Libertas: Privacy-Preserving Computation for Decentralised Personal Data Stores

Rui Zhao[1], Naman Goel[1], Nitin Agrawal[1], Jun Zhao[1], Jake Stein[1], Ruben Verborgh[2], Reuben Binns[1], Tim Berners-Lee[1], and Nigel Shadbolt[1]

[1] University of Oxford, Oxford, UK
[2] Ghent University, Ghent, Belgium

Abstract. Data-driven decision-making and AI applications present exciting new opportunities delivering widespread benefits. The rapid adoption of such applications triggers legitimate concerns about loss of privacy and misuse of personal data. This leads to a growing and pervasive tension between harvesting ubiquitous data on the Web and the need to protect individuals. Decentralised personal data stores (PDS) such as Solid are frameworks designed to give individuals ultimate control over their personal data. But current PDS approaches have limited support for ensuring privacy when computations combine data spread across users. Secure Multi-Party Computation (MPC) is a well-known subfield of cryptography, enabling multiple autonomous parties to collaboratively compute a function while ensuring the secrecy of inputs (*input privacy*). These two technologies complement each other, but existing practices fall short in addressing the requirements and challenges of introducing MPC in a PDS environment. For the first time, we propose a modular design for integrating MPC with Solid while respecting the requirements of decentralisation in this context. Our architecture, Libertas, requires no protocol level changes in the underlying design of Solid, and can be adapted to other PDS. We further show how this can be combined with existing differential privacy techniques to also ensure *output privacy*. We use empirical benchmarks to inform and evaluate our implementation and design choices. We show the technical feasibility and scalability pattern of the proposed system in two novel scenarios – 1) empowering gig workers with aggregate computations on their earnings data; and 2) generating high-quality differentially-private synthetic data without requiring a trusted centre. With this, we demonstrate the linear scalability of Libertas, and gained insights about compute optimisations under such an architecture.

1 Introduction

The Web was created as an open and universal platform for information sharing. Over the years, the Web also enabled the development of useful data-driven technologies. However, an unintended consequence has been that the users today face unprecedented privacy challenges. Users have lost control over what data are being collected about them by centralized online platforms and how these data are used [24], which has created stark information asymmetries and power imbalances, leading to the frequent deprivation of basic rights [51].

Conversely, there exists a growing need for collective information employment in the pursuit of societal well-being. Instances of such scenarios encompass endeavors like enhancing our response to global pandemics by aggregating individuals' health and mobility records [43], addressing climate change through the exchange of energy consumption patterns [5], and even ameliorating working conditions by pooling telemetry data among otherwise isolated gig workers [49, 41].

The challenge that persists is how to unlock the potential of collective data in a decentralized manner, devoid of a central intermediary, all while upholding individual privacy and autonomy. In an ideal decentralized context, the expectations for privacy-preserving computation are very different from that in a centralized context. Notably, user autonomy stands out as a pivotal factor, and the presence of a reliable or trusted central entity can not be assumed. This gives rise to distinct requirements for achieving collective privacy-preserving computation within such a decentralized setting, as depicted in Table 1.

Personal Data Stores (PDS) provide users decentralised data storage and autonomous controls [13, 31, 38] over their personal data, aiming at a paradigm shift to the current centralised platform-led architectures. Solid [38, 27] is an example PDS that embodies semantic technologies like Linked Data, while providing interoperability and user autonomy in one framework. They serve as a good basis for storage and protocols, but only support part of our requirements: they support R2 (Autonomy) with their access control mechanisms. But access control mechanisms can not guarantee R1 (Privacy), when

Requirement	Explanation
R1: Privacy	Keeps input data private during computation, and output of the computation can not be used to infer the input.
R2: Autonomy	Gives meaningful control to the data providers/owners over permitted usage of their data.
R3: Minimum Trust	Requires minimum and well-reasoned trust assumptions.
R4: Scalability	Scales well with the number of data providers.
R5: Generality	Supports all types of computation.

Table 1: Requirements for a collective privacy-preserving computation system in decentralised context without a central trusted party.

computational algorithms (e.g. aggregation) need access to data stored across multiple users/PDS (see Sec 2.1).

On the other hand, algorithmic mechanisms such as *(Secure) Multi-Party Computation* (MPC; also called sMPC or SMC) [46, 26] and *Differential privacy* (DP) [15] can provide R1 (Privacy) and R5 (Generality). An MPC protocol can securely evaluate a function over multiple independent parties' data (i.e. compute output without revealing inputs). Differential privacy [15] aims to alleviate the concerns of reverse inference of input from the output of the computations. However, without an appropriate underlying system, MPC and DP can not provide R2, R3 and R4, and as we will discuss in Sec 2.2, this is indeed the case with existing implementations.

Given the complementary nature of decentralized PDS and algorithmic privacy mechanisms such as MPC and DP, it is natural to ask whether and how these can be combined to satisfy all the requirements listed in Table 1. In this paper, we discuss and address several challenges of employing MPC and DP onto PDS. We pioneer and test an architecture, Libertas, and show how it supports the requirements for a collective privacy-preserving computation system in decentralised contexts. Our implementation is based on Solid, but we will also show how the architecture may be adapted to other PDS systems. Our proposed architecture uses what we call the delegated-decentralised MPC model, which has better scalability (linearly with the number of data providers) compared with the direct-decentralised model. This intuition is verified by benchmarking the two models.

We evaluate the Libertas architecture using realistic use cases, namely gig worker empowerment and synthetic differentially-private data generation, which verified both the technical feasibility and the scalability, showing our approach's wide applicability and potential.

Contributions: To summarise, this paper proposes and validates a novel architecture, Libertas, for efficiently integrating algorithmic privacy techniques, MPC (multi-party computation) and DP (differential privacy), into Solid and similar PDS (personal data stores) systems. Libertas is a novel end-to-end solution to unlock the potential of collective data in decentralised settings. Unlike related work, Libertas achieves this while meeting all the desired requirements (privacy, autonomy, minimum trust, scalability and generality). We empirically verify its technical feasibility and scalability, and provide insights on design considerations of the architecture's applicability through realistic scenarios such as gig worker empowerment and synthetic differentially-private data generation.

2 Background and Related Work

2.1 Personal Data Stores (PDS)

Personal Data Stores (PDS), such as openPDS [13], Solid [38, 27] or Databox [31], support a decentralised data paradigm by allowing users to keep their data in their full control within their own data stores, rather than data being locked away by large platforms. PDS also provides practical data protection-related controls, so that users can set up preferences regarding who can access what and can audit the access to their data. While PDS provides granular control over data access and ensure secure transmission of data, as soon as an application receives data, the privacy faces a challenge. This is the case for any scenario related to facilitating collective data use. For example, if workers were to store their payroll data in their own PDS and want to share this data to discover their group average salary to argue for better pay rates. A traditional PDS approach will fall short of supporting this transaction while preserving privacy – users' privacy may be compromised as their sensitive information may be revealed or misused once

the application has access to their(/everyone's) data in order to perform calculations. In the worst case, the application can retain a copy of the original data from everyone, and do arbitrary further operations without the data subject's knowledge or consent.

