Privacy Integrated Data Stream Queries

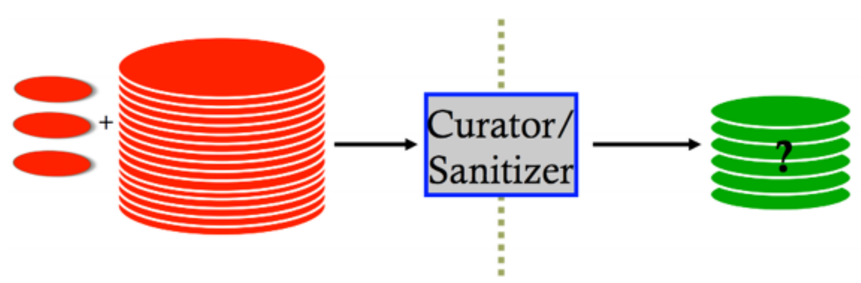
Lucas Waye

Harvard School of Engineering and Applied Sciences, 3rd year PhD

Privacy Tools for Sharing Research Data

A National Science Foundation Secure and Trustworthy Cyberspace Project

INTRODUCTION



Research on differential privacy is generally concerned with examining data sets that are static. Because the data sets do not change, every computation on them produces "one-shot" query results; the results do not change aside from randomness introduced for privacy . There are many circumstances, however, where this model does not apply, or is simply in- feasible. Data streams are examples of non-static data sets where results may change as more data is streamed. Theoretical support for differential privacy with data streams has been researched in the form of differentially private stream- ing algorithms. In this paper, we present a practical frame- work for which a non-expert can perform differentially private operations on data streams. The system is built as an extension to PINQ (Privacy Integrated Queries), a differentially private programming framework for static data sets. The streaming extension provides a programmatic interface for the different types of streaming differential privacy from the literature so that the privacy trade-offs of each type of algorithm can be understood by a non-expert programmer.

Most of the research and techniques of differential privacy have been with respect to a single static database. This precludes many environments where this approach is not feasible. Situations where one would like to avoid holding the entire database include:

•The data is coming in over a long period of time and we would like to perform intermediate computations on it.

•It is technically infeasible to hold the entire data set at one time.

•We would like to offer privacy guarantees against intrusions into the system (e.g. someone breaks into the system containing the database).

The streaming extensions implemented in Streaming PINQ attempt to bring in many of the different views and algorithms of streaming differential privacy from the literature into a programming framework. My goal was not for a complete library based on the literature, but rather to get a good representative sample of different types of streaming algorithms and how the framework would interact with them.

Considerations for a streaming differentially private framework

•When does an algorithm make an output?
•What does an adversary get to observe?
•What data can an algorithm keep as internal state?
•How do we protect individuals? (how do we model data set neighbors?)

CONTRIBUTIONS

•Extended PINQ to support streaming algorithms
•Support for different properties of streaming algorithms Pan-Privacy Continuous Output User-Level Privacy
•Implemented five differentially private streaming algorithms in the Streaming PINQ framework

This work appeared in the proceedings of the 1st International Workshop on Privacy and Security in Programming (PSP), October 2014.

