A note on Privacy-Preserving Measurements Techniques

Sofía Celi

Brave Software

July 25, 2022

Abstract

A very succint and informal note on Privacy-Preserving Measurements (PPM) techniques and schemes.

1 Introduction

Aggregate measurements are a way by which systems (servers, cloud servers: the data collectors) can receive data from a population (a number of users) and compute useful aggregate statistics over them. While the general idea is to be able to only compute aggregation functions; in practice, systems collect private/sensitive data prior to even computing the aggregation. This centralized leakage of private user data poses severe security and privacy risks: attackers can steal and leak (publish) users' sensitive data, servers can misuse said users' data for profit, and intelligence agencies or other entities can use the data for targeted or mass surveillance.

To mitigate these threats, several mechanisms/protocols have been developed: they aim to provide a level of both security and of privacy. They have recently been called Privacy-Preserving Measurements (PPM) [1]. However, it is not easy to compare which technology/protocol/scheme provides which property and with what efficiency. We aim to provide a quick note on that.

Analysing the different protocols aiming to provide private aggregate measurements asks for a need of properly defining security and privacy in their specific context: What type of privacy and security they claim to attain? What constitutes a failure to preserve either? What is the power of the adversary whose goal it is to compromise the properties of the scheme? What auxiliary information is available to the adversary even without access to the user private data in question?

As stated by [Dwo06], a paper by Dalenius [Dal77] stated a desire for something like "semantic security" in the context of this type of data handling: access to a statistical database should not enable anyone to learn anything about a user that could not be learned without access. This idealized strong privacy goal is achieved to a degree by various PPM systems, and it is impossible to fully achieve in the presence of *auxiliary information* (information available to the adversary other than from access to the statistical database). This will be our guiding privacy notion when looking at the schemes.

2 Differential Privacy

As described by [Des21], the core idea of Differential Privacy (DP) is to add some randomness, or noise, in specific parts of the scheme: to the data collected, to the output of the aggregate statistic (or function), or to the aggregation itself. It formalizes the goal that the the risk to one's privacy should not substantially increase as a result of participating in a statistical database (mainly, of participating in a survey). DP assumes that users individually contribute a single row of data to a database D. On D, an algorithm A will be executed to gather some information about the rows, and the output will be shown. DP, informally, guarantees that it is very hard to reconstruct any individual row of data from D when observing the output from the algorithm A.

[1]The IETF has recently created a working group devoted to standardize this privacy-preserving measurements techniques [IET21]

2.1 The models

Note that these models could be applied to any of the schemes listed in this note by removing the need of adding noise at certain points. Note also that beyond DP schemes, an interactive model is rarely used.

Non-interactive In the non-interactive setting, the system that collects users data, a trusted entity (which could be the sever or a proxy), publishes a "sanitized" version of the collected data. "Sanitization" here means "anonymization" or "de-identification". Traditionally, sanitization employs techniques such as data perturbation and sub-sampling, as well as removing identifiers such as IPs and/or user-agents (it can also remove names or specific dates) depending on a local policy. After sanitization, the trusted entity should discard any of the stripped identifiers, and proceed with the aggregation function.

Interactive In the interactive setting, the data collector, again trusted, provides an interface through which users may pose queries about the data, and get noisy answers. [Dwo06]

2.2 Notions

Definition 2.1 (*ε-differential-privacy*)**.** A randomized function K gives ε-*differential-privacy* if for all data sets $D1$ and $D2$ differing on at most one element, and all $S \subseteq Range(K)$: $Pr[K(D1) \in S] \leq exp(\epsilon) \times Pr[K(D2) \in S]$.

What the definition captures is the notion that the output of the function is similar on both data sets if you change or remove the one element. The degree of similarity depends on ε: the smaller it is, the more similar the outputs are. *Randomized response*, a surveying technique developed in the 1960s [War65], is a mechanism that exemplifies this intuition. An example commonly used to describe this technique involves a question on a sensitive topic, such as "Are you a member of the Communist party?". For this question, the survey respondent (the user) is asked to flip a coin in secret, and answer "Yes" if it comes up heads (all times), but tell the truth, either a "Yes" or "No", otherwise (if the coin comes up tails). Using this procedure, each respondent retains very strong deniability for the "Yes" answers, since such answers can be most likely attributable to the coin landing on heads. Respondents can also choose the untruthful answer by flipping another coin in secret, and get strong deniability for both "Yes" and "No" answers.

