Combining Differential Privacy and Secure Multiparty Computation

Martin Pettai, Peeter Laud
{martin.pettai|peeter.laud}@cyber.ee

December 11th, 2015
Introduction

Problem
- Institutions have data about individuals
- Combining the data would allow useful inferences to be made
- But who combines and computes? Privacy!?

A solution
- Use secure multiparty computation (SMC)
  - ...or some other secure computation technique
- This field is currently maturing

New problem
- ✓ The computation is now private
- ✗ But what do the results leak?
Example — the PRIST study

Questions of the study

- Does working during studies affect the drop-out rate of IT students?
- How does dropping out affect the future earnings of IT students?

Implementation

- Get study status records from Ministry of Education
- Get tax records from the Tax Board
- Run statistical tests in our SHAREMIND SMC platform

We were asked if the outputs leak anything.
Differential privacy

- We have a probabilistic data release mechanism $\mathcal{M}$
  - Takes as input a set of records
  - Produces a tuple of values
- $\mathcal{M}$ is $\epsilon$-differentially private if
  - for any $S \subseteq \text{range}(\mathcal{M})$
  - for any two input datasets $T, T'$ differing in one record
  $\Pr[\mathcal{M}(D) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in S]$
Making functions differentially private

- We want to compute $f(T) \in \mathbb{R}^n$ for dataset $T$
- We add random noise $N$ to the output to mask the possible variation in the output due to changing one record
  - **Data release mechanism:** output $f(T) + N$
- Noise magnitude is proportional to the sensitivity $s$ of the function
  - the maximum change in output when one record is changed
- Add noise from $\text{Laplace}(\frac{s}{\epsilon})$
- We get $\epsilon$-differential privacy

This is the **most general** method and studied in the context of SMC

December 11th, 2015
Sample-and-Aggregate mechanism

Applicability
- there’s a distribution $\mathcal{D}$ over records
- $T$ — really a set of i.i.d. samples from $\mathcal{D}$
- $f(T)$ is computed in order to learn the statistic $f(\mathcal{D})$
- $f$ is generically asymptotically normal
  - Not a strong requirement

Mechanism
- Partition $T$ into $T_1, \ldots, T_\ell$ of almost equal size
- Let $O_i = \text{clip}(f(T_i), [\text{left}, \text{right}])$
- Output $\text{avg}\{O_1, \ldots, O_\ell\} + \text{Laplace}((\text{right} - \text{left})/(\ell \cdot \epsilon))$

[Smith, STOC 2011]
Our selection of $f$-s

- count
  - seems completely trivial, but it is not
- average
  - also sum, which is similar
- linear correlation coefficient
- median
Inaccuracies of the mechanism

Three sources of inaccuracy

Estimation error \( f \) is estimated on subsets of \( T \), instead of \( T \) itself

Clipping values of \( f(\{T_i\}) \) outside \([\text{left}, \text{right}]\) are pushed into the range

Noise \ldots by Laplace distribution

- When changing parameters, some sources of inaccuracy may increase and some may decrease
- Sample-and-aggregate may give better accuracy than simple Laplacian noise addition
SMC platform SHAREMIND

- SMC protocol set based on additive secret sharing
  - Three computing parties
    - security against one semi-honest party
  - any number of input and result parties
- Sharing can be over any ring
- Large set of available operations on private data:
  - Integer arithmetic, relational and logical operations, fix-point and floating-point arithmetic and elementary functions, generation and application of permutations
  - Floating-point operations are slower than fix-point operations
- Programming language SecreC for specifying private functionalities
  - Strong support for SIMD operations
Secret-sharing based SMC with **SHAREMIND**

- **Private and public data**
- Each $CP$ stores a copy of public data and performs operations on it
- **Private data** stored in secret-shared manner
- To operate on private data, computing parties run protocols
  - **Inputs**: Shares of inputs to the operation
  - **Outputs**: Fresh shares of the result of the operation
- There are **theorems** about security
Sample-and-aggregate on \textit{Sharemind}

Given

- \texttt{SECREC} code for a statistical function $f$
- block size $\ell$ (as a function of $f$) and clipping range $[\text{left}, \text{right}]$
- desired level $\epsilon$ of differential privacy

Compute $f([T])$ by

Randomly partition $[T]$ into subsets $[T_1], \ldots, [T_\ell]$

\begin{verbatim}
for $i \in \{1, \ldots, \ell\}$ (in parallel) do

$[O_i] \leftarrow f([T_i])$

$b_{1,i} \leftarrow [O_i] < [\text{left}]$ ; $b_{2,i} \leftarrow [O_i] > [\text{right}]$

