The Impact of Data Analysis
Collaborative Data Sharing

Collaboration between departments
- Departments have different access levels

Collaboration between institutions
- Institutions don’t trust each other to keep data private

Public release of data
- Public data might reveal sensitive information about original data
Current Frameworks

Data Owner → Data → Analyst → Analytics Program → Computation Infrastructure → Results
Current Frameworks

Threat 1: Untrusted program
Threat 2: Untrusted infrastructure
Threat 3: Sensitive results
Data Capsules

“Computation results must be differentially private”

Results satisfy original policy

Data Owner

Machine Learning Expert

Machine Learning Model

Residual Policy

Model Query Program

Analyst

Use Control Policy

Query Results

42

42

Query Results

Residual Policy

Machine Learning Program
Helio Framework

Data owner not required to trust program or infrastructure
Progress on Helio
Using Helio

- Data
- Use Control Policy

Analyst → Analytics Program → Helio System → Computation Infrastructure

Helio System with SGX

Use Control Policy
allow Role Researcher
  Declassification DiffPrivacy
  Deployment UCSF + SGX
  SuppressCol dob, name
allow Role Patient($id)
  Condition id = $id
Attribute semantics specified by **concept lattices**

![Diagram of concept lattice with nodes and edges representing roles (Patient, Researcher, Admin), deployment (UCSF, SGX), security (Encrypted, Machine Learning, Differential Privacy), and declassification (Propose-Test-Release, Laplacian Noise)].
Using Helio

Analyst → Analytics Program → Helio System → Computation Infrastructure

Data → Use Control Policy → Helio System

SGX

SGX
Example Program

```
val r1 = sc.load(UCSF_Records).map((_.id, 1)).reduceByKey(_ + _)
val r2 = sc.load(Mayo_Records).map((_.id, 1)).reduceByKey(_ + _)

// compute average number of hospital visits per individual
val r = r1.union(r2).reduceByKey(_ + _).map(_._2).mean
```
Using Helio

Data
Use Control Policy

Helio System
Computation Infrastructure

Analyst
Analytics Program

SGX
Helio System Architecture

SGX wrapper & bridge run user-defined functions inside enclave

Instrumenter & verifier enforce policies
Progress on SGX
Leveraging Secure Hardware: Intel SGX
Helio & SGX

- **Encrypted Input**
  - Constructs enclave, batches streaming input, returns output to Spark

- **Encrypted Output**
  - Decrypts input, invokes UDF on each record, encrypts output

- **SGX Wrapper**

- **SGX Bridge**

- **Decrypted Input**
  - Processes a single input record, produces a single output record

- **Decrypted Output**
Performance Evaluation

Test cluster: 4x Dell Laptop
Performance Evaluation

Running time normalized to Spark

Average overhead: less than 5%
Future Work & Case Studies
Generating Enclave Programs

RDD Query: Arbitrary Scala

```scala
val r1 = sc.load(UCSF_Records).map((_.id, 1)).reduceByKey(_ + _)
val r2 = sc.load(Mayo_Records).map((_.id, 1)).reduceByKey(_ + _)
val r = r1.union(r2).reduceByKey(_ + _).map(_._2).mean
```

Dataframe Query: Embedded DSL + SQL

```scala
val r = r1.union(r2).groupBy("id").count.agg(avg("count")).show
```

Generated Enclave Program: C++

```
```

- Query Arguments
- Decrypted Input
- Trusted built-in function implementation (e.g. `avg`)
- Decrypted Output
Case Study: Health eHeart Study

- Crowd-sourced longitudinal study on heart disease at UCSF Medical School
  - Subjects donate medical records and device data, fill out surveys
  - Goals: 1 million participants, enable collaborative research

- Helio accomplishes Health eHeart collaborative analytics goals
  - We have encoded HeH privacy policies and analytics tasks in Helio
  - HeH plans to use Helio to provide collaborative analytics
Conclusion

Three major threats to secure & private data analytics
No current solution addresses all three

Helio aims to give control to the data owner
Policies protect data capsules
Automatic policy enforcement protects against untrusted analyst
Secure hardware protects against untrusted infrastructure