in the Labeled IO (LIO) system [17] for Haskell. LIO is a drop-in replacement for the Haskell IO monad, extending IO with an internal current label and clearance label. Such labels are lattice ordered (as is typical [18]), with the degenerate case being a secret (high) label and public (low) one. LIO's current label constitutes the least upper bound of the security labels of all values read during the current computation. Effectful operations such as reading/writing from stable storage, or communicating with other processes, are checked against the current label. If the operation's security label (e.g., that on a channel being written to) is lower than the current label, then the operation is rejected as potentially insecure. The clearance serves as an upper bound that the current label may never cross, even prior to performing any I/O, so as to reduce the chance of side channels. Haskell's clear, type-enforced separation of pure computation from effects makes LIO easy to implement soundly and efficiently, compared to other dynamic enforcement mechanisms.

3.7 Oblivious Computation

An 'oblivious computation' is any computation that is free from information leaks which are due to observable differences in the *timing* of the computation or in the *memory-access patterns* of the computation. Oblivious programming languages offer mechanisms for ensuring that all programs written in the language are oblivious by construction. For example, in a traditional programming language the if construct is a source of potential side-channels, consider the following:

```
if (p) {
  <stmts1>
} else {
  <stmts2>
}
```

In the instance where stmts1 and stmts2 are not identical (the common case, as otherwise the conditional branching is unnecessary), it may be possible for a third-party to determine p's value based purely on external observations, such as knowing that stmts2 is a significantly more expensive computation. If p is meant to be secret, this is a significant issue, rendering the computation *not* oblivious. One method of making the above oblivious is by ensuring that regardless of p's value, *both* branches are executed, i.e. all of stmts1 and stmts2 are processed,

but only the results from the semantically necessary branch are used. This ensures that no side-channel information is leaked regarding the value of p. However, this also means that the programming environment must manage side-effects carefully. Consider the following:

```
...

if (p) {
  a = x + y;
} else {
  a = x + z;
}

return a;
```

Attempting to execute the above in an oblivious manner raises the question "what should be returned as a's value?" This can be addressed via several means, having effects that can be 'rolled back', or forbidding side-effects such as assignment or mutation.

3.8 IoT and Virtual Sensors

The increasing use and presence of Internet of Things (IoT) devices has led to privacy concerns for both the consumer of the IoT devices and for people in public spaces. The TIPPERS project, at The University of California, Irvine, is a smart-building developed, in part, to study the privacy implications of IoT devices and their use in a working environment [19]. The TIPPERS system is deployed in the Bren Hall on the UC Irvine campus. The building provides IoT devices that monitor various aspects of the daily use of the building: HVAC, lighting, occupancy, access control, CCTV, and so on. TIPPERS abstracts over these systems and aims to provide privacy-aware querying and analysis of the system data [19].

One such privacy-aware abstraction over the raw IoT systems are an abstraction named SemIoTic, which provides a set of 'virtual sensors' [20]. These virtual sensors aim to provide smart-building application developers with tools for determining high-level building information (occupancy, temperature, energy usage, etc.) without providing access to the raw sensor data, which may (intentionally or not) violate privacy requirements. For example, the raw WiFi access-point data could be used to estimate room occupancy, but that very same data (MAC address connection and disconnect events) can be easily tied to a particular individual [20].

SemIoTic virtual sensors also perform some computation on the raw sensor data in order to provide more meaningful semantic information to the developer. One example of such a computation is occupancy estimation. If an individual's WiFi enabled device connects to an access point that is near, but not in, their office, the virtual sensor code will rate their presence as being more likely to be in their office.

3.9 Sparse Vector Technique

The Sparse Vector Technique (SVT) is a fundamental differential privacy technique that has the distinct feature of answering some queries at no cost while preserving the privacy of individual records. SVT has both interactive (online) and non-interactive (offline) variants. In the former, the queries are not known in advance while in the latter, the queries are available up front. There are various versions of this method available in the literature, and we recall one of them, namely, the Numeric Sparse Vector Technique (NSVT), from [21].

NSVT is an approach for accounting for the privacy effect of revealing numeric valued query answers that lie above a certain threshold while not revealing answer that are under the bound. When numeric values over the bound are published, this incurs some privacy cost. This way, one can have the advantage of getting differentially private answers to a large number of queries assuming there are a smaller number of queries whose fuzzy values are above the threshold—the answer vector is $sparse$.

We reproduce the algorithm 3 developed and presented in [21] (see Section 3.6) by Cynthia Dwork and Aaron Roth: see Figure 2.

NSVT $(\mathcal{D}, \{f_i\}, T, c, \epsilon, \delta)$

Input: a private database $\mathcal{D}, T \in \mathbb{R}, \epsilon, \delta \in [0,1], c \in \mathbb{N}$,

Parameters :

if $\delta = 0$ **then** $\epsilon_1 = \frac{8}{9}\epsilon, \epsilon_2 = \frac{2}{9}\epsilon$ **else** $\epsilon_1 = \frac{\sqrt{512}}{\sqrt{512}+1}\epsilon, \epsilon_2 = \frac{2}{\sqrt{512}+1}\epsilon$.

if $\delta = 0$ **then** $\sigma(\epsilon) = \frac{2c}{\epsilon}$ **else** $\sigma(\epsilon) = \frac{\sqrt{32c \ln \frac{2}{\delta}}}{\epsilon}$.

begin

1 : Set $\hat{T}_0 = T + Lap(\sigma(\epsilon_1)), \text{count} = 0$

2 : **for** each query i **do**

3 : $v_i = Lap(2\sigma(\epsilon_1))$

4 : **if** $f_i(\mathcal{D}) + v_i \geq \hat{T}_{\text{count}}$ **then**

5 : $w_i \leftarrow Lap(\sigma(\epsilon_2))$

Output $a_i = f_i(\mathcal{D}) + w_i$

$\text{count} = \text{count} + 1,\ \hat{T}_{\text{count}} = T + Lap(\sigma(\epsilon_1))$

6 : **else**

7 : **Output** $a_i = \perp$

8 : **end if**

9 : **if** count $\geq c$ **then**

10 : **Halt.**

11 : **end if**

12 : **end for**

end

Figure 2. Numeric Sparse Vector Technique (NSVT) Algorithm

To be more specific, on a given input that consists of private database $\mathcal{D}$, a sequence of queries $\{f_i\}$ (adaptively chosen) whose sensitivity is 1, a threshold T, a cut-off point c, and privacy budget ϵ, the NVST checks if the noisy value $f_i(\mathcal{D}) + v_i$ is greater than or equal the fuzzy version of the threshold $\hat{T}_{\text{count}}$. If the check is succeeded, then it outputs the noisy output $f_i(\mathcal{D}) + w_i$. Otherwise, it outputs $\perp$ as an answer to the query. Of course, the algorithm terminates when the number of values that exceed the threshold reached the cut-point c or all the queries are completed.

3.10 Economic Background

Consider the scenario of an advertiser who wants to learn about a potential customer's private data in order to more accurately target advertisements at them. As examples, the private data might include demographic information (age, location, level of education, approximate annual salary), behavioral data (which stores a customer visits most frequently, what loyalty programs a customer belongs to), or health data. For many of these types of data there is real risk of harm to the individual should that data be publicly disclosed. As such, in this scenario, there is some risk to the individual due to sharing the data with the advertiser. There is also presumably some benefit in the form of better knowledge about what products would be of interest to the consumer. And there is clearly an accrual of value to the advertiser, as better targeted ads are more likely to drive sales.

If we can measure information leakage or privacy risk as well as the utility gained by the consumer and advertiser, then we can approach this scenario from an economic perspective. We might want to know at what point the privacy risk exceeds the value accrued to the consumer. Or we might want to quantify the difference between the consumer and advertiser value so that payments can be offered to close this gap. This last approach of paying for access to private data is particularly common and can even have high societal value, for example when the data analyst is a health researcher and the private data is information relevant to the spread of a disease.

The academic literature has examined a number of scenarios of this form and begun developing a theory of privacy economics. We have contributed to that body of work with a study of using markets to set privacy parameter values [22] and a study of how to design auctions to facilitate the sale of private data [23].

3.11 Human Factors and Privacy Controls

An important consideration for privacy-aware systems is that the end user, whose private data is being analyzed, can understand the risks, protections, and tradeoffs of various approaches to privacy. In work led by Charles River Analytics, we conducted a number of *knowledge elicitation* exercises. In particular, we met with the FBI and Fairfax Virginia Heavy Task Force 1, a disaster response unit. These interviews and visits were summarized and shared with the Mobile CRT in order to inform the design of the RapidGather scenarios that served as an initial application target for implementation and integration efforts.

We also defined a number of scenarios that could serve as uses / tests of the RapidGather technology. These included a chemical spill scenario, a "Inaugural Ball Attack" scenario, and a missing child scenario. These incorporated details and considerations that we learned from the Virginia Task Force and provided a rich set of objectives, private data, and stakeholders.

3.12 Mental Models for Privacy

In work led by Michelle Mazurek, the TAMBA team performed research that supports the development of short in-band messages that can help to improve the functionality of users' mental models for secure messaging apps. We used an iterative, user-centered design process to learn about what concepts were most important and relevant to convey, then to test a set of short

messages to see what worked well. We designed and performed an in-person lab study as well as an online study to assess these attributes.

In the lab study [24], 25 participants were asked about their understanding of end-to-end encryption before and after a tutorial we created, as well as which information they found most useful and surprising. The study procedure consisted of:

- Participants take an initial quiz to explore understanding of confidentiality, integrity, and authenticity
- Participants receive a tutorial we designed, then take the quiz again
- Solicit feedback from participants about the tutorial
- Ask participants to critique real-world app descriptions of end-to-end encryption
- Ask participants to design a short explanation of end-to-end encryption.

In the online study [25] the focus was on assessing user attitudes toward encrypted messaging apps. We showed participants an app-store-style description of a messaging tool, varying the terminology used, whether encryption was on by default, and the prominence of encryption. We collected perceptions of the tool's security guarantees, appropriateness for privacy-focused use by whom and for what purpose, and perceptions of paranoia.

4.0 RESULTS AND DISCUSSION

4.1 Utilizing prob for analysis of Disaster Response Scenario

We used our abstract interpretation based QIF analysis tool (as described in Section 3.2) to analyze the computations involved in the Enterprise CRT Coalitions Scenario. We were able to do this by encoding the database as a set of records and mapping these records to program variables (the native input format for our tool is an imperative program that manipulates a fixed set of variables). Figure 3 is an example of this encoding, and shows the procedure for determining if a given ship has the necessary number of berths for the scenario in question.

```
querydef enough_berths ship amount -> result :
  int ship_berths = 0;
  if ship == 1 then
    ship_berths = ship1_berths
  endif;
  if ship == 2 then
    ship_berths = ship2_berths
  endif;

  int result = 0;
  if (ship_berths >= amount) then
    result = 1
  endif;
```

Figure 3. Prob Encoding of an SQL Query

Note that the potential identities of the ship are made explicit in the procedure. This is a necessary limitation which makes the abstract interpretation approach tractable for the scenario in question. Despite the requirement that the queries and the database schema be known up-front, this approach enabled us to provide useful results and feedback to the Enterprise CRT.

4.2 Decomposition-Assisted Scalability

As the Coalitions Scenario developed; more ships, ports, and additional properties (ship depth, port depth, etc.) became necessary to model. The computational limits of our technique became a more pressing consideration. In the initial formulation, all aspects of the scenario were represented as one large probabilistic polyhedra. This caused large memory usage requirements as the numeric polyhedra library that underpinned our probabilistic polyhedra implementation assumed that all dimensions may relate to each other. However, in our scenario, it is clear that not all dimensions are related to each other by constraints. For example, the number of berths in Ship 5 has no relationship to the number of berths in Ship 6.

Motivated by this observation, we implemented *decomposition* [26]. Decomposition takes the inverse approach to relating dimensions: All dimensions are assumed to be independent unless some operation forces a relational constraint. Figure 4 shows the scalability improvements gained from decomposition. In Figure 4, the y-axis shows the time in seconds, and the x-axis shows the number of dimensions that are represented in the abstract domain.

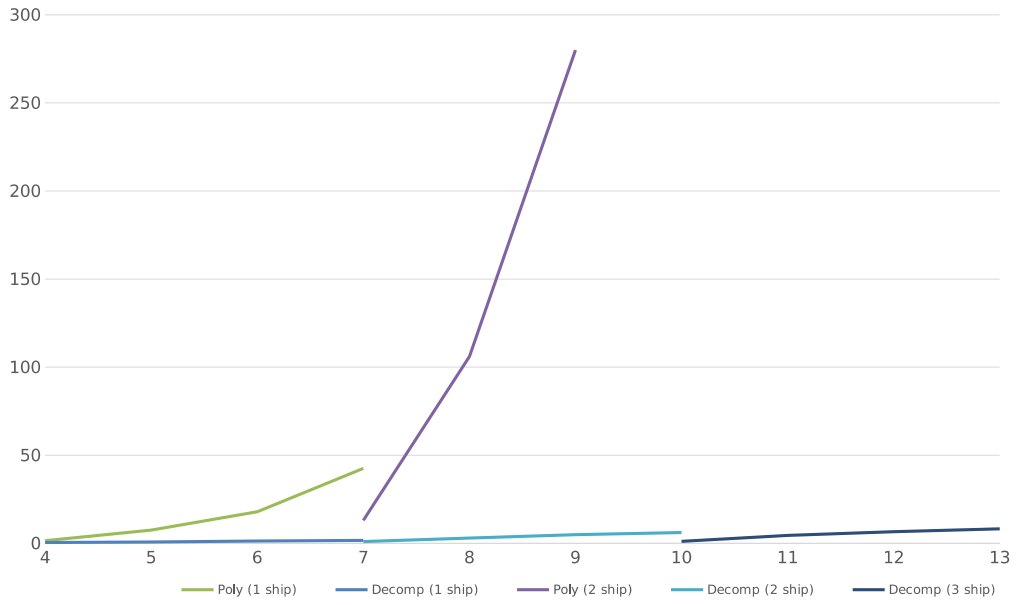


Figure 4. Scalability Improvements with Decomposition

4.3 Improving the scalability of QIF analysis via Sampling and Symbolic Execution

The prior analyses of the Coalition Scenario were based on work by Mardziel et al. [27], which uses a *sound* analysis technique based on abstract interpretation [4]. In particular, this approach involves estimating a query's probability distribution using an abstract domain called a *probabilistic polyhedron* (PP), which pairs a standard numeric abstract domain, such as *convex*

polyhedra [28], with some additional *ornaments*, which include lower and upper bounds on the size of the support of the distribution, and bounds on the probability of each possible secret value. Using PP can yield a precise, yet safe, estimate of the privacy leakage. However, PPs can be very inefficient. Defining *intervals* [5] as the PP's numeric domain can dramatically improve performance, but only with an unacceptable loss of precision.

The goal of the work described in this subsection was to investigate a new approach that ensures a better balance of both precision and performance in vulnerability computation, augmenting PP with two new techniques. In both cases we begin by analyzing a query using the fast interval-based analysis. Our first technique is then to use *sampling* to augment the result. In particular, we execute the query using possible secret values i sampled from the posterior δ' derived from a particular output o_i. If the analysis were perfectly accurate, executing $f(i)$ would produce o_i. But since intervals are overapproximate, sometimes it will not. With many sampled outcomes, we can construct a Beta distribution to estimate the size of the support of the posterior, up to some level of confidence. We can use this estimate to boost the lower bound of the abstraction, and thus improve the precision of the estimated vulnerability.

Our second technique is of a similar flavor, but uses symbolic reasoning to magnify the impact of a successful sample. In particular, we execute a query result-consistent sample *concolically* [29], thus maintaining a symbolic formula (called the *path condition*) that characterizes the set of variable valuations that would cause execution to follow the observed path. We then count the number of possible solutions and use the count to boost the lower bound of the support (with 100% confidence).

Sampling and concolic execution can be combined for even greater precision.

We have formalized and proved our techniques are sound (Sections 4.3.3–4.3.6) and implemented and evaluated them (Sections 4.3.7 and 4.3.8). Using a privacy-sensitive ship planning scenario (Section 4.3.1) we find that our techniques provide similar precision to convex polyhedra while providing orders-of-magnitude better performance. More experiments are needed to see if the approach provides such benefits more generally. Our implementation is freely available at https://github.com/GaloisInc/TAMBA.

4.3.1 Overview

To provide an overview of our approach, we will describe the application of our techniques to a scenario that involves a coalition of ships from various nations operating in a shared region. Suppose a natural disaster has impacted some islands in the region. Some number of individuals need to be evacuated from the islands, and it falls to a regional disaster response coordinator to determine how to accomplish this. While the coalition wants to collaborate to achieve these humanitarian aims, we assume that each nation also wants to protect their sensitive data—namely ship locations and capacity.

More formally, we assume the use of the data model shown in Figure 5, which considers a set of ships, their coalition affiliation, the evacuation capacity of the ship, and its position, given in

terms of latitude and longitude.[2] We sometimes refer to the latter two as a location l, with $l.x$ as the longitude and $l.y$ as the latitude. We will often index properties by ship id, writing capacity(z) for the capacity associated with ship id z, or location(z) for the location.

field	type	range	private?
shipid	integer	1–10	no
nationid	integer	1–20	no
capacity	integer	0–1000	yes
latitude	integer	-900,000–900,000	yes
longitude	integer	-1,800,000–1,800,000	yes

Figure 5. Data Model Used in the Evacuation Scenario

The **evacuation problem** is defined as follows:

Given a target location l and number of people to evacuate n, compute a set of nearby ships s such that $\sum_{z \in s}$ capacity $(z) \geq n$.

