Decentralised, Scalable and Privacy-Preserving Synthetic Data Generation

Vishal Ramesh, Rui Zhao, Naman Goel

University of Oxford

Abstract

Synthetic data is a promising solution for use-cases such as privacy-friendly data release and complementing real data to train machine learning algorithms that are more fair and robust to distribution shifts. However, there is limited work on responsible, ethical and trustworthy synthetic data generation systems. Recently, Zhao et al. (2023) proposed a general-purpose decentralised computation system, called Libertas, that allows individuals to autonomously participate in joint computations over their data without relying on a trusted centre. Libertas uses Solid (Social Linked Data) and MPC (Secure Multi-Party Computation) to achieve this goal. Solid is a specification that lets people store their data securely in decentralised data stores called Pods and control access to their data. MPC refers to the set of cryptographic methods for different parties to jointly compute a function over their inputs while keeping those inputs private. Thus, Libertas can also be used to generate synthetic data from real data of individuals in a responsible way, by ensuring contributor autonomy, decentralisation and privacy. However, the scalability of this system remains limited due to the high computation and communication costs in MPC. In this paper, we show how one can use Trusted Execution Environments (TEEs) to address the scalability challenge. TEEs such as Intel SGX rely on hardware based features for confidentiality and integrity of code and data. We discuss a principled approach for integrating TEEs within the Libertas architecture for scalable differentially private synthetic data generation, and support our analysis with rigorous empirical results on simulated and real datasets and different synthetic data generation algorithms.

1 Introduction

In today's data-driven world, the need for large and diverse datasets is greater than ever. These datasets are instrumental in training machine learning models, conducting research, and fuelling innovation across various domains, from healthcare to finance and beyond. However, the acquisition and sharing of real-world data in a responsible way have become increasingly challenging due to concerns about privacy, security and the potential misuse of sensitive information.

In response to these challenges, synthetic data generation has emerged as a promising solution (Jordon et al. 2022). Synthetic data refers to artificially generated data that mimics the statistical properties of real-world data. It offers a way to balance the need for data-driven insights and open availability with protecting individual privacy (Emam, Mosquera, and Hoptroff 2020). Beyond privacy, synthetic data approaches are also being actively explored to overcome the limitations and shortcomings of real data for building more robust artificial intelligence (Xu et al. 2018; Brown 2020).

There exists a rich body of research dedicated to the algorithmic intricacies of synthetic data generation (Hu et al. 2023). The focus remains on development of novel algorithms for generating synthetic data with properties closely mirroring those of real-world data, addressing challenges such as preserving statistical distributions, correlations, and structural features while guaranteeing individual privacy. However, while these algorithmic aspects are crucial, for synthetic data to be truly responsible, trustworthy, and privacy-preserving, it is necessary to take a holistic perspective. This encompasses not only the algorithms used for data synthesis but also the entire lifecycle of data, starting with the contributors of the real data. Contributor autonomy should be one of the central tenets of responsible synthetic data generation. Contributors, who provide the real data, should have a significant degree of control and decision-making power throughout the synthetic data generation process. This autonomy ensures that their interests and concerns are taken into account, ultimately promoting trust and ethical data handling practices.

Further, a common assumption in most synthetic data generation approaches is the existence of a central curator. There are several problems with this approach. It expects diverse set of contributors to trust a common party to manage their (real) data, which may not be a practical and inclusive assumption. The lack of trust can lead to reduced or dishonest contributions, in turn lowering the quality of data (McSherry and Talwar 2007). The dependency on intermediaries can also introduce vulnerabilities and privacy risks. It can lead to a lack of transparency and accountability in the data generation pipeline.

Our research advances the state-of-the-art in *contributor-centric* and decentralised synthetic data generation approaches. Building upon the work of Zhao et al. (2023), we develop a system that runs differentially private synthetic data generation algorithms in a scalable, decentralised and private manner, while the contributors of original data retain control of their data and their decision to participate in a given synthetic data generation process. Our approach requires no alteration in the synthetic data generation algorithm and thus, provides the same level of accuracy and differential-privacy guarantees as the approaches that assume trusted center (unlike, for e.g., local differential privacy that has worse accuracy-privacy trade-off).

At the heart of this solution lies Solid (derived from *social linked data*), a specification pioneered by Sambra et al. (2016a). Solid lets people store their data securely in decentralised data stores called Pods and lets them control access to their data. Unlike current centralised Web and data architectures, where different applications and organisations collect and keep user data locked away, Solid decouples data from applications. A variety of user data can be stored in Solid pods. Different applications can read and write data to the user pod and provide services using the data, under user specified preferences. This is useful from a privacy perspective but more importantly, people get control over their data (further discussion on Solid in Section 3). Individuals may also be willing to contribute the data in their Pods to generate synthetic data for a variety of causes. However, for synthetic data generation, computations have to be performed over the data of multiple people. This is challenging without requiring a trusted center. The challenge is further complicated by the fact that Solid Pods lack any local computation capability. Zhao et al. (2023) proposed a modular architecture for integrating Secure Multi-Party Computation (MPC) with Solid, enabling arbitrary computations to be performed in a decentralised manner over data stored in Solid Pods. We build upon this recent research and show how differentially-private synthetic data can be generated in a scalable manner without compromising the autonomy and privacy of contributors. Instead of *only* relying on MPC, we offer a more scalable alternative that uses both MPC and Trusted Execution Environments (TEEs). We show how to implement different steps of synthetic data generation algorithms in MPC and TEEs, utilising MPC where it offers the most value and using TEEs elsewhere to overcome MPC's performance challenges. We analyse the scalability and privacy related strengths and limitations of the approach.

In the rest of this paper, we will discuss background concepts, describe the details of our design, explain our choices, and provide empirical evaluation of our system and comparison with baselines. We finally conclude the paper with discussion on future work.

2 Problem Description

A synthetic dataset is a substitute for an original dataset that has the same format and reflects the statistical properties of the original dataset, without reproducing the records in the original. In our settings, different individuals hold their own data records in a decentralised manner. We are interested in generating synthetic data that mimics the properties of a dataset containing records of all individuals who are willing to participate in the synthetic data generation process, without requiring a trusted centre and ensuring that individuals have full autonomy and privacy. During the synthetic data generation process, the original data records should be kept confidential with the respective individuals and not revealed to others. This is also referred to as input privacy. From the synthetic data, an adversary should not be able to make undesired inferences about the original data records or the individuals. This is also referred to as output privacy.

In this paper, we consider synthetic data generation algorithms that provide differential-privacy guarantees (Dwork et al. 2006) for output privacy. However, the algorithms to generate differentially-private synthetic data, by themselves, provide neither decentralisation nor input privacy. We therefore need additional mechanisms. For decentralisation, we need decentralised storage, access control and reduced dependence on a trusted center for the computation steps involved in the synthetic data generation algorithms.

Zhao et al. (2023) proposed a novel architecture, Libertas, to integrate Personal Data Stores with Secure Multi-Party Computation (MPC). Personal Data Stores provide individuals decentralised storage and access control for their data records. Secure Multi-Party Computation (MPC) provides input privacy and allows individuals to participate in arbitrary computations over their data records collectively without relying on a trusted center. Decentralised synthetic data generation with input and output privacy is thus one example of a use-case that can be realised using Libertas. However, the scalability of this approach remains a challenging problem due to high computation and communication costs of the MPC protocols. We address this technical challenge in our work. We first briefly provide background on personal data stores, secure multi-Party computation, Libertas architecture, differential-privacy and the synthetic data generation algorithms that are of interest to us in this paper. Appendix A contains an expanded background section, with details that can not be covered here due to space constraints.

3 Background

Personal Data Stores and Solid

In response to increasing centralisation of data and loss of user autonomy, several decentralised data architectures have been proposed that aim to give users more control over their personal data. One such approach that has gained considerable traction is Personal Data Stores (PDS) (Sambra et al. 2016b; Mansour et al. 2016).

Under this decentralised data paradigm, users store their data in a Pod (Personal Online Datastore). A Pod provides granular control to users over who can access which data as well as secure transmission of data for authorised requests. The apps and services that they use (for e.g. various Web based applications, online social platforms, health service applications etc), also read and write data to the user's Pod. Users can transfer their data from one Pod service provider to another, including the option to self host. This unlocks a whole new paradigm for app design based on interoperability, a key design aspect (Linked Data) of Solid. The decoupling of data and apps reduces the need for expensive data collection across platforms, thereby benefiting developers as well. Pods support both structured and unstructured data.

There are several PDS projects ranging from purely academic implementations to those with commercial offerings, among them openPDS (De Montjoye et al. 2014), Databox (Mortier et al. 2016), and Solid (Mansour et al. 2016). These vary in their protocol design and the demands they place on the features a Pod must support. We focus on Solid because of its open design based on existing Web standards and growing adoption. Appendix A contains more details about Solid.

While Solid provides users control of their data and allows Web applications to read/write based on user preferences, it leaves open a challenging problem. When computations have to be performed over the data of multiple users, Solid does not specify how this can be done. Solid Pods also do not have any local computation capability in the current specification. This problem has recently been addressed by the Libertas architecture proposed by Zhao et al. (2023). Before discussing the Libertas architecture, we provide a brief introduction to Secure Multi-Party Computation (MPC).

