**Helio: Secure & Privacy-preserving Collaborative Data Analytics**

Noah Johnson, Chang Liu, Joseph Near, Elaine Shi, Srinidhi Devadas, Dawn Song

---

**Explosion in Sensitive Data Collection**

More information is collected every day:
- Electronic health records for medical research
- Personal information from mobile devices
- Survey data for infrastructure research
- Location information from vehicles
- Sensor data from IoT devices
- E-mails for spam detection

---

**The Security & Privacy Challenge**

- Increasing security & privacy breaches involving personal data
- Collaborative data-driven research hampered by security & privacy concerns
- Current frameworks lack security & privacy solutions
- Data owner not required to trust analytics program or computation infrastructure

---

**Helio System**

- Data Collection
- Compute server
- Analyst
- Controller
- Dataframe Query: Embedded DSL + SQL

---

**For the Data Owner**

- Security & privacy policy specifies allowed uses of data
- Policy language designed to be accessible for non-programmers
- Policies written once, applied to all programs: no negotiation with analyst required
- Policies consist of nested clauses specifying allowed uses in terms of attributes

---

**For the Analyst**

- Analyst writes Spark program
- Program written as if raw data is available: no annotations or security checks required

```scala
val r1 = sc.load("UCSF_Records").map(_.id).reduceByKey(_. + _)
val r2 = sc.load("Stx_records").map(_.id).reduceByKey(_. + _)

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

---

**Program Analysis & Instrumentation**

- **Policy converter** converts policy into output checks
  - deny Data PatientRecord
  - allow Declassification
  - DifferentialPrivacy
  - Deployment UCSF+SGX
  - suppressCol dob, name

- **Verifier** uses information flow analysis to verify that instrumented program satisfies output checks
  - Concept lattices encode attribute semantics
  - Lattice ordering is used to compare output labels and required checks
  - Example: Declassification attribute:

  ```
  Data ⊑ PatientRecord
  Declassification ⊑ DifferentialPrivacy
  Deployment ⊑ UCSF
  SuppressCol ⊑ dob | name
  ```

- **Output Label**: Declassification DP Mean
- **Output Check**: Declassification ⊑ DifferentialPrivacy
- **Verifier checks**: DP Mean ⊑ DifferentialPrivacy

---

**Future Work: Generating Enclave Programs**

- Dataframe Query: Embedded DSL + SQL

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

- Generated Enclave Program: C++

---

**Architecture Overview**

- Spark Master
- Spark Worker
- Instrumenter
- Instrumented DAG
- DAG & Schedule
- Secure Shuffle Coordinator
- UDF / built-in function

- SGX Wrapper
- SGX Bridge
- Encrypted Input
- Decrypted Output
- Decrypted Input
- Encrypted Output

---

**Performance Evaluation**

- PageRank
- Graph
- KMeans
- Regression
- Workload

---

**SGX Wrapper** constructs enclave, batches streaming input
- SGX Bridge decrypts input, invokes UDF on each record, encrypts output
- UDF processes individual records, produces outputs

---

**Mobile CRT Integration**

- Hello-based compute server provides high-performance, secure and privacy-preserving query responses
  - Accepts private data - policy from mobile devices
  - Answers queries from command center
  - Hello guarantees policy enforcement