$[O_i] \leftarrow [O_i] + b_{1,i} \cdot ([\text{left}] - [O_i]) + b_{2,i} \cdot ([\text{right}] - [O_i])$

end

return $\frac{1}{\ell} \sum_{i=1}^{\ell} [O_i] + \text{Laplace} \left( \frac{[\text{right}] - [\text{left}]}{\ell \cdot \epsilon} \right)$
\end{verbatim}
SMC operations for computing $f([T])$

Randomly partitioning $[T]$ into $\ell$ subsets
- **Sharemind** has protocols for permuting vectors and tables
- Randomly permute the rows of $[T]$
- $[T_i] \leftarrow i$-th $|T|/\ell$ rows of $[T]$

Running $f$ in parallel on all $[T_i]$
- **Secrec** does not yet support `par do i \leftarrow 1, \ldots, n : f(\ldots, i)`
- We parallelize the code of $f$ by hand

Generating Laplace noise
The protocols for elementary functions on private fixed-/floating-point numbers are sufficient for this.
Filtering

- Usually need to filter before aggregating
- A trusted party or a SMC protocol could create a new dataset containing the needed records
  - Leaks the number of matched records
- In secret sharing, we use a mask vector instead
  - Availability bit for each record
- The black box in Sample-and-Aggregate must use mask vectors
  - For median, we replace half of the excluded values with a very small value, and the other half with a very large value
  - The median stays roughly the same
Overhead of differential privacy on Sharemind

**Hardware:** three servers on 1 Gbps LAN

- **count** Around 360 ms. Does not depend on $|T|$. ($\ell = 1$)
  - That’s the time to generate Laplace noise

- **average** $2|T|$ comparisons, $3|T|$ multiplications, noise generation. ($\ell = |T|$)
  - Around 3.5 s for $|T| = 200k$

- **median** $(1 + \ell/|T|)$ times
  - We use Hoare’s $k$-th element finding algorithm

- **correlation** Proportional to $\ell$
  - Around 8 s for $\ell = 1000$
  - Due to extra floating-point operations

This generalizes to other SMC frameworks based on secret sharing.
Privacy budgets

- Composition of $\epsilon_i$-differentially private queries is $(\sum \epsilon_i)$-differentially private
- Global budget: $\sum \epsilon_i \leq B$
  - Queries that use a part of the database still reduce the global budget

Personalized Differential Privacy

- Budget for each row
- Or: budget for each provenance, multiple rows may have the same provenance
  - Need to join tables

[Ebadi et al., POPL 2015]
Implementing in-place budgets

- $T$ has columns $T_{\cdot, \text{budget}}$ and $T_{\cdot, \text{mask}}$

Running query $Q$ with in-place budgets

```
for $i \in \{1, \ldots, |T|\}$ (in parallel) do

   $[a_i] \leftarrow [T_{i, \text{mask}}] \land [T_{i, \text{budget}}] \geq \epsilon$
   \hspace{1cm} $\triangleright a_i \in \{0, 1\}$

   $[b_i] \leftarrow [T_{i, \text{budget}}] - [a_i] \cdot \epsilon$

   $[T_{\cdot, \text{budget}}] := [b]$

   $[r] \leftarrow $ output of $Q$ on $[T]$ with $[\overline{a}]$ as the mask vector

return $[r]$
```

- $Q$ must be $\epsilon$-differentially private

- **Overhead:** $n$ comparisons, multiplications and boolean operations
  - a bit over 10 $\mu$s per row of $T$
On provenance

Processing

$T$
On provenance

Processing

Budgets

2.39
1.75
3.15

T
Implementing provenance budgets

- Need to access records in “Budgets” table according to the provenance identifier
  - No fast SMC protocol for accessing an array element according to a private index
  - Accessing many elements in parallel can be more efficient (per element)
- We have developed and implemented reasonably efficient protocols for handling provenance budgets
  - Some of them similar or complementary to [Laud, PETS 2015]
- Details in the paper
- **Overhead**: significant (an order of magnitude)
  - Major step: sort $T \cup \text{Budgets}$ according to provenance identifiers
Conclusions

- Common mechanisms for differential privacy of outputs can be implemented on top of SMC protocols
- The overheads are generally not prohibitive
- Sharemind’s business developers are happy