Secure Multi-Party Computation (MPC)

Secure multi-party computation (MPC) (Evans et al. 2018) is a set of cryptographic protocols for ensuring that one or more parties can participate in the decentralised computation of a function over some privately held inputs such that no party learns anything about the private input of another party. The only information that can be inferred about the private input is whatever can be inferred from the output of the function alone. [1] More formally, given n parties P_1, ..., P_n, their corresponding inputs x_1, ..., x_n and a function f, an MPC protocol computes $y = f(x_1, ..., x_n)$ without revealing input x_i to party P_j where $i \neq j$.

In the past few decades, there has been significant research on the development of efficient MPC protocols. The cryptographic primitives that are typically used to implement MPC are secret sharing schemes (Shamir 1979), garbled circuits (Yao 1986), and oblivious transfer (Rabin 2005). We focus on secret sharing. Further, MPC protocols differ in their security assumptions regarding the adversary (further background in Appendix A). In our experimental analysis, we use protocols with varying assumptions.

Libertas Architecture by Zhao et al. (2023)

Libertas architecture, proposed by Zhao et al. (2023), extends Solid (while adhering to the Solid protocol) to enable performing computations using MPC on decentralised data held in individuals' Pods. Appendix A provides further details about Libertas and the roles different agents play in this architecture. Libertas addresses three main limitations of Solid. First, a user can allow an agent, that has not been granted access to some data directly, to utilise the data as input in a multiparty computation procedure, without compromising the user privacy. Second, it overcomes the limitations arising due to the lack of computation capability in Solid Pods, by a mechanism to delegate participation in the joint computation while users retain control. Third, under a direct-decentralised model where data providers are also the protocol players, MPC scales poorly as the number of players grow. Conversely, the delegated-decentralised model used in Libertas, with a smaller number of agents acting as players, offers significantly better scalability. But fundamentally, costs due to MPC protocols remain challenging.

Differential-Privacy

Differential privacy (Dwork et al. 2006) is a rigorous mathematical definition of privacy that has become the de facto standard for output privacy. Many tech companies such as Apple, Google (Erlingsson, Pihur, and Korolova 2014), and Microsoft (Ding, Kulkarni, and Yekhanin 2017) have built privacy-aware systems that use differential privacy. National Statistical Offices too have begun releasing censuses that incorporate differential privacy. Differential privacy bounds the impact any one individual can have on the output of an algorithm. More formally, a randomised mechanism $M : D \rightarrow \mathcal{R}$ satisfies (ϵ, δ)-differential privacy (DP) if for all neighboring (differing on one element) datasets $D, D' \in \mathcal{D}$ and all subsets of possible outputs $\mathcal{S} \subseteq \mathcal{R}$,

$$Pr[M(D) \in \mathcal{S}] \leq exp(\epsilon) Pr[M(D') \in \mathcal{S}] + \delta$$

Synthetic Data Generation Algorithms

Any dataset release (including synthetic data) that meaningfully preserves statistics of the original dataset will result in a privacy loss. Thus, there exists a privacy-utility trade-off. We are interested in synthetic data generation algorithms with differential-privacy guarantees for output privacy. There exists several algorithms for differentially-private synthetic data generation. In this paper, we focus on marginal-based approaches to synthetic data generation that have achieved considerable success in practice. Marginal-based algorithms won the 2018 NIST Differential Privacy Synthetic Data Competition, for instance, and performed competitively in other iterations of the contest (McKenna, Miklau, and Sheldon 2021). Further, we focus on workload aware algorithms. A workload is a collection of queries that a synthetic dataset must preserve well. These are the queries that downstream users of the synthetic dataset might be interested in answering. In our experiment analysis, we use the following state-of-the-art algorithms for synthetic data generation: the Multiplicative Weight Exponential Mechanism (MWEM) algorithm (Hardt, Ligett, and McSherry 2012), the Probabilistic Graphical Model (PGM) approach (McKenna, Sheldon, and Miklau 2019), the Relaxed Adaptive Projection (RAP) (Aydore et al. 2021) or Relaxed Tabular approach and the local consistency methods such as Generative Networks with the Exponential Mechanism (GEM) (Liu, Vietri, and Wu 2021) and Approx-Private-PGM (APPGM) (McKenna et al. 2021). Readers can check further discussion about the algorithms in Appendix A; while not critical, some knowledge about these will be useful for easier reading of the rest of the paper.

4 Proposed Approach for Scalable Decentralised Synthetic Data Generation

To address the issue of high computation and communication costs incurred in Libertas, due to its dependence on Secure Multi-Party Computation, we propose an adaptation

[1] Differential-privacy, to be discussed next, addresses this limitation and thus, MPC and differential privacy are complementary privacy mechanisms.

tailored for differentially-private synthetic data generation. Instead of *only* relying on MPC, we propose a more scalable alternative that uses both MPC and Trusted Execution Environments (TEEs) (Jauernig, Sadeghi, and Stapf 2020; Li et al. 2023). A Trusted Execution Environment (TEE) is a secure area in the main processor of a device; examples of TEEs include Intel SGX (Costan and Devadas 2016) or AMD SEV (AMD 2020). We implement different steps of synthetic data generation algorithms in MPC and TEEs, utilising MPC where it offers the most value and using TEEs in other parts of the pipeline to overcome the performance challenges.

Steps

1. Anonymisation: In the first step, data providers can remove personally identifiable information (PII) from data records. As users of Solid can set access control to each individual data item in their records, this step can be trivially performed by the data provider through the access control mechanism.
2. Client-Side Aggregation: The next step is for the chosen encryption agents in the Libertas architecture to run the client code. This involves reading the access authorised data from the Pod of participating data provider and converting it into a histogram representation, possibly performing any binning along the way. This setup allows for settings where each data provider manages more than one data record.
3. MPC-Based Histogram Aggregation: The third step is for the computation agents in the Libertas architecture to run the MPC code. This involves reading shares of the individual histogram representations from the clients and computing the aggregate histograms through an addition of arrays / matrices operation. No agent receives the result (aggregate histograms) yet.
4. Differentially-Private Synthetic Data Generation: The next step is to execute the remaining steps of respective differentially-private synthetic data generation algorithm. We do not perform this step in MPC; more details about this step as in the following text.

Separating Noise Addition and Data Generation from MPC

We have proposed that separating the preliminary, aggregating histogram step (in MPC) from the subsequent steps could improve the scalability of privacy friendly synthetic data generation in decentralised contexts. For example, depending on the synthetic data generation algorithm, these subsequent steps may involve adding noise to the histograms using the appropriate mechanism based on the designated privacy budget to ensure the measurements are differentially-private and then run a generation algorithm such as PGM on the measurements that can be released to the MPC app user. Similarly, in the case of MWEM algorithm, this involves running the iterative algorithm where measure and generate steps are more interweaved. Note that we do not modify the steps of the algorithm or propose a new algorithm. The aggregate histograms that are needed as input in the synthetic data generation algorithm are computed from the individual data points using MPC. This aggregated histogram is then used by the algorithm but the algorithm itself is not implemented in MPC (unlike (Zhao et al. 2023; Pereira et al. 2022)). We therefore inherit the differential-privacy privacy guarantees from the respective algorithm for the final output. Let us now address an intermediate vulnerability that is introduced as a result of this separation from MPC and how we alleviate it.

In particular, between steps 3) and 4), we need to trust a computation agent with the aggregated histogram, who then runs a differentially-private synthetic data algorithm. If this agent is compromised, an adversary can access non-private aggregated histogram.

What the adversary could learn about individuals participating in the data analysis depends on the nature of the dataset. For example, in non-homogeneous populations, identifiable outliers in the histogram could compromise the privacy of participants from certain subgroups. This is of particular concern if it involves protected characteristics or individuals from marginalised groups. To mitigate against these attacks, we use the following methods to strengthen the privacy assurance of our approach.

1. Random Selection In a naive implementation of MPC, all parties receive the same output, i.e. the result of evaluating the arithmetic circuit. However, it is possible for different parties to receive different outputs or no output at all. We delegate the subsequent tasks to a randomly chosen computation party instead, i.e. only the chosen party receives the aggregated histogram output. The random selection is part of the MPC circuit as a joint random number generation procedure and hence adherence is enforced by the MPC protocol used (and with the underlying security parameters as discussed in Appendix A).

Random selection reduces the chance of a malicious party getting the non-private aggregated histogram. Furthermore, it is impossible for the adversary to cheat the random number generation through its choice of inputs. It is worth recalling that MPC still protects the input data (individual records) as long as the protocol security assumptions hold, thereby limiting the attack surface.

2. Trusted Execution Environment A Trusted Execution Environment (TEE) (Jauernig, Sadeghi, and Stapf 2020; Li et al. 2023) is a secure area in the main processor of a device; examples of TEEs include Intel SGX (Costan and Devadas 2016) or AMD SEV (AMD 2020). With appropriate complementary mechanisms, TEEs are a promising approach to maintaining the confidentiality and integrity of code and data located inside them, without relying on a trusted operating system. Therefore, many cloud service providers such as Azure offer TEEs based "confidential computing" feature for sensitive use-cases.

The MPC histogram generation proceeds as before. Additionally, a computation agent is nominated to serve as the *enclave agent*. The enclave agent can be nominated using the MPC based random selection as discussed above.

Before dispatching the code for the remaining synthetic data generation steps to the enclave agent, the encryption party optionally verifies the identity of the enclave through remote attestation. Remote attestation, which allows a remote party to verify the contents of the program that is running in the enclave with a certificate generated by the underlying hardware. After the MPC has concluded, the output (aggregated non-private histogram) is sent to the enclave via a secure channel for executing rest of the steps of differentially-private synthetic data generation process. Typically an extension of SSL/TLS which forces the enclave endpoint to incorporate its attestation proof with its certificate serves as the channel (Knauth et al. 2018).

