

TPDP, WPES and CCS talks

Sebastian Meiser¹, Esfandiar Mohammadi²

1: University College London, United Kingdom, e-mail: s.meiser@ucl.ac.uk

2: ETH Zurich, Switzerland, e-mail: mohammadi@inf.ethz.ch

October 15, 2018

Abstract

Summaries of talks we attended.

1 Invited talk TPDP: Composition, Verification, and Differential Privacy

Speaker: Justin Hsu

After introducing differential privacy in general, we see a few well-known composition results. Post-processing is seen as an instantiation of sequential composition of an (ϵ, δ) -DP mechanism with a $(0, 0)$ -DP mechanism, which is kind of nice. Similarly, local DP is presented as an instantiation of parallel composition.

1.1 Formally verifying privacy

Our goals here are twofold: we want to have *dynamic* verification, i.e., raise errors if DP is violated and *static* verification, i.e., checking DP on all possible inputs. As a side-note, we want to simplify verification by using composition results; composition allows verification techniques to have much better automation.

The Fuzz family of languages allows for describing each type with a metric and, using group privacy ($(\epsilon, 0)$ -DP for $\Delta_f = 1$ implies $(k \cdot \epsilon, 0)$ -DP for $\Delta_f = k$). The description allows for an immediate static analysis that can use both sequential and parallel composition. These languages, so far, cannot deal well with ADP, but the speaker is currently working on that.

Privacy as an approximate coupling Idea: verify privacy by game hops. We somehow relate pairs of sampling instructions to show properties of a pair of inputs, but I'm not sure how exactly we do that. This static-analysis technique seems to be related to how Tim and me modeled differential indistinguishability, i.e., as a property of a pair of programs/machines. They, so far, cannot automatically generate proofs, but can check them.

1.2 Advanced composition

They analyze the old “advanced composition theorem” by Dwork, Rothblum and Vadhan. The speaker says that the analysis is more complicated, since the composition theorem needs to be applied “as a block”, instead of on small units of the program. The speaker then mentions that novel approaches (e.g., by Mironov on RDP) makes formal verification easier, as it allows to analyze a program step-by-step. I think that using our privacy buckets approach, such an analysis could be improved even more and made flexible as well.

1.3 Modularity

The verification approach aims at being able to modularly compose differential privacy mechanisms. Their approach does not solve a fundamental problem that black-box composition of DP mechanisms have: after black-box composition there is potentially too many points where noise is added. So, a more thorough

approach would divide DP mechanisms into summarizing data (e.g., via statistical queries) and adding noise. Then, composition would compose the summarization operations and then only add noise once. Such an approach would deteriorate utility much less.

2 Invited Talk TPDP 2: Deploying Differential Privacy for Learning on Sensitive Data

Speaker: Ulfar Erlingsson

The speaker mentions problems with RAPPOR, mostly that for the quantities of data they require, the privacy guarantees are deteriorating too fast. The main problem seems to be that they have to use local DP mechanisms.

The aim of their Prochlo realization is to combine the local differential privacy mechanisms with some central DP approach.

They notice that if they don't need correlations between data points from the same user, anonymous communication can help them. As a result, they have a partially central approach. They group data based on useful features.

Their approach combines three aspects of uncertainty: the uncertainty introduced by the local DP mechanism (via randomized response), the uncertainty introduced by the combination (anonymity and random shuffling) and the uncertainty applied to the results afterwards (in a centralized DP manner).

2.1 Private neural networks

The speaker identifies the problem of machine learning models, e.g., NNs, memorize the training data. This confirms known results on adversarial ML and works by Dawn Song and Vitaly Shmatikov.

Specifically, they introduce canaries, i.e., specific numbers, and want to check whether the NN is surprised to see this canary or not. In other words, they check whether the NN learnt this canary. Their experiments show that NNs indeed learn such canaries. Learning the canaries seems to reach its worst case early, with the convergence of the model.

The speaker emphasizes that this memoization is different from overfitting, because in spite of memoization generalization works well.