This "randomized responses" technique and the notion of ε-*differential-privacy* achieves privacy of individual data, irrespective of prior (auxiliary) knowledge. In general, it:

- Preserves the privacy of any kind of information.
- Preserves privacy irrespective of any prior knowledge.
- Allows the quantification of the greatest possible information that an attacker can gain.
- Takes into account "insider conspiracy" (participants of a survey can conspire to de-anonymize other participants) which degrades the level of privacy.

In some early schemes, the privacy guarantee degrades if the survey is repeated with the same set of respondents.

There is an slightly "weaker" definition used for Differential Privacy:

Definition 2.2 (*δ-approximate ε-differential-privacy*)**.** A randomized function K gives $\delta-$*approximate* ε-*differential* privacy if for all data sets $D1$ and $D2$ differing on at most one element, and all $S \subseteq Range(K)$: $Pr[K(D1) \in S] \leq exp(\varepsilon) \times Pr[K(D2) \in S] + \delta$

What the *δ-approximate ε-differential-privacy* definition captures is to account for an "small privacy loss", δ.

2.3 Schemes

RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response [EPK14] The RAPPOR scheme uses randomized response techniques to achieve a level of privacy and collection of large amounts of data from users. Contrary to the stated limitation, privacy is not degraded if the survey is repeated with the same set of users. But, this advantage of the scheme opens the avenue to new attackers.

- A "simple" attacker who has access to a single report (one instantiation of data). This attacker will be used in the "one-time" model.
- A "window" attacker who has access to multiple reports over a window of time. This attacker has an extra advantage if the data does not change significantly over the window of time. This attacker will be used in the "longitudinal" model.
- A "complete" attacker who has unlimited access to all reports over an unlimited time.

RAPPOR builds on the idea of memoization and randomization. It provides a framework for one-time and longitudinal privacy protection by executing the randomized response step twice with a memoization step in between. The first step, *Permanent randomized response*, is used to create a "noisy" answer which is memoized by the client and permanently reused in place of the real answer. The second step, *Instantaneous randomized response*, reports on the "noisy" answer over time, eventually completely revealing it.

RAPPOR protects against the 3 types of attackers with a strictly bounded parameter. But, it does not prevent cases where survey collectors manipulate the process and ask *selected users* to report more than once, or when users participate in the same survey from multiple devices/accounts. It does not also strongly preserve longitudinal privacy when the user's value changes rapidly over time.

In the efficiency and accuracy side, each reporting user performs independent coin flips (they individually add randomness), which, in practice, translates to perturbed analysis results induced by the properties of the binomial distribution. This makes the scheme a type of *Local differential privacy*, which can be very costly. The overall magnitude of this added random noise (Gausian noise) can be very large: even in the theoretical best case, the standard deviation grows in proportion to the square root of the survey count ($\sqrt{n}$, where n is the number of participants), and the randomness is in practice higher by an order of magnitude. Thus, if a billion users data are analyzed, then, a common signal from even up to a million reports may be missed. This could be greatly improved if reports are partitioned; but, in turn, this degrades the privacy guarantees.

PROCHLO: Strong Privacy for Analytics in the Crowd [BEM+17] The PROCHLO scheme uses optional randomized response techniques in a *Encode, Shuffle, Analyze* (ESA) architecture. They define DP in the *shuffled model*: the local randomness is augmented by a private channel that randomly permutes a set of user-supplied data, and differential privacy is only required as part of the output of the shuffler [DP20]. The architecture splits entities into different responsibilities:

- **Encoders:** encode and encrypt (in a nested manner) the data collected at client level, and, optionally, add randomness for privacy preservation.
- **Shufflers:** run as a standalone networked service with the aim of masking the data origin and confounding data provenance by eliminating metadata and by shuffling (over long periods of time). They undo one layer of nested encryption and should be trustworthy.
- **Analyzers:** networked services that receive shuffled data batches, and are responsible for analyzing or releasing a database after all the remaining layers of encryption. Their cryptographic key determines the specific analysis outcome and associated privacy protection.

The scheme inherently introduces other kinds of attackers due to the different entities that interact in their architecture. Mainly:

- Analyzer compromise: it can link different data received, and correlate it with auxiliary information.