We provide an overview of our approach in Figure 1, an adapted version of the one presented in Zhao et al. (2023).

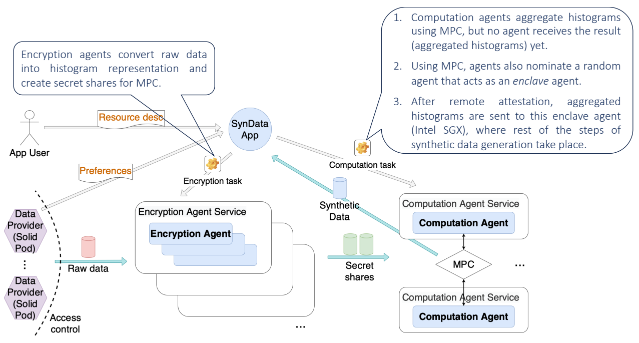

Encryption agents convert raw data into histogram representation and create secret shares for MPC.
1. Computation agents aggregate histograms using MPC, but no agent receives the result (aggregated histograms) yet.
2. Using MPC, agents also nominate a random agent that acts as an *enclave* agent.
3. After remote attestation, aggregated histograms are sent to this enclave agent (Intel SGX), where rest of the steps of synthetic data generation take place.
App User
Resource desc
SynData App
Preferences
Encryption task
Computation task
Data Provider (Solid Pod)
Data Provider (Solid Pod)
Access control
Raw data
Encryption Agent Service
Encryption Agent
Synthetic Data
Secret shares
Computation Agent Service
Computation Agent
MPC
Computation Agent Service
Computation Agent

Figure 1: For generating differentially-private synthetic data from personal data stored in Solid pods, we adapt the Libertas architecture such that MPC is used for histogram aggregation and nominating a random enclave/TEE agent only. Subsequent steps of synthetic data generation are executed inside the TEE (Intel SGX) after remote attestation.

Discussion Compared to MPC, the trust relationship needed in TEEs is that data providers trust the hardware manufacturer. We note that while setting up an enclave all the dependencies must be loaded into the enclave file system ahead of its creation. This limits the libraries that the code running in the enclave can access. Thus for convenience, the services offered by the enclave agent is baked into the architecture, at the cost of versatility. Furthermore, SGX is vulnerable to side channel attacks (Nilsson, Bideh, and Brorsson 2020). However, since the first step in our solution uses MPC, we not only keep the solution decentralised but the risks due to SGX's vulnerabilities are also significantly reduced. Thus, in order to utilise the TEEs within the Liberetas architecture in a principled manner, we separate the computation steps in the algorithm into steps that should be executed in MPC and TEE while balancing the privacy, decentralisation and scalability trade-offs.

Assumptions The proposed approach inherits threat model assumptions from parent technologies and architectures including Solid (Sambra et al. 2016b; Mansour et al. 2016), MPC (Evans et al. 2018), MP-SPDZ (Keller 2020), Libertas (Zhao et al. 2023), differential-privacy (Dwork, Roth et al. 2014) and Intel SGX (Nilsson, Bideh, and Brorsson 2020). The assumptions include but are not limited to, encryption agents and computation agents in the Libertas architecture behaving as specified in the trust preferences of the data providers, and appropriate MPC protocol being used for honest majority or dishonest majority setting.

5 Empirical Evaluation

We implemented the above approach and performed comprehensive experiments to understand the strengths and limitations of the proposed approach. In this section, we will first discuss implementation related details, following by experimental setup and the results.

Implementation Details

Apart from the Solid development framework and the Libertas implementation (Zhao et al. 2023)), we use the following in our *open-source* implementation. [2]

MP-SPDZ MP-SPDZ is a framework for benchmarking multi-party computation tasks (Keller 2020). Similar to (Zhao et al. 2023), we use MP-SPDZ in our experiments because it enables easy testing of different protocols from a single high level program and for its comprehensive metrics.

Zhao et al. (2023) implemented the encryption agents using the client mechanism of the MP-SPDZ framework for secret sharing with players, which in turn is based on the SPDZ protocol (Damgård et al. 2016). We follow the same and we also use it for the enclave agent. Although it does not send (unlike encryption agents) any input to the computation parties, we use the *client* mechanism of MP-SPDZ to allow it to receive the aggregated histogram.

Gramine Gramine is a library OS that is used to run unmodified Linux applications (e.g. the Python interpreter) in an SGX enclave (Tsai, Porter, and Vij 2017). While the Intel SGX SDK exposes a low-level C/C++ interface for writing enclave applications, we use Gramine for portability and faster development cycles. We provide a manifest file with the associated source code to configure the application environment and isolation policies.

Marginal-Based Inference (MBI) The Marginal-Based Inference (MBI) library exposes procedures that take as input noisy measurements (e.g. differentially private marginals) and generates synthetic data (McKenna, Miklau, and Sheldon 2021). While selection of the right queries is an important question (McKenna et al. 2022), we restrict attention to investigating the scalability and MBI provides a generic interface with a range of implemented algorithms for comparison. As with MP-SPDZ, we use MBI for easy testing of different generation algorithms and comparability of benchmarking results.

[2]Source code as well as all the raw data from experimental results can be reviewed anonymously at https://tinyurl.com/aies-269/. The link will be replaced by an open-source github link in the camera-ready version of the paper.

Experimental Setting

Setup Our computational and encryption agents were deployed on a single server connected over a virtual LAN. This was to focus on the computational cost of the approaches while controlling for network factors such as latency and bandwidth that would affect the scalability. We acknowledge that these are important as the agents may be connected over a WAN, hence we report the size of data transfers in each computation.

We deploy 3 computation agent servers and 2 encryption agent servers, unless stated otherwise. This is because for honest majority protocols, 3 is the minimum number of players. Furthermore, Zhao et al. (2023) show that in a delegated-decentralised setting, MPCs with fewer players scale better. Hence, any improvement our approach offers in this setting should also translate to cases with greater number of players.

Platform The experiments were conducted on an Azure DV4 VM running on an Intel Xeon Platinum 8370C (2.8GHz) with 4 vCPUs and 32 GB RAM. This architecture includes support for the SGX instruction set and this VM series comes with SGX enabled in the BIOS. A distribution with Linux kernel of version at least 5.11 is used as it includes support for SGX drivers (we use Ubuntu 22.04).

We increased the maximum number of open files to 65536 as we otherwise quickly run out of file descriptor resources during our experiments. Nevertheless, in certain (e.g. dishonest majority) settings we encounter resource limitations and hence must limit the scope of our experiment to fewer data providers.

Benchmark Datasets For evaluating the proposed approach, we simulate different settings in which data providers have data records stored in their Solid pods. For the first part of our experiments, we use 1-dimensional simulated data records. Please note the distinction between 'simulated data' and 'synthetic data'. In our paper, synthetic data is the data produced by a differentially-private generation algorithm that takes real data from data providers as input. In the first part of our experiments, we simulate this real data using 'simulated data', which would be the input of synthetic data generation algorithm. The data was uniformly sampled from a range of 0 to 20. There are two scenarios that we consider here. The first is the fixed total data setting where we fix the total number of data records at 10000 and evenly distribute it among the data providers. The second is the variable total data setting where we fix the number of data records per data provider at 100. This latter situation gives us between 1000 and 100000 data records (the number of data providers range from 10 to 1000).

For the second part of experiments, where the performance of marginal-based generation algorithms under our approach is benchmarked, we use real-world datasets that are common benchmarks in the synthetic data generation literature. In particular, we rely on the Adult (Kohavi et al. 1996) and the Titanic (Encyclopedia Titanica 2023) datasets. There are a few preprocessing steps that we undertake before storing the data records in Solid Pods. Since many generation algorithms including MWEM assume discrete-valued data, we remove certain continuous values attributes and discretise others. We further remove data records with missing values.

Table 1: A summary of the real-world datasets used in the experiments (after preprocessing). Total domain size was rounded to one significant figure.

Dataset	**Records**	**Dimensions**	**Min-Max Domains**	**Total Domain Size**
Adult	48842	14	2-100	4×10^{17}
Titanic	713	7	2-90	2×10^{5}

We see in Table 1 a summary of the datasets after preprocessing. Both real-world datasets have much larger domains than the simulated data. As we shall see further in this section, the proposed approach scales efficiently in all these cases.

Since the number of instances in Adult (48842) is much larger than the maximum number of data providers we consider (1000), one can consider two different data distribution strategies, much akin to the difference between the variable total and fixed total datasets described before. In the first, each data provider receives a single instance from the Adult dataset. This is more in line with the traditional thinking of a Pod as being controlled by and holding a single individual's data. In the second, the records from Adult are distributed roughly equally among the different data providers.

Client Binning We use a pre-specified number of bins in histogram for inducing a manageable dimensionality. This conversion of individual data points to the bin format can be done either in the encryption server via client code or in the computation server via MPC code. The binning strategy is pre-agreed through client code. Results from (Zhao et al. 2023) find that performing binning in the client code results in a significant performance gain. We use this so-called *client-binning* optimisation in all our experiments involving MWEM.

Reported Metrics

1. **Time:** The total time taken to complete a given MPC computation, including any setup time, for e.g., establishing connections, fetching data, shares of data being received from the encryption server clients etc. While reporting this metric for our approach, we add the time for performing the measure and generate steps in SGX.
2. **Communication Rounds:** The number of rounds of communication required for the players to finish a given computation. This differs from the number of communications which is the number of rounds times the number of players (for the protocols we consider, may differ in other protocols).
3. **Local Data:** The amount of data being transmitted by a single player. Usually this value does not vary greatly between different players for the MPC protocols that we use. We report the local data measured at player 0 for consistency.

