Poirot: Private Contact Summary Aggregation

Chenghong Wang$^{1}_{*}$, David Pujol$^{1}_{*}$, Yanping Zhang$^{1}_{*}$, Johes Bater^{1}, Matthew Lentz^{1,2}, Ashwin Machanavajjhala^{1}, Kartik Nayak^{1}, Lavanya Vasudevan^{3,4}, Jun Yang^{1}
Department of Computer Science, Duke University^{1},VMware Research^{2},
Department of Family Medicine and Community Health, Duke University^{3},
Duke Global Health Institute^{4}

Abstract

Physical distancing between individuals is key to preventing the spread of a disease such as COVID-19. On the one hand, having access to information about physical interactions is critical for decision makers; on the other, this information is sensitive and can be used to track individuals. In this work, we design Poirot, a system to collect aggregate statistics about physical interactions in a privacy-preserving manner. We show a preliminary evaluation of our system that demonstrates the scalability of our approach even while maintaining strong privacy guarantees.

1 Introduction

COVID-19 is a contagious disease that is known to spread rapidly through person-to-person contact. To curb its spread, many state governments announced lockdowns, and offices and schools were temporarily shut down. While such measures help stabilize the infection rate, these measures cannot be implemented indefinitely.

For organizations to reopen, ideally, they would need relevant data about adherence to physical distancing [3]. This will inform their decisions on measures and interventions to reduce the risk of contracting the disease. For instance, such information can help assess the effectiveness of current policies such as opening at a reduced scale and also act as a feedback loop to inform new policies. They can also help identify *hotspots* and to better allocate sanitation resources. Individuals can also benefit from such information since it helps them understand how they are adhering to social distancing policies. On the other hand, information required to track physical distancing is very sensitive since it can reveal users' location trajectories. Thus, there is a tension between obtaining such physical distancing information and ensuring individuals' privacy.

The goal of our work, Poirot, is to collect necessary information about individuals' physical interactions in a privacy-preserving manner to provide actionable information to decision-makers and the individuals themselves. Our work measures physical interactions through the number of contact events between individuals; such measurements are directly related to the disease's spread and, thus, a crucial enabler to decision-making. To ensure privacy, we collect and release aggregate statistics of contact events that cannot be linked back to any individual. Thus, Poirot can provide utility to organizations and individuals with little loss of privacy.

Outline. In this paper, we describe the overall design of Poirot in Section 2, elaborate on one aspect of the system, the differentially-private release of statistics in Section 3, and present a preliminary evaluation in Section 3.

Authors: $*$ denotes equal contribution.

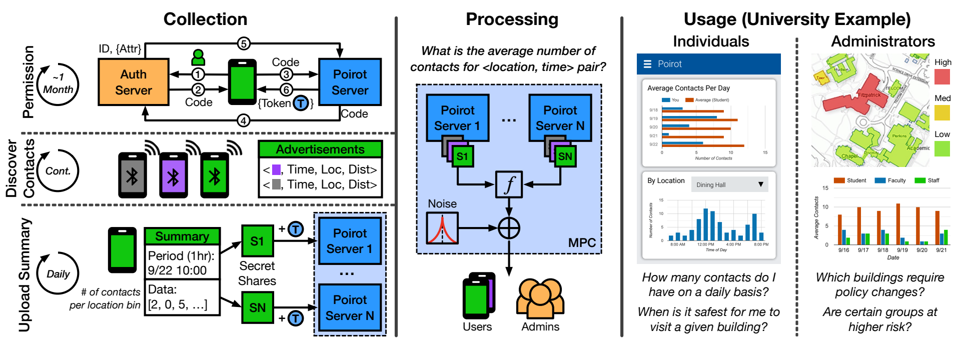


Figure 1: Overview of the Poirot workflow and architecture, broken down into three phases: collection, processing, and usage.

2 Design

As shown in Figure 1, the key actors involved in Poirot are users (and their smartphones), administrators, authentication servers, and some compute servers. Poirot protects sensitive data sent by individual users from the administrators who receive released statistics, any subset of the compute servers, and authentication servers. We assume that both users and servers are semi-honest, meaning they honestly participate in the protocol but may attempt to derive sensitive information. Users provide accurate information, without attempting to maliciously modify the Poirot app. Authentication servers validate user credentials and provide proof of identity to the compute servers. Finally, compute servers faithfully execute a secure multi-party protocol to evaluate queries. Compute servers may learn the total number of days for which a user has uploaded their data and which users are participating. We also assume that all but one compute server may collude with each other.