Some PDS research provides mechanisms to ensure privacy in certain situations. The openPDS includes sub-divided personal data stores including access controls hosted locally or in the cloud [13]. Databox [31] similarly provides a mix of local and remote data stores which are configured to allow only authorised applications to access personal data. Both also outline the importance of privacy-preserving aggregation within individual PDS, but do not specifically test performing privacy preserving data aggregation tasks among *a set of users*. Databox showed the potential for distributed privacy-preserving machine learning with their implementation [50] of Federated Learning (FL) among a group of data providers, orchestrated by a central server. However, that requires the central server and inherits properties and requirements for FL and requires redeveloping when the algorithm or model changes. OpenPDS suggested the possibility of using MPC for retrieving aggregated results from multiple PDS [13] but, to the best of our knowledge, did not put the suggestion to test. On the contrary, [4] considered it impractical for combining PDS with MPC, and proposed an architectural vision to use trusted execution environments (TEE) as a solution. In general, a mechanism for privacy-preserving data access and processing is needed above and beyond the present features of PDS systems.

Solid [38] is proposed as a hybrid (inter-)personal data and knowledge store approach based on Linked Data and existing Web protocols, allowing users (and applications) to interoperably store and retrieve data, permitting federation with other users' PDS (called *Pods* in Solid), while preventing vendor lock-in of either PDS hosts or applications. In this work, we explore the implementation of decentralised privacy-preserving computation in a Solid-based architecture rather than Databox or OpenPDS because of its open and standard-based design.

2.2 Privacy-Preserving Mechanisms

There are several approaches proposed to preserve data privacy [29] applicable in diverse settings. *Data modifications* rely on obfuscation- or perturbation-based approaches [20, 3] and modify or sanitise user data so that it cannot be linked to specific individuals. *Data minimisation* approaches aim to achieve the optimal computational results by adjusting the computational model and minimising the volume data required. Federated learning (FL) is a related machine-learning specific approach that aims to train machine learning models over distributed data sets while limiting the movement of data to central servers by only transmitting model updates [25, 45]. Similarly, Meurisch et al.[28] propose a privacy-preserving platform for the subsequent personalisation of pre-trained ML models with personal data stored in PDS [28] using secure enclaves like Intel SGX [11]. *Data encryption* approaches use encrypted user data, ensuring integrity and confidentiality when sharing data. There are two major data encryption approaches relevant to our work, homomorphic encryption (HE) [18] and (Secure) Multi-Party Computation (MPC) [35]. HE is able to analyse or manipulate encrypted data without revealing the data; however, it is limited by its low computational efficiency and limited operations. MPC encompasses a class of cryptographic protocols that rely on the secure evaluation of a function over sensitive data shared across multiple parties. MPC has the benefits of not losing precision and performing any type of computation, providing a promising option for privacy-preserving computation over dispersed data sources. We will briefly present some properties of MPC below, and discuss the challenge of using MPC in a decentralised setting.

Multi-Party Computation (MPC) Given an environment with n parties $P_1 \cdots P_n$, their corresponding inputs $x_i \cdots x_n$ and a function f, an MPC protocol computes $y = f(x_1 \cdots x_n)$ without revealing any input x_i to a party P_j $(i \neq j)$. Traditionally, two security notions have been considered for MPC [33] — *semi-honest security* and *malicious security*. Protocols with semi-honest security are relatively more efficient and protect against passive attackers that do not deviate from the protocol. In contrast, protocols with malicious security also provide security from attackers that may deviate from the protocol. The number of parties that an attacker could corrupt or compromise is another factor of security in the contexts involving multiple computation parties. Protocols assuming an *honest majority* ensure security under the assumption that fewer than half the parties could be corrupted by an attacker. In a *dishonest majority*, the same assumption does not hold. MPC protocols are generally realised using a combination of primitives such as oblivious transfer [36], garbled circuits (GC) [47] and secret sharing schemes [39, 7]. Here, we focus on additive [8] and Shamir secret sharing [39]. To understand the main results in this

paper, knowing the details of MPC is not a requirement and therefore, we skip those in the interest of brevity.

In prior work, MPC has been mostly explored in settings where each computation party has direct access to data. Work on combining MPC with distributed data sources partially addresses this issue: Mohassel and Zhang [30] proposed MPC-based protocols for specific AI algorithms but they only experimented with distributing user data among two non-colluding servers; Bonawitz et al. [9] proposed an efficient model to securely perform FL over multiple users, but assumed a (single) trustworthy server as with FL in general; Rouhani et al. [37] use GC to securely perform scalable Deep Learning execution over distributed data from individuals, but is also constrained to the properties (e.g. performance) of GC; work on Private Set Intersection like [17] and [2] involved many autonomous parties, with or without a central server's help, while intensively exploiting properties of the specific task. Secure Aggregation

Despite their success in increasing the number of data sources, little has been discussed on who provides these computation parties, and their relationship to the data subjects, particularly for general MPC that supports a wide-range of computation tasks. Essentially, this implies a centralised trust of an organisation providing or choosing these computing parties. This does not reflect the ethos of decentralisation in the context of PDS, where users are empowered with choice, and each user / data provider is an autonomous party.

Differential Privacy (DP) Output privacy represents a distinct privacy aspect that complements the privacy protection provided by MPC. While MPC ensures that the inputs used in a computation remain undisclosed, output privacy goes a step further by preventing reverse inference based on the revealed computation results. Differential privacy, a formal mathematical concept introduced by Dwork [14], plays a crucial role in restricting the disclosure of private information contained in a database when employing a computation algorithm. In simpler terms, an algorithm is considered differentially private if an external observer, upon observing its output, remains unable to determine whether a specific individual's information was utilized during the computation process. In the paper, we show with an example how this complementary notion of privacy can be implemented in our proposed solution.

3 MPC in Decentralised Settings

3.1 Direct- and Delegated-Decentralised Models

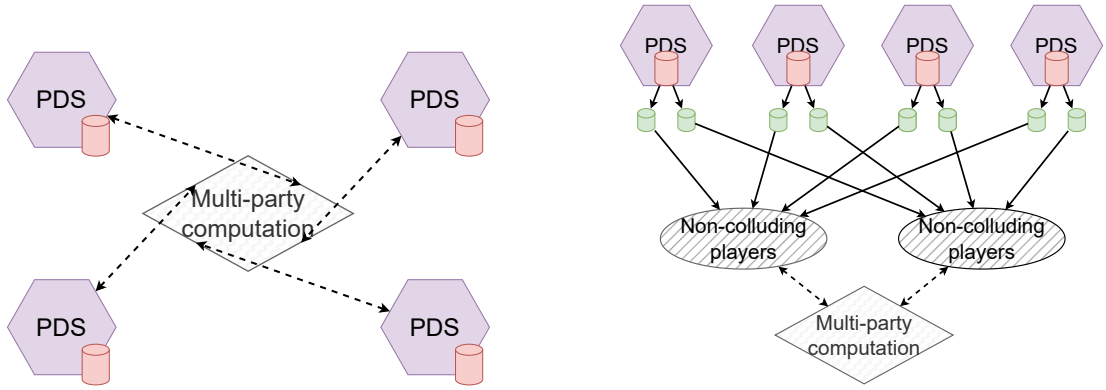
In this section, we focus on *general* MPC protocols that support a wide range of computations, to satisfy the generality (R5) requirement. The decentralised computation models for MPC can be classified into two broad categories:

1. *Direct-Decentralised:* (Fig. 1a) In this setting, each *data provider* is a computation party (*player*), following an MPC protocol to carry out secure computation;
2. *Delegated-Decentralised:* (Fig. 1b) In this setting, *data providers* are different from *players*. Data providers send secret shares of data to these players (such that no single player can make sense of the data independently). The *players* perform the main MPC computation between one another.

