PrivApprox's Privacy Analysis and Proofs

Do Le Quoc[†]*, Martin Beck*[†]*, Pramod Bhatotia*[*]
Ruichuan Chen[‡]*, Christof Fetzer*[†]*, and Thorsten Strufe*[†]
[†]TU Dresden [*]The University of Edinburgh [‡]Nokia Bell Labs

In this report, we present the privacy analysis and proofs of PRIVAPPROX system [8]. PRIVAPPROX is designed for privacy-preserving stream analytics on distributed users' private dataset. The system consists of four main components: clients, proxies, aggregator, and analysts. At a high-level, the system works as follows: a query published by an analyst is distributed to clients via the aggregator and proxies. Clients answer the analyst's query locally over the users' private data using a privacy-preserving mechanism. Clients' answers are transmitted to the aggregator via anonymizing *proxies*. The *aggregator* aggregates received answers from the clients to provide privacy-preserving stream analytics to the analyst.

Before we explain in details the privacy analysis and proofs of PRIVAPPROX, we present the system model assumed in this system.

1 System model

1.1 Threat model

Analysts are potentially malicious. They may try to violate the PRIVAPPROX privacy model; i.e., de-anonymize clients, build profiles through linkage of requests and answers, or de-randomize (remove added noise from) the answers.

Clients are also potentially malicious. They could generate false or invalid responses to distort the query result sent to the analyst. However, we do not defend against Sybil attacks [3], which is beyond the scope of this work [10].

Proxies are also assumed to be potentially malicious. They may transmit messages between clients and the aggregator in contravention of the system protocols. PRIVAPPROX has at least two proxies, while we assume that also at least two proxies do not collude with each other.

The *aggregator* is assumed to be Honest-but-Curious (HbC): the aggregator faithfully conforms to the system protocol, but may try to exploit the information about clients. The aggregator does not collude with any proxy, nor the analyst.

Finally, we assume that all end-to-end communications use authenticated and confidential connections (are protected by long-lived TLS connections), and no system component could monitor all network traffic.

1.2 Privacy model

Privacy properties of PRIVAPPROX include: *(i)* zero-knowledge privacy, *(ii)* anonymity, and *(iii)* unlinkability.

All aggregate query results in the system are independently created under *zero-knowledge privacy* guarantees. The chosen privacy metric *zero-knowledge privacy* [6] builds upon differential privacy [4] and provides a tighter bound on privacy guarantees compared to differential privacy. Informally, zero-knowledge privacy states that essentially everything that an adversary can learn from the output of an zero-knowledge private mechanism could also be learned using aggregate information. *Anonymity* means that no system components can associate query answers or query requests with a specific client. Finally, *unlinkability* means that no system component can join any pair of query requests or answers to the same client, even to the same anonymous client.

2 Proofs

2.1 Zero-knowledge privacy

We show that PRIVAPPROX achieves ϵ_{zk}-zero-knowledge privacy and prove a tighter bound for ϵ_{dp}-differential privacy, than what generally follows from zero-knowledge privacy [6]. The basic idea is that all data from the clients is already differentially private due to the use of randomized response. Furthermore, the combination with pre-sampling at the clients makes it zero-knowledge private as well. Following the privacy definitions, any computation upon the results of differentially, as well as, zero-knowledge private algorithms is guaranteed to be private again.

In the following paragraphs we show that:

- Independent and identically distributed (IID) sampling decomposes easily and is self-commutative. See Lemma 2.1.
- Sampling and randomized response mechanisms commute. See Lemma 2.2.
- Pre-sampling and post-sampling can be traded arbitrarily around a randomized response mechanism. See Corollary 2.3.
- A ϵ_{zk}-zero-knowledge privacy bound for PRIVAPPROX system. See Theorem 2.4
- A ϵ_{dp}-differential privacy bound for PRIVAPPROX system. See Theorem 2.5
- Our differential privacy bound is tighter than the general differential privacy bound derived from a zero-knowledge private algorithm. See Proposition 2.6.

Intuitively, differential privacy limits the information that can be learned about any individual i by the difference occurring from either including i's sensitive data in a differentially private computation or not. Zero-knowledge privacy on the other hand also gives the adversary access to aggregate information about the remaining individuals denoted as D_{-i}. Essentially everything that can be learned about individual i can also be learned by having access to some aggregate information upon D_{-i}.