4. **Global Data:** The total amount of data exchanged during all communications between players over the course of the MPC computation.

Communication rounds, local and global data are standard metrics provided by the MP-SPDZ framework itself; we did not measure these metrics using a separate code or tool.

Empirical Analysis

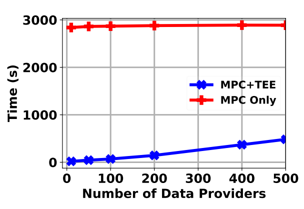
We now present results from experiments, comparing our proposed hybrid approach using MPC and SGX versus MPC only, for MWEM across various simulated datasets and MPC protocols with different adversarial assumptions. The MPC only implementation is taken from Libertas (Zhao et al. 2023), while the MPC and SGX implementation is ours.

We first discuss the results obtained on the simulated data and MASCOT protocol. We run the MASCOT protocol to simulate a dishonest majority setting (Keller, Orsini, and Scholl 2016).

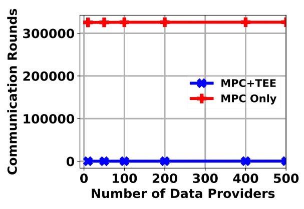
Figure 2 shows the results for the setting in which we fix the total number of data points among all data providers at 10000 and distribute them equally among them. MWEM is run with the client-binning optimisation using a fixed number of bins (10), epsilon value 2 ($\epsilon = 2$), and 30 iterations ($T = 30$). The reported time includes the time in communication setup. We observe that running MWEM with our approach takes less than 500 seconds, while MWEM with MPC only takes almost 3000 seconds even for just 10 data providers. While the computation time in both settings grows linearly with data providers, the faster growth in the MPC+TEE setting can be attributed to initial setup time where shares are received from the encryption agents, which increases with more data providers and may dominate a smaller computation time. Zhao et al. (2023) also discuss computation time and setup times separately and show that computation time indeed grows much slowly than the total time (computation + setup time). We also observed that the behavior w.r.t. local and global data may vary with the implementation of the MPC program but generally, grows slowly with the number of data providers. We zoom into the graphs by showing one observation point in Table 2. In the MPC only setting, the amount of global data exchanged to complete the computation is nearly 360 GB, compared to the much more manageable size of 357 MB in our approach. This is an orders of magnitude improvement.

Table 2: MASCOT with 10000 data itemso divided among 100 data providers. Time is rounded to 0.01; ± indicates standard deviation. MWEM on fixed total dataset (simulated) with 10 bins, $\epsilon = 2$, $T = 30$.

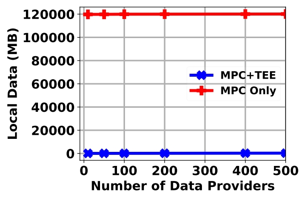
Approach	Time(s)	Rounds	Local Data(MB)	Global Data(MB)
MPC Only	2860.68± 9.29	325796	119936	359514
MPC+TEE	73.65 ± 14.96	208	118.98	356.87



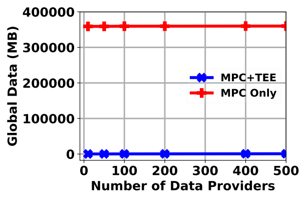
(a) Time



(b) Communication Rounds



(c) Local Data



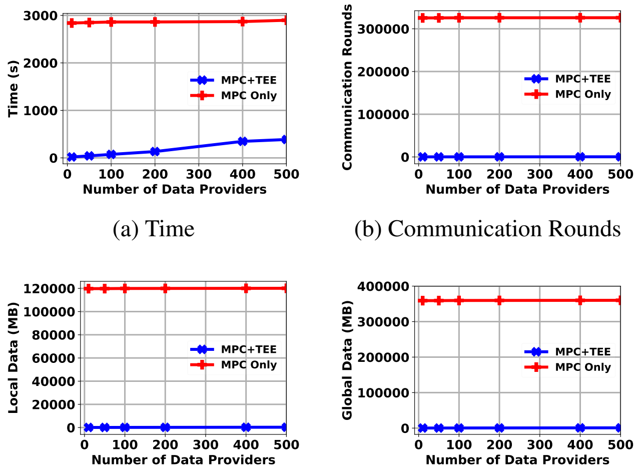
(d) Global Data

Figure 2: Comparison of MPC Only and MPC+TEE Approaches [MWEM; MASCOT protocol, fixed total data (simulated) i.e. 10000 data points divided equally among data providers; 10 bins, $\epsilon = 2$, $T = 30$.]

Figure 3 show the corresponding results for the setting when we fix the total number of data points per data providers at 100. Figure 2 and 3 show very similar results due to the fact that the number of data points contributed by data provider does not matter as much in our implementation because the client code converts raw data from data providers into histogram representation and the rest of the process only deals with the histogram representation. Therefore, the additional cost is mostly limited to the process of converting raw data into histogram representations and all further dominating costs are nearly independent of the number of data points.

We next draw attention to Figure 4 where we show the time taken with the number of iterations in the MWEM algorithm, an important hyper-parameter in any iterative algorithm. Clearly, time increases at a much faster rate in the MPC only approach compared to the hybrid MPC+TEE approach. The reason for similarity in the fixed data and variable data setting is the same as discussed previously.

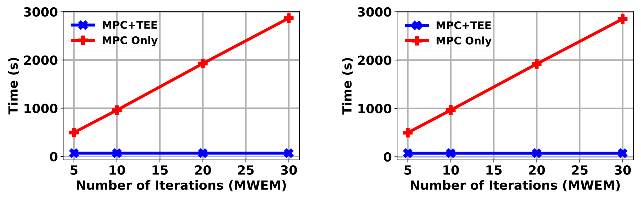
In Figure 5, we have plotted the error in MWEM, measured as the distance between the generated data distribution and the actual data distribution, against the number of iterations. The code was run without MPC or SGX as it does not affect the measured value. We use a dataset with a skewed distribution for this experiment. This is because we initialise MWEM with a uniform distribution and thus a skewed dataset would require more iterations for it to converge. We observe that the error continues to fall even at 140 iterations where it is around 0.1, suggesting that there is room for further convergence. This is an important property because in the MPC only setting, we found even at a modest 30 iterations, the amount of time approached 50 minutes and the amount of global data exchanged was 360 GB. Our results suggest that running it for further iterations would be impractical. Indeed if we double the number of



(a) Time (b) Communication Rounds

(c) Local Data (d) Global Data

Figure 3: Comparison of MPC Only and MPC+TEE Approaches [MWEM; MASCOT protocol, variable total data (simulated) i.e. 100 data points per provider; 10 bins, $\epsilon = 2$, $T = 30$.]



(a) Fixed total data (simulated) i.e. 10000 data points divided among 100 data providers.

(b) Variable Total Data (simulated) i.e. 100 data points per provider.

Figure 4: Comparison of time in MPC Only and MPC+TEE Approaches, for different number of iterations of MWEM. MASCOT protocol, 100 data providers, 10 bins, $\epsilon = 2$, $T = 100$.

bins to 20, as we do in figure on the right, we notice the convergence is slower. The appropriate number of bins in a practical implementation would depend on the number of attributes and the domain size but it is reasonable to expect that real world datasets, such as Adults and Titanic that we use in the next section, would preclude us from using a MPC only approach.

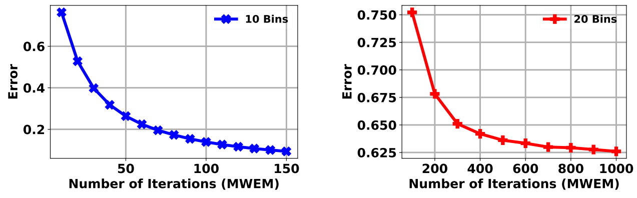
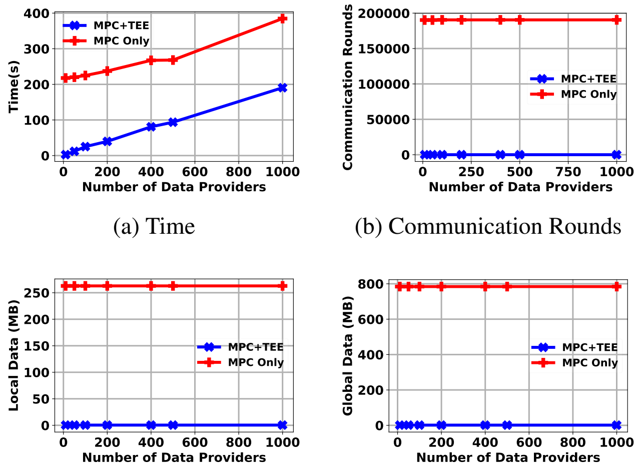


Figure 5: MWEM error i.e. difference between true and generated distribution, with different number of iterations of the algorithm on a skewed dataset using 10 bins and $\epsilon = 2$.