Our goal is to solve this problem in a way that minimizes the vulnerability to the coordinator of private information, i.e., the ship locations and their exact capacity. We assume that this coordinator initially has no knowledge of the positions or capabilities of the ships other than that they fall within certain expected ranges.

If all members of the coalition share all of their data with the coordinator, then a solution is easy to compute, but it affords no privacy. Figure 6 gives an algorithm the response coordinator can follow that does not require each member to share all of their data. Instead, it iteratively performs queries *atleast* and *nearby*. These queries do not reveal precise values about ship locations or capacity, but rather admit ranges of possibilities. The algorithm works by maintaining upper and lower bounds on the capacity of each ship i in the array berths. Each ship's bounds are updated based on the results of queries about its capacity and location. These queries aim to be privacy preserving, doing a sort of binary search to narrow in on the capacity of each ship in the operating area. The procedure completes once its solution determines the minimum required capacity is reached.

[2] We give latitude and longitude values as integer representations of *decimal degrees* fixed to four decimal places; e.g., 14.3579 decimal degrees is encoded as 143579.

```
(* s = #ships; n = #evacuees; l = island loc.; d = min. proximity to l *)
    let berths = array.make s (0,1000)
    let is_solution () = sum (array.map fst berths) >= n
    let mid (x,y) = (x + y) / 2
    let <@\textit{atleast(z,b)}@> = capacity(z) >= b
    let <@\textit{nearby(z,l,d)}@> =|loc(z).x - l.x| + |loc(z).y - l.y| <= d
    while true do
      for i = 0 to s do
        let ask = mid berths[i]
        let ok = <@\textit{atleast}@>(i,ask) && <@\textit{nearby}@>(i,l,d)
        if ok then  berths[i] <- (ask, snd berths[i])
        else        berths[i] <- (fst berths[i], ask)
        if is_solution () then return berths
      done
    done
```

Figure 6. Algorithm to Solve the Evacuation Problem for a Single Island

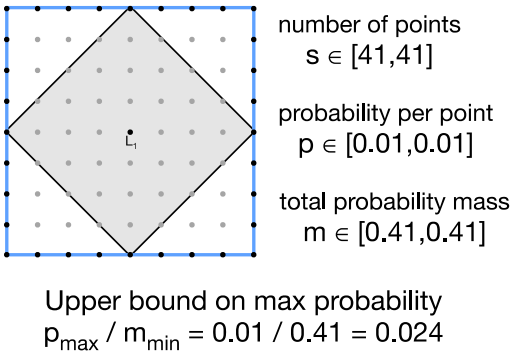
4.3.2 Improving precision with sampling and concolic execution

Using the technique proposed by Mardziel et al. [27] (described in Section 3.2), we can calculate an estimate of the QIF properties of this scenario. Here, we will describe how the guarantees of this approach (that the result is always a safe upper-bound) can lead to significant inaccuracies.

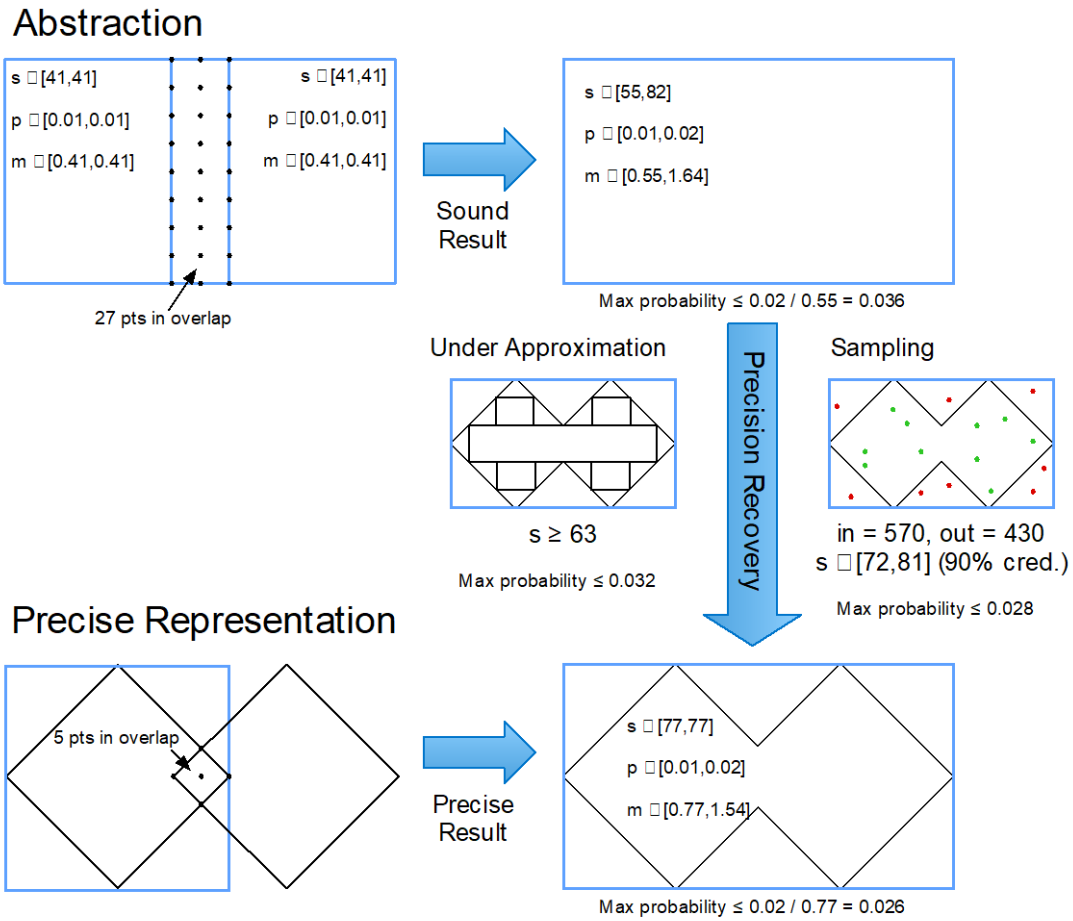
In Figure 7(a), the parameters s, p, and m are precise. However, as additional operations are performed, these quantities can accumulate imprecision. For example, suppose we are using intervals for the shape domain, and we wish to analyze the query $\text{Nearby}(z, L_1, 4) \vee \text{Nearby}(z, L_2, 4)$ (for some nearby point L_2). The result is produced by analyzing the two queries separately and then combining them with an *abstract join*; this is shown in the top row of Figure 7(b). Unfortunately, the result is very imprecise. The bottom row of Figure 7(b) illustrates the result we would get by using convex polyhedra as our shape domain. When using intervals (top row), the vulnerability is estimated as 0.036, whereas the precise answer (bottom row) is actually 0.026. Unfortunately, obtaining this precise answer is far more expensive than obtaining the imprecise one.

This work presented two techniques that can allow us to use the less precise interval domain but then *recover* lost precision in a relatively cheap post-processing step. The effect of our techniques is shown in the middle-right of Figure 7(b). Both techniques aim to obtain better lower bounds for s. This allows us to update lower bounds on the probability mass m since m_{min} is at least $s_{\text{min}} \cdot p_{\text{min}}$ (each point has at least probability p_{min} and there are at least s_{min} of them). A larger m means a smaller vulnerability.

The first technique we explore is *sampling*, depicted to the right of the arrow in Figure 7(b). Sampling chooses random points and evaluates the query on them to determine whether they are in the support of the posterior distribution for a particular query result. By tracking the ratio of points that produce the expected output, we can produce an estimate of s, whose confidence increases as we include more samples. This approach is depicted in the figure, where we conclude that $s \in [72,81]$ and $m \in [0.72,1.62]$ with 90% confidence after taking 1000 samples, improving our vulnerability estimate to $V \leq \frac{0.02}{0.72} = 0.028$.



(a) Probabilistic polyhedra



(b) Improving precision with sampling and underapproximation (concolic execution)

Figure 7. Computing Vulnerability (Max Probability) Using Abstract Interpretation

The second technique we explore is the use of *concolic execution* to derive a *path condition*, which is a formula over secret values that is consistent with a query result. By performing *model counting* to estimate the number of solutions to this formula, which are an underapproximation of the true size of the distribution, we can safely boost the lower bound of s. This approach is depicted to the left of the arrow in Figure 7(b). The depicted shapes represent discovered path condition's disjuncts, whose size sums to 63. This is a better lower bound on s and improves the vulnerability estimate to 0.032.

These techniques can be used together to further increase precision. In particular, we can first perform concolic execution, and then sample from the area not covered by this underapproximation. Importantly, Section 4.3.8 shows that using our techniques with the interval-based analysis yields an orders of magnitude performance improvement over using polyhedra-based analysis alone, while achieving similar levels of precision, with high confidence.

4.3.3 Syntax and Semantics

This section presents the core language—syntax and semantics—in which we formalize our approach to computing vulnerability.

Core Language and Semantics:

The programming language we use for queries is given in Figure 8. The language is essentially standard, apart from $\mathsf{pif}\ q\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2$, which implements probabilistic choice: S_1 is executed with probability q, and S_2 with probability $1-q$. We limit the form of expressions E so that they can be approximated by standard numeric abstract domains such as convex polyhedra [28]. Such domains require linear forms; e.g., there is no division operator and multiplication of two variables is disallowed.[3]

Variables	x	$\in$	$\mathbf{V}$
Integers	n	$\in$	$\mathbb{Z}$
Rationals	q	$\in$	$\mathbb{Q}$
States	σ	$\in$	$\Sigma \stackrel{\text{def}}{=} \mathbf{V} \rightharpoonup \mathbb{Z}$
Distributions	δ	$\in$	$\mathbb{D}\Sigma \stackrel{\text{def}}{=} \Sigma \to \mathbb{R}_{+0}$
Arith.ops	*aop*	$::=$	$+ \mid \times \mid -$
Rel.ops	*relop*	$::=$	$\leq \mid < \mid = \mid \neq \mid \cdots$
Arith.exps	E	$::=$	$x \mid n \mid E_1\ aop\ E_2$
Bool.exps	B	$::=$	$E_1\ relop\ E_2 \mid B_1 \wedge B_2 \mid B_1 \vee B_2 \mid \neg B$
Statements	S	$::=$	$\mathsf{skip} \mid x := E \mid S_1\ ;\ S_2 \mid \mathsf{while}\ B\ \mathsf{do}\ S \mid$
			$\mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \mid \mathsf{pif}\ q\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2$

Figure 8. Core Language Syntax

[3] Relaxing such limitations is possible—e.g., polynominal inequalities can be approximated using convex polyhedra [30]—but doing so precisely and scalably is a challenge.

We define the semantics of a program in terms of its effect on (discrete) distributions of states. States σ are partial maps from variables to integers; we write domain(σ) for the set of variables over which σ is defined. Distributions δ are maps from states to nonnegative real numbers, interpreted as probabilities (in range [0,1]). The denotational semantics considers a program as a relation between distributions. In particular, the semantics of statement S, written $[[S]]$, is a function of the form $\mathbb{D}\Sigma \to \mathbb{D}\Sigma$; we write $[[S]]\delta = \delta'$ to say that the semantics of S maps input distribution δ to output distribution δ'. Distributions are not necessarily normalized; we write $\| \delta \|$ as the probability mass of δ (which is between 0 and 1). We write $\dot{\sigma}$ to denote the point distribution that gives σ probability 1, and all other states 0.

The semantics is standard and not crucial in order to understand our techniques. See Clarkson et al. [31] or Mardziel et al [27] for detailed explanations.

4.3.4 Computing Vulnerability: Basic procedure

This section explains the basic approach by which we can use probabilistic polyhedra to compute *vulnerability*, i.e., the probability of the most probable point of the posterior distribution. Improvements on this basic approach are given in the next two sections.

Our convention will be to use a_1, $\mathrm{s}_1^{\min}$, $\mathrm{s}_1^{\max}$, etc. for the components associated with probabilistic polyhedron P_1. In the program S of interest, we assume that secret variables are in the set T, so input states are written σ_T, and we assume there is a single output variable r. We assume that the adversary's initial uncertainty about the possible values of the secrets T is captured by the probabilistic polyhedron P_0 (such that domain(P_0) $\supseteq T$).

Computing vulnerability occurs according to the following procedure.

1. Perform abstract interpretation: $\langle\langle S \rangle\rangle P_0 = P$
2. Given a concrete output value of interest, o, perform abstract conditioning to define[4] $P_{r=o} \stackrel{\text{def}}{=} (P \wedge r = o)$.

The vulnerability V is the probability of the most likely state(s). When a probabilistic polyhedron represents one or more true distributions (i.e., the probabilities all sum to 1), the most probable state's probability is bounded by $\mathrm{p}^{\max}$. However, the abstract semantics does not always normalize the probabilistic polyhedron as it computes, so we need to scale $\mathrm{p}^{\max}$ according to the total probability mass. To ensure that our estimate is on the safe side, we scale $\mathrm{p}^{\max}$ using the *minimum* probability mass: $V = \frac{\mathrm{p}^{\max}}{\mathrm{m}^{\min}}$. In Figure 7(b), the sound approximation in the top-right has $V \leq \frac{0.02}{0.55} = 0.036$ and the most precise approximation in the bottom-right has $V \leq \frac{0.02}{0.77} = 0.026$.

[4] We write $P \wedge B$ and not $P \mid B$ because P need not be normalized.

4.3.5 Improving precision with sampling

We can improve the precision of the basic procedure using sampling. First, we introduce some notational convenience:

$$P_T \stackrel{\text{def}}{=} P \wedge (r = o) \downharpoonright T$$
$$P_{T+} \stackrel{\text{def}}{=} P_T \text{ revised polyhedron with confidence } \omega$$

P_T is equivalent to step 2, above, but projected onto the set of secret variables T. P_{T+} is the improved (via sampling) polyhedron.

After computing P_T with the basic procedure from the previous section we take the following additional steps:

1. Set counters α and β to zero.

2. Do the following N times (for some N, see below):

    a. Randomly select an input state $\sigma_T \in \gamma_{\mathbb{C}}(a_T)$.

    b. "Run" the program by computing $[[S]]\sigma_T = \delta$. If there exists $\sigma \in \text{support}(\delta)$ with $\sigma(r) = o$ then increment α, else increment β.

3. We can interpret α and β as the parameters of a Beta distribution of the likelihood that an arbitrary state in $\gamma_{\mathbb{C}}(a_T)$ is in the support of the true distribution. From these parameters we can compute the *credible interval* $[p_L, p_U]$ within which is contained the true likelihood, with confidence ω (where $0 \leq \omega \leq 1$). A credible interval is essentially a Bayesian analogue of a confidence interval and can be computed from the cumulative distribution function (CDF) of the Beta distribution (the 99% credible interval is the interval $[a, b]$ such that the CDF at a has value 0.005 and the CDF at b has value 0.995). In general, obtaining a higher confidence or a narrower interval will require a higher N. Let result $P_{T+} = P_T$ except that $\mathrm{s}_{T+}^{\min} = p_L \cdot \#(a_T)$ and $\mathrm{s}_{T+}^{\max} = p_U \cdot \#(a_T)$ (assuming these improve on $\mathrm{s}_T^{\min}$ and $\mathrm{s}_T^{\max}$). We can then propagate these improvements to $\mathrm{m}^{\min}$ and $\mathrm{m}^{\max}$ by defining $\mathrm{m}_{T+}^{\min} = \mathrm{p}_T^{\min} \cdot \mathrm{s}_{T+}^{\min}$ and $\mathrm{m}_{T+}^{\max} = \mathrm{p}_T^{\max} \cdot \mathrm{s}_{T+}^{\max}$. Note that if $\mathrm{m}_T^{\min} > \mathrm{m}_{T+}^{\min}$ we leave it unchanged, and do likewise if $\mathrm{m}_T^{\max} < \mathrm{m}_{T+}^{\max}$.

At this point we can compute the vulnerability as in the basic procedure, but using P_{T+} instead of P_T.

Consider the example of Section 4.3.2. In Figure 7(b), we draw samples from the rectangle in the top-right. This rectangle overapproximates the set of locations where s might be, given that the query returned true. We sample locations from this rectangle and run the query on each sample. The green (red) dots indicate true (false) results, which are added to α (β). After sampling $N = 1000$ locations, we have $\alpha = 570$ and $\beta = 430$. Choosing $\omega = .9$ (90%), we compute the credible interval [0.53,0.60]. With $\#(a_T) = 135$, we compute $[\mathrm{s}_{T+}^{\min}, \mathrm{s}_{T+}^{\max}]$ as $[0.53 \cdot 135, 0.60 \cdot 135] = [72,81]$.

There are several things to notice about this procedure. First, observe that in step 2b we "run" the program using the point distribution $\dot{\sigma}$ as an input; in the case that S is deterministic (has no pif statements) the output distribution will also be a point distribution. However, for programs with pif statements there are multiple possible outputs depending on which branch is taken by a pif. We consider all of these outputs so that we can confidently determine whether the input state σ could ever cause S to produce result o. If so, then σ should be considered part of P_{T+}. If not, then we can safely rule it out (i.e., it is part of the overapproximation).