The speaker emphasizes that there are edge cases that the NNs seemingly have to memorize exceptions, because there does not seem to be a rule to characterize these edge cases. These exceptions are not necessarily important for the final model, as they contradict the general paradigm of machine learning: to learn general properties of samples and not exceptions. This makes differential privacy a natural fit for ML.

We now get an introduction into the privacy-preserving stochastic gradient descent (SGD) [?] work and the PATE paper [?].

In their case study, they use prior knowledge and use a strong bayesian prior.

3 Local differential privacy for evolving data

Speaker: Matthew Joseph, collaboration with Aaron Roth, Jonathan Ullman and Bo Waggoner

Paper: <https://arxiv.org/pdf/1802.07128.pdf>

The main difficulty they tackle is evolving data, i.e., data that can change over time. The simplest solution in terms of differential privacy would be to simply have a randomized response for every user in every round. While this is very simple, privacy deteriorates over time. To moderately improve it, you can apply subsampling, i.e., ask different subsets of users every time. This is better in terms of privacy, but the error is still large. Local sparse vectors (?) would allow for nice privacy guarantees, but require an exponentially larger number of samples.

The solution lets users “vote” to change statistics, i.e., they construct an adaptive mechanism.

In each epoch t , each user creates a localized bit, then generates a vote for or against updating and then depending on the average vote the analyst decides whether or not the estimate should be updated (requiring more interaction) or not.

Each user can guess whether or not the global estimate has changed significantly. We aren't completely sure on the maths here, but apparently each user locally simulates what the estimate should be (in the current round) and compare that to the previous global estimate. If these don't differ much, the user is content; if they do differ significantly, the user votes for a change.

If a change occurred, the analyst collects the changed estimates from the users to update the global estimate. Otherwise the analyst just keeps the previous estimate.

I'm not completely convinced yet by the proof intuition for why their mechanism satisfies differential privacy; a closer look into their paper ¹ might be in order to understand what exactly they are doing and why that is privacy-preserving, particularly towards the data analyst that collects the votes and combines the estimates.

4 WPES: TightRope: Towards Optimal Load-balancing of Paths in Anonymous Networks

Speaker: Hussein Darir (University of Illinois at Urbana-Champaign), collaboration with Hussein Sibai (University of Illinois at Urbana-Champaign), Nikita Borisov (University of Illinois at Urbana-Champaign), Geir Dullerud (University of Illinois at Urbana-Champaign), and Sayan Mitra (University of Illinois at Urbana-Champaign)

The paper is about Tor; some Tor circuits tend to be slow. Knowing the current state of the network (in terms of congestion and usage) leaks too much information, but maybe differential privacy can help us here. The goal is to use load balancing with locally optimal mechanisms and this mechanism should then be differentially private.

They assume that each user only holds a single static circuit that is active all the time.

Each relay computes the ratio between their capacity over the number of paths going through it. Then they choose the relay with the minimal ratio, called the bottleneck relay (it is most congested). The path(s) going through the bottleneck relay are tagged with the ratio of the bottleneck relay. Then from the other relays, the capacity used by the path through the bottleneck relay is subtracted from these relays' capacities, and then this path is removed. The algorithm then is repeated to compute the bandwidth available to each path. They evaluate this and find (unsurprisingly) that the bandwidths are unfairly distributed over paths.

Their (non-private) algorithm now takes the existing bandwidth capacities and ratios into account and also computes the ratio where one more path is added to the relay. They find the most congested node, i.e., the one where the ratio would be worst if we added one. We then remove this relay (and paths through it; and update the ratios). This process is repeated until we can build a path. Using this technique makes path selection much more fair, but violates privacy.

To achieve differential privacy, they need up-to-date statistics. I'm not completely sure how they achieve that; they seem to sample many times, i.e., have each user locally simulate how the network might have changed? I'm not quite sure what exactly they do and what privacy guarantees they achieve, but their evaluation shows that still some improvement over a simple random choice (without optimization) is possible.