PINQ

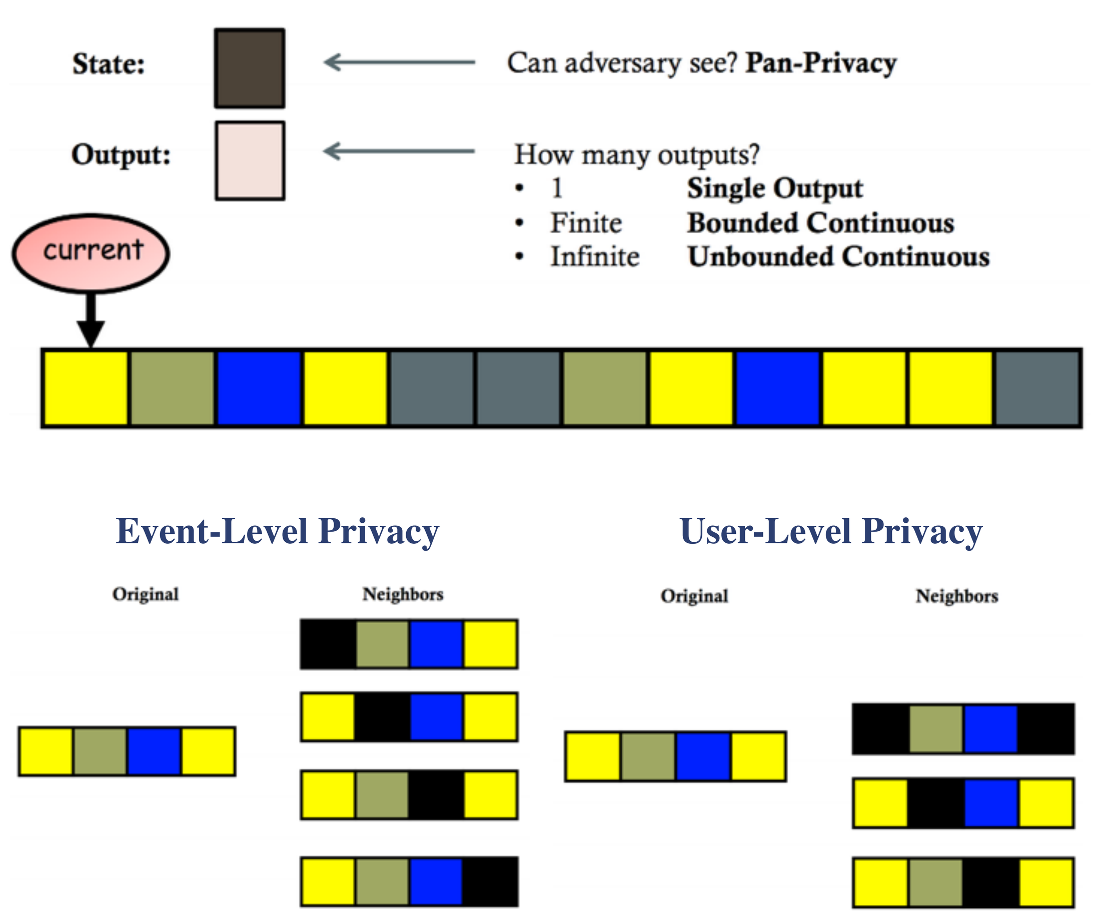
static data set **event-level or user-level privacy?**

```
var tweets = ReadAllSavedTweets("saved_tweets.txt");
var agent = new PINQAgentBudet(1.0);
var data = new PINQueryable<Tweet>(tweets, agent);

double tweetsFromNY = data
    .Where(tweet => tweet.Location.State == "NY")
    .NoisyCount(1.0);

Console.WriteLine("Tweets from New York: " + tweetsFromNY);
```

answer returned immediately

PINQ (**P**rivacy **IN**tegrated **Q**uery) [3] is a programming framework built on top of LINQ [4] in C#. It uses the Laplacian Noise mechanism to provide differential privacy guarantees. The privacy budget is enforced via an agent that is attached to the private query mechanism and is notified every time a differentially private operation takes place. We extend PINQ to support streaming data sets.

Streaming Algorithm Parameters



Internal State
•**Pan-Privacy:** the adversary may observe the internal state of the algorithm (usually at most 1 time).
Number of Outputs
•**Single Output:** the algorithm can only make one output.
•**Bounded Continuous Output:** the algorithm makes at most *n* outputs (where *n* is given as a parameter that affects accuracy).
•**Unbounded Continuous Output:** the algorithm can make an unbounded number of outputs.
Neighboring Streams
•**Event-Level Privacy:** adjacent streams differ by one event.
•**User-Level Privacy:** adjacent streams differ by all events that a particular user produces. *In the diagram above, different colors represent events from different users.*

STREAMING PINQ

```
// streaming data provider
var tweets = new AllTweetsFireHose();
// streaming-based privacy Agent
var agent = new EventLevelPrivacyBudget(1.0);
var data = new StreamingQueryable<Tweet>(tweets, agent);

// returns handle to streaming alg, rather than answer
var tweetsFromNY = data
    .Where(tweet => tweet.Location.State == State.NY)
    .RandomizedResponseCount(1.0);

// callback when output is made by algorithm
tweetsFromNY.OnOutput = (c =>
    Console.WriteLine("Tweets from New York: " + c));

// process 5,000 events
tweetsFromNY.ProcessEvents(5000);
```

Streaming PINQ is my extension of PINQ to support streaming differentially private algorithms. It is implemented in roughly 1000 lines of C# code. The system is meant to "look and feel" like PINQ. Many of the classes are entirely new and do not rely on the PINQ object model directly, but the same coding style is adopted for the programmer's ease of use and understanding.

Key Classes

•**`StreamingQueryable`** - wrapper class around a private stream in much the same way as `PINQueryable` is in PINQ. It supports transformations on the data and keeps track of active streaming algorithm subscribers to the private stream data.

