Role-Based Ecosystem Model for Design, Development, and Deployment of Secure Multi-Party Data Analytics Applications

Andrei Lapets Mayank Varia Azer Bestavros Frederick Jansen

Department of Computer Science, Boston University
111 Cummington Mall
Boston, MA USA 02215
{lapets, varia, best, fjansen}@bu.edu

Abstract

Individuals and organizations face a tension between (1) the explosion in the amount of valuable data that can be collected and processed and (2) the liability of possession and the threat of exposure of data (which may be sensitive) due to malicious actors, criminal enterprises, and software errors. These threats can lead entities to protect their data throughout its lifecycle, discouraging them from sharing it (or even assessing if sharing has value). Consequently, opportunities to benefit from collaborative data analysis are lost. Secure multi-party computation (MPC) can recover these opportunities and empower both individuals and organizations to benefit from collective data aggregation and analysis in contexts where data sharing is encumbered by confidentiality concerns, legal restrictions, or corporate policies. Theoretical constructs for MPC have been known for 35 years, with several existing software frameworks designed over the past 10 years. Successful examples of deploying MPC for social good include tax fraud detection, disease surveillance, and pay equity assessment.

Our own experiences deploying MPC indicate that the technology is beyond ready for transition and deployment in the real world *for appropriate scenarios and at suitable scales*. MPC's benefits are often subtle and identifying compatible scenarios that would benefit from MPC is a multi-disciplinary array of challenges. Many difficulties and opportunities remain in terms of both the accessibility and the scalability of the candidate solutions for a given scenario. How can the community ensure that further research and development efforts lead to building blocks that will have the flexibility necessary to fit idiosyncratic real-world use cases?

Based on our insights, we advocate for the construction of a collection of production-quality, modular, open-source components that can support a broad *ecosystem* in which organizations and developers can rapidly spin up applications that employ MPC (or related technologies) to protect security-sensitive data, perform privacy-preserving computations, and enable new opportunities for collective data analysis that are currently inhibited or disincentivized by legal, institutional, or economic constraints. Such an ecosystem can allow and incentivize data owners and a diverse assortment of service providers to leverage sensitive data in deriving new insights that serve participant goals and/or the public interest.

In addition to its security benefits, the ecosystem vision facilitates separation of responsibilities and areas or expertise by decoupling the work of software engineers, lawyers, data analysts, communications infrastructure providers, cloud providers, and others. Crucial ingredients for a realization of such an ecosystem include modular design of functionalities that enable delivery of MPC services, composable security analyses of such functionalities, policy-agnostic programming and static analysis techniques that enable modularity and scalability, and accessible and scalable production-quality software applications that utilize MPC functionalities.

1 Introduction and Overview

Enabling Collaborative Data Analytics. We advocate for the development and implementation of an open-source, accessible, modular, web-based secure multi-party analytics infrastructure built upon and drawn from a rich body of existing work on secure multi-party computation (MPC). Such an infrastructure can enable individuals, researchers, businesses, government agencies, and policymakers to use modern frameworks and development environments to build and deploy applications that allow cooperating parties to compute analytics securely over data belonging to multiple entities without the expense, liability, or privacy loss associated with contributors submitting that data to a single trusted entity. Real-world applications enabled by such an infrastructure can empower their users by enabling new forms of collaborative data analysis that are currently encumbered by legal, corporate, or individual restrictions and data sharing preferences.

To provide the security guarantees expressed in the vision described above, an infrastructure must expose (as a collection of modular components and functionalities) secure multi-party computation, a cryptographic technique that allows independent parties to jointly compute a shared result without revealing their private inputs to the computation. MPC has been an active area of cryptography research for more than 35 years [14, 62, 63, 68]. The past decade has seen intense focus on improving MPC algorithms and designing MPC frameworks that perform quickly enough on specific functions and algorithms likely to be of interest in practice [11, 24, 51, 53, 56]. Due to these efforts, MPC is a mature technology that is ripe for transition.

Secure Multi-Party Computation as an Accessible Service. Real-world applications of MPC can be extremely valuable in a number of settings, with significant payoff to society. One specific example that we have targeted in our preliminary work and to which we refer throughout this report is the use of payroll data from multiple institutions to shed light on pay inequities [18, 21, 48, 52]. Other compelling uses of MPC are numerous; applications include disease surveillance [38], electricity trading markets [12], scientific discovery [4], smart-cities [6], genomics [3], homeland and cyber security [33], global advanced persistent threat identification in corporate network data [9], and prevention of satellite collisions [45].

However, despite a small number of successful deployments of MPC for social good in areas such as tax fraud detection [22], the impact of MPC remains fairly limited to proof-of-concept studies. Adoption of secure MPC is in part hindered by the fact that the existing frameworks are closed source or are engineered to optimize for metrics that will not necessarily drive adoption in practice. Many efforts compete on computational performance for small-scale data, a domain in which all modern frameworks are adequate. Meanwhile, they neglect human-scale performance costs: frameworks that require users to become familiar with new domain-specific languages and to set up compatible hardware infrastructures can be burdensome for software engineers and IT administrators to deploy (and/or impractical for laymen to adopt).