Let $San()$ be a sanitizing algorithm, which takes a database D of sensitive attributes a_i of individuals $i \in \mathrm{P}$ from a population P as input and outputs a differentially private or zero-knowledge private result $San(D)$. For brevity, we write $San_A(D)$ for the output of the adversary A with arbitrary external input z and access to $San(D)$. Similarly, we omit the explicit usage of the external information z as input to the simulator S, as well as the total size of the database. See [5] Definition 1 and 2 for the extended notation. Let $O \subseteq Range(San_A)$ be any set of possible outputs. ϵ_{dp}-differential privacy can be defined as

$$\Pr[San_A(D) \in O] \leq e^{\epsilon_{dp}} \cdot \Pr[San_A(D_{-i}) \in O] \tag{1}$$

while ϵ_{zk}-zero-knowledge privacy is defined as

$$\Pr[San_A(D) \in O] \leq e^{\epsilon_{zk}} \cdot \Pr[S(T(D_{-i}), |D|) \in O]. \tag{2}$$

Before proving the desired properties, we need to introduce some notation. Let $D = \{a_i\}$ be a database of sensitive attributes of individuals $i \in \mathrm{P}$. For ease of presentation and without loss of generality we restrict the individual's sensitive attribute to a boolean value $a_i \in \{0, 1\}$ and $\mathrm{D} = a_{i'}$ for all $i' \in \mathrm{P}$. Furthermore, let $\mathsf{D}(\mathrm{D}) = \{U : U \subseteq \mathrm{D}\}$ be the super-set of all possible databases and $Sam(D, u)$: - $\mathsf{D}(\mathrm{D}) \times (0, 1) \to \mathsf{D}(\mathrm{D})$ be a randomized algorithm that i.i.d. samples rows or individuals with their sensitive attributes from database D with probability s without replacement. Let $San(D, p, q)$: $\mathsf{D}(\mathrm{D}) \times (0, 1) \times (0, 1) \to \mathsf{D}(\mathrm{D})$ be a two-coin randomized response algorithm that decides for any individual i' in database D with probability p if it should be part of the output. If it is not included in the output, the result of tossing a biased coin (coming up heads with probability q) is added to the output.

Lemma 2.1. (Decompose and commute sampling) *Let $s = uv$ with $s, u, v \in (0, 1)$ being sampling probabilities for a sampling function $Sam()$. It follows that $Sam()$ can be composed and decomposed easily and is self-commutative.*

$$\begin{aligned} Sam(D, s) &\approx Sam(Sam(D, u), v) \\ &\approx Sam(Sam(D, v), u). \end{aligned}$$

Proof. Let Sam_u, Sam_v be sampling algorithms that sample rows i.i.d. from a database with probability u and v respectively. By applying $Sam_u(D)$, any row in D has probability u of being sampled. The probability for any row in D being sampled by Sam_v is equivalently v. Using function composition the probability for any row in D being sampled by $Sam_s = (Sam_u \circ Sam_v)(D)$ is

$$s = uv. \tag{3}$$

From multiplication being commutative ($u \cdot v = v \cdot u$) follows that Sam_u and Sam_v commute, that is $Sam_u \circ Sam_v = Sam_v \circ Sam_u$. This is true for deterministic functions and can easily be extended to randomized functions described as random variables, as random variables are commutative under addition and multiplication. For ease of presentation and without loss of generality we keep the notion of functions instead of random variables. Let $Sam_s(D) = Sam(D, s)$ be a sampling function that

samples rows i.i.d. from a given database D with probability s. Decomposing sampling function $Sam_s()$ with probability s into two functions with probabilities u and v follow from Equation (3). It also follows that two sampling functions with probabilities u, v can be composed into a single sampling function with sampling probability s.

□

Lemma 2.2. (Commutativity of sampling and randomized response) *Given a sampling algorithm $Sam()$ and a randomized response algorithm $San()$, the result of the pre-sampling algorithm $F_{pre}(D, s, p, q) = San(Sam(D, s), p, q)$ is statistically indistinguishable from the result of the post-sampling algorithm $F_{post}(D, s, p, q) = Sam(San(D, p, q), s)$. It follows that sampling and randomized response commute under function composition: $Sam \circ San = San \circ Sam$.*

Proof. For any individual i having $a_i \in D$ we have to consider eight different possible cases. In case the sampling algorithm $Sam()$ decides to not sample i, it obviously doesn't matter if it gets removed before the randomized response algorithm is run of afterwards. We thus condition on $Sam()$ to include i in the output.