Second, we only update the size parameters of P_{T+}; we make no changes to $\mathrm{p}_{T+}^{\min}$ and $\mathrm{p}_{T+}^{\max}$. This is because our sampling procedure only determines whether it is *possible* for an input state to produce the expected output. The probability that an input state produces an output state is already captured (soundly) by p_T so we do not change that. This is useful because the approximation of p_T does not degrade with the use of the interval domain in the way the approximation of the size degrades (as illustrated in Figure 7(b)). Using sampling is an attempt to regain the precision lost on the size component (only).

Finally, the confidence we have that sampling has accurately assessed which input states are in the support is orthogonal to the probability of any given state. In particular, P_T is an abstraction of a distribution δ_T, which is a mathematical object. Confidence ω is a measure of how likely it is that our abstraction (or, at least, the size part of it) is accurate.

We prove (in our extended report [32]) that our sampling procedure is sound:

$$\textit{If } \delta_0 \in \gamma_{\mathbb{P}}(P_0),\ \langle\!\langle S \rangle\!\rangle\, P_0 = P, \textit{ and } [\![S]\!]\delta_0 = \delta \textit{ then } \delta_T \in \gamma_{\mathbb{P}}(P_{T+}) \textit{ with confidence } \omega \textit{ where}$$

$$\begin{aligned} \delta_T &\stackrel{\text{def}}{=} \delta \wedge (r = o) \downharpoonright T \\ P_T &\stackrel{\text{def}}{=} P \wedge (r = o) \downharpoonright T \\ P_{T+} &\stackrel{\text{def}}{=} P_T \textit{ sampling revised with confidence } \omega. \end{aligned}$$

4.3.6 Improving precision with concolic execution

Another approach to improving the precision of a probabilistic polyhedron P is to use concolic execution. The idea here is to "magnify" the impact of a single sample to soundly increase $\mathrm{s}^{\min}$ by considering its execution *symbolically*. More precisely, we concretely execute a program using a particular secret value, but maintain symbolic constraints about how that value is used. This is referred to as *concolic* execution [29]. We use the collected constraints to identify all points that would induce the same execution path, which we can include as part of $\mathrm{s}^{\min}$.

We begin by defining the semantics of concolic execution, and then show how it can be used to increase $\mathrm{s}^{\min}$ soundly.

(Probabilistic) Concolic Execution:

Concolic execution is expressed as rewrite rules defining a judgment $\langle \Pi, S \rangle \rightarrow_{\pi}^{p} \langle \Pi', S' \rangle$. Here, Π is pair consisting of a concrete state σ and symbolic state ζ. The latter maps variables $x \in \mathrm{V}$ to *symbolic expressions* $\mathcal{E}$ which extend expressions E with *symbolic variables* α. This judgment indicates that under input state Π the statement S reduces to statement S' and output state Π' with

probability p, with *path condition* π. The path condition is a conjunction of boolean symbolic expressions $\mathcal{B}$ (which are just boolean expressions B but altered to use symbolic expressions $\mathcal{E}$ instead of expressions E) that record which branch is taken during execution. For brevity, we omit π in a rule when it is true.

The rules for the concolic semantics are given in Figure 9. Most of these are standard, and deterministic (the probability annotation p is 1). Path conditions are recorded for if and while, depending on the branch taken. The semantics of pif q then S_1 else S_2 is non-deterministic: the result is that of S_1 with probability q, and S_2 with probability $1 - q$. We write $\zeta(B)$ to substitute free variables $x \in B$ with their mapped-to values $\zeta(x)$ and then simplify the result as much as possible. For example, if $\zeta(x) = \alpha$ and $\zeta(y) = 2$, then $\zeta(x > y + 3) = \alpha > 5$. The same goes for $\zeta(E)$.

We define a *complete run* of the concolic semantics with the judgment $\langle \Pi, S \rangle \Downarrow_{\pi}^{p} \Pi'$, which has two rules:

$$\langle \Pi, \mathsf{skip} \rangle \Downarrow_{\mathsf{true}}^{1} \Pi$$

$$\frac{\langle \Pi, S \rangle \longrightarrow_{\pi}^{p} \langle \Pi', S' \rangle \quad \langle \Pi', S' \rangle \Downarrow_{\pi'}^{q} \Pi''}{\langle \Pi, S \rangle \Downarrow_{\pi \wedge \pi'}^{p \cdot q} \Pi''}$$

A complete run's probability is thus the product of the probability of each individual step taken. The run's path condition is the conjunction of the conditions of each step.

The path condition π for a complete run is a conjunction of the (symbolic) boolean guards evaluated during an execution. π can be converted to disjunctive normal form (DNF), and given the restrictions of the language the result is essentially a set of convex polyhedra over symbolic variables α.

$$\begin{array}{l}
\langle (\sigma, \zeta), x := E \rangle \longrightarrow^{1} \langle (\sigma[x \mapsto \sigma(E)], \zeta[x \mapsto \zeta(E)]), \mathsf{skip} \rangle \\
\langle (\sigma, \zeta), \mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \rangle \longrightarrow_{\zeta(B)}^{1} \langle (\sigma, \zeta), S_1 \rangle \qquad \text{if } \sigma(B) \\
\langle (\sigma, \zeta), \mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \rangle \longrightarrow_{\zeta(\neg B)}^{1} \langle (\sigma, \zeta), S_2 \rangle \quad \text{if } \sigma(\neg B) \\
\langle \Pi, \mathsf{pif}\ q\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \rangle \longrightarrow^{q} \langle \Pi, S_1 \rangle \\
\langle \Pi, \mathsf{pif}\ q\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \rangle \longrightarrow^{1-q} \langle \Pi, S_2 \rangle \\
\langle \Pi, S_1\ ;\ S_2 \rangle \longrightarrow_{\pi}^{1} \langle \Pi', S_1'\ ;\ S_2 \rangle \quad \text{if } \langle \Pi, S_1 \rangle \longrightarrow_{\pi}^{1} \langle \Pi', S_1' \rangle \\
\langle \Pi, \mathsf{skip}\ ;\ S \rangle \longrightarrow^{1} \langle \Pi, S \rangle \\
\langle \Pi, \mathsf{while}\ B\ \mathsf{do}\ S \rangle \longrightarrow_{\zeta(B)}^{1} \langle \Pi, S\ ;\ \mathsf{while}\ B\ \mathsf{do}\ S \rangle \qquad \text{if } \sigma(B) \\
\langle \Pi, \mathsf{while}\ B\ \mathsf{do}\ S \rangle \longrightarrow_{\zeta(\neg B)}^{1} \langle \Pi, \mathsf{skip} \rangle \qquad \text{if } \sigma(\neg B)
\end{array}$$

Figure 9. Concolic Semantics

Combining Sampling with Concolic Execution:

Sampling can be used to further augment the results of concolic execution. The key insight is that the presence of a sound under-approximation generated by the concolic execution means that it is unnecessary to sample from the under-approximating region. Here is the algorithm:

1. Let $a = a_0 \sqcap (\bigsqcup_i a_i)$ be the under-approximating region.

2. Perform sampling per the algorithm in Section 4.3.5, but with two changes:

   – if a sampled state $\sigma_T \in \gamma_{\mathbb{C}}(a)$, ignore it

   – When done sampling, compute $s_{T+}^{\min} = p_L \cdot (\#(a_T) - \#(a)) + \#(a)$ and $s_{T+}^{\max} = p_U \cdot (\#(a_T) - \#(a)) + \#(a)$. This differs from Section 4.3.5 in not including the count from concolic region a in the computation. This is because, since we ignored samples $\sigma_T \in \gamma_{\mathbb{C}}(a)$, the credible interval $[p_L, p_U]$ bounds the likelihood that any given point in $a_T \backslash a$ is in the support of the true distribution.

For our example, concolic execution indicated there are at least 41 points that satisfy the query. With this in hand, and using the same samples as shown in Section 4.3.5, we can refine $s \in [74,80]$ and $m \in [0.74,0.160]$ (the credible interval is formed over only those samples which satisfy the query but fall outside the under-approximation returned by concolic execution). We improve the vulnerability estimate to $V \leq \frac{0.02}{0.0.74} = 0.027$. These bounds (and vulnerability estimate) are better than those of sampling alone ($s \in [72,81]$ with $V \leq 0.028$).

The statement of soundness and its proof can be found in the extended technical report [32].

4.3.7 Implementation

We have implemented our approach as an extension of Mardziel et al. [27], which is written in OCaml. This baseline implements numeric domains a via an OCaml interface to the Parma Polyhedra Library [33]. The counting procedure $\#(a)$ is implemented by LattE [34]. Support for arbitrary precision and exact arithmetic (e.g., for manipulating $\mathrm{m}^{\min}$, $\mathrm{p}^{\min}$, etc.) is provided by the mlgmp OCaml interface to the GNU Multi Precision Arithmetic library. Rather than maintaining a single probabilistic polyhedron P, the implementation maintains a *powerset* of polyhedra [35], i.e., a finite disjunction. Doing so results in a more precise handling of join points in the control flow, at a somewhat higher performance cost.

We have implemented our extensions to this baseline for the case that domain a is the interval numeric domain [5]. Of course, the theory fully applies to any numeric abstract domain. We use Gibbs sampling, which we implemented ourselves. We delegate the calculation of the beta distribution and its corresponding credible interval to the ocephes OCaml library, which in turn uses the GNU Scientific Library. It is straightforward to lift the various operations we have described to the powerset domain. All of our code is available at https://github.com/GaloisInc/TAMBA.

4.3.8 Experiments

To evaluate the benefits of our techniques, we applied them to queries based on the evacuation problem outlined in Section 4.3.1. We found that while the baseline technique can yield precise answers when computing vulnerability, our new techniques can achieve close to the same level of precision far more efficiently.

Experimental Setup:

For our experiments we analyzed queries similar to $Nearby(s, l, d)$ from Figure 6. We generalize the *Nearby* query to accept a set of locations L—the query returns true if s is within d units of any one of the islands having location $l \in L$. In our experiments we fix $d = 100$. We consider the secrecy of the location of s, $Location(s)$. We also analyze the execution of the resource allocation algorithm of Figure 6 directly; we discuss this in Section 4.3.8.3.

We measure the time it takes to compute the *vulnerability* (i.e., the probability of the most probable point) following each query. In our experiments, we consider a single ship s and set its coordinates so that it is always in range of some island in L, so that the concrete query result returns true (i.e. $Nearby(s, L, 100) = true$). We measure the vulnerability following this query result starting from a prior belief that the coordinates of s are uniformly distributed with $0 \leq \text{Location}(s).x \leq 1000$ and $0 \leq \text{Location}(s).y \leq 1000$.

In our experiments, we varied several experimental parameters: *analysis method* (either P, I, CE, S, or CE+S), *query complexity* c; *AI precision level* p; and *number of samples* n. We describe each in turn.

Analysis method

We compared five techniques for computing vulnerability:

P: Abstract Interpretation (AI) with convex polyhedra for domain a (Section 4.3.4),

I: AI with intervals for a (Section 4.3.4),

S: AI with intervals augmented with sampling (Section 4.3.5),

CE: AI with intervals augmented with concolic execution (CE) (Section 4.3.6), and

CE+S: AI with intervals augmented with both techniques (Section 4.3.6.2)

The first two techniques are due to Mardziel et al. [27], where the former uses convex polyhedra and the latter uses intervals (aka boxes) for the underlying polygons. In our experiments we tend to focus on P since I's precision is unacceptably poor (e.g., often vulnerability = 1).

Query complexity. We consider queries with different L; we say we are increasing the *complexity* of the query as L gets larger. Let $c = |L|$; we consider $1 \leq c \leq 5$, where larger L include the same locations as smaller ones. We set each location to be at least $2 \cdot d$ Manhattan distance units away from any other island (so diamonds like those in Figure 7(a) never overlap).

Precision. The precision parameter p bounds the size of the powerset abstract domain at all points during abstract interpretation. This has the effect of forcing joins when the powerset grows larger than the specified precision. As p grows larger, the results of abstract interpretation are likely to become more precise (i.e. vulnerability gets closer to the true value). We considered p values of $1, 2, 4, 8, 16, 32,$ and 64.

Samples taken. For the latter three analysis methods, we varied the number of samples taken n. For analysis CE, n is interpreted as the number of samples to try per polyhedron before giving up trying to find a "valid sample."[5] For analysis S, n is the number of samples, distributed proportionally across all the polyhedra in the powerset. For analysis CE+S, n is the combination of the two. We considered sample size values of $1{,}000 - 50{,}000$ in increments of $1{,}000$. We always compute an interval with ω =99.9% confidence (which will be wider when fewer samples are used).

System description. We ran experiments varying all possible parameters. For each run, we measured the total execution time (wall clock) in seconds to analyze the query and compute vulnerability. All experiments were carried out on a MacBook Air with OSX version 10.11.6, a 1.7GHz Intel Core i7, and 8GB of RAM. We ran a single trial for each configuration of parameters. Only wall-clock time varies across trials; informally, we observed time variations to be small.

Results:

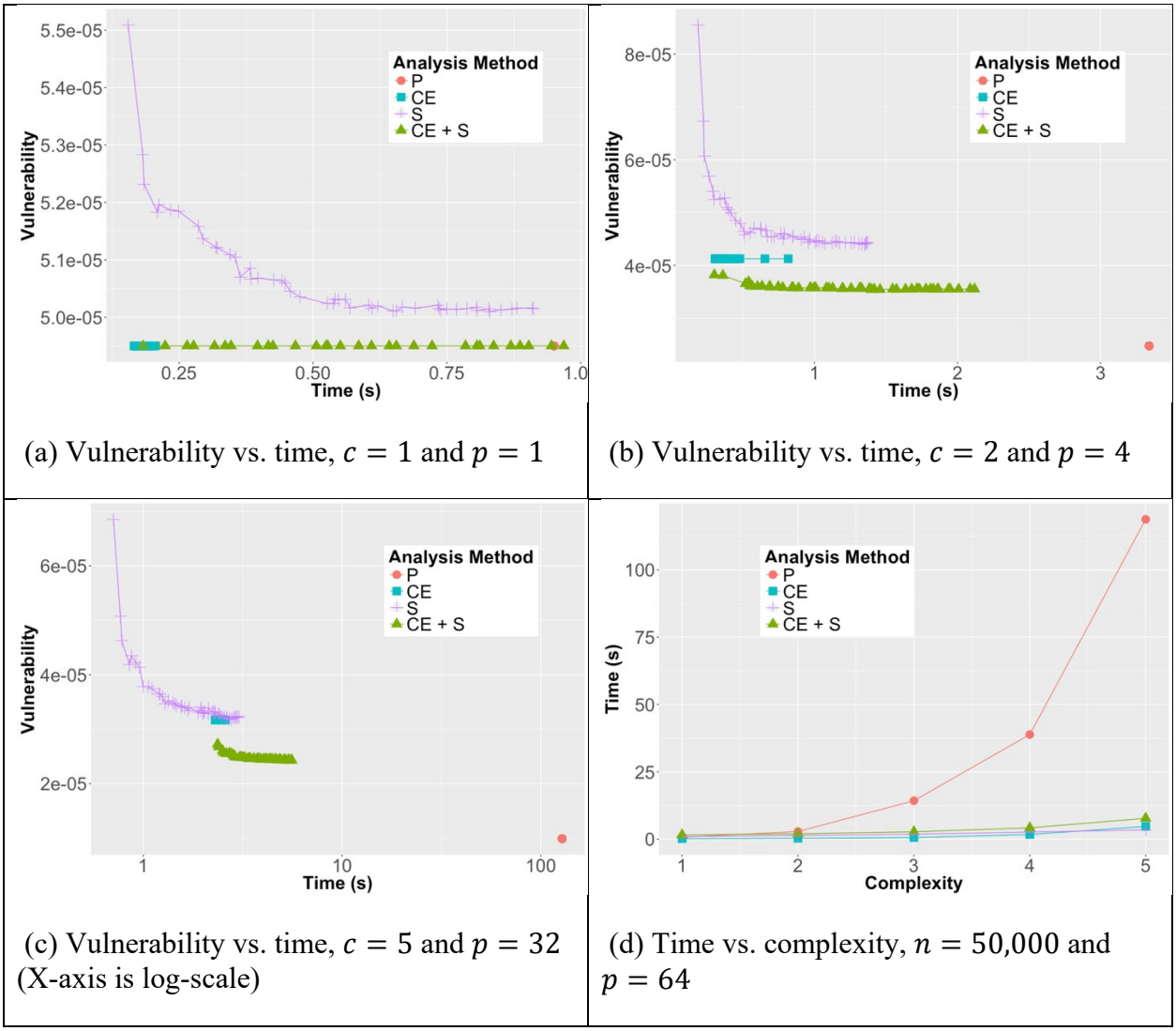
Figure 10(a)–(c) measure vulnerability (y-axis) as a function of time (x-axis) for each analysis. These three figures characterize three interesting "zones" in the space of complexity and precision. The results for method I are not shown in any of the figures. This is because I always produces a vulnerability of 1. The refinement methods (CE, S, and CE+S) are all over the interval domain, and should be considered as "improving" the vulnerability of I.

In Figure 10(a) we fix $c = 1$ and $p = 1$. In this configuration, baseline analysis P can compute the true vulnerability in ~ 0.95 seconds. Analysis CE is also able to compute the true vulnerability, but in ~ 0.19 seconds. Analysis S is able to compute a vulnerability to within $\sim 5 \cdot e^{-6}$ of optimal in ~ 0.15 seconds. These data points support two key observations. First, even a very modest number of samples improves vulnerability significantly over just analyzing with intervals. Second, concolic execution is only slightly slower and can achieve the optimal vulnerability. Of course, concolic execution is not a panacea. As we will see, a feature of this configuration is that no joins take place during abstract interpretation. This is critical to the precision of the concolic execution.