The direct-decentralised model reflects the traditional interpretation of MPC – each computation party (*player*) holds their data and performs computation. The data provider is the same as the player, and the only exchange is between different players. On the other hand, the delegated-decentralised model separates the computation parties and the data providers, leading to a different relationship.

3.2 Challenge in Decentralised PDS Context

As briefly mentioned earlier in Sec 2.2 existing MPC literature places an emphasis on the security-related assumptions about different computation parties. That lays a necessary foundation for the Privacy (R1) requirement. However, there is little to no discussion on who specifies the group of parties and the prescribed security properties. In a platform-based setting (e.g. [6]), this is often assumed as a given because the platform will determine the parties and their rights on the behalf of users. Users do not normally have control over how their data is used, apart from the Terms and Conditions which they seldom read [32]. Thus, *trust* is still centralised to the platform, despite using MPC. This contradicts with Minimum Trust (R3) and Autonomy (R2) of the requirements.



(a) Direct-Decentralised MPC (b) Delegated-Decentralised MPC

Fig. 1: MPC Models in decentralised settings. Data provider is denoted here as PDS.

In a decentralised context, such as that with PDS, not only the data storage is decentralised, but the aforementioned central trusted party is also removed, to ensure individual privacy and autonomy over data use. Therefore, the platform as conceived in a centralised setting no longer exists, no longer orchestrates the permission acquisition of data subjects, and no longer predefines the set of computing facilities and their security features. Aligning with the ethos of decentralisation and PDS, it is the data owners/providers who should possess the autonomy and ability to express the preferences of their trust, and control who has access to their data. Therefore, it requires an appropriate alternative mechanism for permission establishment and, more importantly, a mechanism for computing facility selection.

Performing a computation task, including MPC, always requires an initiating or controlling application (the MPC App). There is a clear distinction between *data providers* and the *App user*: a *data provider* is someone (or someone's PDS) who contributes data to MPC computation; the *App user* is a person who uses the MPC App to perform computation over multiple data providers' data. Thus, the App user is not necessarily a data provider, and, more importantly, the App user does not necessarily represent the interests or preferences of all data providers.

OpenFL Platform?

Therefore, three interweaving questions need to be answered for performing MPC: a) who will carry out the computation; b) why are these parties selected; c) what are their security properties?

3.3 Utilising MPC on PDS

A naive way to execute MPC in the decentralised PDS-based setting would let the MPC App developer(s) determine these facilities, similar to that in a centralised setting by using the App user's machine and/or servers provided by the App developers. However, from the data providers' perspectives, these computation parties will not be trustworthy as they can easily collude, unless the data provider fully trusts the App user, which is a very strong requirement and is rarely the case.

同样是传统FL的困境，即对运行的FL服务的不信任问题

Another way is to follow the direct-decentralised model: use the PDS (or a dedicated server for each of them, same below) as the computation parties in MPC. Because each data provider trusts its own PDS, at the minimum a dishonest-majority protocol can be used. Further, if an appropriate punishment or incentive mechanism exists (e.g. retaining social relationships with peers or blacklist), data providers would want to maintain their reputation, and therefore can generally form an honest-majority group. However, beside the technical requirements (capability to accept custom computation), this also involves a performance issue – the number of data providers (thus *players*) could be very large. As we will show later (in Sec 5.1), this scales poorly. That is why we would like to avoid this model.

MPC的问题：计算开销大并且不好扩展

A third approach uses the delegated-decentralised model. The secret sharing of data between the data providers and the computation parties (players) can be realised using a mechanism like the "*client*" approach in [12], which can handle corrupted participants. For *players*, we can expect internal or external services providing agents as player candidates, and each data provider to express their trust. Therefore, an algorithm can be used to select a subset from them as the actual players. As to be discussed later in Sec 4.3, a subset of the trusted agents of all data providers can form a non-colluding or even honest-majority group (e.g. intersection of everyone's trusted agents).

The delegated-decentralised model is also expected to scale better than the direct-decentralised model because fewer *players* participate in the computation, which is verified by our benchmark (Sec 5.1). With this in mind, the next section explains the architecture we have developed for employing delegated-decentralised MPC with PDS. This approach allows for the utilisation of data providers' preferences for trusted participants, while maintaining compatibility with existing protocols.

4 Libertas: Architecture for Privacy-Preserving Decentralised Computation

To address the aforementioned challenges and requirements, we propose an extended architecture for Solid [38], called **Libertas** [3], illustrated in Figure 2. Solid is a well-known PDS system focusing on interoperability with a modular architecture and a clear separation of roles. Our architecture takes advantage of existing mechanisms such as authorisation and access controls, and is compatible with underlying protocols. Though we demonstrate a Solid-based architecture, we also briefly discuss how the proposed architecture can be adapted to other PDSs.

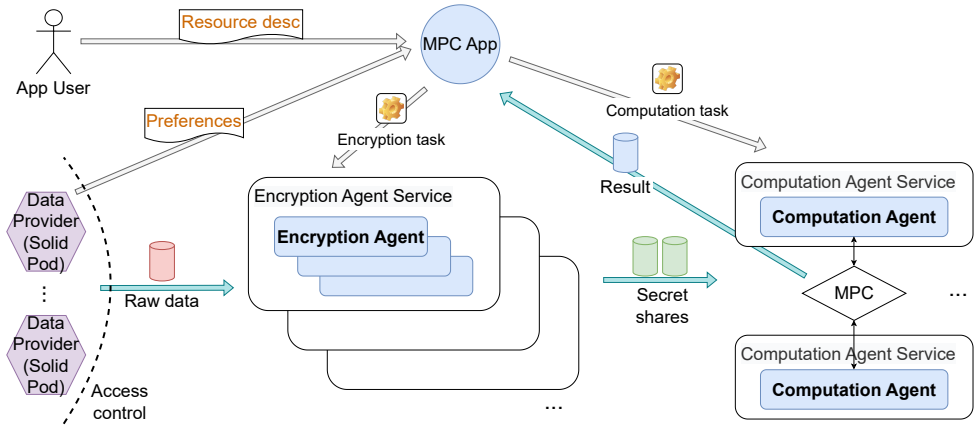


Fig. 2: Libertas: A modular architecture for integrating MPC with Solid. Three dots denote possibly more repetitions. Architectural core components are coloured.

有点类似contract-based FL的一种可能实现

4.1 Architecture Design

Core components The core components are coloured in Figure 2. In Solid, users store their data in their own Pods. The MPC App initiates the computation and sends relevant tasks to the agents who carry out further computation; the *encryption agents* (as the *clients* of MPC) read data from Pods, and send secret shares of the data to the computation agents; the *computation agents* (as the *players* of MPC) perform the main MPC computation amongst one another; finally, the MPC App obtains the results. This process has a slight difference to that depicted in Figure 1b – the Solid Pods do not hold computation power, so the *encryption agents* are made explicit in addition to the PDS, to act as the clients.