1. Let us first consider the case that $San()$ outputs the real value for individual i. As $Sam()$ is fixed to output i independent of its value, there is no difference between F_{pre} and F_{post}.

2. In case $San()$ outputs a randomized answer $Sam()$ again is not influenced by the outcome of any of the coin tosses and passes i along to the output. This is of course also independent of the actual randomized result.

This concludes the proof that sampling and randomized response are independent regarding their order of execution and thus commute. □

Corollary 2.3. (Arbitrary sampling around randomized response) *Let $s = uv$ for $s, u, v \in (0, 1)$ be sampling probabilities for a sampling function $Sam()$ and $San()$ be a two-coin randomized response mechanism with probabilities (p, q). Sampling can be arbitrarily traded between pre-sampling and post-sampling around the randomized response mechanism San.*

$$\begin{aligned} San(Sam(D, s), p, q) &\approx Sam(San(Sam(D, u), p, q), v) \\ &\approx Sam(San(D, p, q), s). \end{aligned}$$

Proof. This follows directly from applying Lemma 2.1 and Lemma 2.2. □

We will now give a bound on ϵ_{zk} for the privacy of PRIVAPPROX system under the zero-knowledge privacy setting, as well as derive a tighter bound for (ϵ_{dp})-differential privacy, than the bound that generally follows from zero-knowledge privacy.

Theorem 2.4. (ϵ_{zk}-zero-knowledge privacy) *Let A be an algorithm that applies sampling with probability s, together with a two-coin randomized response algorithm using probabilities (p, q). A is ϵ_{zk}-zero-knowledge private with*

$$\epsilon_{zk} = ln\left(s\frac{2-s}{1-s}\left(\frac{p+(1-p)\,q}{(1-p)\,q}\right)+(1-s)\right). \tag{4}$$

Proof. From [5], Theorem 1 follows that a (k, ϵ_{rr})-crowd-blending private mechanism combined with a pre-sampling using probability s achieves ϵ-zero-knowledge privacy with

$$\epsilon_{zk} = ln\left(s \cdot \left(\frac{2-s}{1-s} e^{\epsilon_{rr}}\right) + (1-s)\right).$$

We omit the description for the additive error δ, which can be derived equivalently from [5] Theorem 1. Following Proposition 1 from [5] every ϵ_{rr}-differentially private mechanism is also k, ϵ_{rr}-crowd-blending private, thus randomized response being an ϵ_{rr}-differentially private mechanism, also satisfies (k, ϵ_{rr})-crowd-blending privacy with $k = 1$. Combining both results with Equation 8 in the technical report [8]) $\epsilon_{rr} = ln\left(\frac{p+(1-p)q}{(1-p)q}\right)$ gives an

$$\epsilon_{zk} = ln\left(s \cdot \left(\frac{2-s}{1-s}\left(\frac{p+(1-p)q}{(1-p)q}\right)\right) + (1-s)\right)$$

zero-knowledge private mechanism for randomized response combined with pre-sampling. Using Corollary 2.3 we can replace pre-sampling with a combination of pre- and post-sampling (with probabilities u, v respectively and $s = u \cdot v$) while keeping ϵ_{zk} fixed. We thus have

$$\epsilon_{zk} = ln\left(uv\frac{2-uv}{1-uv}\left(\frac{p+(1-p)\,q}{(1-p)\,q}\right) + (1-uv)\right).$$

□

If we do not aim at achieving zero-knowledge privacy, we can fall back to differential privacy using the result from [6], Proposition 3, which states that any ϵ-zero-knowledge private algorithm is also 2ϵ-differentially private. Using the results from sampling secrecy [7], which achieve a privacy boost by applying pre-sampling before using a differentially private algorithm, we derive a tighter bound for differential privacy, than what follows generally from zero-knowledge privacy.