In Figure 10(b) we fix $c = 2$ and $p = 4$. In contrast to the configuration of Figure 10(a), the values for c and p in this configuration are not sufficient to prevent all joins during abstract interpretation. This has the effect of taking polygons that represent individual paths through the program and joining them into a single polygon representing many paths. We can see that this is the case because baseline analysis P is now achieving a better vulnerability than CE. However,

[5] This is the N parameter from Section 4.3.6.

one pattern from the previous configuration persists: all three refinement methods (CE, S, CE+S) can achieve vulnerability within $\sim 1 \cdot e^{-5}$ of P, but in $\frac{1}{4}$ the time. In contrast to the previous configuration, analysis CE+S is now able to make a modest improvement over CE (since it does not achieve the optimal).



(a) Vulnerability vs. time, $c = 1$ and $p = 1$

(b) Vulnerability vs. time, $c = 2$ and $p = 4$

(c) Vulnerability vs. time, $c = 5$ and $p = 32$ (X-axis is log-scale)

(d) Time vs. complexity, $n = 50{,}000$ and $p = 64$

Figure 10. Experimental Results

In Figure 10(c) we fix $c = 5$ and $p = 32$. This configuration magnifies the effects we saw in Figure 10(b). Similarly, in this configuration there are joins happening, but the query is much more complex and the analysis is much more precise. In this figure, we label the X axis as a log scale over time. This is because analysis P took over two minutes to complete, in contrast to the longest-running refinement method, which took less than 6 seconds. The relationship between the refinement analyses is similar to the previous configuration. The key observation here is that, again, all three refinement analyses achieve within $\sim 3 \cdot e^{-5}$ of P, but this time in 4% of the time (as opposed to $\frac{1}{4}$ in the previous configuration).

Figure 10(d) makes more explicit the relationship between refinements (CE, S, CE+S) and P. We fix $n = 50{,}000$ (the maximum) here, and $p = 64$ (the maximum). We can see that as query complexity goes up, P gets exponentially slower, while CE, S, and CE+S slow at a much lower rate, while retaining (per the previous graphs) similar precision.

Evacuation Problem:

We conclude this section by briefly discussing an analysis of an execution of the resource allocation algorithm of Figure 6. In our experiment, we set the number of ships to be three, where two were in range $d = 300$ of the evacuation site, and their sum-total berths (500) were sufficient to satisfy demand at the site (also 500). For our analysis refinements we set $n = 1000$. Running the algorithm, a total of seven pairs of *Nearby* and *Capacity* queries were issued. In the end, the algorithm selects two ships to handle the evacuation.

Table 1 shows the time to execute the algorithm using the different analysis methods, along with the computed vulnerability—this latter number represents the coordinator's view of the most likely nine-tuple of the private data of the three ships involved (x coordinate, y coordinate, and capacity for each). We can see that, as expected, our refinement analyses are far more efficient than baseline P, and far more precise than baseline I. The CE methods are precise but slower than S. This is because of the need to count the number of points in the DNF of the concolic path conditions, which is expensive.

Table 1. Analyzing a 3-ship Resource Allocation Run

Resource Allocation (3 ships)		
Analysis	Time (s)	Vulnerability
P	Timeout (5 min)	N/A
I	0.516	1
CE	16.650	$1.997 \cdot 10^{-24}$
S	1.487	$1.962 \cdot 10^{-24}$
CE+S	17.452	$1.037 \cdot 10^{-24}$

Discussion:

The queries considered in Figure 10 have two features that contribute to the effectiveness of our refinement methods. First, they are defined over large domains, but return true for only a small subset of those values. For larger subsets of values, the benefits of sampling may degrade, though concolic execution should still provide an improvement. Further experiments are needed to explore such scenarios. Second, the example in Figure 10 contains short but complex queries. A result of this query structure is that abstract interpretation with polyhedra is expensive but sampling can be performed efficiently. The evacuation problem results in Table 10 provide some evidence that the benefits of our techniques also apply to longer queries. However, it may still be possible to construct queries where the gap in runtime between polyhedral analysis and sampling is smaller, in which case sampling would provide less improvement.

4.3.9 Conclusions of this work

Quantitative information flow is concerned with measuring the knowledge about secret data that is gained by observing the answer to a query. This work presented a combination of static analysis using probabilistic abstract interpretation, sampling, and underapproximation via concolic execution to compute high-confidence upper bounds on information flow. Preliminary experimental results are promising and suggest that this approach can operate more precisely and efficiently than abstract interpretation alone.

4.4 QIF-Supported Workflow Adaptation

Another core TAMBA result was the demonstration of the ability to adapt workflows based on online leakage computation. To demonstrate this technique, we use the problem of coalition collaboration for aid delivery without revealing private information to demonstrate this technique.

4.4.1 HADR Aid Delivery Problem

The Humanitarian Assistance and Disaster Relief (HADR) aid delivery problem that we consider involves two categories of participants:

- N *Aid Provider* nations, each of which has some number S_n of ships with aid (e.g., food, medicine) to be delivered
- a single *Aid Recipient* nation that has P ports to which aid can be delivered

Collectively, these participants need to formulate a schedule for delivering aid on board the Aid Provider ships to ports belonging to the Aid Recipient, ensuring delivery prior to a specified deadline. As summarized in Figure 11, a solution involves assigning to each Aid Provider ship: a port to which its aid should be delivered, a berth at the port, and a docking time. These assignments are subject to various constraints on ship and port characteristics to ensure physical compatibility between the ship and the port/berth, schedule availability of the berth, and the ability for the ship to completing the docking before the assigned deadline. We further seek an assignment that optimizes according to the following load-balancing criteria.

For a **Ship** to be scheduled to dock at a given **Docking Time** at a **Berth** belonging to a **Port**, the following must hold:

Port

- **ship-draft** $\leq$ **port-harbor-depth**
- ship-cargo-amount $\leq$ **port-cargo-offload-capacity**
- **ship-earliest-arrival** $= \frac{\text{EarthDistance}(\textbf{ship-location}, \text{port})}{\textbf{ship-maxspeed}}$
- **ship-earliest-arrival** $\leq$ deadline

Berth in Port

- **ship-length** $\leq$ **berth-length**

Docking Time at Berth

- **ship-earliest-arrival** $\leq$ docking-time $\leq$ deadline
- **berth-availability** (pairs of **berth-occupied-start** and **berth-occupied-end**) for berth at docking-time

Figure 11. Constraints in the Scheduling Problem

Optimization Criteria: Load-balancing across ports. Let $Assigned(\text{Port}_i)$ designate the number of aid provider ships assigned to aid recipient port Port_i. An optimal solution is a set of assignments that minimizes

$$\text{MAX}_{j,k}(\mid Assigned(\text{Port}_j) - Assigned(\text{Port}_k) \mid)$$

Generating a solution to this scheduling problem requires information from the various parties about their assets (ships, ports), some of which they would prefer not to share with other coalition members. Table 2 summarizes this private data. In both Table 2 and Figure 11 the problem data is color-coded, with blue used for private data belonging to an Aid Provider nation and green for private data belonging to the Aid Recipient nation. As the coloring clearly shows, determining a solution requires combining private information from multiple parties, which motivates our use of secure multi-party computation within our scheduling algorithm.

Table 2. Summary of Private Data by Data Owner

Aid Provider	Aid Recipient	
Ship Info	*Port Info*	*Berth Info*
ship-location **ship-length** **ship-draft** **ship-maxspeed**	**port-available** **port-cargo-offload-capacity** **port-harbor-depth**	**berth-length** **berth-occupied-start** **berth-occupied-end**

4.4.2 Scheduling Workflow

Creating a single Multi-Party Computation (MPC) circuit to solve the full optimized scheduling problem in a general way is not practical at this point in time. For this reason, our solution approach consists of a workflow that decomposes the scheduling task into the following sequence of steps.

Step A. Collect relevant inputs:

- Aid Provider: determine ports that can be reached by each ship before the deadline D
- Aid Recipient: select ports to which aid will be accepted

Step B. Determine ports that satisfy joint ship/harbor physical compatibility constraints:

- Aid Provider ship draft is compatible with the harbor depth in the port
- Aid Recipient port has sufficient offload capacity

Step C. Determine all berths within each feasible port from Step B that satisfy joint ship/berth compatibility and berth availability constraints:

- Aid Provider ship fits within the berth
- Aid Recipient berth is available at planned ship arrival time

Step D. Schedule the aid ships across possible ship-port-berth-arrival-time options (from Step C) in accordance with the optimization goal of load-balancing across ports.

Step A is a simple information gathering/filtering task performed locally by individual participants. Step B requires computation over private inputs from both the Aid Provider and the Aid Recipient. Steps C and D combine private inputs with intermediate results from earlier steps. For these reasons, we use three secure multi-party circuits within our workflow:

- A two-party circuit for the physical compatibility test in Step B (Aid Provider and Aid Recipient)
- A two-party circuit for the viability test in Step C (Aid Provider and Aid Recipient)
- An N-party circuit for optimizing berth allocation in Step D (all N Aid Providers)

The circuits for Steps B and C are straightforward but must be run for each possible ship/port (Step B) and ship/berth (Step C) combination. The circuit for Step D is more complex. Because it is not possible to implement a truly optimized solution in the MPC circuit model, we instead opted for the following greedy approach to load balancing across ports.

1. Initialization: set the list of ship-port-berth solutions to be empty and the working set of options to be the set of ship-port-berth-unload time entries from Step C
2. Select a port P with the fewest number of assignments and for which there remain options
3. Select earliest ship-port-berth-unload time entry for P and add to the solutions list
4. Remove from the set of options all entries for the ship and berth selected in Step 3
5. Repeat steps 2-4 until no more ship-port-berth solutions remain
6. Output the solution list

We use the Lumen agent technology (described in [36]) to provide an adaptive workflow capability for executing this scheduling process. Although we depict only one workflow here, more generally our adaptive workflow capability draws from a library of alternative approaches to a range of aid distribution scheduling problems, enabling solution approaches to be matched to the specifics of a given situation. For example, we have defined alternative workflows that embed different scheduling and optimization strategies and that make use of secure multi-party computation in different ways as a means of investigating tradeoffs between efficiency and privacy.

4.4.3 Modeling Workflows

A QIF analysis begins by transforming a program (such as our aid distribution workflow) into a model that represents the relationship between the (private) inputs and outputs. Specifically, these models are based on information-theoretic *channels*, which we use to construct a mapping from prior distributions on private data to distributions on posterior distributions [37], [38].

A ‘prior’ can be thought of as the initial set of beliefs that an adversary has about a system. An adversary may have a prior over the lengths of naval ships (e.g., between 10 and 1,500 feet long), or the possible locations of the ships. If an adversary has no reason to believe one value is more likely than any other then the prior is a uniform distribution over the space of secrets, hence it is known as a *uniform* prior.

We use these models to support "predictive-mode" adaptive workflows in which we reason about what an adversary could learn from any possible set of private inputs, as well as "posterior-mode" in which we determine how much an adversary may learn from some specific concrete result.

4.4.4 Supporting Workflow Adaptation

Our predictive-mode adaptation strategy is based on QIF *predictive leakage*, which attempts to predict how much will be leaked about the private data before the computation is run on the concrete private values. We can also approximate predictive leakage using Monte Carlo simulation, permitting the use of these metrics even in scenarios where operational requirements make it infeasible to complete the precise predictive leakage within a desired time frame. Incorporating predictive leakage metrics into our workflows enables identification of potentially higher-risk situations where it may be preferable to adapt or halt the workflow (based on some policy) rather than participating in a computation that may reveal too much.

Our posterior-mode adaptation strategy is based on QIF *dynamic leakage*, which takes into account the actual results of a computation. As opposed to the predictive leakage's assessments, that average all possible private input values, dynamic leakage enables our workflows to incorporate more accurate assessments of what was actually revealed in the specific ongoing workflow.

In resource-constrained systems, it is useful to be able to predict how much of a given resource would be used if a certain action were to be executed. Private data can similarly be viewed as a form of resource for which predictive analysis can be applied to assess the impact of planned or possible actions.

The ability to predict the future vulnerability can be implemented in several ways, the practicality and efficiency of different methods depends heavily on the constraints of the system, imposed by the state-space and the adversarial model.[6]

In our system we have chosen an approximate method that enables an analyst to calculate a histogram over the possible vulnerability outcomes without any access to the private data. This method uses Monte Carlo simulation to run our analysis over randomly sampled points from the space of possible private data values.

Figure 12 shows the results of a Monte Carlo simulation of the predictive vulnerability of running Steps A+B (using 13788 samples[7]). The vertical dotted line shows the median value ($6.847603e-6\%$) of all the samples. This method of approximation provides analysts with various options. In a scenario where the preservation of privacy is paramount, the analyst may focus on the right-hand side of the figure, where the potential vulnerability is higher, 0.03039514%, though still unlikely.

[6] In the QIF literature, this ability is known as *Static Leakage Analysis* [38], [39].

[7] The sampling procedure is time based, hence the seemingly arbitrary number of samples.

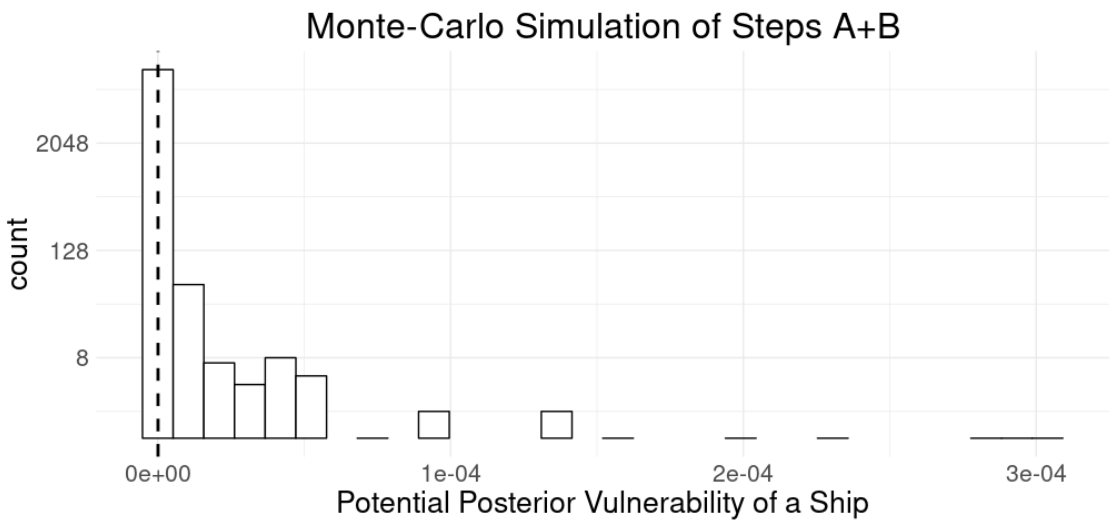


Figure 12. Predictive Vulnerability of Steps A+B

This method also adapts well to use cases where an analyst may have access to *some* of the private data. For example, an analyst for one of the coalition partner nations might have access to that nation's private data, but not the private data of the ports. Using this same method of sampling from the space of (unknown) private data, such an analyst would be able to approximate the future vulnerability of their data if they were to respond to a query.

Figure 13. Posterior Belief over the Position of Ship #9 in the HADR Scenario at Step D

4.4.5 Posterior Vulnerability for Steps A+B

Once a step is taken, we can calculate the *posterior vulnerability* of the private data. Unlike the predictive analysis in the previous section, this analysis is 'exact' in that the real vulnerability cannot be more than what the analysis reports.

Ship #9 Position $6.847603e-6\%$

In this case, the posterior vulnerability coincides with the median of the predictive vulnerability. This is not too surprising as the median was also the most likely value by a significant margin.

4.4.6 Predictive Vulnerability for Step C

Figure 14 shows the Monte Carlo simulation for vulnerability of a ship after Step C is completed.[8] The median predicted value in this instance, $6.644298e-4\%$, is two orders of magnitude higher than the vulnerability after Steps A+B alone. This makes intuitive sense as much more information is revealed after Step C that can be used to infer a ship's private data. Unsurprisingly, the maximum sampled predictive vulnerability is also substantially higher: 0.2012072%.

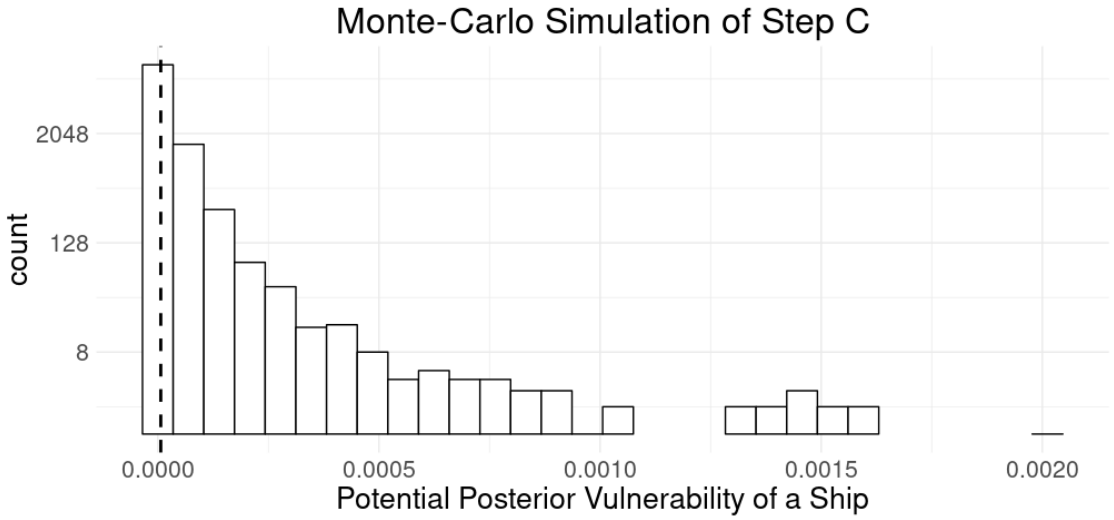


Figure 14. Predictive Vulnerability of Step C

4.4.7 Posterior Vulnerability for Steps C

As with the posterior vulnerability for Steps A+B, the posterior vulnerability for Step C is based on a sound analysis using the real results of the workflow step, and are not simulated as in the predictive vulnerability.