Data-provider-centric configuration Prior to any computation, each data provider provides a *preference file* denoting their trust[4] on a list of (encryption and computation) agents[5]. As files are protected by Solid's access control mechanism (Web Access Control [10]), the data provider also needs to grant relevant permissions. In doing so, the provider makes the preference file readable to the MPC App or the App user, and grants the trusted encryption agents (identified by their WebID [44]) read permission to their data. Note that these steps only need to be performed once, and can stay the same for all future MPC tasks, as long as the preference does not change. If the data provider decides to distrust anyone, they can simply revoke the access permission from their Solid Pod, which is enough to block future access.

[3] Our prototype implementation is available at `https://github.com/OxfordHCC/libertas`.
[4] Principles on this are discussed in *trust and agent selection*.
[5] More precisely, the *services* providing such agents. Their relationship will be explained later in *separation of duty*.

App usage The App user supplies a *resource description*, containing a list of data (resources) and the relevant *preferences* from each data provider. Assuming with right permission, the MPC App reads the preference files, determines the encryption agents and computation agents (discussed later), sends relevant directives (and MPC tasks) to them, waits for the computation to finish, and obtains the result. The encryption task describes how to securely share the data with each player; the computation task describes how to perform the main computation among computation agents. Intuitively, the encryption task and the computation task should contain the list of computation agents (instances of such agents, obtained from the services) to allow connection establishment. We use MP-SPDZ [21] as the MPC framework, and use its supported *client* mechanism, based on a variant of SPDZ protocol [12], for the secret share of data. In our implementation, the tasks are sent as source code, and the agents will (compile and) execute them after receiving. In addition, the App can also customise the settings for the MPC computation, such as the number of computation agents (and methods to determine them), MPC protocol and MPC parameters, taking the advantage of MP-SPDZ which supports a wide range of MPC protocols and parameters.

Trust and agent selection The preference files from data providers are the sole source for the MPC App to select relevant agents from. In our prototype, we take the simplest view: each preference file contains a list of trusted encryption agents and a list of trusted computation agents. It is worth nothing that the trust of encryption agents and computation agents are different: the encryption agent must be fully trustworthy (by the data provider), while the computation agents just need to be semi-honest. This is a natural result because encryption agents have access to raw data, and computation agents only receive secure shares of data. Of course, the computation agents should be non-colluding – if all of them collude (or a significant portion of them collude, depending on the exact protocol), they can combine their shares and reconstruct the original data. Such agent services can be from public or private providers. For example, the PDS host can deploy an encryption agent service in parallel with the PDS service itself, which is easy to gain trustworthiness by the data providers. For the MPC App, choosing the encryption agent for each data provider is easy: just choose one of the trusted agents of that data provider; choosing the same encryption agent for different data providers is allowed. However, the computation agents should be unique, so the selection process is more complicated. There can be different approaches for this, and we implemented two extreme versions: take (a subset of) the intersection of the trusted computation agents of all data providers, and take the union. More details are discussed in Sec 4.3. In general, the exact appropriateness is context-dependent, which we do not discuss in this paper.

4.2 Additional Properties

Separation of duty The Libertas architecture separates the duties of different stakeholders, making all components reusable across tasks. Both types of agents can be reused by different MPC applications, because they only provide the computing (and data fetching) *facility* rather than that logic. That is why we envision the existence of agent services for the data providers to choose from. The actual computation is from the MPC App, which provides the logic (and parameters) for the computation. As the agent services are separate from App, each App does not need to worry about the computation facility, and can directly reuse existing agents (specified in the preferences by data providers). Besides, because the App user is not necessarily a data provider, this gives data providers the flexibility to allow third parties to use the data for rightful computation, e.g. platforms, worker unions, or the government. Access control is put into place to avoid undesired access; the security mechanism and the secure protocol of MPC secures the raw data from being seen by the App user.

Variants As discussed above, the Libertas architecture does not require changes to the underlying Solid protocol, and utilises existing mechanisms. From a generic point of view, this architecture is not bound to a specific ecosystem or framework, both for the PDS and MPC framework, but only assumes some basic properties of them, such as the availability of an access control mechanism. To suit different systems, the different core components in the architecture may be merged, resulting in different derived architectures, with different tradeoffs. In particular, the encryption agent may be merged with the data store (PDS), freeing the user from the task of seeking and assigning trustworthiness to encryption agents; on the other hand, this has drawbacks such as making the PDS service more complex and thus no longer consistent with their vanilla protocols. This could be an interesting exploration for some PDS systems with computation capabilities, such as openPDS. One may also explore the possibility and implication of integrating the computation agents with the PDS services, and randomly sample a subset of them by the MPC App. And of course, the MPC framework can be swapped, while the rest of the architecture stays the same.

低耦合系统

4.3 Threat Model and Assumptions

We assume the network transmission is secure. This can be achieved by using standard practices such as SSL. Our chosen MPC framework, MP-SPDZ, also secures the data transmission between all MPC parties using SSL. We also assume the computation result/output can be used by the App user, e.g. because it reveals no sensitive information of the input data. This can be achieved by using Differential Privacy as we show in the evaluation.

There are four major parties in the Libertas architecture, namely the data provider, the App (user), the encryption agent, and the computation agent. We assume the existence of unique identifiers for each party in the architecture, such as using WebID. In our work, the focus is to support autonomy, and thus we assume the data providers are always honest, and make sensible choices of agents to the best of their knowledge – they only express trusted agents if they trust them. We also assume the agents behave as expressed in the preferences. More specifically, we assume that an encryption agent trusted by a data provider will always be fully trustworthy and honest. Similarly, we assume that a computation agent trusted by a data provider will be semi-honest (in the worst case), from the point of view of this data provider. However, other data providers may not trust this agent to this degree. Therefore, as we will discuss later in the section, a selection algorithm will need to select computation agents to participate in MPC, given the trust preferences of different data providers, forming different security properties.

In a computation, the chosen MPC protocol determines the implications of malicious computation agents (if involved), which has been intensively discussed by the relevant research [26, 22]. We do not discuss this here, but readers are encouraged to consult the security assumptions (and therefore guarantees) of each individual protocol. The same is also true for interaction between encryption agents and computation agents, as determined by the chosen delegated-decentralised protocol, i.e. the client [12] mechanism in our implementation.

A random App (user) on the market can contain malicious behaviours. Therefore, we take advantage of Solid's access control mechanism to only allow trusted Apps (and users) to access preference files. Without the preference files, it is infeasible for an untrusted App to pick the right encryption agent from the wide range of available encryption agents and therefore cannot initiate computation on that user's data. Thus, we assume an authorised App will obey the protocol to avoid being distrusted, i.e. semi-honest.

The encryption agent is provided full access to the input data, and therefore we require it to be fully trusted by the data provider. The access control mechanism can prevent untrusted agents to access the data, and the data providers should make sensible choices. As discussed earlier, an easier solution would be if the encryption agent were provided by or merged with the PDS service, which the data provider always trusts. However, by doing so, the wide range of encryption agents would also be eliminated, a desired feature that helps to avoid malicious Apps (and users). Therefore, this should also require that encryption agents to check the identity of the App (and user), and to verify if it is trusted by the data provider.