- Shuffler compromise
- Analyzer and shuffler collusion: they will see which users contribute what data.
- Encoder compromise and collusion

The scheme also formalises the properties that differential privacy should guarantee:

- **Robustness to auxiliary information:** independence of prior knowledge.
- **Preservation under post-processing:** once DP is executed on data, it cannot be undone.
- **Composability and graceful degradation:** which should happen when the same user repeatedly responds to the survey and changes the response either dramatically or slowly.

Note that PROCHLO expects honest execution of each entity (and, for this reason, uses trusted hardware and software enclaves).

3 Prio [CGB17]

Prio is a scheme for private aggregation that aims to provide privacy, robustness, and scalability. Prio works with a small number of servers, and, as long as one of them is honest, the system leaks nearly nothing about users data, except for what the aggregate statistic itself reveals. It aims to work with a large number of clients and assumes (and expects) the existence of authenticated encrypted channels between them (and with the servers).

3.1 Properties

- **Robustness:** (only maintained if all servers are honest) a coalition of malicious clients cannot affect the output of the function only by misreporting (willingly or not) their data values.
- **Anonymity:** the adversary cannot pinpoint which honest client submitted which data even when the adversary controls all (or a subset) of other clients, chooses and injects data, or controls all but one server.
- ***f*-*privacy*:** for an aggregation function, f, an adversary, who controls any number of clients and all but one server, learns nothing about the honest clients' data x_i, except what they can learn from $f(x_1...x_n)$. Depending on the aggregation function, Prio either provides this property in its strongest notion (*strong f-privacy*) or a "weaker" version that depends on what the function leaks (*$\tilde{f}$-privacy*).

The full definition of *f-privacy* is as follows:

Definition 3.1 (*f-privacy*)**.** Assuming there are s servers and n clients, for every subset of at most $s-1$ servers and every subset of at most n clients there exists an efficient simulator that, for every choice of client inputs $(x_1, ..., x_n)$, takes as input:

- the public parameters of the scheme,
- the indices of the adversarial clients and servers,
- oracle access to the adversarial participants, and
- the value $f(x_1, ..., x_n)$

and outputs a simulation of the adversarial participants' view of the scheme run whose distribution is computationally indistinguishable from the distribution of the adversary's view of the honest scheme run.

3.2 Prior knowledge

The next sections will use some concepts that the reader might be unfamiliar with. We will briefly explain them now:

- **Gossiping:** a mechanism that allows for state sharing in distributed systems.
- **Secret-Sharing:** a cryptographic tool that allows a specific user to "share" a private value x in a vector of values $[x] = ([x]_1, ..., [x]_n)$ in such a way that any subset of these values reveals nothing about x, but all values can be used together to reconstruct x in full. In arithmetic secret-sharing, $[x]_i$ are (usually) random values in a ring Z_M subject to the constrain $\sum_i [x_i] = x \mod M$. In boolean secret-sharing, $[x]_i$ are (usually) random L-bit integers subject to the constrain $\bigoplus_i [x_i] = x$.

3.3 Scheme

We will explain the scheme with the example of sum of one-bit values as the aggregation function (as in the original Prio paper). In this case, each client holds a one-bit integer x_i and the servers want to compute the sum of the clients' values

$$\sum_i x_i.$$

The public parameter is a prime p and all parties hold an arithmetic circuit representing a predicate $Valid : F^L \to F$ (where F is some finite field and L is a modest leakage function).

The scheme proceeds as:

- **Upload:** Each client i splits its value x_i into s shares, one per server, using a secret-sharing scheme. Each client also creates a proof string. The client then sends, over an encrypted and authenticated channel, one share of its submission and the proof string to each server.
- **Aggregate:** Upon receiving a share, each server checks that the received value is well formed by using the received *secret-shared non-interactive proofs* (a construction that uses arithmetic circuits). The servers, then, gossip between themselves to determine that the "full" proof is valid. If it is valid, they proceed. Each server j holds an accumulator value $A_j \in F_p$, initialized at zero. Upon receiving a share from the *ith* client (and checking that it is valid), the server adds the uploaded share into its accumulator: $A_j \leftarrow Aj + [x_i]_j \in F_p$.
- **Publish:** Once the servers have received a share from each client, they publish their accumulator values. Computing the sum of the accumulator values $\sum_j A_j \in F_p$ outputs the desired sum $\sum_i x_i$ of the clients' data.