Ship #9 Position $2.049806e-4\%$

[8] Note that the range of the x-axis has changed in order to better display the data.

In the case of Step C, the posterior vulnerability for Ship #9 is *lower* than the median of the predictive result ($6.644298e-4\%$). From an analyst's perspective, this could mean that Ship #9 has revealed even less about its private data than the 'average' ship would in this scenario.

Why Step D doesn't leak

Step D has no meaningful consequences on the vulnerability of the private data for any stakeholder in the HADR scenario *if the result of Step C has been observed by the adversary*. The reason for this is that Step D's algorithm can be computed completely from the results of previous steps in the workflow, i.e. it does not require the values of the private data directly. Interestingly, this point reinforces an important aspect of QIF analysis: even though Step C's result was computed with private data, the vulnerability metrics from the QIF analysis of Step C take into account *any possible use* of the result. Therefore, additional computation over the results of Step C (or any prior step) does not affect the vulnerability of private data.

4.4.8 Privacy-aware Workflow Adaptation

The vulnerability assessment computed by the QIF capability provides insights to data owners as to the security of private information that they wish to protect. We exploit these insights within our workflow manager to adapt the scheduling process in order to ensure adherence to privacy objectives. More specifically, we use the QIF capability in two ways: in a *predictive mode* to estimate the amount of leakage associated with potentially performing a particular query or task and in a *posterior mode* to track actual leakage based on the specific values that a query or task returns.

When executing a particular task our workflow manager invokes the predictive mode to estimate leakage. If the estimate of aggregate leakage for designated private data does not exceed set thresholds, then the workflow proceeds and the posterior leakage analysis is invoked to determine actual leakage values. If the estimate does exceed the threshold then the workflow is either terminated or (if possible) modified via a *remediation strategy* to keep leakage below the threshold.

The idea behind a remediation strategy is to modify the problem or state in ways that will likely reduce impacts on private data. For example, our aid delivery problem requires computing the reachability of a port by a given deadline. Knowing that a given ship can reach the port by that deadline reveals information about the combination of ship position and maximum speed. One simple remediation strategy is to postpone the deadline, which will reveal less information about the position and speed values (e.g., the fact that a ship can reach a port by a given deadline reveals something about the lower bound for its max-speed; a later deadline introduces greater uncertainty as to what that speed might be by decreasing that lower bound). Our Lumen-based adaptive workflow engine includes such remediation strategies to enable adaptivity based on QIF predictive analyses.

Privacy thresholds are implemented using an existing policy framework within Lumen that was developed previously to enable users to impose boundaries on the behaviors of autonomous agents [40]. The privacy policies have the general form:

`Keep below` **`<percentage>`** `the probability of knowing` **`<private-data>`** `within` **`<tolerance>`**

Below we show two examples used in the system, one for the Aid Provider nation and one for the Aid Recipient nation.

Aid Provider Sample Policy:

"Keep below 10 % the probability of knowing the location of my ships within 50 NMs"

Aid Recipient Sample Policy:

"Keep below 20 % the probability of knowing the port harbor depth within 40 feet"

Figure 15 shows sample vulnerability assessments for two types of private data (max-speed, location) for a select set of ships belonging to an individual Aid Provider nation. The top image shows the initial vulnerabilities of the data, prior to performing any computations; the bottom image shows the vulnerabilities after workflow completion. The display shows the QIF-derived vulnerability level as a colored bar representing the adversary's likelihood of guessing the private data within the specified tolerance. The vertical line bisecting the display for each piece of private data marks the policy-prescribed threshold of acceptability for the vulnerability. We note that the initial vulnerabilities for the ship locations are non-zero but so small as to not be perceptible in the image.

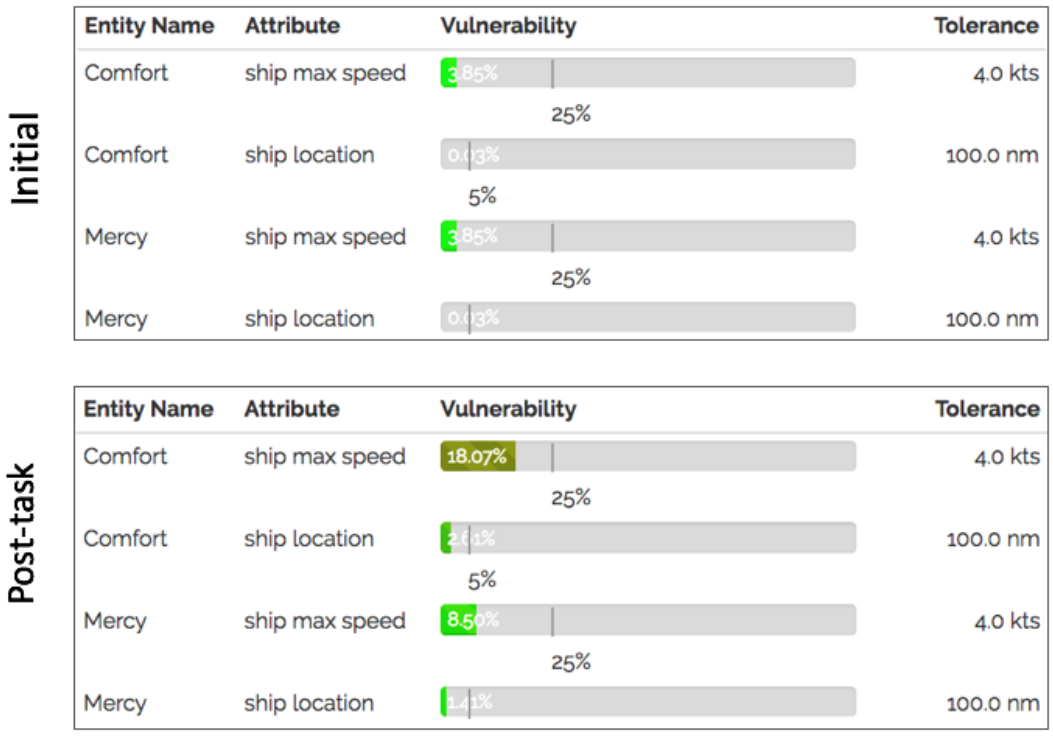


Figure 15. Comparison of Initial (Prior) and Post-Task (Posterior) Vulnerability Assessments

4.5 Programming Language Enforcement of Privacy

4.5.1 LWeb

LWeb extends Yesod (a web-server framework for Haskell programs) with LIO-style IFC enforcement. The implementation has two parts. As a first step, we generalize LIO to support an arbitrary underlying monad by making it a monad transformer, applying it to Yesod's core monad. Then we extend Yesod operations to incorporate label-based policies that work with this extended monad.

LMonad:

LIO as a monad transformer. LMonad generalizes the underlying IO monad of LIO to any monad m. In particular, LMonad is a monad transformer LMonadT l m that adds the IFC operations to the underlying monad m, rather than making it specific to the IO monad.

The new monad has the following operations:

```
label     :: (Label l, Monad m) => l -> a -> LMonadT l m (Labeled l a)
unlabel   :: (Label l, Monad m) => Labeled l a -> LMonadT l m a
runLMonad :: (Label l, Monad m) => LMonadT l m a -> m a
```

The label operation assigns a label to a value. The unlabel operation removes a label, but ensures that the appropriate contextual information is tracked within the current LMonadT scope. runLMonad runs the provided computation, ensuring that all invariants are maintained.

A major component of this result is the full mechanization and verification that this system properly enforces IFC properties. Full details of this mechanization and proof can be found in the relevant paper [41].

4.5.2 A Language for Probabilistically Oblivious Computation

As discussed in Section 3.7 traditional execution of programming languages does not protect against certain kinds of known side-channel attacks. lambda-obliv, is a probabilistic programming language for oblivious computation, where the type system ensures that all valid programs are oblivious. Furthermore, lambda-obliv guarantees that *revelations* (when hidden choices are permitted to be observed by an adversary), do not communicate information to the adversary by ensuring that all revealed values adhere to a uniform distribution. These guarantees are enforced by lambda-obliv's type system, which uses *affine types* for randomly generated values, ensuring that each randomly generated number cannot be freely copied.

The details of this type system and the semantics of lambda-obliv are provided in the relevant paper [42].

4.6 Sparse Vector Technique for DP

The online multiplicative weights algorithm has been presented in [43] by Hardt et al. The algorithm is an instantiation of the Sparse Vector Technique in the interactive setting, where the threshold T is replaced with the difference between the actual answer and the estimation of the

query output computed using the multiplicative weights on the histogram of the database values. As described in Section 3.9, SVT is applied and gets the output of fuzzy values for the queries whose values are above the threshold. This reveals a point whenever the estimation is not accurate enough. The multiplicative weights estimate is then updated and SVT is applied again. This process repeats until all the queries are answered or the cut-off point is reached.

Roth et al. observed that this is a specific instance of a more general class of algorithms called *iterative database constructions* [44]. In the work of Hardt et al., the multiplicative weights algorithm was used to can be swapped estimate database contents based on query results. But this algorithm can be swapped out with any other approach to database approximation and [44] provides a framework for such constructions. We implemented this framework with a database estimation procedure based on the abstraction described in Section 4.8. This enables the use of the sparse vector approach to differentially-private answering of online database queries while scaling beyond two database columns (the general limit for multiplicative weights-based approaches).

4.7 By-hand Analysis of IoT CRT's Virtual Sensors

It is not immediately clear *how well* virtual sensors are able to provide privacy to the occupants and users of the TIPPERS system. Working with the TIPPERS team, TAMBA was able to construct a model of a virtual sensor algorithm. This model is shown in Figure 16. TAMBA then encoded this model into a prob query, in order to determine the quantitative information flow properties of this algorithm.

```
user  = MAC address owner (if known)
ap    = wifi access point
rooms = rooms covered by ap

isOfficePresent = size({() | x <- rooms, x == office_of(user)}) >= 1
nMR             = count({x | x <- rooms, isMeetingRoom(x)})


if(isOfficePresent)
    dc = 0.1 / (|rooms| - 1)
else
    if(nMR > 0)
        dc = 0.2 / (|rooms| - 1)  // should be (|rooms| - nMR)
    else
        dc = 1 / |rooms|

confidenceFor(room,isOfficePresent,rooms):
    case isOfficePresent && nMR > 0:
        Office: confidence = 0.5
        Mtg Rm: confidence = 0.4 / nMR
        else: confidence = dc
    case isOfficePresent && nMR = 0:
        Office: confidence = 0.9
        else: confidence = dc
    case not(isOfficePresent) && nMR > 0:
        Mtg Rm: confidence = 0.8 / nMR
        else: confidence = dc
    case not(isOfficePresent) && nMR = 0:
        all: confidence = dc
```

Figure 16. Occupancy Algorithm

As discussed in Section 3.8, the virtual sensor output is meant to preserve the privacy of the occupants themselves. However, because the output of the algorithm can vary due to the specific individuals in question (i.e. whether they have an office nearby), this algorithm causes some info about those individuals to leak. In particular, by looking at the probability number that results from the virtual presence sensor, users are able to distinguish whether office data was used in the processing of the occupancy data point. Given public side-channel information on which office belongs to which person, users can then infer whose personal location data is being displayed (i.e. de-anonymize this data point). This analysis was presented to the TIPPERS team as a way to inform the design of future virtual sensors.

4.8 Privacy and SQL analysis

The analysis of privacy-related properties of SQL queries was a significant aspect of the TAMBA effort. prob, as described above, has the advantage that the initial size of the data set is a static property of the program being analyzed. SQL, in contrast, *abstracts* these properties out of the program, which required new techniques in order for any metrics to be possible. The resulting tool is db-priv, which uses the probabilistic abstract domains described above as a starting point.

DB-Priv allows an analyst to provide an initial abstract representation of a database, and a sequence of queries; after each query is analyzed, DB-Priv emits a number of statistics summarizing the information leakage and privacy impacts of broadly revealing the result of that query.

As with prob, DB-Priv supports rudimentary belief revision, allowing users to propagate knowledge about a single concrete result into the model, to be used by the predictive analysis for subsequent queries.

The majority of DB-Priv's contributions come in the design of the relational scheme abstract domain and the corresponding abstract interpretation, which allow it to make intelligent trade-offs between the accuracy of its results and the speed with which it provides them. This allows it to stay within the bounds of making sound predictions of privacy impacts in a timely manner.

4.8.1 The Relational Scheme Abstract Domain

The Relational Scheme Abstract Domain used by DB-Priv is considerably more complex than that of prob, due to the high-dimensional properties of databases as compared to program states containing scalar values. Unlike prob, db-priv must also abstract over possible *table-sizes*. Using a naive generalization of the prob abstract domain would result in an exponential blowup of states, making the analysis of SQL intractable.

There are several models of concrete databases, db-priv uses the following:

- A database is a mapping from relation names to relations
- A relation is a bag (multiset) of tuples over a common, statically known schema
- A schema is a mapping from attribute names to attribute domains (set of admissible values)
- A schema is a mapping from attribute names to attribute domains (set of admissible values)

For each of the above, db-priv maintains an *abstract* representation. In each case, an abstract construct is a concise representation of a conservative overapproximation of a set of corresponding concrete entities. This follows from the theoretical definition of abstract interpretation, which defines two functions – the abstraction function $alpha: Concrete \rightarrow Abstract$, and the concretization function $gamma: Abstract \rightarrow Concrete$ such that for any set of concrete databases D (or relation R, tuple T, etc.), $D \subseteq gamma(alpha(D))$.

While any abstraction which satisfies the above property would be an admissible abstraction, db-priv maintained the following representations:

- An abstract database is a mapping from relation name to abstract relation, one for each relation in the underlying concrete database
- An abstract relation is a pair of an abstract tuple (block) map and a data structure maintaining per-block metadata to support each application/operating-mode

- An abstract tuple or block is an instance of DB-Priv's underlying numeric abstract domain. Each dimension of a block corresponds to a column in the respective concrete table, while the domain of that dimension corresponds to that column's own domain, as specified by the table's schema

4.8.2 Per-block Metadata

db-priv supports a number of different applications and modes of operation by maintaining different sets of per-block metadata. The splitting and merging of the individual blocks themselves are common to all applications and governed by the rules of the underlying abstract interpretation. However, the per-block metadata are maintained in accordance with their own rules, which ensure that the result is always a conservative approximation in the appropriate direction.

db-priv maintains the following pieces of per-block metadata, either directly, or indirectly through derivation:

Block Lengths: For each abstract relation, DB-Priv maintains a separate numeric domain where each dimension represents the length of a block in the abstract relation. This allows a db-priv abstract relation to model a set of concrete relations, each comprising tuples drawn from the various abstract blocks, with the number of such tuples limited by the constraints on the size of that block. The use of a relational numeric domain allows the abstract interpretation of filter predicates to interact with block sizes in a predictable way: if a block is split into two sub-blocks on the basis of a predicate over its columns, the sizes of the two sub-blocks are constrained to be between zero and the length of the original block, and to sum up to that length.

Block lengths are an integral part of information leakage analysis; the number of bits of information leaked to an adversary about a secret database is related directly to the number of concrete databases ruled out as being infeasible, on observation of the concrete result of a query. Block lengths allow db-priv to compute an upper bound on the number of such databases.

Block Histories: For each block in an abstract relation, DB-Priv maintains a history of that block, in terms of the queries that the block has participated in. This is critical in order to support a variety of differential privacy applications, where the budget consumption of all tuples modeled by a block is a function of the number of queries the block has participated in.

Block histories also play a role in the block-coalescing optimization described in the next section, which allows DB-Priv to trade accuracy for performance for future queries in a sequence.

Block Weights: For each block in an abstract relation, DB-Priv maintains a weight for that block, describing the number of tuples from that block that may be expected to exist in the concrete relation modeled by that block, as a fraction of the size of the block. For most real-world applications, the number of tuples in a relation is extremely small compared to the number of tuples that satisfy the schema of that relation; therefore, the weight is generally an extremely small number.

Maintaining block weights allows DB-Priv to report an expected number of rows in the result of a query; in addition, an analyst may communicate a revised estimate for this quantity, which forces DB-Priv to revise the weights for all blocks in the result of that query.

4.8.3 Numeric Abstract Domains in db-priv

The leaf-level constructs in all of db-priv's representations are numeric abstract domains, a placeholder term for any structure capable of representing approximations of sets of numbers. Following the approach from prob, db-priv uses the polyhedral abstract domain provided by the ELINA library as the numeric domain underlying all of its representations.