Theorem 2.5. (ϵ_{dp}-differential privacy) *Let A be an algorithm that applies sampling with probability s, followed by a two-coin randomized response algorithm using probabilities (p, q). A is ϵ_{dp}-differentially private with*

$$\epsilon_{dp} = ln\left(1 + s\left(\frac{p+(1-p)\,q}{(1-p)\,q} - 1\right)\right). \tag{5}$$

Proof. We use the result from [1], Proof of Lemma 3, which bounds an ϵ_{rr}-differential private algorithm combined with pre-sampling using probability s by $\epsilon_{dp} = ln(1 + s(exp(\epsilon_{rr}) - 1))$. Let $\epsilon_{rr} = ln\left(\frac{p+(1-p)q}{(1-p)q}\right)$ be the bound derived for randomized response, we get

$$\epsilon_{dp} = ln\left(1 + s\left(\frac{p+(1-p)\,q}{(1-p)\,q} - 1\right)\right).$$

Applying Corollary 2.3 we derive an ϵ_{dp} bound for the combination of pre-sampling, randomized response and post-sampling of:

$$\epsilon_{dp} = ln\left(1 + (uv)\left(\frac{p + (1-p)\,q}{(1-p)\,q} - 1\right)\right).$$

□

Proposition 2.6. (Tighter ϵ_{dp}-differential privacy bound) *The bound ϵ_{dp} for differential privacy of a sampled randomized response system derived in Theorem 2.5 is tighter than ϵ_{zk}-differential privacy, which is again tighter than the general $2\epsilon_{zk}$-differential privacy bound that follows from ϵ_{zk}-zero-knowledge privacy [6].*

We directly proof Proposition 2.6 by comparing ϵ_{dp} from Theorem 2.5 with ϵ_{zk} from Theorem 2.4. As we want to prove a bound that is tighter than ϵ, we drop the factor of 2. This is possible because a ϵ-differentially private algorithm is also 2ϵ-differentially private. If we succeed in proving a bound tighter than ϵ, then 2ϵ-differential privacy is trivially fulfilled.

Proof. Proposition 3 from [6] states that every ϵ-zero-knowledge private algorithm is also 2ϵ-differentially private. Using Theorem 2.4 we get a ϵ_{zk}-differentially private system with $2\epsilon_{zk} = 2ln\left(s\frac{2-s}{1-s}\left(\frac{p+(1-p)q}{(1-p)q}\right) + (1-s)\right)$. Theorem 2.5 proves a bound of $\epsilon_{dp} = ln\left(1 + s\left(\frac{p+(1-p)q}{(1-p)q} - 1\right)\right)$. Let $e^{\epsilon_{rr}} = \frac{p+(1-p)q}{(1-p)q}$. Putting together Theorem 2.5, Theorem 2.4, Proposition 2.6 and Proposition 3 [6] we have:

$$\begin{aligned}
ln\left(1 + s\left(\frac{p + (1-p)\,q}{(1-p)\,q} - 1\right)\right) &< ln\left(s\frac{2-s}{1-s}\left(\frac{p + (1-p)\,q}{(1-p)\,q}\right) + (1-s)\right) \\
s\left(\frac{p + (1-p)\,q}{(1-p)\,q} - 1\right) &< \frac{2-s}{1-s}s\left(\frac{p + (1-p)\,q}{(1-p)\,q} - 1\right) \\
s\left(e^{\epsilon_{rr}} - 1\right) &< \frac{2-s}{1-s}s\left(e^{\epsilon_{rr}} - 1\right) \\
1 &< \frac{2-s}{1-s}
\end{aligned}$$

As $s \in (0,1)$ is the sampling parameter with a minimal right side for $s_{min} = \text{argmin}_{s\in(0,1)}\left(\frac{2-s}{1-s}\right) = 0$ the above inequality becomes $1 < 2$, which holds and concludes the proof. □

2.2 Relation of differential privacy and zero-knowledge privacy

Zero-knowledge privacy and differential privacy describe the advantage ϵ of an adversary in learning information about individual i by using an output from an algorithm $San()$ running over database D containing sensitive information $a_i \in D$ of individual i compared to using a result of a second — possibly different — algorithm $San'()$ running over D_{-i}. Zero-knowledge privacy is a strictly stronger privacy metric through the additional access to aggregate information of the remaining database D_{-i} compared to differential privacy [6]. By intuition, as differential privacy is a special case of zero-knowledge privacy and the adversary aims at maximizing its advantage, the advantage