The computation agent does not get full access to input data, but only the secret shares of inputs from encryption agents. If the fraction of corrupted computation agents is larger than the security assumption of the MPC protocol, data breach could happen. Therefore, we inherit assumptions on computation agents from the chosen MPC protocol. In the best case, if the chosen computation agents are trusted by all data providers, they will form an honest-majority group, and an honest-majority MPC protocol (e.g. Shamir) would suffice for the computation. Therefore, using *intersection* for computation agent selection creates a strong security guarantee, as well as performance expectation, with an honest-majority MPC protocol. Similarly, in weaker scenarios, when the intersection is less than the required number of computation agents, alternative protocols can be used, up until a covert (e.g. ChaiGear or CowGear [23]) or malicious protocol (e.g. MASCOT [22]) protocol, to tolerate and detect dishonest behaviours in this context, at the cost of performance.

On the contrary, if none of the data providers share trusted computation agents, either the computation can not proceed, or it can proceed by choosing from (a subset of) the union of all computation agents, with a different security property. For example, we can randomly choose from the union, and use a dishonest-majority malicious MPC protocol like MASCOT, which terminates the computation if a malicious behaviour is detected. This guarantees privacy, given that not all the computation agents are corrupted. This is of probability $\prod_{i=0}^{m-1} \frac{k-i}{n-i} \leq (\frac{k}{n})^m$ for random choice, where n is the number of agents in the union of trusted computation agents of all data providers, k is the number of corrupted agents in the union and m is the number of chosen computation agents. With a sufficiently large group of data providers and a limited k, this probability is close to zero. Therefore, assuming all data providers make

sensible choices, any data provider would accept that even if a few trusted computation agents trusted by others are malicious, the security assumption of MASCOT normally holds, and allow performing the computation. This exhibits the worst case scenario, while a plausible mechanism still exists for users to gain trust. Future work can explore alternative selection algorithms for better suitability in different scenarios, utilizing context-specific information.

5 Empirical Evaluation

5.1 Scalability of the Two MPC Models

Using the same framework, MP-SDPZ [21], we compared the scalability of the two MPC models: direct- and delegated- decentralised MPC. In the benchmarks, each data provider has an array of data, and uses MPC to compute an element-wise operation over all data, and finally sum these results[6]. Our benchmarks cover different computation operations (sum and multiplication), parameters (array sizes and numbers of parties) and protocols (Shamir and MASCOT) in the two models, on a server with 2x 8core Intel E5-2650v2 (2.6GHz) CPUs and 48GB RAM. The main factors/metrics recorded are time, rounds of communications, and data transmission.

The results are shown in Figure 3. In summary, we observed and confirmed the following:

- In all settings, computation cost of delegated-decentralised MPC grows linearly with the number of data providers, while that of direct-decentralised MPC grows polynomially[7].
- Cost for more complex computation (multiplication) grows faster than simpler computation (sum). Sometimes this is in magnitudes of difference (i.e. multiplication vs sum for direct-decentralised).
- Protocols with stronger security guarantees require significantly more resources, and are more costly than more complex computational operations. For example, sum operation in MASCOT is more costly than multiplication in Shamir.

From the results, we conclude that the delegated-decentralised model is more appropriate for a significant decentralised context such as Solid. This can be intuitively explained: increasing the number of data providers equals to increasing players in a direct-decentralised model, thus leading to more rounds of transmissions, and number of communications and data transmission in each round; while more data providers in a delegated-decentralised model results in more clients, but does not change the number of players inter-communicating to perform the main computation.

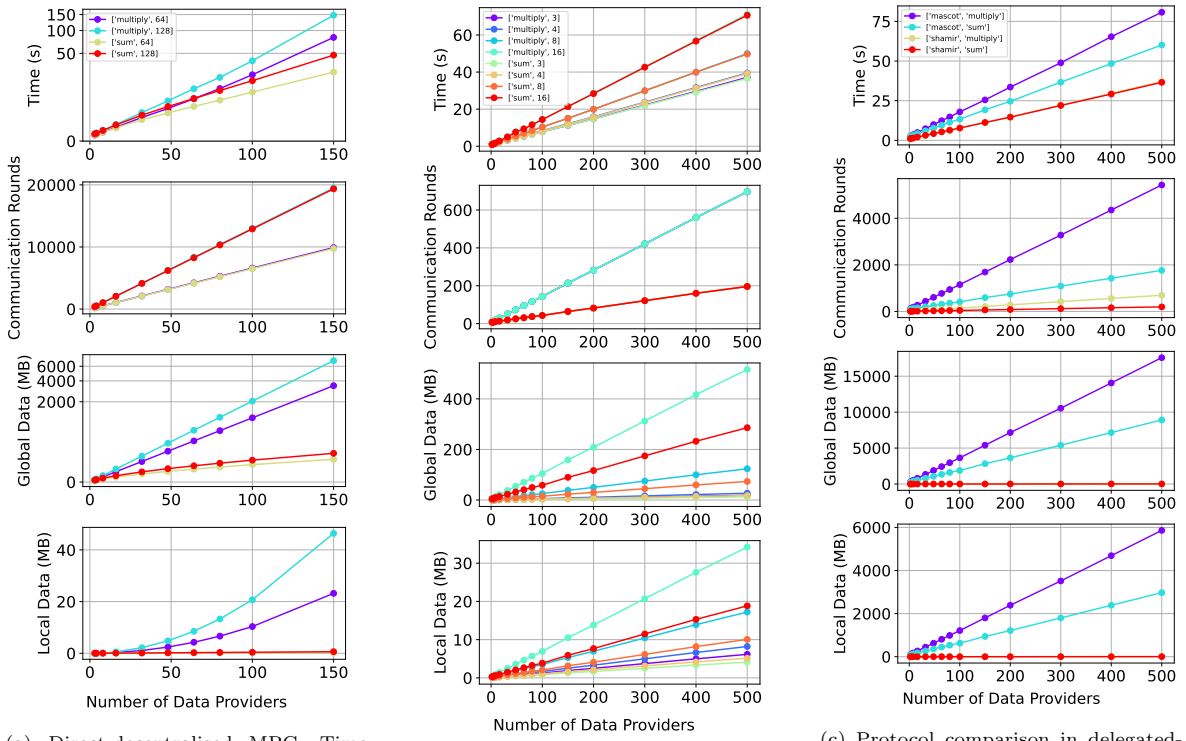
We also found each server can run a sufficient number of players/clients, without impacting the performance. We did not specifically test the maximum capacity of the server hardware, but stopped at the shown numbers (150 players for direct-decentralised and 500 clients for delegated-decentralised) because of soft operating system limits (esp. number of file descriptors). This demonstrates the possibility of running dedicated servers, for the agents proposed in the Libertas architecture.

Finally, in our benchmark, the time required for even basic operations is not ignorable. It is more an issue for protocols with stronger security guarantees. For this reason, the state-of-the-art MPC (or the implementation of MP-SPDZ) is perhaps less ideal for tasks requiring real-time processing in large decentralised contexts. But it is worth noting that we did not fine-tune parameters, and the time includes establishing connections between all parties. They will all impact the performance; further research in MPC may also improve performance in the future. On the other hand, for tasks that require no real-time processing, this is less of a concern, especially for the tasks where privacy is more important than efficiency (e.g. medical records or highly sensitive and high-stake financial data). The use cases to be discussed below provide some directions.