ELINA is a high-performance library capable of representing polyhedra in multidimensional spaces, with a number of optimizations that prove useful in db-priv's performance profile.

db-priv's use of ELINA (or polyhedral domains more generally) is largely an implementation detail; any numeric domain may be used, bringing with it the corresponding accuracy/performance trade-off.

4.8.4 Input and Outputs

There are two kinds of inputs to any invocation of db-priv: the abstract database description, and a query sequence

An Abstract Database Description is a textual description of an initial abstract database, as introduced in the previous section. db-priv accepts on the command-line a path to such a file in JSON format, containing a single abstract database, each containing a number of relations, and in turn, a block map and metadata segments. The schema of the abstract database need not be specified explicitly; it is inferred from the database description and must be consistent across all blocks in a given relation. Simple examples of abstract database descriptions may be found in db-priv's examples directory with extension "adb.json."

A Query Sequence is a sequence of SQL statements, one per line delimited by semicolons, fed to db-priv through either a file up-front, or incrementally by the analyst. db-priv currently supports a restricted dialect of SQL consisting of selections (filters) over the rows of a table, projections of all attributes, or (equivalently) of an aggregate function (COUNT, SUM, MIN, MAX, AVG). Any query not conforming to this dialect will be rejected with an appropriate error message describing the problem.

db-priv processes queries one at-a-time, starting with an initial model ingested from the provided abstract database description, and modifying it over the course of abstract interpretation. After each query has been evaluated, the following statistics are output:

- The resulting blocking structure, an abstract database which is semantically equivalent to the input, but with the additional property that the result of the query just processed consists of a whole number of blocks in this new representation. This blocking structure is arrived at through abstract interpretation, ensuring that query predicates that intersect with an existing block generate multiple blocks in the output representation

- As the blocks in the output representation may be derived from a variety of decomposition operations of previously existing blocks, and only a subset of those blocks are contained in the output of the query, db-priv outputs a comprehensive block history summary, detailing a block-by-block enumeration of the queries they have participated in. The size of the history for a given block therefore may be treated as the budget consumption for any tuple in that block

- To assist with identifying possible difference attacks, db-priv outputs maximum impact blocks: blocks resulting from splits over all queries to-date, which are the smallest among all blocks, regardless of their participation in the results of any query or set of queries

- For each block, db-priv outputs the size of that block in the tuple-space; in addition, db-priv also outputs a weighted size of each block, as well as the weighted sum of all blocks in the result of the given query, denoting the expected number of result rows

4.8.5 Interactive Weight Revision

When run in interactive mode, db-priv pauses after each query is evaluated, allowing the analyst to observe the generated statistics. In particular, db-priv allows the analyst to input into the system a revised expected result count, to enable db-priv to better model the information gained by an adversary when observing the result of the query.

If a revised expected result count is provided, db-priv incorporates the information into its model by revising the weights of all blocks in the result proportional to their respective sizes. The analysis of subsequent queries in the sequence will use these revised weights to deliver more accurate estimates of result sizes.

4.8.6 Performance and Optimizations

db-priv relies on a number of optimization techniques to achieve an acceptable level of performance, while delivering results accurate enough for regular use.

The use of convex polyhedral abstract domains is itself a design decision rooted in performance considerations; convex polyhedra explicitly trade improved performance of operations such as intersections and enumerations, for an accuracy loss in disjunctive convex hull operations. In particular, db-priv makes use of the ELINA library's decomposition techniques to ensure that computational complexity is tied more closely to the complexity of the constraints of a polyhedron, rather than its dimensionality.

One of the drawbacks of using a single convex polyhedron to represent a region of the tuplespace is a loss of accuracy from convex hull operations during intermediate steps of an abstract interpretation. db-priv can optionally use a precise-splitting technique to alleviate this concern by representing non-convex regions as multiple independent convex polyhedra, removing this source of inaccuracy for select operations.

db-priv employs an optional block-coalescing mode, where the accuracy of maintaining a large number of granular blocks may be explicitly traded off for the benefit of maintaining fewer blocks' worth of metadata. When activated, DB-priv enumerates a list of geometrically adjacent

pairs of blocks, and coalesces the pair nearest to each other in a chosen metric space. Available metrics are the weight and budget metrics, as described above. The analyst may also specify a threshold which db-priv must reach by iteratively coalescing blocks.

4.9 Bayesian Evaluations of DP Systems

One of our initial investigations was into the performance of factored inference on a time-series person localization problem based on the UCI dataset. As previously described, this approach employed a simple Markov model assuming independence across individuals. Unsurprisingly, person localization was not possible while employing this weaker prior.

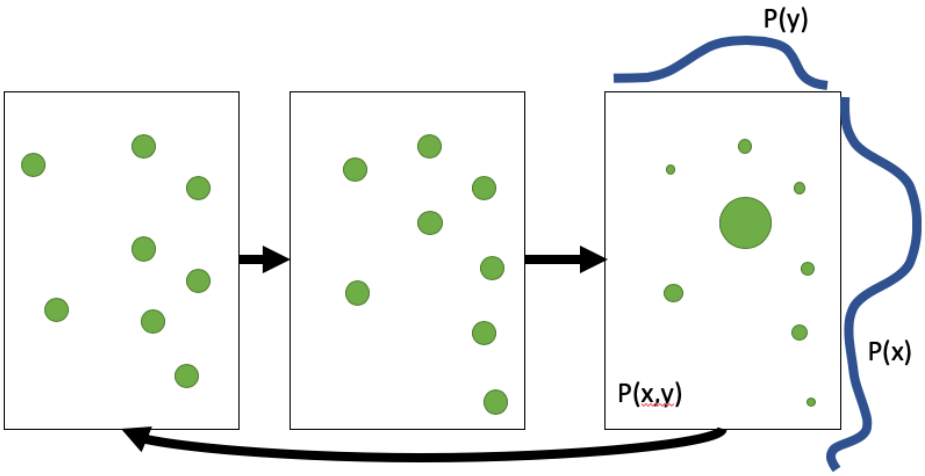


Figure 17. A Conceptual Depiction of Factored Particle Filtering (FPF)

Figure 18 shows an example of the "guessing" probability with Factored Particle Filtering (FPF) inference applied to a Laplace mechanism on the UCI dataset. This plot shows that guesses rarely exceed random chance in their accuracy.

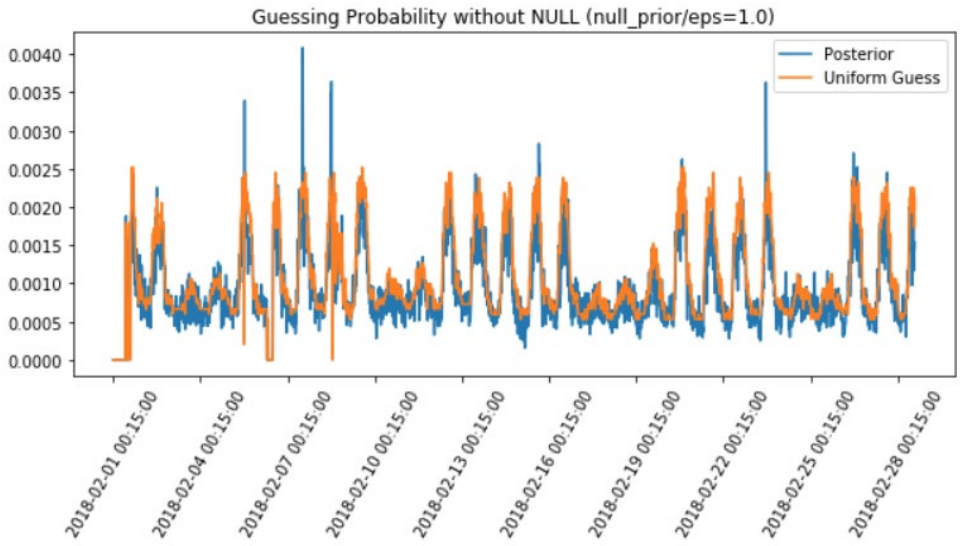


Figure 18. Example of the "Guessing" Probability with Factored Particle Filtering

To accomplish tractable posterior evaluation, we developed several novel extensions of particle filtering. This includes an implementation of what we call Stochastic Factored Particle Filtering as well as the design of Structure Inference Factored Particle Filtering. These algorithms enable accurate evaluation of the high-dimensional posteriors required in time-series estimation by enabling the specification of, and in the future, the learning of, the independence relationships between the many variables required.

4.9.1 Deep Learning Approaches to Prior Construction

To construct a structured prior for person localization, we augmented the variational autoencoder, a standard architecture used in deep generative modeling, to work with sequential data. The input data consists of batched vectors of time series distributions over room presence. The architecture first encodes the batched data to an initial latent state z_0 by outputting parameters of a multivariate Gaussian and then sampling. Then, a recurrent neural network constructs sequences of latent states $z_1, \ldots, z_t$ by constructing new parameters of a multivariate Gaussian from the previous latent state and then sampling. The decoder architecture is trained to construct the observed data from the latent states, transforming the sequence $z_0, \ldots, z_t$ into the observed sequence of categorical distributions over room presence $x_0, \ldots, x_t$ by applying a final softmax layer.

A high-level process overview is shown in Figure 19, which shows the application of the recurrent autoencoder architecture. An encoder converts batches of shape (num_people, time, room) data into a latent state. A transition model transitions the latent state over subsequent time steps. A decoder model reconstructs the observed data from each latent state sequentially.

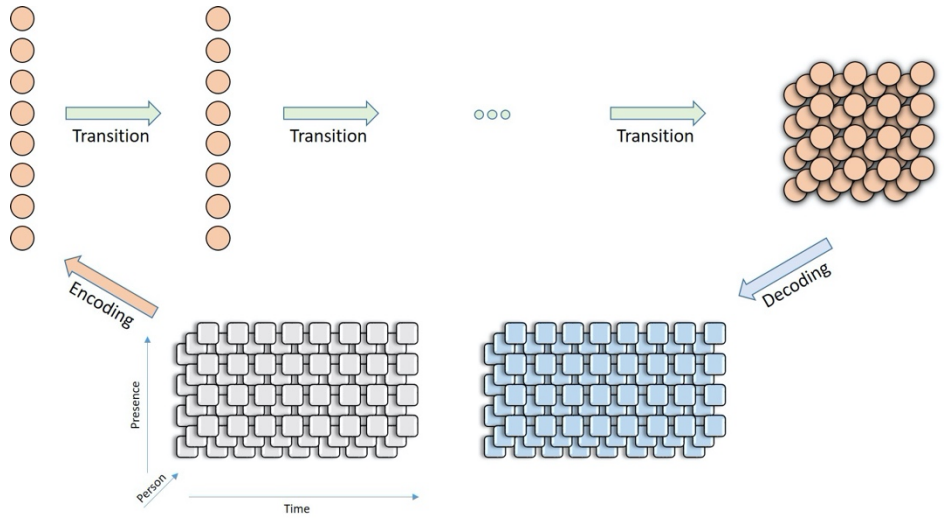


Figure 19. Application of the Recurrent Autoencoder Architecture

At a low level, the encoder consists of a sequence of convolution, max pool, and dense layers. The transition model consists of a similar mixture, with the addition of residual connections and permutation invariant operations. The transition architecture is motivated by the concept of modeling correlations between activity patterns in room transitions. Intuitively, for the number of unique individuals present in the dataset, we expect that certain activity patterns are correlated across groups of individuals. To represent this assumption, the permutation invariant layers

compute batch local "global states" which are designed to keep track of correlation information (see Figure 20). At each step of the transition model, we first sample from a multivariate Gaussian distribution with parameters output from the previous step. After a number of transitions equal to time steps in the observed data, each step data is transformed by the decoder into a categorical distribution over room presence using convolution and softmax.

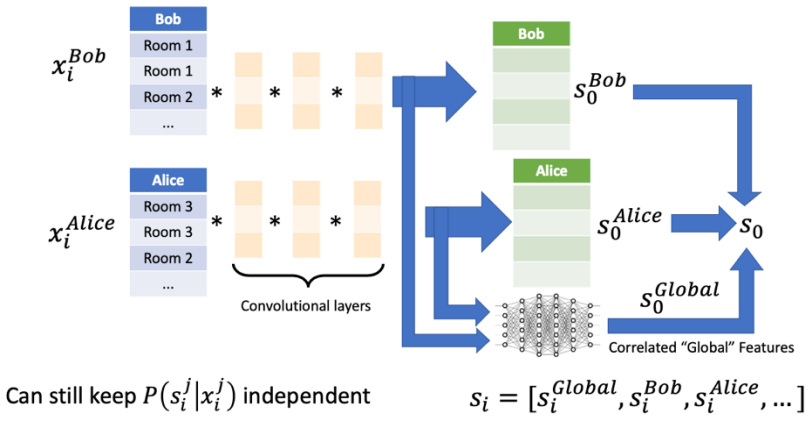


Figure 20. Bottlenecking Cross-Individual Correlation via Global Features

To train the model to capture the correlation information, we performed data preprocessing to identify seasonality. A time series plot of a smoothed version of the data is shown in Figure 21. Figure 21 shows a time series plot of total presence in building in 3-hour increments. The data presents clear seasonality fluctuations between weekday and weekend, as expected. Of the 19056 entities observed in the building, the maximum present in the building at any 3-hour window is approximately 850. This is one possible indicator of low signal-to-noise ratio.

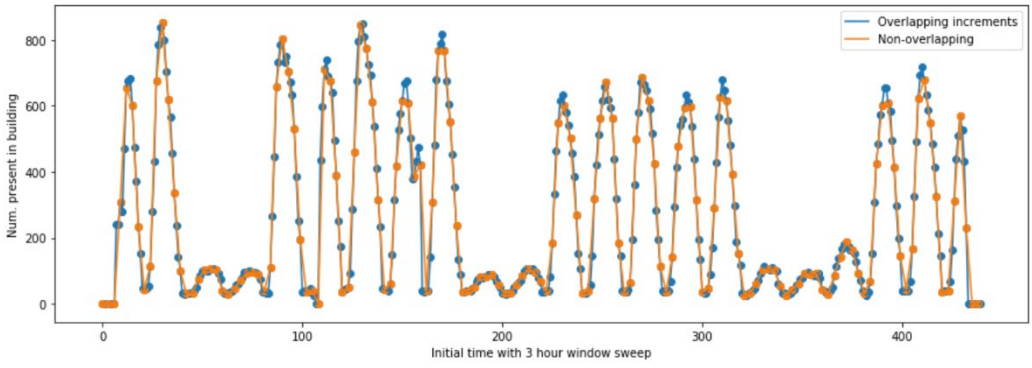


Figure 21. Time Series Plot of Total Presence in Building

Additionally, we constructed a similarity matrix to use as a sampling prior over people who are present in the same rooms at the same time. The structure of a subset of this matrix is show in Figure 22. Specifically, Figure 22 show the scaled similarity matrix between the top 1000 most present entities in the building. Diagonal block structure indicates strong intergroup correlation.

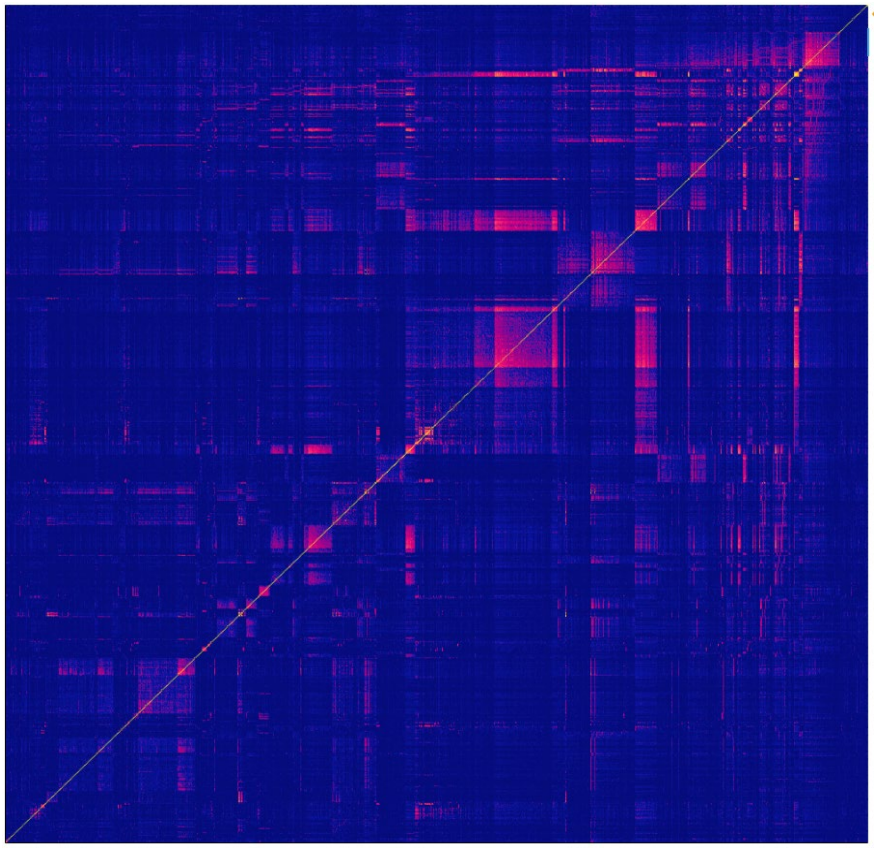


Figure 22. Scaled Similarity Matrix

Employing a heuristic for group selecting in training is critical, as it is not possible to fit all individuals into the training set at one time. One solution would be to sample random subsets of individuals, but this would be unlikely to contain multiple individuals with correlated behavior. Instead, we use a heuristic that individuals spending significant time in the same room are likely to exhibit correlated behavior (a subtly distinct notion). The rows and columns of the similarity matrix represent similarity in presence pattern between people (who index the matrix). From this similarity matrix, we biased the sampling process to fill each batch with people who are similar to an initial seed person. To draw the group, we sample the seed, and then draw people without replacement from a normalized categorical distribution constructed by applying softmax to the rows of the similarity matrix.