So far we have discussed results obtained using the MASCOT protocol (simulating dishonest majority setting). We



(a) Time (b) Communication Rounds

(c) Local Data (d) Global Data

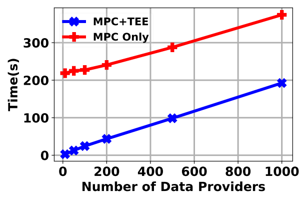
Figure 6: Comparison of MPC Only and MPC+TEE Approaches [MWEM; SHAMIR protocol, fixed total data (simulated) i.e. 10000 data points divided equally among data provoiders; 10 bins, $\epsilon = 2$, $T = 100$.]

now report the results for the Shamir protocol to simulate an honest majority setting. Results are shown in Figures 6 and 7 for fixed total data and variable total data settings using the simulated dataset. Being a more efficient protocol due to relaxed adversarial assumptions, we see some clear differences in the behaviour of the metrics under both approaches. Consequently we note that values for time, global and local data are significantly lower than in Mascot. This is despite the fact that the number of communication rounds (around 190000) is not that far off from the rounds in the Mascot setting (approximately 320000), indicating that communication is noticeably cheaper in an honest majority setting. We also observe that the time for both MPC only and MPC+TEE grows linearly at roughly the same rate in this case.

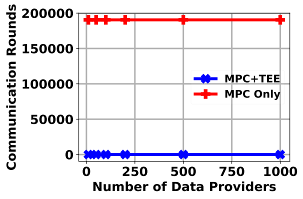
We observe in figure 8 that time taken grows much slower in the hybrid MPC+TEE approach, with the number of iterations of the MWEM algorithm. A similar trend, although not plotted, was observed for local and global data as well, with global data exceeding 1 GB at 140 iterations in the MPC only approach.

Benchmarks on Real Datasets

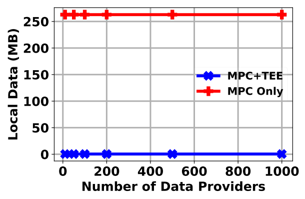
Having carefully analysed the difference between the MPC only and the MPC+TEE approaches using simulated datasets, we now benchmark the MPC+TEE approach using real-world datasets. In particular, we benchmark the performance of different synthetic data generation algorithms such as PGM and Local Consistency (`mbi.FactoredInference` and `mbi.LocalInference`) in the proposed MPC+TEE approach with common datasets like Adults and Titanic. In the figures in this section, the results of the MPC+TEE approach will also be more clearly visible since we will not include the MPC only approach which would increase the scale of Y-axis in the figures.



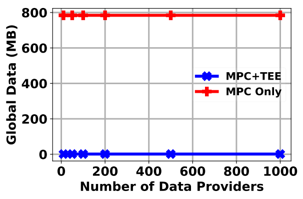
(a) Time



(b) Communication Rounds

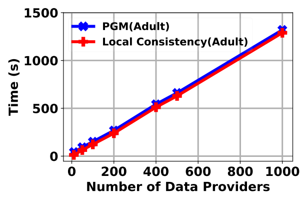


(c) Local Data

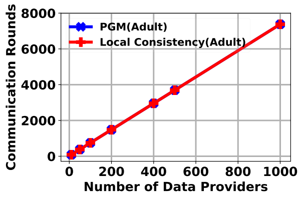


(d) Global Data

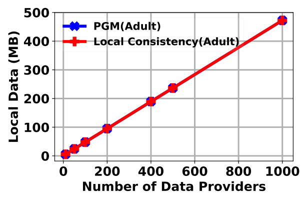
Figure 7: Comparison of MPC Only and MPC+TEE Approaches [MWEM; SHAMIR protocol, variable total data (simulated) i.e. 100 data points per provider; 10 bins, $\epsilon = 2$, $T = 100$.]



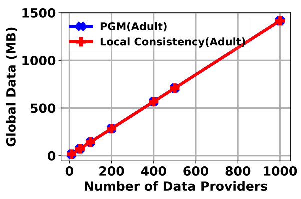
(a) Time



(b) Communication Rounds

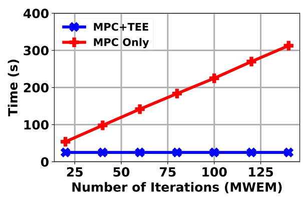


(c) Local Data

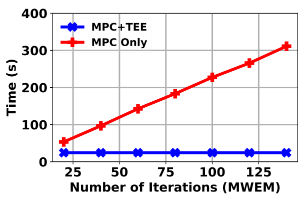


(d) Global Data

Figure 9: Performance of the MPC+TEE approach using PGM and Local Consistency generation algorithms (30 iterations); Adult dataset; SHAMIR protocol, one data point per provider.]



(a) Fixed total data (simulated) i.e. 10000 data points divided among 100 data providers.



(b) Variable Total Data (simulated) i.e. 100 data points per provider.

Figure 8: Comparison of time in MPC Only and MPC+TEE Approaches, for different number of iterations of MWEM. SHAMIR protocol, 100 data providers, 10 bins, $\epsilon = 2$, $T = 100$.

We first benchmark the performance of the MPC+TEE approach with the PGM algorithm on 10 two-dimensional marginals of the Adult dataset using the Shamir protocol. The reason we did not consider a larger number of marginals is because of memory limitations we encountered on our compute while running PGM rather than the scalability of the approach to real world high-dimensional datasets. Each data provider represented a single record from Adults. PGM was run for 30 iterations with no change to the other hyperparameters. The results are shown in Figure 9. Despite the large number of attributes (14), many of which have a large domain (maximum 100), our approach keeps the computation time at less than 1400 seconds even with 1000 data providers. While the amount of data exchanged grows linearly, it still remains at a manageable 1400 MB global data for 1000 providers.

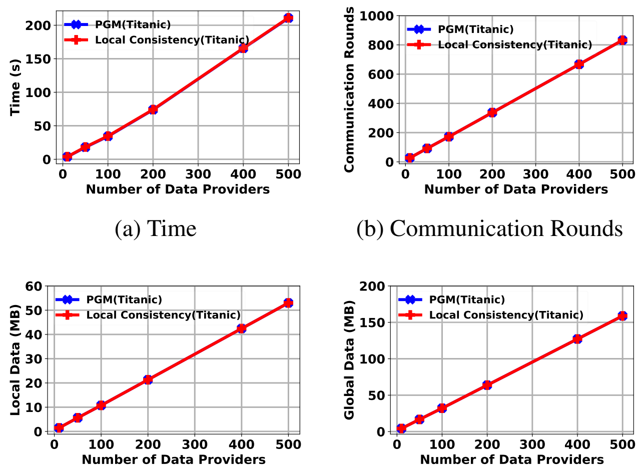
We also benchmark the performance of the MPC+TEE approach with PGM on the Titanic dataset using the Shamir protocol. We consider all two-dimensional marginals over the 7 attributes resulting in 21 histograms in total. Again PGM is run for 30 iterations. Each data provider represents a single record from Titanic. Results are shown in Figure 10. Despite considering a greater number of marginals than Adult, we see that it scales better on Titanic across all metrics. This can be attributed to the larger domain size of attributes. As seen in figure 10a the time taken for 500 data providers is around 210 seconds, compared to over 660 seconds in 9a for Adult.

Figures 9 and 10 also show the performance using Local Consistency method. While the differences are not significant to be visible in the figures, we observed that on both the Adult and the distributed Adult dataset, Local Consistency based generation takes approximately 30 seconds less time than PGM. On the other hand in the Titanic dataset, Local Consistency based inference takes about 1-1.5 seconds longer than PGM. We also observed, for example, that Local Consistency does not run into the same memory limitations that we encountered when considering more marginals in PGM. The communication rounds, local data and global data does not vary between PGM and Local Consistency in our experiments due to our implementation (steps in MPC do not differ).

Discussion

One of the takeaways from the results presented above is that the MPC only approach is very costly compared to the proposed MPC+TEE approach, particularly in the dishonest majority settings. Dishonest majority setting is important because if the data providers do not trust common computation agents, a dishonest majority protocol like MASCOT is unavoidable. The difference in cost is significant even in the honest majority setting. While all experiments were conducted in a LAN setting, a truly decentralised implementation of Solid+MPC would be over the internet where such costs are major concerns.

The other takeaway is that the proposed approach is robust to a range of parameters beyond just the number of data



(a) Time (b) Communication Rounds

(c) Local Data (d) Global Data

Figure 10: Performance of the MPC+TEE approach using PGM and Local Consistency generation algorithms (30 iterations); Titanic dataset; SHAMIR protocol, one data point per provider.]

providers. We can see from Table 1 that the domain size of the Adult dataset is 2×10^{12} times larger than that of Titanic. When considering just the attributes that are considered in marginal queries, the domain size of Adult comes at 8.5×10^6, still 40 times that of Titanic. However, when we look at the maximum time in our proposed MPC+TEE approach and PGM, Adult is only just over 6 times that of Titanic. Another axis of comparison is to look at the size of the histograms rather than the domain of attributes that are queried. For the marginal queries we considered, this gives us 14739 for Adult vs 3285 for Titanic. From this we observe a linear growth in metrics as the histogram size increases. Considering that in large datasets the number of low-dimensional histograms is likely to be much smaller than the number of data providers, the number of marginals considered does not appear to be a bottleneck. Indeed, as we noted previously, it was the PGM generation algorithm that prevented us from considering more marginals. This is important because training on a greater number of marginals increases the versatility of the synthetic dataset by giving accurate results on a wider range of tasks.