5.2 Libertas Evaluation in Real-World Use-Cases

To demonstrate how our implementation of Libertas can deliver impact, we evaluate how our implementation can support realistic use cases, such as, gig worker empowerment, and differentially private synthetic dataset generation from decentralised data sources.

[6] E.g. in two players setting with multiplication as operation, this is equivalent to computing the dot-product.
[7] Quadratically for sum; cubically for multiplication.



(a) Direct-decentralised MPC. Time and global data are in cubic-root scale. Legend shows the computation and array size.

(b) Delegated-decentralised MPC. Legend shows the computation and number of players. Array size is 128.

(c) Protocol comparison in delegated-decentralised MPC, with 3 players and array size 128. Legend shows the protocol and computation.

Fig. 3: Benchmark results of two MPC models in different settings

Gig Worker Empowerment Gig workers are vulnerable to be unfairly treated by their work platform. They already self-organise to collectively understand their working conditions through various online channels, but risks the exposure of sensitive information and trusting moderators [48]. Several recent studies have highlighted that the preservation of workers' personal privacy is critical for their uptake of any new paradigm, as sharing salary information or work patterns may jeopardise their earnings or competitiveness amongst other workers [49],[34]. We envision Libertas can facilitate this, by providing each worker a Pod and performing privacy-protecting collective computation. We implemented an "average wage" circuit, calculating the average wage from individual earnings. We conducted experiments with different numbers of data providers (10 - 1000) to understand the scalability pattern.

Synthetic Dataset Generation with Differential Privacy As discussed in Section 2.2, while MPC ensures that the inputs used in a computation remain undisclosed, further privacy protection is required for the computation results. Differential Privacy [16, 15] gives a formal definition of privacy. In simple words, it means a function or algorithm, $\mathcal{M}$, produces indistinguishable outputs for two datasets that only differ by a single entry, bounded by the privacy parameter ϵ. Therefore, it protects that the output cannot be used to infer if a particular data record exists in the dataset, or if a particular data provider contributed to the computation (assuming only contributing one data record).

In this paper, we consider the example of differentially-private synthetic data generation. There exist several algorithms for generating synthetic data with differential privacy guarantees. The algorithms have various strengths and limitations [42, 40], which is an active area of research. We propose a novel idea of implementing these algorithms in the Libertas architecture for decentralised personal data stores. This idea has several merits:

1. Because privacy protection at both input end and output end is ensured, users may be encouraged to participate in synthetic data generation and help create more high-quality and privacy-friendly open synthetic data for common good.

2. Because running queries and analysis on differentially-private synthetic data is privacy-friendly, we only have to use costly MPC once to generate the synthetic data and then make this synthetic data available for running queries normally. This is a much more scalable approach than using costly MPC for every query and analysis directly on sensitive personal data stored in personal data stores.

Libertas ensures that the differentially-private synthetic data can be generated without requiring a trusted center and without the need to add noise to individual data points, while respecting user autonomy at the same time.

We implemented the classic Multiplicative Weights and Exponential Mechanism (MWEM) algorithm [19] for differential privacy as an MPC circuit[8], and initiate computation through our prototype App. MWEM is an iterative algorithm for constructing a synthetic dataset that is close to the original dataset in answering the queries. In the experiment, data (and preferences) are distributed in different resources under different containers, simulating different Pods.

We consider the following settings: 1) fix the number of total amount of data points (10000), and evenly distribute them among data providers; 2) fix the number of data points per data provider (100). We experimented with different numbers of data providers, ranging from 10 to 1000. The MPC MWEM circuit converts the data to histogram, using a fixed number of bins (10), and performs the MWEM algorithm with 60 randomly pre-generated queries for 30 iterations ($T = 30$) and epsilon value 1 ($\epsilon = 1$). In addition, we performed another set of benchmarks with a simple optimisation (Setting 3): let the clients create (local) histograms and send them instead of raw data (instead of players creating histograms after receiving data).

Experimental Settings We deployed 3 computation agent servers and 1 encryption agent server, connected over (virtual) LAN. Each computation agent is on its own server, reflecting the non-colluding requirement of computation agents discussed previously. All servers have 2x Quad core Intel E5520 (2.2GHz) CPU and 12GB RAM. Data is deployed on a Solid server running Community Solid Server [1], separated from the agent servers. We use Shamir protocol, with the default parameters from MP-SPDZ. We record the relevant factors for the computation on the 1st player. Specifically, we record the time for the whole job (*full-time*)[9], as well as the time after all connections are established (*comp-time*)[10]. Each task is repeated for a minimum of 10 times to obtain the average result in case of random fluctuations.

Results Fig 4 and 5 show the experiment results in different settings. As can be observed, they all have a linear growth trend of all factors with the increase of number of data providers. That matches the linear trend we observed in the platform-agnostic benchmark for delegated-decentralised MPC (Sec 5.1), confirming the general scalability of the architecture.

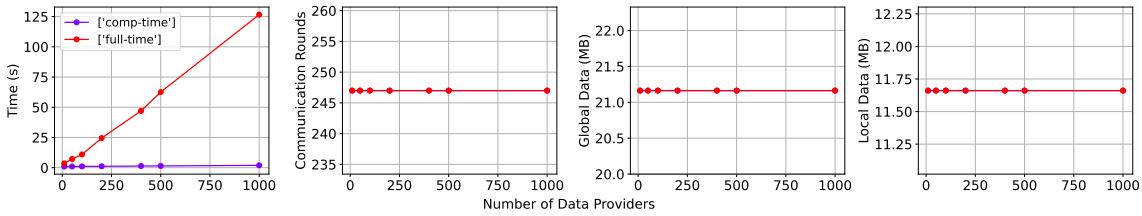


Fig. 4: Results for average wage computation in Libertas in the gig workers scenario.

Zooming into Fig 5, by contrasting Setting 1 (green lines) with Setting 2 (blue lines), we observe that the amount of total data impacts overall computation cost significantly. On the other hand, the red lines show that the resource requirement for Setting 3 (client-binning optimisation) decreased to even

[8] We implemented MWEM of 1-D dataset (integers), taking the final distribution as output, and do not perform mini-iterations during the multiplicative weights update step. With the produced distribution, one can sample a synthetic dataset.

[9] More accurately, this is from when the players connected to each other, and until the finish of the computation.

[10] This eliminates the time for encryption agents to prepare and download data, and establish connection with computation agents.

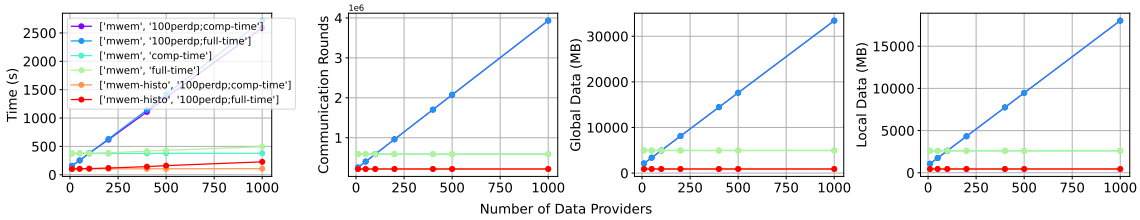


Fig. 5: Results for differentially-private synthetic data generation (MWEM) computation in Libertas. In the legend, `mwem` means the basic MWEM implementation, while `mwem-histo` means with the client-binning optimisation; `100perdp` means each data provider provides 100 data points.

lower than that for Setting 1. This shows that a naive implementation may easily encounter performance bottlenecks with large number of data providers, such as in the process of creating histogram from raw data. While some simple optimisation such as the client-binning can significantly lower the cost.