To train the model, we deployed a version of gradient descent called ADAM on the variational objective, shown below

$$\text{Loss} = E[\log P(X|y)] - \frac{1}{2}$$

The variational objective consists of a reconstruction term and a likelihood term. The likelihood term encourages the distribution of the latent state (which is a multivariate Gaussian with mean μ and covariance matrix Σ constructed by the encoder) to be close to a multivariate normal distribution. In this way, the likelihood term acts as regularizer for the distribution parameters learned by the encoder. The reconstruction term (the expectation) is the cross entropy between

the categorical distribution predicted by the model and the data. After training, the model represents a structural prior on room presence and, correspondingly, person localization within the building.

By the conclusion of this line of research, we had constructed a training pipeline for this architecture, as well as completed preliminary tests on the results of sampling with a similarity bias. However, we were not able to prevent the network from overfitting to transitions out of the building. To investigate this phenomenon, we trained a simple Markov model on batches of size 64, for time increments beginning at 12 PM and ending at 12 AM. The log transition matrix learned by a model trained on repeated samples from a single day, as well as random samples across all days (global), are shown below in Figure 23. The column index represents transitions from a room, the row index represents transitions to a row. The top row is out of the building.

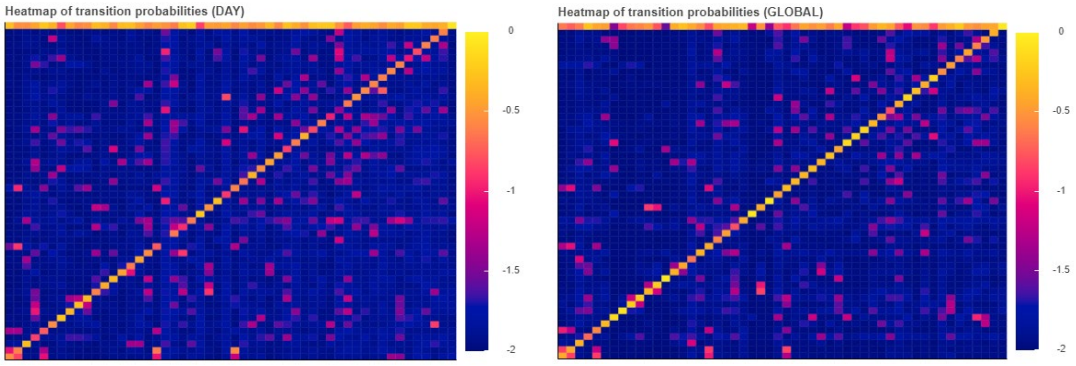


Figure 23. Transition Matrix with Log Probabilities

This simple structural prior approximates the empirical data transition matrix. These results indicate that a large portion of transition probability typically clusters around transitions out of the building. Accounting for this bias in training would be a critical component of future work for constructing strong prior models in this domain.

We believe further investigation on the impact of strong priors in privacy attacks is merited. While the construction of strong priors and the application of them in inference are challenging problems, we believe partial solutions to these problems are or will soon be within the grasp of sophisticated adversaries. Given the operational significance of future DP system deployment, further investigation is justified in this area.

In practice, we believe approaches like the ones we've developed are reasonable benchmarks for DP system deployment. Unfortunately, without external assumptions, it is difficult, if not impossible, to prove the impossibility of inference against an arbitrarily strong prior. A reasonable pragmatic measure would be to employ local sensitive data in the construction of a strong prior. If privacy protection is maintained empirically then that is reasonable assurance that the system can be deployed, as adversaries would need to develop priors superior to those trained on data in the absence of any data. Note that this does not ensure that attacks are impossible – strong, hand built prior models built on domain expertise are always a possible threat.

4.10 Equilibrium in Privacy Economics

Consider a marketplace for privacy, as described in Section 3.10. One might naively expect that as the level of privacy protection is increased, the value to the data consumer goes down, the value to the individual the data describes goes up, and the natural price of the data would decrease. However, while these effects may dominate initially, when we consider how different participants change their behavior as privacy levels are altered, some surprising effects emerge once the system reaches equilibrium.

We studied these equilibrium models and described some of these unexpected outcomes [45]. For example, when the privacy level increases, this can actually result in an increase in value of the data to an advertiser as well as leading to an increase in advertiser profit. In general, the relationship between these values can be non-monotonic and even discontinuous in the input privacy level.

4.11 Differential Privacy and Privacy Budgets

Differentially privacy continues to gain traction in real-world deployments involving collection and analysis of sensitive personal information. Yet many of these deployments fall outside formally-studied bounds. For example, industry deployments of differential privacy commonly run algorithms designed for single-use multiple times as data changes over time. We performed work to bridge these gaps between theory and practice, developing techniques that allow for more efficient accounting of privacy loss in real-world situations. We published 35 papers on advanced privacy algorithms, analysis, and applications over the course of the Brandeis program, but there are three capabilities in particular that we want to highlight here.

First, we developed a technique that allows differential privacy to be applied in a manner that targets an accuracy goal instead of a privacy goal. While privacy theorems generally provide an analytic characterization of the privacy / accuracy tradeoff that could be used to convert from an accuracy goal to a privacy goal, these theorems are over-approximate, and only provide bounds on the accuracy. The actual accuracy may vary widely. Multiple instances of the algorithm can be run at different epsilon values to determine which gets closest to the accuracy goal, but this incurs privacy cost linear in the number of times the algorithm is run. We devised a method for using correlated noise to perform this search with privacy cost logarithmic in the number of epsilon values searched [46]. This makes the problem of targeting an accuracy goal feasible with minimal privacy overhead.

This makes solving the common problem of privately tracking evolving aggregate statistics practical within reasonable privacy bounds.

Second, we developed a technique for tracking slowly-evolving statistics. As an example, consider the set of "most popular websites". This will change over time but will be relatively stable day-to-day. Such questions can be answered privately using well-known techniques but privacy cost accumulates linearly in the number of times the “top websites” query is executed. If this query is run daily, then the privacy budget is quickly exhausted. Instead, we can monitor the results of the query and devised an update a release mechanism that only incurs privacy cost when the statistic *changes*. Since changes are assumed to be rare, this significantly decreases

privacy cost over a given period of time. Additionally, this mechanism works in the *local model*, which does not require a trusted party to aggregate data [47].

Third, we showed how heuristics can be used to accelerate privacy-preserving algorithms in much the same manner that heuristics are commonly used to accelerate constraint solving and machine learning [48]. In fact, the existence of effective heuristics is a key factor in the success of those fields since many of the problems faced are NP-hard. This work brings the benefit of these heuristics to privacy-preserving algorithms in a principled manner.

4.12 Modern Data Reconstruction and Other Attacks

Any development of privacy policies, privacy analysis, or privacy-preserving technology should take into consideration the various forms of attacks against privacy. Knowledge of possible attacks not only provides a mechanism for testing tools and technologies, but can also *inform* the development of new technologies. For example, the discovery of side-channel attacks inspired the development of the theory oblivious computation (discussed in Section 3.7) and its applications. In this section we provide an overview of modern data reconstruction attacks and other forms of attacks on privacy so that the knowledge of these attacks similarly informs new techniques and technologies.

Dinur and Nissim [49] give a theoretical discussion on the trade-off between privacy and utility. The authors represent databases as bit-strings which respond to subset-sum queries. They define a notion, called perturbation, which is the absolute error from the actual answer to the noisy answer of a mechanism. In order for a database to be private, the authors find that the total perturbation must be at least on the order of the square root of the number of bits in the database. This result shows that an exact mechanism is not private, and that privacy should be ensured by perturbing answers to queries.

Song, Ristenpart, and Shmatikov [50] describe adversarial algorithms to train machine learning (ML) models which can covertly exfiltrate training data by embedding the training data in the generated model. In their attack model, the adversary will hand an ML training algorithm to the data owner. While the trained model will still perform, the adversary can extract the exfiltrated data from the model by consuming the model either as a white-box, in which the adversary extracts the training data by observing the specially-selected model parameters, or as a black-box, in which the adversary extracts training data by running the model on various inputs. As a result, it is ill-advised to run untrusted algorithms on private data.

Yeom et. al. [51] also explore attacks on recovering training data from trained machine learning models. Unlike with Song, Ristenpart, and Shmatikov [50], however, Yeom et. al. focus on extracting training data from models which were trained using benign algorithms. They find that overfitting a model is one way to leak the training set, but it is still possible to leak information even if no overfitting occurs. Entities which train machine learning models should know that private data ‘only being used to train a machine learning model’ is insufficient to ensure privacy.

Mironov [52] explores ways in which the implementation of differential privacy algorithms can be insecure due to floating point error. In particular, because methods to generate random floating-point numbers tend not to be cryptographically secure, and because sampling from

floating point distributions is approximate, the least-significant-bit of noise-infused floating point outputs can indicate that certain private inputs were infeasible, resulting in a loss of privacy. Differential privacy implementers should be wary of the fact that floating point numbers merely approximate real numbers, and that real-valued formulas should not be implemented verbatim, especially in a cryptographic context.

Calandrino et. al. [53] consider the privacy implications of recommender systems. Amazon's product recommendation system is an example of such a system. It consumes information about which products users are buying, and outputs suggestions for what they should buy next. The authors find that, despite intuitions to the contrary, this system may reveal private information about individuals. Observing only the public recommendations made by the system is sufficient to recover private information. As a result, it is important to consider the information leakage of any mechanism that consumes private data. The fact that a mechanism only releases aggregate information isn't sufficient to ensure that the mechanism is secure.

Narayanan and Shmatikov [54] describe a de-anonymization attack on the Netflix recommendation dataset by combining Netflix data with IMDB data (Internet Movie Data-Base). By finding IMDB users that had rated movies similarly to a Netflix user, the authors would be able to conclude that the accounts likely corresponded to the same individual. Their work was one of the early examples to highlight that removing PII (Personally Identifiable Information) from a dataset is insufficient to protect the privacy of users. Furthermore, this example demonstrates that it is likely infeasible to build an accurate prior distribution which models an adversary's knowledge. As a result, it is desirable to only rely on techniques which are secure regardless of an adversary's knowledge (such as differential privacy).

Hayes et. al [55] describe methods of determining whether a sample was part of a set used to train a generative model. They describe both black-box (which run the model without observing its internals) and white-box attacks (which have access to internal parameters of the model). In general, the attack works because models tend to over-fit to the data they're trained on. This enables attackers to discover the training set by searching for places where the model is over-fitted. Because generative model training algorithms, on their own, do not hide the training data, it would be desirable to explore external privacy protection techniques to use to either pre-process the training data or post-process the trained model, in order to assure the privacy of the training set.

Song and Shmatikov [56] describe another attack which can be used to identify data used to train a machine learning model. They discuss the phenomenon of *overlearning* in which a model may be able to detect features aside from those that the model's creator intended it to detect. By observing the model's ability to detect ancillary features, an attacker can learn about what data the model was trained on. To avoid these issues, when training a model, it's best to employ privacy-preserving technologies in conjunction with the model training algorithms.

4.13 User-centered privacy studies

4.13.1 Virginia Task Force

In an effort to better understand the utility/privacy trade-off from the perspective of all stakeholders, the TAMBA team met with the Virginia Task Force (VTF). The VTF is a group trained in acting as first-responders to domestic and international disaster response and search and rescue efforts. The meeting was organized with the Mobile CRT in order to better understand what information would be useful for first-responders in the event of a disaster taking place in a civilian area (many of whom carry mobile phones). This information was used to inform the notion of utility in further user-centered privacy studies.

4.13.2 TIPPERS

The Charles River Analytics team performed user studies related to the perception of risk vs. benefit for the use of IoT systems. These user studies were in the context of the UC Irvine TIPPERS system. The results of these user studies are divided along several axes: data type, precision, benefits, risks, perceived trade-offs.

Dataype: Utilities (usage, settings and state).

The tracking of building utilities is a major application area of IoT systems. The CRA study was concerned with two levels of precision for utility tracking: Room-level (individual offices) and area-level (labs, entire floors, common spaces). Users were able to reason about a significant risk/benefit trade-off when confronted with these two levels of precision. Any monitoring of room-level utility use is likely to have privacy implications for individuals, particularly in the case of single-occupant rooms (such as faculty offices). However, uncoupling utility use from people tracking means that attribution will have more errors. The user conclusions stated that while area-based tracking can still reveal individual or group 'patterns of life', the data allows for strong optimization of utility usage, which provides significant benefits, and that the tracking of room-level utility usage would need to consider individual privacy preferences.

Dataype: Locations of people.

The tracking of people was also split into two levels of accuracy for the purpose of the user study: individual tracking and group tracking. The user's viewed location data on individuals as 'clearly private', citing potential leaks of schedules and activities that would infringe on personal privacy. However, the users also noted that group tracking *if combined with room level utility tracking* would also allow for the same inference of individual data. The user-proposed solution was to allow for an opt-in system that allows individuals to determine what their location data is used for (personal climate control, for instance). Differential privacy was cited as a possible technical solution for the tracking of groups.

When combining utility and location tracking a few conclusions were reached:

- Sharing area location may not be enough information to infer what service is used, decreasing the benefit of utility tracking

- The tracking of room-level may create some risk in tracking how individuals are using services, but can be coupled with differential privacy to reduce that risk

4.14 Mental Models for End-to-End Encrypted Communications

We collected valuable user feedback from our in-person study about non-expert users' understanding of end-to-end encryption, and describe the details in a paper [24] to appear in the 2020 European Workshop on Usable Security (EuroUSEC). The outcomes of this work include the recommendations summarized as follows:

- Participants place the most importance on understanding the level of confidentiality.

- Detailing risks that distinguish end-to-end encryption from point-to-point encryption helps users understand the relevant threat models.

- Adding technical details to descriptions introduced a strong risk of misunderstandings.

- Calling attention to what end-to-end encryption cannot protect against helps users better understand the protections that exist.

- It may not be worth it to explain integrity and authenticity: these are nuanced concepts and users consider them less important.

- Simple explanations free of jargon are best.

We also designed and conducted an online user study [25] with 357 participants to analyze how a messaging tool's description of its encryption features impacted participant perceptions. We found that describing a messaging tool as "encrypted" or "military-grade encrypted" – as opposed to "secure" – resulted in participants perceiving the tool as more appropriate for sending information that is sensitive. However, describing the same messaging tool as "end-to-end encrypted" did not show any effects. Further, we found that participants saw messaging tools that supported encryption by default more secure against adversaries than tools with encryption that they need to turn on. We also observed some association between perceptions of paranoia relating to secure communication tools and specific phrasings of encryption (e.g., participants were more willing to think that users of a "military-grade encrypted" messaging tool were paranoid). Finally, we see that users' own psychological paranoia levels affect how secure they think privacy-sensitive communication tools are against adversaries, how much utility for privacy they provide, and how paranoid they think using such tools are.

Based on this work, we recommend that designers of secure communication tools turn ubiquitous security features on-by-default, and carefully chose how they describe their tools as to not be vague but also not seem overly technical and mysterious.

4.15 Publications

Below is the list of relevant papers and other publications from the TAMBA team supported by the Brandeis program.