6 Related Work

Other than the background covered in Section 3, we note two other lines of related work. The first is local differential privacy methods (Zhao et al. 2020; Cormode et al. 2018; Wang et al. 2019; Qin et al. 2017; Ren et al. 2018). The accuracy of query results in the local model is typically orders of magnitude lower for the same privacy cost and the same query, compared to central differential privacy (Dwork, Roth et al. 2014). On the other hand, we use central differential privacy using other privacy-enhancing technologies such as MPC and TEE and therefore, the privacy-utility tradeoff is the same as in central model. We show that one can limit MPC use to strictly necessary for addressing the scalability challenge. The second line of work either proposes to use blockchains for decentralised data and access control or focuses on synthetic data generation process or other computations from distributed data. Examples from this very large body of work include (Zyskind, Nathan, and Pentland 2015; Zyskind, Nathan et al. 2015; Golob et al. 2023; Pereira et al. 2022; Veeraragavan and Nygård 2023; Hynes et al. 2018). The weaknesses of blockchains based approaches include 1) for personal data, blockchains are considered too transparent. Off-chain hosting of data, which is often suggested as an alternative to address this concern, does not really solve the problem but merely delegates it. 2) the guarantees provided by blockchains such as immutability are not required (and may even be undesired) in many use-cases and come with huge costs. On the other hand, we use Solid (a personal data store) which provides everything we need and is based on open Web protocols.

7 Conclusions and Future Work

In this paper, we advance state-of-the-art in contributor-centric approach to responsible synthetic data generation. Personal data stores such as Solid provide individuals ultimate control over their data. Users keep their data in Pods (Personal Online Datastores). The apps and services that they use (for e.g. various Web based applications, online social platforms, health service applications etc), read and write data to the user's Pod. A Pod provides granular control to users over who can access which data. In the proposed approach for synthetic data generation, the real data remains with individuals in their Solid pods. Individuals use Solid's access control mechanism to decide whether they want to participate in the synthetic data generation process (for example, by taking into account what is the purpose of synthetic data generation and which real data is required as input). We show how participating individuals can then use secure multi-party computation (MPC) and Intel SGX for running differentially-private synthetic data generation algorithms in a scalable way. Our comprehensive experiments in different experimental settings support our thesis.

In future work, it would be interesting to explore other personal data stores such as openPDS and Databox with different capabilities than Solid. We expect to see similar trends on those platforms under similar experiments, but there could be interesting variations in the cause of performance bottlenecks and worth investigating. Similarly there are other frameworks for MPC, many derived from MP-SPDZ and some independent (Duan 2020). While communication costs impose a natural challenge, more efficient implementations of the underlying protocols could also contribute to more scalable synthetic data generation. Finally, we have focused on tabular data in this work whereas Solid also has support for unstructured data such as text and images. There has been much success in private synthetic data generation of images using GANs and diffusion models, such as PATE-GAN (Jordon, Yoon, and Van Der Schaar 2018). An interesting direction for future inquiry would be to improve their scalability in a decentralised setting.

References

AMD. 2020. AMD SEV-SNP: Strengthening VM isolation with integrity protection and more. *White Paper, January*, 53: 1450–1465.

Andrei Sambra; Henry Story; and Tim Berners-Lee. 2014. WebID 1.0. https://www.w3.org/2005/Incubator/webid/spec/identity/.

Aydore, S.; Brown, W.; Kearns, M.; Kenthapadi, K.; Melis, L.; Roth, A.; and Siva, A. A. 2021. Differentially private query release through adaptive projection. In *International Conference on Machine Learning*, 457–467. PMLR.

Brown, A. 2020. Synthetic Data Promises Fair AI And Privacy Compliance, But How Exactly Does It Work? [Online; accessed 29-September-2023].

Cormode, G.; Jha, S.; Kulkarni, T.; Li, N.; Srivastava, D.; and Wang, T. 2018. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, 1655–1658.

Costan, V.; and Devadas, S. 2016. Intel SGX explained. *Cryptology ePrint Archive*.

Damgård, I.; Damgård, K.; Nielsen, K.; Nordholt, P. S.; and Toft, T. 2016. Confidential benchmarking based on multiparty computation. In *International Conference on Financial Cryptography and Data Security*, 169–187. Springer.

De Montjoye, Y.-A.; Shmueli, E.; Wang, S. S.; and Pentland, A. S. 2014. openpds: Protecting the privacy of metadata through safeanswers. *PloS one*, 9(7): e98790.

Ding, B.; Kulkarni, J.; and Yekhanin, S. 2017. Collecting telemetry data privately. *Advances in Neural Information Processing Systems*, 30.

Duan, P. 2020. Introduction to Secure Collaborative Intelligence (SCI) Lab. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, 1–1.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, 265–284. Springer.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407.

Emam, K. E.; Mosquera, L.; and Hoptroff, R. 2020. *Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data*. "O'Reilly Media, Inc.". ISBN 978-1-4920-7271-3.

Encyclopedia Titanica. 2023. Encyclopedia Titanica. Accessed on: 2023.

Erlingsson, Ú.; Pihur, V.; and Korolova, A. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 1054–1067.

Evans, D.; Kolesnikov, V.; Rosulek, M.; et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3): 70–246.

Golob, S.; Pentyala, S.; Dowsley, R.; David, B.; Larangeira, M.; De Cock, M.; and Nascimento, A. 2023. A Decentralized Information Marketplace Preserving Input and Output Privacy. In *Proceedings of the Second ACM Data Economy Workshop*, 1–6.

Group, S. C. 2023a. Solid Protocol. [Online; accessed 16-August-2023].

Group, S. C. 2023b. Web Access Control. [Online; accessed 28-August-2023].

Hardt, M.; Ligett, K.; and McSherry, F. 2012. A simple and practical algorithm for differentially private data release. *Advances in neural information processing systems*, 25.

Hu, Y.; Wu, F.; Li, Q.; Long, Y.; Garrido, G. M.; Ge, C.; Ding, B.; Forsyth, D.; Li, B.; and Song, D. 2023. SoK: Privacy-Preserving Data Synthesis. *arXiv preprint arXiv:2307.02106*.

Hynes, N.; Dao, D.; Yan, D.; Cheng, R.; and Song, D. 2018. A demonstration of sterling: a privacy-preserving data marketplace. *Proceedings of the VLDB Endowment*, 11(12): 2086–2089.

Inrupt. 2023. Inrupt Documentation. [Online; accessed 16-August-2023].

Jauernig, P.; Sadeghi, A.-R.; and Stapf, E. 2020. Trusted execution environments: properties, applications, and challenges. *IEEE Security & Privacy*, 18(2): 56–60.

Jordon, J.; Szpruch, L.; Houssiau, F.; Bottarelli, M.; Cherubin, G.; Maple, C.; Cohen, S. N.; and Weller, A. 2022. Synthetic Data–what, why and how? *arXiv preprint arXiv:2205.03257*.

Jordon, J.; Yoon, J.; and Van Der Schaar, M. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*.

Keller, M. 2020. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 1575–1590.

Keller, M.; Orsini, E.; and Scholl, P. 2016. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 830–842.

Knauth, T.; Steiner, M.; Chakrabarti, S.; Lei, L.; Xing, C.; and Vij, M. 2018. Integrating remote attestation with transport layer security. *arXiv preprint arXiv:1801.05863*.

Kohavi, R.; et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, 202–207.

Li, X.; Zhao, B.; Yang, G.; Xiang, T.; Weng, J.; and Deng, R. H. 2023. A Survey of Secure Computation Using Trusted Execution Environments. *arXiv preprint arXiv:2302.12150*.

Lindell, Y. 2020. Secure multiparty computation (MPC). *Cryptology ePrint Archive*.

Liu, T.; Vietri, G.; and Wu, S. Z. 2021. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems*, 34: 690–702.

Mansour, E.; Sambra, A. V.; Hawke, S.; Zereba, M.; Capadisli, S.; Ghanem, A.; Aboulnaga, A.; and Berners-Lee, T. 2016. A demonstration of the solid platform for social web applications. In *Proceedings of the 25th international conference companion on world wide web*, 223–226.

McKenna, R.; and Liu, T. 2022. A simple recipe for private synthetic data generation. DifferentialPrivacy.org. https://differentialprivacy.org/synth-data-1/.

McKenna, R.; Miklau, G.; and Sheldon, D. 2021. Private-PGM. GitHub repository. Available at https://github.com/journalprivacyconfidentiality/private-pgm-jpc-778/tree/v2021-10-04-jpc.

McKenna, R.; Miklau, G.; and Sheldon, D. 2021. Winning the NIST Contest: A scalable and general approach to differentially private synthetic data. *arXiv preprint arXiv:2108.04978.*

McKenna, R.; Mullins, B.; Sheldon, D.; and Miklau, G. 2022. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677.*

McKenna, R.; Pradhan, S.; Sheldon, D. R.; and Miklau, G. 2021. Relaxed marginal consistency for differentially private query answering. *Advances in Neural Information Processing Systems*, 34: 20696–20707.

McKenna, R.; Sheldon, D.; and Miklau, G. 2019. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, 4435–4444. PMLR.

McSherry, F.; and Talwar, K. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, 94–103. IEEE.

Mortier, R.; Zhao, J.; Crowcroft, J.; Wang, L.; Li, Q.; Haddadi, H.; Amar, Y.; Crabtree, A.; Colley, J.; Lodge, T.; et al. 2016. Personal data management with the databox: What's inside the box? In *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, 49–54.

Nilsson, A.; Bideh, P. N.; and Brorsson, J. 2020. A survey of published attacks on Intel SGX. *arXiv preprint arXiv:2006.13598.*

Pereira, M.; Pentyala, S.; De Cock, M.; Nascimento, A.; and de Sousa, R. 2022. Secure Multiparty Computation for Synthetic Data Generation from Distributed Data. In *NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research.*

Qin, Z.; Yu, T.; Yang, Y.; Khalil, I.; Xiao, X.; and Ren, K. 2017. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 425–438.