They did not analyze the impact their work has on the anonymity provided by Tor; just the impact of releasing their histogram.

5 WPES: ClaimChain: Improving the Security and Privacy of In-band Key Distribution for Messaging

Speaker: Wouter Lueks (Ecole Polytechnique Fédérale de Lausanne), collaboration with Bogdan Kulynych (Ecole Polytechnique Fédérale de Lausanne), Marios Isaakidis (University College London), George Danezis (University College London), and Carmela Troncoso (Ecole Polytechnique Fédérale de Lausanne)

Paper: <https://arxiv.org/abs/1707.06279>

¹available online: <https://arxiv.org/pdf/1802.07128.pdf>

The classical PKI problem is: how does Alice find Bob’s key? They analyze how to distribute keys for an email structure that allows for encrypted communication. Whenever someone sends an email, they want to attach some small datastructure that includes claims about their own key and about their friends’ keys. Moreover, they want to hide whose keys they are communicating from unauthorized people. They intend to do that by adding some access control mechanism, specifying who can access these records. Finally, they want to get non-equivocation, i.e., sending around different keys supposedly belonging to the same entity, accessible by different people.

Their Approach They store all claims (of the form “entity, key”) into an encrypted dictionary. To hide the indexes of the dictionary, they put a verifiable random function in there, i.e., some apparently random value; they communicate this value to people of interest. They then put a proof that they computed the VRF correctly into the encrypted part. They then chain these claims together with a hash chain. This basically is the ClaimChain that can be attached to emails.

6 WPES: What’s a little leakage between friends? (Short)

Speaker: Sebastian Angel, collaboration with David Lazar, Ioanna Tzialla

Paper: <https://arxiv.org/abs/1809.00111>

Metadata-private messaging systems allow communication without leaking metadata to providers, servers or users not involved in the communication. In the simplest case, everyone sends packets to everyone else. Existing MPM systems avoid this broadcast by limiting the number of messages per round. Clients now have to start scheduling their own communication (and potentially wait until their messages are being sent). To actually communicate, you can have a dialing round in which you negotiate a communication round.

They then analyze what happens if a malicious user tries to gain information about whether others are talking: the idea is that an adversary can ask a user whether they want to talk; since every user only has a limited number of (real) messages per round, they can see whether the user is “free” or “already in conversation”. To counter this, they envision a private answering machine, that hides from callers whether or not a user is talking; moreover eventually a caller needs to get through and it should be efficient, i.e., the capacity of the answering machine should be significantly smaller than full broadcast. Their proposal has several limitations, including that everyone needs to have a max number of friends m ; then statically map callers to rounds (mod m). Drawbacks are potential limitation and leakage of the number of friends and an increased latency (if the answering machine has low capacity or the user many friends).

7 WPES: DynaFlow: An Efficient Website Fingerprinting Defense Based on Dynamically-Adjusting Flows (Short)

Speaker: David Lu (MIT PRIMES), collaboration with Sanjit Bhat (MIT PRIMES), Albert Kwon (MIT), and Srinivas Devadas (MIT)

Paper: <https://dl.acm.org/citation.cfm?id=3268960>

The paper aims at countering website fingerprinting, particularly when people are using Tor. As we know, Tor is vulnerable to traffic analysis attacks, including website fingerprinting attacks. The talk considers an open-world scenario for fingerprinting. Existing defenses, e.g., supersequence defenses over-approximate all websites within an anonymity set and make all these websites look the same. Similarly, constant flow defenses enforce a constant transmission rate, which introduces a significant overhead as well.

Their approach morphs traffic into fixed bursts: o outgoing and i incoming packets, for fixed values o and i . Moreover, they vary the inter-packet timing interval: t changes every b bursts, t is chosen from some set $\{t_1, \dots, t_k\}$ and up to a adjustments are allowed.²

They evaluate their approach against a couple of existing attacks, using both a “medium security” and a “high security” setting. Their results look promising: the overheads are only 28% of time overhead and about 110% of bandwidth overhead.

²I’m not sure within which time frame a is measured