- **Metric-Free Individual Fairness in Online Learning**. Bechavod, Y., Jung, C., Wu, Z.S., *arXiv preprint arXiv:2002.05474 2020*
- **Causal Feature Discovery through Strategic Modification**. Bechavod, Y., Ligett, K., Wu, Z.S., Ziani, J., *arXiv preprint arXiv:2002.07024 2020*
- **Guaranteed Validity for Empirical Approaches to Adaptive Data Analysis**. Rogers, R., Roth, A., Smith, A., Srebro, N., Thakkar, O., Woodworth, B., *Manuscript 2019*
- **A New Analysis of Differential Privacy's Generalization Guarantees**. Jung, C., Ligett, K., Neel, S., Roth, A., Sharifi-Malvajerdi, S., Shenfeld, M., *ITCS 2019*
- **Exponential Separations in Local Differential Privacy**. Joseph, M., Mao, J., Roth, A., *SODA 2019*
- **Differentially Private Objective Perturbation: Beyond Smoothness and Convexity**. Neel, S., Roth, A., Vietri, G., Wu, Z.S., *Manuscript 2019*
- **Equal Opportunity in Online Classification with Partial Feedback**. Bechavod, Y., Ligett, K., Roth, A., Waggoner, B., Wu, Z.S., *NeurIPS2019*
- **Average Individual Fairness: Algorithms, Generalization and Experiments**. Kearns, M., Roth, A., Sharifi-Malvajerdi, S., *NeruIPS (Oral Presentation) 2019*
- **Privately Learning Thresholds: Closing the Exponential Gap**. Kaplan, H., Ligett, K., Mansour, Y., Naor, M., Stemmer, U., *COLT 2020*
- **A Necessary and Sufficient Stability Notion for Adaptive Generalization**. Ligett, K., Shenfeld, M., *NeurIPS 2019*
- **The Role of Interactivity in Local Differential Privacy**. Joseph, M., Mao, J., Neel, S., Roth, A., *FOCS 2019*
- **How to Use Heuristics for Differential Privacy**. Neel, S., Roth, A., Wu, Z.S., *FOCS 2018*
- **Differentially Private Fair Learning**. Jagielski, M., Kearns, M., Mao, J., Oprea, A., Roth, A., Sharifi-Malvajerdi, S., Ullman, J., *ICML 2018*
- **Access to Population-Level Signaling as a Source of Inequality**. Immorlica, N., Ligett, K., Ziani, J., *FAT* 2019*
- **Fair Algorithms for Learning in Allocation Problems**. Elzayn, H., Jabbari, S., Jung, C., Kearns, M., Neel, S., Roth, A., Schutzman, Z., *FAT* 2019*

- **The Downstream Effects of Affirmative Action**. Kannan, S., Roth, A., Ziani, J., *FAT* 2019*
- **An Empirical Study of Rich Subgroup Fairness for Machine Learning**. Kearns, M., Neel, S., Roth, A., Wu, S., *FAT* 2019*
- **Local Differential Privacy for Evolving Data**. Joseph, M., Roth, A., Ullman, J., Waggoner, B., *NIPS 2018*
- **Online Learning with an Unknown Fairness Metric**. Gillen, S., Jung, C., Kearns, M., Roth, A., *NIPS 2018*
- **A Smoothed Analysis of the Greedy Algorithm for the Linear Contextual Bandit Problem**. Kannan, S., Morgenstern, J., Roth, A., Waggoner, B., Wu, S., *NIPS 2018*
- **Mitigating Bias in Adaptive Data Gathering via Differential Privacy**. Neel, S., Roth, A., *ICML 2018*
- **Fair Algorithms for Infinite and Contextual Bandits**. Joseph, M., Kearns, M., Morgenstern, J., Neel, S., Roth, A., *AIES 2018*
- **Accuracy First: Selecting a Differential Privacy Level for Accuracy Constrained ERM**. Ligett, K., Neel, S., Waggoner, B., Wu, S., *NIPS 2017*
- **Meritocratic Fairness for Cross-Population Selection**. Kearns, M., Roth, A., Wu, S., *ICML 2017*
- **Fairness in Reinforcement Learning**. Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., Roth, A., *ICML 2017*
- **Max-Information, Differential Privacy, and Post-Selection Hypothesis Testing**. Rogers, R., Roth, A., Smith, A., Thakkar, O., *FOCS 2016*
- **Putting Peer Prediction Under the Micro(economic)scope and Making Truthtelling Focal**. Kong, Y., Ligett, K., Schoenebeck, G., *WINE 2016*
- **Computer Aided Verification in Mechanism Design**. Barthe, G., Gaboardi, M., Arias, E.J.G., Hsu, J., Roth, A., Strub, P., *WINE 2016*
- **Fairness in Learning: Classic and Contextual Bandits**. Joseph, M., Kearns, M., Morgenstern, J., Roth, A., *NIPS 2016*
- **Learning from Rational Behavior: Predicting Solutions to Unknown Linear Programs**. Jabbari, S., Rogers, R., Roth, A., Wu, Z.S., *NIPS 2016*
- **Privacy Odometers and Filters: Pay as you Go Composition**. Rogers, R., Roth, A., Ullman, J., Vadhan, S., *NIPS 2016*

- **Private Algorithms for the Protected in Social Network Search**. Kearns, M., Roth, A., Wu, Z.S., Yaroslavtsev, G., *Proceedings of the National Academy of Sciences (PNAS) 113 2016*

- **Watch and Learn: Optimizing from Revealed Preferences Feedback**. Roth, A., Ullman, J., Wu, Z.S., *STOC 2016*

- **Do Prices Coordinate Markets?**. Hsu, J., Morgenstern, J., Rogers, R., Roth, A., Vohra, R., *STOC 2016*

- **Privacy-aware Adaptive Scheduling for Coalition Operations**. Myers, K.L., Lee, T., Tam, L., Calderon Trilla, J.M., Davis, B., Magill, S., *SPARK 2019*

- **LWeb: Information Flow Security for Multi-Tier Web Applications**. Parker, J., Vazou, N., Hicks, M., *POPL 2019*

- **Evaluating Design Tradeoffs in Numeric Static Analysis for Java**. Wei, S., Mardziel, P., Ruef, A., Foster, J.S., Hicks, M., *ESOP 2018*

- **Evaluating Fuzz Testing**. Klees, G.T., Ruef, A., Cooper, B., Wei, S., Hicks, M., *CCS 2018*

- **What's the Over/Under? Probabilistic Bounds on Information Leakage**. Sweet, I., Calderon Trilla, J.M., Scherrer, C., Hicks, M., Magill, S., *POST 2018*

- **Decomposition Instead of Self-Composition for Proving the Absence of Timing Channels**. Antonopoulos, T., Gazzillo, P., Hicks, M., Koskinen, E., Terauchi, T., Wei, S., *PLDI 2017*

- **Laziness Boxes You In: Inferring Information Leakage via Forced Thunks**. Calderon Trilla, J.M., Magill, S., *OBS (POPL) 2017*

- **Privacy Technologies for Controlled Information Sharing in Coalition Operations**. Myers, K., Ellis, T., Lepoint, T., Moore, R., Archer, D., Denker, G., Lu, S., Magill, S., Ostrovsky, R., *KSCO 2017*

- **Quantifying Vulnerability of Secret Generation using Hyper-Distributions**. Alvim, M.S., Mardziel, P., Hicks, M., *POST 2017*

- **A Counterexample-guided Approach to Finding Numerical Invariants**. Nguyen, T., Antopoulos, T., Ruef, A., Hicks, M., *Technical Report 2016*

- **Verifying Typeclasses in Liquid Haskell**. Liu, Y., Parker, J., Hicks, M., Vazou, N., *In submission POPL 2021*

- **A Language for Probabilistically Oblivious Computation**. Darais, D., Sweet, I., Liu, C., Hicks, M., *POPL 2020*

- **Coverage Guided Property Based Testing**. Lampropoulos, L., Hicks, M., Pierce, B.C., *OOPSLA 2019*

- **Learning to Prune: Speeding Up Repeated Computations**. Alabi, D., Kalai, A.T., Ligett, K., Musco, C., Tzamos, C., Vitercik, E., *COLT 2019*

- **Adaptive Learning with Robust Generalization Guarantees**. Cummings, R., Ligett, K., Nissim, K., Roth, A., Wu, Z.S., *COLT 2016*

- **Secrecy, Flagging, and Paranoia Revisited: User Attitudes Toward Encrypted Messaging Apps**. Akgul, O., Abu-Salma, R., Bai, W., Mazurek, M.L., Redmiles, E.M., Ur, B., *To be submitted July 2020*

- **Improving Non-Experts' Understanding of End-to-End Encryption: An Exploratory Study**. Bai, W., Pearson, M., Kelley, P.G., Mazurek, M.L., *European Workshop on Usable Security 2020*

- **Improving Mental Models of End-to-End Encrypted Communication**. Akgul, O., Bai, W., *(Presentation) Black Hat 2020*

5.0 CONCLUSIONS

In this report we have described our work on the DARPA Brandeis program. Reasoning about the privacy of certain data and the use of privacy-preserving systems is difficult. This research has resulted in an improved set of techniques for this challenging task, often resulting in novel, peer-reviewed, contributions to the academic literature. In addition to theoretical work, many of these techniques were implemented in proof-of-concept software artifacts which established their feasibility. These tools were then exercised on other Brandeis efforts, principally the Coalition Scenario from the Enterprise CRT and the use of Virtual Sensors from the IoT CRT. This work demonstrates that reasoning about privacy and privacy-preserving technologies is feasible and worthwhile, and that tool support is within reach for many systems.

6.0 REFERENCES

[1] I. Sweet, J. M. Calderón Trilla, C. Scherrer, M. Hicks, and S. Magill, "What's the over/under? Probabilistic bounds on information leakage," *CoRR*, vol. abs/1802.08234, 2018, [Online]. Available: http://arxiv.org/abs/1802.08234.

[2] J. A. Goguen and J. Meseguer, "Security policies and security models," in *IEEE Symposium on Security and Privacy*, 1982, pp. 11–20.

[3] G. Smith, "Recent developments in quantitative information flow."

[4] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," 1977.

[5] P. Cousot and R. Cousot, "Static determination of dynamic properties of programs," 1976.

[6] A. Miné, "The octagon abstract domain," 2001.

[7] P. Mardziel, S. Magill, M. Hicks, and M. Srivatsa, "Dynamic enforcement of knowledge-based security policies," 2011.

[8] R. Giacobazzi and F. Ranzato, "Optimal domains for disjunctive abstract interpretation," *Science of Computer Programming*, vol. 32, nos. 1-3, pp. 177–210, 1998.

[9] D. Monniaux, "Analyse de programmes probabilistes par interprétation abstraite," Thèse de doctorat, Université Paris IX Dauphine, 2001.

[10] F. Biondi, M. Enescu, A. Heuser, A. Legay, K. Meel, and J. Quilbeuf, "Scalable approximation of quantitative information flow in programs."

[11] M. Backes, B. Köpf, and A. Rybalchenko, "Automatic discovery and quantification of information leaks," in *IEEE Symposium on Security and Privacy*, 2009, pp. 141–153.

[12] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, 2009, pp. 19–30.

[13] C. Gould, Z. Su, and P. Devanbu, "JDBC checker: A static analysis tool for sql/jdbc applications," in *Proceedings. 26th International Conference on Software Engineering*, 2004, pp. 697–698.

[14] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao, "A static analysis framework for detecting SQL injection vulnerabilities," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, 2007, vol. 1, pp. 87–96.

[15] M. K. Gupta, M. Govil, and G. Singh, "Static analysis approaches to detect sql injection and cross site scripting vulnerabilities in web applications: A survey," in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, 2014, pp. 1–5.

[16] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 5–19, 2003.

[17] D. Stefan, A. Russo, J. C. Mitchell, and D. Mazières, "Flexible dynamic information flow control in Haskell," in *Proceedings of the 4th ACM Symposium on Haskell*, 2011, pp. 95–106.

[18] D. E. Denning, "A lattice model of secure information flow," *Communications of the ACM*, vol. 19, no. 5, pp. 236–243, 1976.

[19] S. Mehrotra, A. Kobsa, N. Venkatasubramanian, and S. R. Rajagopalan, "TIPPERS: A privacy cognizant IoT environment," in *IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM Workshops)*, 2016, pp. 1–6.

[20] R. Yus, G. Bouloukakis, S. Mehrotra, and N. Venkatasubramanian, "Abstracting interactions with IoT devices towards a semantic vision of smart spaces," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2019, pp. 91–100.

[21] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, nos. 3-4, pp. 211–407, 2013, doi: 10.1561/0400000042.

[22] J. Hsu *et al.*, "Differential privacy: An economic method for choosing epsilon," in *IEEE 27th Computer Security Foundations Symposium*, 2014, pp. 398–410.

[23] A. Ghosh and A. Roth, "Selling privacy at auction," in *Proceedings of the 12th ACM Conference on Electronic Commerce*, 2011, pp. 199–208, doi: 10.1145/1993574.1993605.

[24] W. Bai, M. Pearson, P. G. Kelley, and M. L. Mazurek, "Improving non-experts' understanding of end-to-end encryption: An exploratory study."

[25] O. Akgul, R. Abu-Salma, W. Bai, M. Mazurek, E. Redmiles, and B. Ur, "Secrecy, flagging, and paranoia revisited: User attitudes toward encrypted messaging apps."

[26] G. Singh, M. Püschel, and M. Vechev, "Fast polyhedra abstract domain," in *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, 2017, pp. 46–59, doi: 10.1145/3009837.3009885.

[27] P. Mardziel, S. Magill, M. Hicks, and M. Srivatsa, "Dynamic enforcement of knowledge-based security policies using probabilistic abstract interpretation," *Journal of Computer Security*, vol. 21, pp. 463–532, Oct. 2013.

[28] P. Cousot and N. Halbwachs, "Automatic discovery of linear restraints among variables of a program," 1978.

[29] K. Sen, D. Marinov, and G. Agha, "CUTE: A concolic unit testing engine for C," 2005.

[30] R. Bagnara, E. Rodríguez-Carbonell, and E. Zaffanella, "Generation of basic semi-algebraic invariants using convex polyhedra," 2005.

[31] M. R. Clarkson, A. C. Myers, and F. B. Schneider, "Quantifying information flow with beliefs," *JCS*, vol. 17, no. 5, pp. 655–701, 2009.

[32] I. Sweet, J. M. C. Trilla, C. Scherrer, M. Hicks, and S. Magill, "What's the over/under? probabilistic bounds on information leakage (extended version)," *CoRR*, vol. abs/1802.08234, Feb. 2018, [Online]. Available: https://arxiv.org/abs/1802.08234.

[33] R. Bagnara, P. M. Hill, and E. Zaffanella, "The Parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Sci. Comput. Program.*, vol. 72, Jun. 2008.

[34] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, J. Tauzer, and R. Yoshida, "LattE." https://www.math.ucdavis.edu/~latte/, 2008.

[35] R. Bagnara, P. M. Hill, and E. Zaffanella, "Widening operators for powerset domains," *ijsttt*, vol. 8, no. 4, pp. 449–466, 2006.

[36] K. Myers *et al.*, "Learning by demonstration technology for military planning and decision making: A deployment story," 2011.

[37] G. Smith, "Quantifying information flow using min-entropy," in *Proceedings of the 2011 eighth international conference on quantitative evaluation of systems*, 2011, pp. 159–167, doi: 10.1109/QEST.2011.31.

[38] G. Smith, "On the foundations of quantitative information flow," in *International conference on foundations of software science and computational structures*, 2009, pp. 288–302.

[39] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith, "Measuring information leakage using generalized gain functions," in *2012 IEEE 25th Computer Security Foundations Symposium*, 2012, pp. 265–279.

[40] K. Myers and D. Morley, "Policy-based Agent Directability," in *Agent autonomy*, H. Hexmoor, C. Castelfranchi, and R. Falcone, Eds. Kluwer, 2003, pp. 143–162.

[41] J. Parker, N. Vazou, and M. Hicks, "LWeb: Information flow security for multi-tier web applications," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[42] D. Darais, I. Sweet, C. Liu, and M. Hicks, "A language for probabilistically oblivious computation," Jan. 2020.

[43] M. Hardt and G. N. Rothblum, "A multiplicative weights mechanism for privacy-preserving data analysis," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 2010, pp. 61–70.

[44] A. Gupta, A. Roth, and J. Ullman, "Iterative constructions and private data release," in *Proceedings of the 9th International Conference on Theory of Cryptography*, 2012, pp. 339–356, doi: 10.1007/978-3-642-28914-9_19.

[45] R. Cummings, K. Ligett, M. M. Pai, and A. Roth, "The strange case of privacy in equilibrium models," in *Proceedings of the 2016 ACM Conference on Economics and Computation*, 2016, p. 659, doi: 10.1145/2940716.2940740.

[46] M. Joseph, A. Roth, J. Ullman, and B. Waggoner, "Local differential privacy for evolving data," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 2381–2390.

[47] K. Ligett, S. Neel, A. Roth, B. Waggoner, and Z. S. Wu, "Accuracy first: Selecting a differential privacy level for accuracy-constrained erm," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 2563–2573.

[48] S. V. Neel, A. L. Roth, and Z. S. Wu, "How to use heuristics for differential privacy," in *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019, pp. 72–93.

[49] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2003, pp. 202–210.

[50] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 587–601.

[51] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, pp. 268–282.

[52] I. Mironov, "On significance of the least significant bits for differential privacy," 2012, [Online]. Available: https://www.microsoft.com/en-us/research/publication/on-significance-of-the-least-significant-bits-for-differential-privacy/.

[53] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, "'You might also like': Privacy risks of collaborative filtering," in *IEEE Symposium on Security and Privacy*, 2011, pp. 231–246.

[54] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *IEEE Symposium on Security and Privacy*, 2008, pp. 111–125.

[55] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "LOGAN: Membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[56] C. Song and V. Shmatikov, "Overlearning reveals sensitive attributes," *arXiv preprint arXiv:1905.11742*, 2019.

7.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AFRL	Air Force Research Laboratory
AI	Abstract Interpretation
CBMC	C Bounded Model Checker
CCTV	Closed Circuit Television
CE	Concolic Execution
CRA	Charles River Analytics
CRT	Collaborative Research Team
DARPA	Defense Advanced Research Projects Agency
DNF	Disjunctive Normal Form
DNN	Deep Neural Networks
DP	Differential Privacy
ELINA	ETH Library for Numerical Analyis
FBI	Federal Bureau of Investigation
FPF	Factored Particle Filtering
HADR	Humanitarian Assistance and Disaster Relief
HVAC	Heating, Ventilation, and Air Conditioning
IFC	Information-Flow Control
IO	Input/Output
IoT	Internet of Things
JSON	Javascrip Object Notation
LIO	Labeled IO
MAC	Medium Access Control
MPC	(Secure) Multi-Party Computation
PP	Probabilistic Polyhedron
RAM	Random Access Memory
SAT	Boolean SATisfiability Problem
SCALE-MAMBA	Secure Computation Algorithms from Leuven - Multiparty AlgorithMs Basic Argot
SMT	Satisfiability Modulo Theories
SQL	Structured Query Language
SVT	Sparse Vector Technique

TAMBA	Testing and Modeling of Brandeis Artifacts
TIPPERS	Testbed for IoT-based Privacy-Preserving PERvasive Spaces
UCI	University of California Irvine