Rabin, M. O. 2005. How to exchange secrets with oblivious transfer. *Cryptology ePrint Archive.*

Ren, X.; Yu, C.-M.; Yu, W.; Yang, S.; Yang, X.; McCann, J. A.; and Philip, S. Y. 2018. *LoPub*: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9): 2151–2166.

Sambra, A.; Mansour, E.; Hawke, S.; Zereba, M.; Greco, N.; Ghanem, A.; Zagidulin, D.; Aboulnaga, A.; and Berners-Lee, T. 2016a. Solid: A Platform for Decentralized Social Applications Based on Linked Data. *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*

Sambra, A. V.; Mansour, E.; Hawke, S.; Zereba, M.; Greco, N.; Ghanem, A.; Zagidulin, D.; Aboulnaga, A.; and Berners-Lee, T. 2016b. Solid: a platform for decentralized social applications based on linked data. *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*

Shamir, A. 1979. How to share a secret. *Communications of the ACM*, 22(11): 612–613.

Tsai, C.-C.; Porter, D. E.; and Vij, M. 2017. {Graphene-SGX}: A Practical Library {OS} for Unmodified Applications on {SGX}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 645–658.

Veeraragavan, N. R.; and Nygård, J. F. 2023. Securing Federated GANs: Enabling Synthetic Data Generation for Health Registry Consortiums. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, 1–9.

W3C. 2014. RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/.

Wang, N.; Xiao, X.; Yang, Y.; Zhao, J.; Hui, S. C.; Shin, H.; Shin, J.; and Yu, G. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 638–649. IEEE.

Xu, D.; Yuan, S.; Zhang, L.; and Wu, X. 2018. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, 570–575. IEEE.

Yao, A. C.-C. 1986. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, 162–167. IEEE.

Zhao, R.; Goel, N.; Agrawal, N.; Zhao, J.; Stein, J.; Verborgh, R.; Binns, R.; Berners-Lee, T.; and Shadbolt, N. 2023. Libertas: Privacy-Preserving Computation for Decentralised Personal Data Stores. .

Zhao, Y.; Zhao, J.; Yang, M.; Wang, T.; Wang, N.; Lyu, L.; Niyato, D.; and Lam, K.-Y. 2020. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal*, 8(11): 8836–8853.

Zyskind, G.; Nathan, O.; and Pentland, A. 2015. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471.*

Zyskind, G.; Nathan, O.; et al. 2015. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE security and privacy workshops*, 180–184. IEEE.

A Appendix: Expanded Background

Personal Data Stores

In response to increasing centralisation of data and loss of user autonomy, several decentralised data architectures have been proposed that aim to give users more control over their personal data. One such approach that has gained considerable traction is Personal Data Stores (PDS) (Sambra et al. 2016b; Mansour et al. 2016).

Under this decentralised data paradigm, users store their data in a Pod (Personal Online Datastore). A Pod provides granular control to users over who can access which data as well as secure transmission of data for authorised requests. The apps and services that they use (for e.g. various Web based applications, online social platforms, health service applications etc), also read and write data to the user's Pod. Users can transfer their data from one Pod service provider to another, including the option to self host. This unlocks a whole new paradigm for app design based on interoperability, a key design aspect (Linked Data) of Solid. The decoupling of data and apps reduces the need for expensive data collection across platforms, thereby benefiting developers as well. Pods support both structured and unstructured data.

There are several PDS projects ranging from purely academic implementations to those with commercial offerings, among them openPDS (De Montjoye et al. 2014), Databox (Mortier et al. 2016), and Solid (Mansour et al. 2016). These vary in their protocol design and the demands they place on the features a Pod must support. We focus on Solid because of its open design based on existing Web standards and growing adoption.

Solid

The Solid specification produced by the W3C Solid Community Group is the authoritative set of guidelines on the Solid protocol (Group 2023a). Here we give a brief overview with a focus on what is necessary for a client app developer.

WebID A WebID (Andrei Sambra, Henry Story, and Tim Berners-Lee 2014) is represented as an Internationalised Resource Identifier (IRI) used to identify a specific agent (person or organisation) in Solid. An example of what a WebID might appear as is `https://eg.pod.provider/profile/card#me`. To share data with someone else, the user who controls the Pod must associate sharing preferences of that resource with the WebID of the party with which they wish to share their data with. The WebID is provided by an Identity Provider and provides the authentication function. Most providers use the OpenID Connect protocol (OIDC) to prove ownership of a WebID. Authorisation is handled by the Web Access Control (WAC) system based on an Access Control List model for decentralised use cases (Group 2023b). Anyone or anything that accesses data in a Solid Pod uses a unique ID, authenticated by a decentralised extension of OpenID Connect. Solid's access control system (WAC) uses these IDs to determine whether a person or application has access to a resource in a Pod.

Reading/Writing Resources Resources may be structured (RDF data (W3C 2014)) or unstructured files. Further they may also be container resources, which are analogous to directories in a file system. Although the terminology may vary by Pod providers, it is generally possible for a user to grant the following permissions to a specific resource, enabling authorised client apps to interact with the data. As described above, *read*, *write* and *modify* permission may be granted to an associated WebID. It is also possible to grant public access.

Resources in Solid are assigned an IRI or a Uniform Resource Identifier (URI). While it is possible for clients to add, delete and modify resources in a Pod using primitive HTTP requests described in the specification, one typically uses a client library such as the Inrupt JavaScript library to perform these actions (Inrupt 2023).

While Solid provides users control of their data and allows Web applications to read/write based on user preferences, it leaves open a challenging problem. When computations have to be performed over the data of multiple users, Solid does not specify how this can be done. Solid Pods also do not have any local computation capability in the current specification. This problem has recently been addressed by the Libertas architecture proposed by Zhao et al. (2023). Before discussing the Libertas architecture, we provide a brief introduction to Secure Multi-Party Computation (MPC).

Secure Multi-Party Computation (MPC)

Secure multi-party computation (MPC) (Evans et al. 2018) is a set of cryptographic protocols for ensuring that one or more parties can participate in the decentralised computation of a function over some privately held inputs such that no party learns anything about the private input of another party. The only information that can be inferred about the private input is whatever can be inferred from the output of the function alone. [3] More formally, given n parties P_1, ..., P_n, their corresponding inputs x_1, ..., x_n and a function f, an MPC protocol computes $y = f(x_1, ..., x_n)$ without revealing input x_i to party P_j where $i \neq j$.

In the past few decades, there has been significant research on the development of efficient MPC protocols. The cryptographic primitives that are typically used to implement MPC are secret sharing schemes (Shamir 1979), garbled circuits (Yao 1986), and oblivious transfer (Rabin 2005). We focus on secret sharing, discussed as follows.

Shamir Secret Sharing A secret sharing scheme is a method for a dealer to share a secret s among n parties so that any subset of $t + 1$ or more parties can reconstruct the secret but fewer parties can learn nothing about the secret. Such a scheme is called a $(t + 1)$-out-of-n-threshold secret sharing scheme. Shamir's secret sharing (Shamir 1979) is an example of secret sharing methods. Shamir provides information-theoretic security, i.e. it is secure against a computationally unbounded adversary as long as the threshold has not been reached.

MPC protocols also differ in their security assumptions regarding the adversary, discussed as follows

Security Parameters A party is said to be corrupted if it is under the control of an adversary who wishes to learn the private input of a participant. Corrupted parties may collude

[3] Differential-privacy, to be discussed next, addresses this limitation and thus, MPC and differential privacy are complementary privacy mechanisms.

at any stage of the MPC computation. The two main parameters that define the power of the adversary are: *adversarial behaviour* and *number of corrupted parties*. There are three main types of adversarial behaviours considered in the literature: 1) semi-honest or passive adversaries, 2) malicious or active adversaries, and 3) covert adversaries. The adversaries differ in whether and how they deviate from the protocol. Broadly, the number of corrupted parties is categorised as honest majority and dishonest majority.

Shamir Protocol for MPC Most honest majority protocols for MPC such as Shamir involve representing the function we wish to compute as an arithmetic (or sometimes Boolean) circuit (Lindell 2020). All parties begin with this circuit and further can communicate securely with each other. An arithmetic circuit is composed from addition and multiplication gates. MPC consists of following steps. In the first *input sharing* step, each party shares its inputs with the other parties using Shamir secret sharing. In the second *circuit evaluation* step, the parties evaluate the gates one at a time from the input to the output. We want to maintain the invariant that for each gate, the evaluated output is a $(t+1)$-out-of-n share of the true value of that gate. In the third *output reconstruction* step, the parties now possess shares of the output which they may reconstruct it by sending their shares to each other and interpolating. Note that it is possible for different parties to obtain different or even no output by only sending shares to relevant agreed parties.

Libertas Architecture by Zhao et al. (2023)

Libertas architecture, proposed by Zhao et al. (2023), extends Solid (while adhering to the Solid protocol) to enable performing computations using MPC on decentralised data held in individuals' Pods. Apart from the *data provider* (Pod owner), they introduce three new roles, that of the MPC *app*, the *encryption agent*, and the *computation agent*. We discuss them briefly below and an overview is shown in Figure 11.