3.4 Attacks and Limits

Prio, in its original design, has some efficiency considerations and some limitations. Mainly:

- Server-to-server communication (for when servers are gossiping, for example) grows neither with the complexity of the verification circuit $Valid$ nor with the size of the value x; but, server-to-client communication grows linearly with the size of the $Valid$ circuit. Furthermore, client-side computation increases linearly with the size of users' data.
- Certain aggregation functions might leak "extra" information other than the output of the function. For example, when computing a variance, it leaks the expectation $E[X]$.
- It does not provide *robustness* in the face of faulty servers, which could be a desirable goal in some scenarios.
- It relies on arithmetic circuits, which use finite-field multiplication, addition, and multiplication by constant gates. This makes the size of the proofs grow, which could be detrimental for client-to-server communication.
- It only allows for numerical data inputs.

Furthermore, the original design lists some possible (but not so practical) attacks:

- Selective denial-of-service attack: a network adversary prevents all honest clients except one from being able to contact the Prio servers. The adversary can then learn this selected clients data.
- Intersection attack: the adversary observes the output of $f(x_1, ..., x_n)$ of a run of the Prio scheme with n honest clients. The adversary then forces the nth honest client offline and observes a next scheme run, in which the servers compute $f(\tilde{x_1}, ..., \tilde{x_{n-1}})$. If the clients' data are the same over time ($x_i = \tilde{x_i}$), then the adversary learns the difference between the two runs, which could reveal client n data x_n. Note that this is prevented with DP composability.

3.5 Reducing Leakage

Prio-based schemes allow for identity leakage: IP addresses or HTTP user agents can be seen. This, in turn, means that the link between client identity and their input is unbroken. In the cases where maintaining this link is unuseful, Prio-based schemes can be combined with Oblivious HTTP [TW21] to anonymize the data.

3.6 Variations

The variations of Prio mainly improve its efficiency (especially, in regards to the proof computation and transmission), but seem to preserve the same properties and notions of attackers (there has not been many formal analysis of these schemes but the intuition is that they should inherit from the original Prio). They all also only allow for numerical inputs.

Prio+ [AGJ+21] Prio+ works in the same way as the original design of Prio, but it uses boolean circuits instead of the arithmetic ones for the proofs. This technique reduces the client's computational burden by up to two orders of magnitude (or more depending on the aggregation function) while keeping servers costs similar to those of the original Prio. The usage of boolean circuits is for proving that users' data falls within a correct range, and, later, if needed, the circuit can be converted to an arithmetic one to compute other kind of aggregation functions. While the usage of boolean circuits is very efficient, the conversion from boolean to arithmetic (for functions other than AND, OR, MAX, and MIN) can be costly, and overall only moderately increase efficiency when compared to the original Prio.

Prio2 [AG21] Prio2 works in the same way as the original design of Prio, but extends the original scheme to achieve a degree of differential privacy to the output via amplification of small noise at the users' side. The different communication entities of the original scheme have been detailed to execute specific functionalities. Prio2, for example, uses the notion of a "Leader", which is an entity responsible for coordinating (while not being able to neither see nor modify shares): receives encrypted shares, distributes them to the helpers, and orchestrates the process of verifying and computing the aggregation. "Helpers" (or "Facilitators") are responsible for executing the scheme as instructed by the leader. It further uses a variant of the original secret-shared non-interactive proofs, which improves the client-to-server communication cost; but it is only specified for their case.

Prio3 [BBC+19, GPRW22] Prio3 works in the same way as the original design of Prio and follows the variations introduced by Prio2 (without the explicit usage of DP). The main goal of Prio3 is to generalize the techniques of Prio2 for all kinds of aggregation functions (they use an arithmetic circuit called "Fully Linear Proof (FLP)"). It is worth noting also that with Prio3, the leader can start aggregating data proactively before all data in a batch is received.

This scheme is currently considered for standardization at the IETF [GPRW22].