2.1 Workflow

Collection. Users must first obtain *permission* to participate in the system, which happens on a recurring basis (e.g., monthly). This process follows an OAuth flow whereby the user provides an identity to Poirot via an authorization service. The app and Poirot server participate in a blind signature [1] protocol to generate tokens for uploading data; this decouples identity from data, as the server can only validate a token but not identify the user who generated it. Tokens can encode attributes of the user (e.g., student) based on the key used to sign the tokens (which the user can verify).

A contact event occurs when two user devices remain in close proximity for a period of time, where the distance and duration are specified by a policy (e.g., 6ft and 15min for COVID-19). To *discover contact events*, the app uses Bluetooth to exchange identifiers with nearby participating devices. Unlike existing contact tracing applications, we are interested in detecting contacts regardless of whether either user has tested positive for COVID-19. To maintain long-term unlinkability of a device's identifiers over time while supporting long-running contacts (e.g., hours), identifiers are changed every epoch (nominally 15min) but both the previous and current epoch identifiers are broadcast.

Every day, the app computes and uploads aggregate summaries of all recorded contact events for that day. The aggregate summary contains a vector of counts, where each entry corresponds to a location bin (e.g., county in a state). The app generates secret-shared versions, one per compute server, which ensures that the compute servers do not learn the contents of individual summaries.

Processing. At this point, the compute servers have secret-shared contact summaries for all participating users for the previous day. The servers participate in a multi-party computation (MPC) protocol [5, 11] to compute aggregate statistics over the secret-shared summaries from all users. As part of the MPC protocol, Poirot adds noise to satisfy differential privacy (DP) [4] such that the released statistics are accurate while not revealing sensitive information.

Usage. Poirot disseminates the aggregate statistics computed in the previous phase for *use* by individual users and administrators. In our example, we consider a university setting to highlight the utility in being able to answer common questions for both individuals and administrators. University administrators, for instance, may change policies to reduce crowds in certain locations according to the number of contact events. Users can understand how safe they are being, and may change their behavior to avoid visiting certain locations (e.g., dining hall) at times of high traffic.

3 Processing on the Servers to Release Differentially-Private Statistics

In this section, we elaborate on the *processing* step, focusing on the differentially-private release of statistics. Each day, the app uploads secret-shared aggregate summaries of all recorded contact events to the compute servers. Since the servers receive these summaries from multiple users, they first synchronize with each other to ensure that they have shares corresponding to the same sequence of users for a given day. Next, the servers process these summaries using MPC in *batches*, maintaining secret-shared aggregates after processing each batch. Whenever statistics are to be released to administrators, they employ the differentially-private mechanism described in Algorithm 1. For simplicity, we use two servers in our description (and experiments) and use the noise addition approach from [9]. This can be extended to $N > 2$ semi-honest servers, in which case we can use the approach by Narayan and Haeberlen [8] to bound the added noise. We also assume that there is one batch containing synchronized information about all users and the statistics are released the following day after all (or most) users have uploaded their summaries. In practice, some users may also upload their contact summaries a few days later (e.g., due to bad network connection); we can extend the algorithm to release statistics again after taking into consideration the privacy loss.

Algorithm 1: Computing differentially-private average contact events for each bin under MPC

```
Input: $\mathbf{c}^j, \mathbf{p}^j, \mathbf{c}^{\max}, \epsilon$
– $\mathbf{c}^j$ is a vector over bins storing number of contact events for a user $j$
– $\mathbf{p}^j$ is a vector over bins describing whether there was a contact event for a user $j$
– $\mathbf{c}^{\max}$ is a vector containing the maximum possible contact events per bin
– $\mathbf{c}^j$ and $\mathbf{p}^j$ are secret-shared to servers $\mathcal{S}_A$ and $\mathcal{S}_B$, $\mathbf{c}^{\max}$ is known in the clear
Output: A vector over bins containing differentially-private average number of contacts $\tilde{\mathbf{o}}$
Protocol:
– Each server $\mathcal{S}_x$ samples two noise vectors $\mathbf{v}_x$ and $\mathbf{v'}_x$, where $v_i^x \sim Lap(\frac{2\cdot c_i^{\max}}{\epsilon})$ and $v_i'^x \sim Lap(\frac{2}{\epsilon})$
Using MPC, servers $\mathcal{S}_A$ and $\mathcal{S}_B$ jointly compute:
– aggregate vectors $\mathbf{c} = \sum_j c_i^j$ and $\mathbf{p} = \sum_j p_i^j$
– noisy aggregate vectors $\tilde{\mathbf{c}}$ where $\tilde{c}_i = c_i + v_i^A + v_i^B$, and $\tilde{\mathbf{p}}$ where $\tilde{p}_i = p_i + v_i'^A + v_i'^B$
Each server receives $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{p}}$ in the clear and outputs $\tilde{\mathbf{o}}$ where $\tilde{o}_i = \tilde{c}_i/\tilde{p}_i$ for each bin $i$
```