Further, we observe that full-time (time starting from the beginning until the end of computation) grows faster than comp-time (time for main computation only). We refer to this difference as the *setup time*, which increases with number of data providers because of additional time for encryption agents to prepare and establish connection with computation agents. This could be due to Pods' performance, network, operating system, or other factors, applying to all methods dealing with distributed data providers; it could also be due to MP-SPDZ's implementation. If separating this and only considering the compute time, we observe that the time has a slow growth with number of data providers for both Setting 1 (e.g. 374.9s for 10 data providers vs 379.4s for 1000 data providers) and Setting 3 (e.g. 101.6s for 10 data providers vs 106.6s for 1000 data providers). This slow computation growth trend demonstrates the promising potential for Libertas to scale to significantly many data providers, given the setup time can be optimised outside MPC main computation.

In real-world deployment, the network condition between computation agents will also affect the performance. Task-specific optimisations may significantly improve the performance, such as the one we showed with client-binning. More generally, this may include optimising communication rounds and data transmission by take into account the exact computations involved (in this case MWEM algorithm), the code and the MPC protocol.

The gig worker empowerment scenario demonstrated a similar growth trend (Fig 4). One key difference is that the setup time dominates the overall time when the number of data providers is large. This shows that setup cost should be taken into account if the computation is rather simple, for Libertas or MPC in decentralised context in general.

In general, our evaluation shows that the proposed framework has a good scalability – the computation cost scales linearly with respect to the number of data providers, consistent with that from our discovery in the platform-agnostic benchmark; it can be significantly optimised based on the specific computational task being implemented. We consider the result provides a promising demonstration of the technical feasibility and scalability of Libertas's implementation with Solid, as well as wide applicability over diverse computation tasks.

6 Conclusion and Future Work

Privacy-preserving computation is a critical component for decentralised data architectures, where users can exercise their digital autonomy and benefit from collective value of data, all without the cost of losing control of personal data privacy. In this paper, we addressed various challenges of building an end-to-end solution for this problem. We proposed a novel architecture called Libertas for integrating privacy-preserving computation mechanisms like secure multi-party computation and differential privacy with personal data stores in a modular fashion. We also discussed a novel prototype implementation of this architecture in Solid, while keeping compatibility with existing protocols, through the so-called delegated-decentralised model, which provides better scalability as verified by our empirical evaluations. As discussed in Sec 4, the proposed solution possesses several features that will benefit different stakeholders, and variants of it can be adapted to different PDS systems. Further, we evaluated our proposed architecture using two realistic scenarios, synthetic dataset generation with differential privacy and gig

worker empowerment, demonstrated the wide applicability of our architecture to high-impact privacy-preserving computation use-cases; the empirical results also verified the scalability, as well as shedding lights on possible routes of optimisation when used in production. Our work provides a promising direction for empowering users with privacy-preserving and autonomy-respecting collective computation, thus incentivising adoption of decentralised web technologies like Solid, which is crucial to a privacy-friendly and mutually beneficial web and data ecosystem.

We also note a few limitations of our current work, and expect future work to be taken to better address different scenarios. It would be useful to optimise the performance, particularly in a WAN setting, and evaluate the technology in production settings. It will also be beneficial to explore alternative ways to manage and express trust of agents, such as different levels of trust or dynamic preferences based on personnel, MPC computations, protocols, etc. Further work may also explore variants of the architecture such as integrating components with underlying data architectures; automated selection between MPC models and/or protocols for optimal balance between efficiency and privacy is also an interesting direction.

Acknowledgement

This work is a part of the Ethical Web and Data Infrastructure in the Age of AI (EWADA) project, funded by Oxford Martin School.
In addition, we express our special thanks to Prof. Malcolm Atkinson for his helpful suggestions in improving the presentation.

References

1. Community Solid Server (May 2023), URL https://github.com/CommunitySolidServer/CommunitySolidServer
2. Abadi, A., Dong, C., Murdoch, S.J., Terzis, S.: Multi-party Updatable Delegated Private Set Intersection. In: Eyal, I., Garay, J. (eds.) Financial Cryptography and Data Security, pp. 100–119, Lecture Notes in Computer Science, Springer International Publishing, Cham (2022), ISBN 978-3-031-18283-9, https://doi.org/10.1007/978-3-031-18283-9_6
3. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pp. 308–318 (2016)
4. Anciaux, N., Bonnet, P., Bouganim, L., Nguyen, B., Pucheral, P., Sandu Popa, I., Scerri, G.: Personal Data Management Systems: The security and functionality standpoint. Information Systems **80**, 13–35 (Feb 2019), ISSN 0306-4379, https://doi.org/10.1016/j.is.2018.09.002, URL https://www.sciencedirect.com/science/article/pii/S0306437918304022
5. Baeck, P., Reynolds, S.: Using collective intelligence to address climate change (Jul 2020), URL https://www.themj.co.uk/Using-collective-intelligence-to-address-climate-change/218184
6. Bell, J., Gascon, A., Ghazi, B., Kumar, R., Manurangsi, P., Raykova, M., Schoppmann, P.: Distributed, private, sparse histograms in the two-server model. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (2022)
7. Blakley, G.R.: Safeguarding cryptographic keys. In: Managing requirements knowledge, international workshop on(AFIPS), p. 313, IEEE Computer Society (Dec 1979), https://doi.org/10.1109/AFIPS.1979.98, URL https://www.computer.org/csdl/proceedings/afips/1979/5087/00/50870313-abs.html, place: New York
8. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: European symposium on research in computer security, pp. 192–206 (2008), ISBN 978-3-540-88313-5, https://doi.org/10.1007/978-3-540-88313-5_13, tex.organization: Springer
9. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
10. Capadisli, S.: Web Access Control (Jul 2022), URL https://solid.github.io/web-access-control-spec/
11. Costan, V., Devadas, S.: Intel sgx explained. Cryptology ePrint Archive (2016)
12. Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential Benchmarking Based on Multiparty Computation. In: Grossklags, J., Preneel, B. (eds.) Financial Cryptography and Data Security, pp. 169–187, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2017), ISBN 978-3-662-54970-4, https://doi.org/10.1007/978-3-662-54970-4_10