4 STAR [DSQ+21]

A different approach for being able to compute aggregation functions without learning individual data is to use the privacy notion of *k-anonymity* [Swe02]. As defined by [Swe02]:

Definition 4.1 (*k-anonymity*)**.** . Let $RT(A_1, ..., A_n)$ be a table and QI_{RT} be the quasi-identifier associated with it. RT is said to satisfy *k-anonymity* if and only if each sequence of values in $RT[QI_{RT}]$ appears with at least k occurrences in $RT[QI_{RT}]$.

In a broader sense and more related to the problem at hand, the idea is to learn only data sent by *k-clients* (*k-heavy-hitters*). The server, then, only learns any data from a client if there are at least $k-1$ other clients submitting it. This approach prevents the data collector from learning uniquely identifying (or uniquely co-occurring patterns of) data from a unique client. This specific definition, that works on a $k-1$ notion, could be referred to as *k-anonymity* **threshold aggregation systems**.

One type of these schemes is STAR [DSQ+21], which prioritizes efficiency. In STAR, each client constructs a ciphertext of their data (and any auxiliary data), using an encryption key derived deterministically from either i) any randomness present in the client; and ii) additional randomness provided by a "randomness server" (note that this server never leans any of the clients data). Then, the client sends: the ciphertext, a k-*out-of-N* secret share of the randomness used to derive the encryption key, and a deterministic tag informing the server which shares to combine. The aggregation server, in turn, organizes the shares into subsets depending on the specified tags, and recovers the encryption keys from those subsets of size $\geq K$.

STAR aims to:

- Be implemented practically and easily by a wide array of projects: given its low financial costs and usage of "boring" cryptography.
- Client privacy: as defined by *k-anonymity* threshold aggregation.
- Correctness: the system must provide correct aggregation.
- Low financial costs: the scheme has low monetary costs (and it is the first one to take this concern into account).
- Achievable trust requirements: data aggregation systems that rely on multi-round interactions with a non-colluding party are expensive to maintain.
- Avoid usage of trusted hardware.
- Limit cryptographic complexity.
- Allow for other kinds of inputs other than numerical (unlike the Prio-based schemes).

4.1 Attacks and Limits

STAR does not try to prevent:

- Prevention of Sybil attacks, which are present in all threshold aggregation protocols.
- Leakage-free cryptographic design.

Note also that the aggregation server learns which clients share the same data: it leaks the subsets of clients that share equivalent measurements. STAR, nevertheless, avoids prefix-based leakage.

4.2 Reducing leakage in STAR

As stated, in STAR, the aggregation server could learn equivalent measurements that clients share. It also allows for identity leakage: the link between client identity and their input is unbroken. In the cases where maintaining this link is unuseful, STAR can be combined with:

- Oblivious proxies: an oblivious/anonymizing proxy will strip client identifying information (like IP addresses) from HTTP requests in such a way that the aggregation server learns nothing about the client identity. Note that using some of these anonymizing proxies (like Tor) incur on performance overheads.
- Oblivious HTTP: a proxy with similar anonymization properties as the above but with little performance overhead [TW21]. Such proxies are intended to be standardized by the IETF, and to be run by independent entities.

Scheme	Type of data	Privacy Notion	Robustness Notion	Trust
Prio	Only numeric	f-*privacy* (or $\tilde{f}$-*privacy*)	Preserved only in the face of adversarial clients	At least, one server should be honest
Prio+	Only numeric (validation is boolean)	f-*privacy* (or $\tilde{f}$-*privacy*)	Preserved only in the face of adversarial clients	At least, one server should be honest
Prio2	Only numeric	f-*privacy* (or $\tilde{f}$-*privacy*)	Should inherit from Prio (needs review)	At least, one server should be honest
Prio3	Only numeric	f-*privacy* (or $\tilde{f}$-*privacy*)	Should inherit from Prio (needs review)	At least, one server should be honest
STAR	All types	Threshold k-anonymity	In all cases up to a leakage parameter	All parties can be untrusted
POPLAR	All types	f-*privacy* (or $\tilde{f}$-*privacy*)	Preserved in the face of adversarial clients, and up to a leakage parameter for one malicious server.	At least, one server should be honest

Figure 1: Comparison of properties of different schemes.