•**`StreamingAlgorithm`** - encodes the mechanism for streaming differential privacy. The base class provides functionality to interact with the data stream to receive events. The base class also has functionality for the client of the algorithm to get outputs from the algorithm.

•**`PINQStreamingAgent`** - Agents are responsible for enforcing that ε-differential privacy is preserved for the stream. There are two inheriting classes that enforce either user-level privacy or event-level privacy. In the user-level privacy agent, privacy can never be returned to the stream. On the other hand, when viewing the stream with event-level privacy, the agent only needs to make sure that at most ε is "learned" for each event.

IMPLEMENTED ALGORITHMS

Algorithm	Privacy	Number of Outputs	Pan-Private	Additive Error
Buffered Average*	Event-Level	Single	No	$O(1)$
Randomized Response Count* [8]	Event-Level	Continuous Unbounded	Yes	$O(\sqrt{T})$
Binary Counter [8]	Event-Level	Continuous Bounded	No	$O((\log T)^{1.5})$
User Density [7]	User-Level	Single	Yes	α w.p. $1-\beta$
User Density Continuous [2, 7]	User-Level	Continuous Bounded	Yes	6α w.p. $1-\beta$

The table above shows the implemented algorithms in Streaming PINQ. Note that ε is removed from accuracy measurements. α and β are user-defined parameters to the algorithm. Algorithms with an asterisk (*) denote known optimal accuracy for their listed properties. Buffered Average is simply adding just enough Laplace noise to achieve differential privacy. Randomized Response Count's error match the theoretical lower bound from [2]'s negative result, given its properties (pan-private and continuous observation). Pan-Privacy is with respect to one intrusion.

FUTURE WORK

Performance Testing
- Are some data sets particularly amenable to existing differentially private algorithms?
  - Twitter messages seemed to be captured well
  - Can we get "acceptable" accuracy for real-world applications?

Language Work
- Large base of trusted code
  - Programming framework provides no help in assuring new streaming algorithm implementations are safe.
- C# seems to be the wrong choice of language
- What are the basic primitives for streaming and can we encode that into the language? How do we encode the different notions of privacy?

Theoretical Work
•Including timing of events in the model
  - Stock trade made after hours → institutional trader
  - Predictive mitigation for timing attacks

•Forgetful attacker for user-level privacy
  - Adjacency bounded to the last k events
  - Do the new models allow us to come up with new algorithms?

CONCLUSION

I present an extension to PINQ to support differentially private streaming algorithms. The framework is flexible to allow the data owners and data analysts decide which algorithms should be used based on their needs. In one case, a data analyst might want a very accurate result for user density, but would have to decrease the number of intermediate outputs seen. In another case, if a data owner wants to enforce Pan-Privacy then some algorithms cannot be used. This framework is meant for non-adversarial users. There is no formal checks on an implemented algorithm's advertised guarantees. Another caveat is that this framework is susceptible to timing attacks. I hope that this framework can serve as both a practical implementation for streaming algorithms, as well as providing a base for further exploration of new algorithms and streaming models.

REFERENCES

[1] Cynthia Dowrk. Differential Privacy. In ICALP, pages 1-12. Springer, 2006.
[2] Cynthia Dwork, Moni Naor, Toniann Pitassi and Guy N. Rothblum, Differential Privacy Under Continual Observation, STOC 2010
[3] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. Commun. ACM, 53(9):89–97, September 2010.
[4] Microsoft. Linq (language integrated query).
[5] Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In Proceedings of the 15th ACM SIGPLAN international conference on Functional programming, ICFP '10, pages 157–168, New York, NY, USA, 2010. ACM.
[6] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: security and privacy for mapreduce. In Proceedings of the 7th USENIX conference on Networked systems design and implementation, NSDI'10, pages 20–20, Berkeley, CA, USA, 2010. USENIX Association.
[7] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In In Proceedings of ICS, 2010.
[8] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. ACM Trans. Inf. Syst. Secur., 14(3):26:1–26:24, November 2011.
[9] Andreas Haeberlen, Benjamin C. Pierce, and Arjun Narayan. Differential privacy under fire. In Proceedings of the 20th USENIX conference on Security, SEC'11, pages 33–33, Berkeley, CA, USA, 2011. USENIX Association.

CRCS Center for Research on Computation and Society at Harvard School of Engineering and Applied Sciences
BERKMAN CENTER FOR INTERNET & SOCIETY AT HARVARD UNIVERSITY
The Institute for Quantitative Social Science at Harvard University
DATA PRIVACY LAB