Theorem 1. *Algorithm 1 satisfies ϵ-differential privacy.*

Proof. The MPC protocol can be abstracted as two mechanisms: $\mathcal{M}_1$, which computes the aggregated contacts over all bins, and $\mathcal{M}_2$, which counts how many individuals are present in each bin. As each bin is $(\epsilon/2)$-DP and all bins are disjoint, we can apply parallel composition [7] to show that both $\mathcal{M}_1$ and $\mathcal{M}_2$ are each $(\epsilon/2)$-DP. Furthermore, since $\mathcal{M}_1$ and $\mathcal{M}_2$ are both Laplace mechanisms with privacy parameter $\epsilon/2$, by applying sequential composition [7], we show that the MPC protocol satisfies $(\epsilon/2 + \epsilon/2)$-DP. Each server computes the final result with differentially-private outputs $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{p}}$ as a post-processing step, thus maintaining the same differentially-private guarantee. □

4 Evaluation

We have discussed how Poirot collects, processes, and utilizes aggregate contact event summaries in a privacy-preserving manner. In this evaluation, we focus on a key performance aspect: the scalability of the privacy-preserving computation of aggregate statistics.

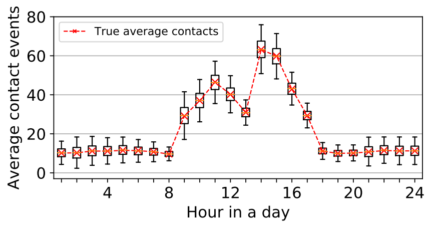
Case studies. Our first case study captures a university setting, where members of the community (e.g., students, faculty) use the app as classes resume during the pandemic. We choose Duke University as our example, setting the number of users to 20K and the number of location bins to 256 (one per campus building). In the second case, we consider a state-wide setting, where all residents participate and the state government is interested in per-county statistics. We chose North Carolina (NC) as our example, setting the number of users to 10M and the number of location bins to 100 (one per county in the state). In the third case, we evaluate the accuracy of Poirot on a real-world interaction data from the Copenhagen Networks Study [10].

Configuration. For the Poirot servers, we use two Google Cloud Platform servers in the us-central1-a zone running Ubuntu Linux, provisioned with 2 vCPUs, 32 GB memory, 10 GB storage, and 1 Gbps network bandwidth. For the app, we use a Samsung Galaxy A20e smartphone running Android 10. We implement the MPC protocol using arithmetic circuits in the ABY framework [2].

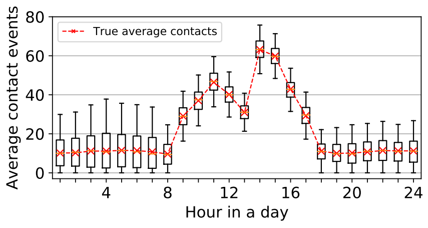
Case	Number of Bins Location	Time	Users	Execution Time App (ms)	Server (s)
`Duke`	256	24	20K	366.1 ± 8.9	94.3 ± 0.4
`NC`	100	1	10M	6.0 ± 4.4	776.1 ± 1.7
`Copenhagen`	1	24	705	1.68 ± 2.35	0.015 ± 0.0

Table 1: Performance for computing aggregate statistics.