4.3 Related Schemes

POPLAR [BBCG+21] POPLAR is a scheme very similar to Prio which allows for finding the most popular strings among a collection of clients, as well as counting the number of clients that hold a given string. POPLAR targets a similar problem as STAR: the *private heavy-hitters* problem and the *private subset-histogram problem* (in which the servers want to count how many clients hold strings in a particular set without revealing the set to the clients). The former problem means that: there are an unbounded number of clients and a small set of "collectors" (servers); each client holds a string, and, for some threshold $t \in N$, the servers recover every string that more than t clients hold.

POPLAR requires two non-colluding data-collection servers that n clients communicate with. It preserves client privacy as long as one of the two servers is honest (note that the server(s) may collude with an unbounded number of malicious clients). It preserves correctness against any number of malicious clients.

The scheme requires clients to secret-share or distribute a point function (-"incremental distributed point functions"- evaluating to 1 on their chosen value, and 0 elsewhere) between the two servers. The servers then combine shares of multiple point functions and reveal the heavy-hitters. The scheme is not very efficient: for 400,000 clients each holding a 256-bit string, for example, it takes the two servers 54 minutes to compute the *k-heavy-hitters*. It also leaks all heavy-hitting prefixes, and all information leaked by the multi-set of honest client inputs. In order to reduce leakage, the authors recommend using local DP.

This protocol is also considered for standardization at the IETF [GPRW22].

5 Comparison

While all the schemes presented achieve a degree of the intial privacy definition ("access to a statistical database should not enable anyone to learn anything about a user that could not be learned without access"), there are difficult to compare and to attest the exact level at which they approach this definition. Note also, that they all present a common leakage: identity-linkage attacks, unless they use anonimization/oblivious proxies/mechanisms.

In this section, we will only compare the Prio-based schemes and the "heavy-hitters" ones, as, the majority of them, claim to be able to apply DP on top. The comparisons can be seen in Figure 1 and Figure 2.

Scheme	**Leakage**	**Expressive Functionality**	**Efficiency**	**Monetary Cost**
Prio	Depending on the aggregation function (on the majority, no leakage)	Can be used with multiple aggregation functions	Slow client-to-server communication	
Prio+	Depending on the aggregation function (on the majority, no leakage)	Can be used with multiple aggregation functions (but it is costly to translate boolean circuits to arithmetic)	Improved client-to-server communication	
Prio2	Depending on the aggregation function (on the majority, no leakage)	Can be used with multiple aggregation functions	Improved client-to-server communication	
Prio3	Depending on the aggregation function (on the majority, no leakage)	Can be used with multiple aggregation functions	Improved client-to-server communication	
STAR	The server learns which clients share the same measurement	Limited use of different aggregation functions	Fast	Cheap
POPLAR	Leakage of all heavy-hitting prefixes	Limited use of different aggregation functions	Slow client-to-server communication	Costly

Figure 2: Comparison of costs, functionality and leakage of different schemes.

6 User Expectations

While all of the described schemes provide a degree of privacy, they still work with data provided by users. On many occasions, this data is provided without the explicit knowledge or consent from the users. Furthermore, users know little about the privacy properties of said schemes, or even if an specific scheme is used.

As stated in [CKR21], in regards to DP, "while DP is mathematically elegant and computationally efficient, it can be difficult to understand. Not only is DP typically defined mathematically, the privacy protections provided by DP are not absolute and require contextualization. DP does not provide binary privacy (i.e., private or not private), but instead provides a statistical privacy controlled by unitless system parameters that are difficult to interpret (...). Additionally, DP can be deployed in different security models, and the choice of model has significant impact on the types of adversarial behavior the system can tolerate."

While there has been some insights on the legal and ethical concerns and opportunities of DP [NW18, CD18, OK20], in the design of these schemes, the voice of the end-user is notably absent (this absentee is also present in the design of the other schemes presented). It begs to ask: Do users understand that their data is collected in a privacy-preserving manner? Can users consent to sharing or remove themselves from a system that uses x scheme? Do they understand the notion of privacy that is given by an x scheme? Do they know the used scheme and the limitations of it?

The findings of [CKR21] suggest that users care about data disclosure and the privacy of it; but, giving them a "random" definition of DP does not make them more willing to share their data. We argue that the same finding can be applied to other schemes: users seem to care about the privacy of their collected data but fail to understand the degree/type/notion of privacy given by a system.