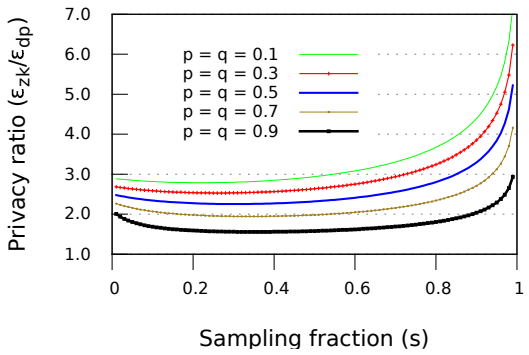

Figure 1: Ratio of $\frac{\epsilon_{zk}}{\epsilon_{dp}}$ depending on the sampling parameter s for different values p and q.

of an adversary in the zero-knowledge model is at least as high and possibly higher than the advantage of an adversary in the differential privacy model: $\epsilon_{zk} \geq \epsilon_{dp}$. Figure 1 draws the ratio $\frac{\epsilon_{zk}}{\epsilon_{dp}}$ between the zero-knowledge privacy level ϵ_{zk} and the differential privacy level ϵ_{dp} given identical parameters p, q and s. Put differently, as the adversary is allowed to do more in the zero-knowledge model, the privacy level is lower, which is reflected by a higher ϵ_{zk} value compared to the differential privacy level ϵ_{dp} — given identical system parameters.

2.3 Anonymity

We make the following assumptions to achieve the remaining two privacy properties:

(A1) At least two out of the n proxies are not colluding.

(A2) The aggregator does not collude with any of the proxies.

(A3) The aggregator and analysts cannot — at the same time — observe the communication around the proxies.

(A4) The adversary, seen as an algorithm, lies within the polynomial time complexity class.

To provide anonymity, we require that no system component (proxy, aggregator, analyst) can relate a query request or answer to any of the clients. To show the fulfillment of that requirement we take the view of all three parties.

a) A *proxy* can of course link the received data stream to a client, as it is directly connected. However, as the data stream is encrypted, it would need to have the plaintext query request or response for the received data stream. To get the plaintext the proxy would either need to break symmetric cryptography, which breaks assumption **(A4)**, collude with *all* other proxies for decryption, which breaks assumption **(A1)** or collude with the aggregator to learn the plaintext, which breaks assumption **(A2)**.

b) Anonymity against the *aggregator* is achieved by source-rewriting, which is a standard anonymization technique typically used by proxies and also builds the basis for anonymization schemes [2, 9]. To break anonymity the aggregator must be a global, passive attacker, which means that he is able to simultaneously listen to incoming and outgoing traffic of any proxy. This would violate assumption **(A3)**. The other possibility to bridge the proxies is by colluding with any of them — breaking assumption **(A2)**.

c) The *analyst* knows the query request, but doesn't get to learn the single query answers. He needs to collude with the aggregator, to see single responses. Thus the problem reduces to breaking anonymity from the view of the aggregator. Collusion with the aggregator and any proxy would break assumption **(A2)**. Collusion with up to $n - 1$ proxies reduces to breaking anonymity from the proxy view.

2.4 Unlinkability

Unlinkability is provided by the source-rewriting scheme as in anonymity. Breaking unlinkability on any *proxy* is similar to breaking anonymity, as the proxy would need to get the plaintext query. The *aggregator* only gets query results, but no source information, as this is hidden by the anonymization scheme. The query results sent by the clients also do not contain linkable information, just identically structured answers without quasi-identifiers. The view of the *analyst* doesn't receive responses, so it must collude with either a proxy or the aggregator, effectively reducing to the same problem as described above.

References

[1] Differential privacy and the secrecy of the sample, Sept. 2009.

[2] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[3] J. R. Douceur. The Sybil Attack. In *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[4] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, part II (ICALP)*, 2006.

[5] J. Gehrke, M. Hay, E. Lui, and R. Pass. Crowd-blending privacy. In *Proceedings of the 32th Annual International Conference on Advances in Cryptology (CRYPTO)*, 2012.

[6] J. Gehrke, E. Lui, and R. Pass. Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy. In *Theory of Cryptography*, 2011.

[7] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What Can We Learn Privately? *SIAM J. Comput.*

[8] D. L. Quoc, M. Beck, P. Bhatotia, R. Chen, C. Fetzer, and T. Strufe. Privacy pre-

serving stream analytics: The marriage of randomized response and approximate computing. https://arxiv.org/abs/1701.05403, 2017.

[9] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 1998.

[10] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao. Defending against sybil devices in crowdsourced mapping services. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2016.