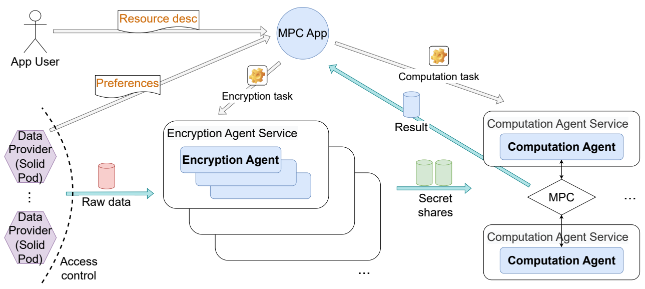


Figure 11: Libertas Architecture for integrating Solid & MPC (Zhao et al. 2023)

Data Provider Before the MPC computation is initiated, the data providers must create a *preference file* which lists the computation and encryption agents that each of them are willing to delegate the computation and encryption tasks. Further, this must be made readable to the MPC App(s) and/or App users via their associated WebIDs. The trusted encryption agents are granted read access to the relevant data. This is all that is expected of the data provider, who need not be active during subsequent computations.

MPC App The user of the MPC App provides a *resource description*, containing a list of data resources and the associated preferences of data providers as URIs. The app reads preference files, determines the encryption and computation agents that match the delegation preferences of data providers and sends out the corresponding tasks. It then waits for the computation to finish before receiving the results from the relevant agent.

Encryption Agent The encryption task instructs the encryption agent where and how to read the data and how to send secret shares of the data to the players (computation agents). The task is sent as raw source code, allowing for verification before compilation/execution to protect against malicious actors. This is implemented using the client mechanism of the MP-SPDZ framework for secret sharing with players, which in turn is based on the SPDZ protocol (Damgård et al. 2016).

Computation Agent The computation agents execute the program on the secret shared inputs based on the protocol of choice. One of the strengths of the Libertas architecture is that both agents are reusable and generic in the computation service they provide. We will take advantage of this in our proposed solution by proposing additional roles for the agents to strengthen the privacy guarantees of our synthetic data generation approach.

Libertas addresses three main limitations of Solid. First, a user can allow an agent, that has not been granted access to some data directly, to utilise the data as input in a multiparty computation procedure, without compromising the user privacy. Second, it overcomes the limitations arising due to the lack of computation capability in Solid Pods, by a mechanism to delegate participation in the joint computation while users retain control. Third, under a direct-decentralised model where data providers are also the protocol players, MPC scales poorly as the number of players grow. Conversely, the delegated-decentralised model used in Libertas, with a smaller number of agents acting as players, offers significantly better scalability. But fundamentally, costs due to MPC protocols remain challenging.

Differential-Privacy

Differential privacy (Dwork et al. 2006) is a rigorous mathematical definition of privacy that has become the de facto standard for output privacy. Many tech companies such as Apple, Google (Erlingsson, Pihur, and Korolova 2014), and Microsoft (Ding, Kulkarni, and Yekhanin 2017) have built privacy-aware systems that use differential privacy. National Statistical Offices too have begun releasing censuses that incorporate differential privacy. Differential privacy bounds the impact any one individual can have on the output of an algorithm. More formally, a randomised mechanism $M : D \rightarrow \mathcal{R}$ satisfies (ϵ, δ)-differential privacy (DP) if for all neighboring (differing on one element) datasets $D, D' \in \mathcal{D}$ and all subsets of possible outputs $\mathcal{S} \subseteq \mathcal{R}$,

$$Pr[M(D) \in \mathcal{S}] \leq exp(\epsilon)Pr[M(D') \in \mathcal{S}] + \delta$$

Synthetic Data Generation Algorithms

Any dataset release (including synthetic data) that meaningfully preserves statistics of the original dataset will result in a privacy loss. Thus, there exists a privacy-utility trade-off. We are interested in synthetic data generation algorithms with differential-privacy guarantees for output privacy. There exists several algorithms for differentially-private synthetic data generation. In this paper, we focus on marginal-based approaches to synthetic data generation that have achieved considerable success in practice. Marginal-based algorithms won the 2018 NIST Differential Privacy Synthetic Data Competition, for instance, and performed competitively in other iterations of the contest (McKenna, Miklau, and Sheldon 2021).

A marginal is a key statistic that captures low dimensional structure in a high dimensional data distribution (McKenna, Miklau, and Sheldon 2021). Let a dataset D consist of N records and each record contain sensitive information about one individual. Let the set of d attributes be $\mathcal{A} = \{A_1, ..., A_d\}$ where attribute A_i has domain Ω_i. We assume that the domain Ω_i is finite, $|\Omega_i| = n_i$. We denote the set of all possible datasets by $\mathcal{D}$. A marginal for a set of attributes $C \subseteq \mathcal{A}$ is a histogram that counts the number of occurrences of each combination of values in C. We use the terms histogram and marginal interchangeably.

Further, we focus on workload aware algorithms. A workload is a collection of queries that a synthetic dataset must preserve well. These are the queries that downstream users of the synthetic dataset might be interested in answering.

Select-Measure-Generate Paradigm The *select-measure-generate* paradigm for synthetic data generation consists of the following three steps (McKenna, Miklau, and Sheldon 2021):

- *Select* a set of queries (low-dimensional marginals) based on the expected workload.
- *Measure* the selected queries with a noise addition mechanism to obtain private marginals.
- *Generate* synthetic data that explains the noisy measurements well.

Due to the post-processing property of differential-privacy (Dwork, Roth et al. 2014), given an output from a differentially-private function, we can perform further data analysis on it without any additional loss in privacy. Therefore, privacy analysis in the above Select-Measure-Generate paradigm can be focused on the first two steps.

MWEM Algorithm The Multiplicative Weights and Exponential Mechanism (MWEM) algorithm is an iterative algorithm for constructing a synthetic dataset whose answers to queries is close to that on the original dataset (Hardt, Ligett, and McSherry 2012). MWEM takes as input the original dataset D and a set Q of linear queries.[4] Besides this, two parameters of interest are epsilon value ϵ and the number of iterations T. We refer the reader to (Hardt, Ligett, and McSherry 2012) for a pseudo-code of the MWEM algorithm. The algorithm produces a distribution A over $\mathcal{D}$ such that the difference between $q(A)$ and $q(D)$ is small. MWEM repeatedly samples a query for which the difference is still large and updates the weight that A places on each record x. MWEM satisfies ϵ-differential privacy by leveraging the exponential mechanism (McSherry and Talwar 2007) to sample query, and the Laplace mechanism (Dwork et al. 2006) to add noise in the query results.

Other Marginal-Based Algorithms The challenge that marginal-based generation algorithms aim to overcome is that the problem quickly becomes intractable when optimising over distributions which best explain the generated marginals (McKenna and Liu 2022). Consider a dataset with a large number of attributes $\mathcal{A}$ and each attribute having large (discrete) domain. Even enumerating all possible entries would be very memory demanding.

PGM A common way to overcome this is to restrict attention to certain joint distributions that have tractable representations. Probabilistic Graphical Models (PGM) represent the possible space of distributions of synthetic data as a graphical model P_θ (McKenna, Sheldon, and Miklau 2019). The parameter vector θ is much smaller than the distribution P, allowing us to overcome the curse of dimensionality. PGM work well in practice having won the 2018 NIST Synthetic Data Competition and its follow up Temporal Map Competition (McKenna, Miklau, and Sheldon 2021).

Relaxed Tabular An alternative approach called Relaxed Adaptive Projection (RAP) restricts attention to so-called pseudo-distributions that can be represented in a relaxed tabular format (Aydore et al. 2021). The format is similar to a one-hot encoding (although entries need not be 0 or 1) and the size of this table is a tunable hyperparameter. Although convergence to the optimum is not guaranteed due to its non-convexity, it too works well in practice.

Local Consistency Local Consistency methods such as Generative Networks with the Exponential Mechanism (GEM) (Liu, Vietri, and Wu 2021) and Approx-Private-PGM (APPGM) (McKenna et al. 2021) work by imposing local consistency constraints on the noisy marginals, as opposed to searching over the space of possible distributions. The scalability of local consistency (much like relaxed tabular) depends on the size of the largest marginal as opposed to the size of the junction tree in PGMs, thereby resulting in better scalability at the cost of relying on heuristics.

B Minor Details Omitted from Section 5

A problem we encountered during our experimentation was that the compilation time of our MPC code was quite long. This often resulted in the client-side code making a connection to a not as of yet open port. This was fixed by adding

[4] A linear query (also called counting or statistical query) is specified by a function q mapping data records to the interval $[-1, 1]$. The answer of a linear query on a dataset D, denoted by $q(D)$, is the sum $\sum_{x \in D} q(x) \cdot D(x)$

sleep statements in certain places (it did not affect the comparison results though). We considered the MPC app sending over compiled byte code instead of the raw source code as a potential fix. But in the end, we followed the same strategy as (Zhao et al. 2023) and agreed that the ability to examine the raw source code by an end user offered a useful protection against malicious actors. Nevertheless, synchronisation between the encryption and computation agents regarding when the MPC program is ready to accept a client connection is a challenge that must be overcome for wider adoption.

We also note that we did not benchmark the performance of PGM and Local Consistency with the MPC only approach. This is because the ecosystem for running MPC programs is still developing and as such there is no mechanism to take an existing library such as MBI and compile it to MPC code. This is exacerbated by the many dependencies of MBI on different ML libraries. Even MP-SPDZ, which includes machine learning capabilities, only implements a limited range of functionality from popular ML libraries. Extrapolating from our results from MWEM, we can reasonably expect the MPC only approach to scale poorly, but this would be an interesting direction for further investigation.