This misunderstanding seems not to be a failure of the user, but rather a failure on the system's side (and of the design of the scheme) on creating a proper explanation (or, sometimes, on providing an explanation at all) of the privacy given. Schemes as the ones presented must provide user-friendly explanations of themselves (and the privacy they give), and allow for easy user consent and removal from their systems if wanted. As detailed in regards to Content Moderation [KKL+22], schemes must emphasize user agency; must be explicit about the exact properties they guarantee; and any change to either the scheme/property needs user notification, consent and opt-out. This must be an ingrained consideration of the schemes rather than an application-specific or architectural option.

Furthermore, the ability to opt-out and disclosure of what the data will be used for must always be present. Even if a scheme preserves privacy, that individual preservation of privacy is not enough for users to be willing to participate. The "privacy-preserving" data can be used for, while not targeting users individually, targeting groups (surveys meant to target non-male genders, for example). In these cases, individual privacy is preserved but group privacy is not.

7 Acknowledgements

Many thanks to Alex Davidson, Damien Desfontaines and Shivan Kaul Sahib for all the reviews, feedback and support on this informal note.

References

[AG21] Apple and Google. Exposure notification privacy-preserving analytics (enpa). White Paper, 2021. https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ENPA_White_Paper.pdf.

[AGJ+21] Surya Addanki, Kevin Garbe, Eli Jaffe, Rafail Ostrovsky, and Antigoni Polychroniadou. Prio+: Privacy preserving aggregate statistics via boolean shares. Cryptology ePrint Archive, Report 2021/576, 2021. https://eprint.iacr.org/2021/576.

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Heidelberg, August 2019.

[BBCG+21] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. Cryptology ePrint Archive, Paper 2021/017, 2021. https://eprint.iacr.org/2021/017.

[BEM+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, oct 2017.

[CD18] Rachel Cummings and Deven Desai. The role of differential privacy in gdpr compliance. 2018.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 259–282, Boston, MA, March 2017. USENIX Association.

[CKR21] Rachel Cummings, Gabriel Kaptchuk, and Elissa M. Redmiles. "I need a better description": An investigation into user expectations for differential privacy. *CoRR*, abs/2110.06452, 2021.

[Dal77] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15(429-444):2–1, 1977.

[Des21] Damien Desfontaines. Why differential privacy is awesome. Personal website, 2021. https://desfontain.es/privacy/differential-privacy-awesomeness.html.

[DP20] Damien Desfontaines and Balázs Pejó. SoK: Differential privacies. *PoPETs*, 2020(2):288–313, April 2020.

[DSQ$^+$21] Alex Davidson, Peter Snyder, E. B. Quirk, Joseph Genereux, and Benjamin Livshits. STAR: distributed secret sharing for private threshold aggregation reporting. *CoRR*, abs/2109.10074, 2021.

[Dwo06] Cynthia Dwork. Differential privacy (invited paper). In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 1–12. Springer, Heidelberg, July 2006.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1054–1067. ACM Press, November 2014.

[GPRW22] Tim Geoghegan, Christopher Patton, Eric Rescorla, and Christopher Wood. Privacy preserving measurement. IETF draft, 2022. https://www.ietf.org/archive/id/draft-gpew-priv-ppm-01.html.

[IET21] IETF. Privacy preserving measurement (ppm) working group. IETF working group, 2021. https://datatracker.ietf.org/wg/ppm/about/.

[KKL$^+$22] Seny Kamara, Mallory Knodel, Emma Llansó, Greg Nojeim, Lucy Qin, Dhanaraj Thakur, and Caitlin Vogus. Outside looking in: Approaches to content moderation in end-to-end encrypted systems, 2022.

[NW18] Kobbi Nissim and Alexandra Wood. Is privacy privacy? *Philosophical Transaction of the Royal Society A*, 376(2128), 2018.

[OK20] Daniel L. Oberski and Frauke Kreuter. Differential Privacy and Social Science: An Urgent Puzzle. *Harvard Data Science Review*, 2(1), jan 31 2020. https://hdsr.mitpress.mit.edu/pub/g9o4z8au.

[Swe02] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, oct 2002.

[TW21] Martin Thomson and Christopher Wood. Oblivious HTTP. IETF draft, 2021. https://www.ietf.org/archive/id/draft-thomson-ohai-ohttp-00.txt.

[War65] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. PMID: 12261830.