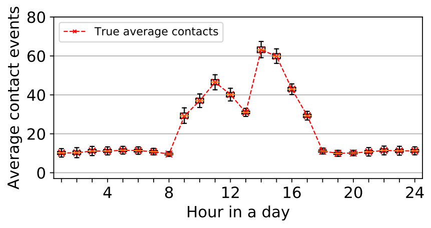
Results. In Table 1, we show the average and standard deviation for the execution time of MPC without differential privacy, both for the app (1000 trials) and the server (10 trials). There is limited app-side computation to generate the secret shares, with 256 bins for 24 time periods taking 366.1ms on average. Likewise, server-side computation is also efficient; even for 10M users in our North Carolina example, the MPC protocol completes in just 776.1s. These results demonstrate the scalability for large deployment scenarios.



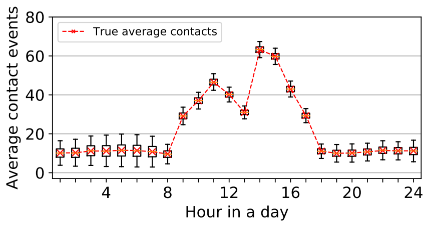
(a) $\epsilon = 0.1$, hourly-max



(b) $\epsilon = 0.1$, all-time-max



(c) $\epsilon = 0.5$, hourly-max



(d) $\epsilon = 0.5$, all-time-max

Figure 2: Comparing true averages with the distributions of differentially-private averages.

To study the accuracy of the released statistics, we use the Copenhagen Networks Study dataset which contains contact events across times in the day for a period of 28 days. We split the daily contact summary into 24 hourly bins; due to lack of location data, we treat all contact events as occurring single location bin. Each user can have multiple contact events in an hour, and the sensitivity of the average is the maximum possible number of contact events. To compute our results, we consider two settings: (i) all-time-$\max$: choosing sensitivity as $1.5\times$ the maximum number of contact events for any user in any day, and (ii) hourly-$\max$: choosing sensitivity as $1.5\times$ the maximum number of contact events for any user in an hour. The second option is reasonable because we treat each location-time bin as disjoint. Figure 2 plots the true daily averages along with distribution of differentially-private noisy averages computed over 1000 runs (shown by the box plot with 1st and 99th percentile whiskers). We make two observations. First, the distribution of noisy averages are closer to the true averages when we choose hourly-$\max$ (especially for $\epsilon = 0.1$). This shows the advantage of choosing an appropriate maximum based on the hour of the day. Second, we see that our choice of ϵ provides different privacy versus accuracy trade-offs, with lower ϵ values providing better privacy but lower accuracy, and vice versa. Poirot allows the administrator to precisely configure this trade-off to satisfy their specific use cases.

5 Conclusion

Physical distancing is key to managing the spread of contagious diseases such as COVID-19. Our insight is that aggregate statistics over contact events detected between individuals can serve as actionable information while providing strong privacy guarantees for individual users. Poirot uses a combination of multi-party computation and differential privacy techniques to preserve user privacy, while scaling to large deployment scenarios (e.g., state-wide). We envision Poirot as a useful tool that complements contact tracing applications [6] by providing information to enable proactive (rather than reactive) actions.

References

[1] D. Chaum. Blind Signature System. In *Advances in Cryptology*, pages 153–153. Springer, 1984.

[2] D. Demmler, T. Schneider, and M. Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *Network and Distributed System Security Symposium (NDSS)*, 2015.

[3] K. Drakopoulos. *The Logic Around Contact Tracing Apps Is All Wrong*, 2020. https://www.wired.com/story/opinion-the-logic-around-contact-tracing-apps-is-all-wrong/.

[4] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In S. Halevi and T. Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 2019.

[6] G. Kaptchuk, D. G. Goldstein, E. Hargittai, J. M. Hofman, and E. M. Redmiles. *How good is good enough for COVID19 apps?*, 2020. https://arxiv.org/pdf/2005.04343.pdf.

[7] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2009.

[8] A. Narayan and A. Haeberlen. Djoin: Differentially private join queries over distributed databases. In *Symposium on Operating Systems Design and Implementation (OSDI)*, 2012.

[9] A. Roy Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha. Crypt: Crypto-assisted differential privacy on untrusted servers. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2020.

[10] P. Sapiezynski, A. Stopczynski, D. D. Lassen, and S. Lehmann. Interaction data from the copenhagen networks study. *Scientific Data*, 6(1):1–10, 2019.

[11] A. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science (SFCS)*, pages 160–164. IEEE, 1982.