13. De Montjoye, Y.A., Shmueli, E., Wang, S.S., Pentland, A.S.: openpds: Protecting the privacy of metadata through safeanswers. PloS one **9**(7), e98790 (2014)
14. Dwork, C.: Differential privacy. In: International colloquium on automata, languages, and programming, pp. 1–12, Springer (2006)
15. Dwork, C.: Differential Privacy: A Survey of Results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) Theory and Applications of Models of Computation, pp. 1–19, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2008), ISBN 978-3-540-79228-4, https://doi.org/10.1007/978-3-540-79228-4_1
16. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography, pp. 265–284, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2006), ISBN 978-3-540-32732-5, https://doi.org/10.1007/11681878_14
17. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004, pp. 1–19, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2004), ISBN 978-3-540-24676-3, https://doi.org/10.1007/978-3-540-24676-3_1
18. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing, pp. 169–178 (2009)
19. Hardt, M., Ligett, K., Mcsherry, F.: A Simple and Practical Algorithm for Differentially Private Data Release. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2, pp. 2339–2347, NIPS'12, Curran Associates Inc., Red Hook, NY, USA (2012)
20. Ji, Z., Lipton, Z.C., Elkan, C.: Differential privacy and machine learning: a survey and review. arXiv preprint arXiv:1412.7584 (2014)
21. Keller, M.: MP-SPDZ: A Versatile Framework for Multi-Party Computation. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1575–1590, ACM, Virtual Event USA (Oct 2020), ISBN 978-1-4503-7089-9, https://doi.org/10.1145/3372297.3417872, URL https://dl.acm.org/doi/10.1145/3372297.3417872
22. Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 830–842, CCS '16, Association for Computing Machinery, New York, NY, USA (2016), ISBN 978-1-4503-4139-4, https://doi.org/10.1145/2976749.2978357, URL http://doi.org/10.1145/2976749.2978357
23. Keller, M., Pastro, V., Rotaru, D.: Overdrive: Making SPDZ Great Again. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, pp. 158–189, Lecture Notes in Computer Science, Springer International Publishing, Cham (2018), ISBN 978-3-319-78372-7, https://doi.org/10.1007/978-3-319-78372-7_6
24. Kollnig, K., Binns, R., Kleek, M.V., Lyngs, U., Zhao, J., Tinsman, C., Shadbolt, N.: Before and after GDPR: tracking in mobile apps. Internet Policy Review **10**(4), 30 (2021)
25. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
26. Lindell, Y.: Secure multiparty computation. Communications of the ACM **64**(1), 86–96 (2020), ISSN 0001-0782, https://doi.org/10.1145/3387108, URL http://doi.org/10.1145/3387108
27. Mansour, E., Sambra, A.V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., Aboulnaga, A., Berners-Lee, T.: A demonstration of the solid platform for social web applications. In: Proceedings of the 25th international conference companion on world wide web, pp. 223–226 (2016)
28. Meurisch, C., Bayrak, B., Mühlhäuser, M.: Privacy-preserving ai services through data decentralization. In: Proceedings of The Web Conference 2020, pp. 190–200 (2020)
29. Meurisch, C., Mühlhäuser, M.: Data protection in ai services: a survey. ACM Computing Surveys (CSUR) **54**(2), 1–38 (2021)
30. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP), pp. 19–38, IEEE (2017)
31. Mortier, R., Zhao, J., Crowcroft, J., Wang, L., Li, Q., Haddadi, H., Amar, Y., Crabtree, A., Colley, J., Lodge, T., et al.: Personal data management with the databox: What's inside the box? In: Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking, pp. 49–54 (2016)
32. Obar, J.A., Oeldorf-Hirsch, A.: The biggest lie on the Internet: ignoring the privacy policies and terms of service policies of social networking services. Information, Communication & Society **23**(1), 128–147 (Jan 2020), ISSN 1369-118X, https://doi.org/10.1080/1369118X.2018.1486870, URL https://doi.org/10.1080/1369118X.2018.1486870
33. Oded, G.: Foundations of cryptography: volume 2, basic applications (2009)
34. Open Data Institute, Projects by IF: Perceptions of "Bottom-up Data Institutions": A report on research findings and key learnings for ODI. Tech. rep., Open Data Institute, London (2022)
35. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. ACM Sigkdd Explorations Newsletter **4**(2), 12–19 (2002)
36. Rabin, M.O.: How to exchange secrets with oblivious transfer. Tech. rep. (1981), URL https://www.iacr.org/museum/rabin-obt/obtrans-eprint187.pdf

37. Rouhani, B.D., Riazi, M.S., Koushanfar, F.: Deepsecure: Scalable provably-secure deep learning. In: Proceedings of the 55th annual design automation conference, pp. 1–6 (2018)
38. Sambra, A.V., Mansour, E., Hawke, S., Zereba, M., Greco, N., Ghanem, A., Zagidulin, D., Aboulnaga, A., Berners-Lee, T.: Solid: a platform for decentralized social applications based on linked data. MIT CSAIL & Qatar Computing Research Institute, Tech. Rep. (2016)
39. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979), ISSN 0001-0782, https://doi.org/10.1145/359168.359176, URL http://doi.org/10.1145/359168.359176
40. Stadler, T., Oprisanu, B., Troncoso, C.: Synthetic data–anonymisation groundhog day. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 1451–1468 (2022)
41. Stein, J.M.L., Vizgirda, V., Van Kleek, M., Binns, R., Zhao, J., Zhao, R., Goel, N., Chalhoub, G., Albayaydh, W.S., Shadbolt, N.: 'You are you and the app. There's nobody else.': Building Worker-Designed Data Institutions within Platform Hegemony. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, pp. 1–26, CHI '23, Association for Computing Machinery, New York, NY, USA (2023), ISBN 978-1-4503-9421-5, https://doi.org/10.1145/3544548.3581114, URL https://doi.org/10.1145/3544548.3581114
42. Tao, Y., McKenna, R., Hay, M., Machanavajjhala, A., Miklau, G.: Benchmarking differentially private synthetic data generation algorithms. arXiv preprint arXiv:2112.09238 (2021)
43. Troncoso, C., et al.: Decentralized Privacy-Preserving Proximity Tracing (2020), _eprint: 2005.12273
44. Virginia Balseiro, Timea Turdean, Jeff Zucker: Solid WebID Profile (Jun 2022), URL https://solid.github.io/webid-profile/
45. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
46. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), pp. 160–164 (Nov 1982), https://doi.org/10.1109/SFCS.1982.38, iSSN: 0272-5428
47. Yao, A.C.C.: How to generate and exchange secrets. In: Foundations of computer science, 1986., 27th annual symposium on, pp. 162–167 (1986), tex.organization: IEEE
48. Yao, Z., Weden, S., Emerlyn, L., Zhu, H., Kraut, R.E.: Together But Alone: Atomization and Peer Support among Gig Workers. Proceedings of the ACM on Human-Computer Interaction **5**(CSCW2), 1–29 (2021), publisher: ACM New York, NY, USA
49. Zhang, A., Boltz, A., Wang, C.W., Lee, M.K.: Algorithmic Management Reimagined For Workers and By Workers: Centering Worker Well-Being in Gig Work. In: CHI Conference on Human Factors in Computing Systems, pp. 1–20, ACM, New Orleans LA USA (Apr 2022), ISBN 978-1-4503-9157-3, https://doi.org/10.1145/3491102.3501866, URL https://dl.acm.org/doi/10.1145/3491102.3501866
50. Zhao, Y., Haddadi, H., Skillman, S., Enshaeifar, S., Barnaghi, P.: Privacy-preserving activity and health monitoring on databox. In: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, pp. 49–54, EdgeSys '20, ACM, Heraklion Greece (Apr 2020), ISBN 978-1-4503-7132-2, https://doi.org/10.1145/3378679.3394529, URL https://dl.acm.org/doi/10.1145/3378679.3394529
51. Zuboff, S.: The age of surveillance capitalism: The fight for a human future at the new frontier of power: Barack Obama's books of 2019. Profile books (2019)