Our own experience creating and successfully deploying an MPC solution for a concrete application reflects thoughts expressed by other researchers in the community [69]: "Secure computation is a general scheme; in reality one has to choose an application, starting from a very real business need, and build the solution from the problem itself choosing the right tools, tuning protocol ideas into a reasonable solution, balancing security and privacy needs vs. other constraints: legal, system setting, etc." To meet the needs of our users, we rejected the most algorithmically expressive MPC solutions available in the literature [41] and adopted a variant of the *simplest* of protocols [32]: just expressive enough for the application at hand while being *comprehensible* enough to fuel adoption among corporate officers, legal representatives, and rank and file employees after a few brief meetings with slideshows and whiteboards. We also found that the software platform and IT infrastructure inflexibilities and limitations (legacy systems, restrictive policies, firewalls, and so on) required the most lightweight solution: a simple browser-based application that could accommodate the familiar look and feel of a spreadsheet, with transparent open-source code to enable outside auditing [48, 49].

We also agree with the assertion [69] that only after a solution addresses a concrete need is it appropriate to "Understand, employ, and generalize useful routines: Building more general routines and secure computation software packages of actual business value may, therefore, be a result of collecting various examples of actual useful deployments first, and then creating a common API/software packages based on actual use and experience, in a bottom up fashion." Having reached this stage in our work, we propose a model that will help us (and others) proceed to this next step, as well as to replicate this successful sequence for new use cases and new concrete needs.

2 Role-Based Ecosystem for MPC

Transitioning MPC techniques to practice in a way that drives adoption necessitates a multi-faceted approach that begins well before the software engineering efforts commence. MPC's social benefits cannot be realized unless the decision makers that ultimately choose to adopt MPC (*i.e.*, executives, directors, and legal advisors) possess a clear and confident understanding of exactly what role they (and other entities) play in the process, as well as how MPC protects their sensitive data and mathematically guarantees compliance with data sharing restrictions. Once a solution is accepted, it should ideally be easily and rapidly deployable at little or no cost, and should not necessitate changes to existing software and hardware infrastructures. Finally, it should in all aspects support the needs of the end users that must interact with the solution.

One way to introduce MPC capabilities into practical scenarios in a manageable way is to separate existing functionalities into distinct, modular, reusable, and composable components. These components should individually satisfy concrete needs identified by early adopters of MPC facing real-world use cases. These can then be generalized through incorporation of alternative and complementary techniques and software systems found in the literature. Finally, they can be applied in the construction and deployment of novel MPC-based solutions. Our prior and ongoing work reflects our own efforts to advance in this manner, and the envisioned ecosystem would deliver functionalities that make executing such an approach possible.

Functionalities as Modular and Composable Roles. An effective infrastructure should consist of a collection of libraries and standardized application and service components that implement functionalities. The functionalities themselves can be subdivided into distinct *roles* that might be inhabited by different *agents* (such as servers, mobile devices, desktops, human users, organizations, and so on). In a given MPC solution, each agent can inhabit multiple roles.

In typical models of MPC deployment (supported by existing tools found in the literature and in practice), it is often the case that an assumption is made at the time of framework design as to how various kinds of parties (*e.g.*, data contributor, compute party, data analyst, and so on) are involved in the protocol. This, in turn, determines the architecture of the applications that can be constructed (*e.g.*, the data contributor must also act as a compute party, or all parties act as the data analyst that also receives the final result of the computation).

We advocate a clean separation of the duties and functionalities that may be required for an MPC application into roles that are distinct, modular, and composable. The design of ecosystem components (including libraries and application components) can be organized around these roles, and should not predetermine which roles the agents (*i.e.*, servers, devices, application instances, and human users) must inhabit. Ultimately, functionalities can then be exposed via a framework and API as roles that are coupled with information about which agents can inhabit those roles and which needs they can satisfy.

We briefly enumerate the roles we have identified in preliminary work and note that in a given application, each agent may inhabit more than one of the roles simultaneously.

- A *data analyst* specifies the analytics to be computed on data. This is normally done by the analyst with full knowledge of the schema of the data, but not necessarily with advance knowledge about who owns the data or what data sharing constraints apply to that data (in other words, the data analyst's algorithm specification may be *policy-agnostic*).

- A *data contributor* is the user (an individual or an organization) of an application that supplies the sensitive data to be analyzed. Note that the data contributor may not know in advance the particular analyses that others may want to apply to the contributed data (in other words, any policies the data contributor may be able to specify may be *analysis-agnostic* at the time the data is being contributed). Note also that a contributor may not have the sophistication necessary to deploy virtual machines or servers when participating in a protocol (for example, it may be an individual using a mobile device or a web browser).

- A *security* or *policy expert* can provide expertise on how to specify appropriate policies over the data schema (*e.g.*, given legal restrictions, best practices, or constraints chosen by the data contributors). The policy expert may have the expertise to participate in the development of an application, or may have no such expertise and would require an accessible way to express policies on contributed data that the application would then enforce.

- A *compute service provider* has the capacity to deploy and maintain computational resources to allow analytics to be computed (*e.g.*, on data that may already be secret-shared). Once again, for simple applications, the service provider may only have a mobile device or web browser to contribute; in more complex scenarios, the service provider maybe be a large organization or even an entire cloud provider. Such a service provider may also act as a proxy or aid for data contributors or analysts that have limited resources [34].

- A *code distributor* makes available the actual application (or the specification of the particular computation) to be executed by compute parties on data provided by the data contributors. Since the integrity of the computation (and its conformance to the policy specialist's constraints) relies on the trustworthiness of the delivery mechanism, it should be possible to federate trust among code distributors by allowing multiple entities to inhabit this role.

- The *analysis recipient* is the party that receives the result of computing the secure multi-party analytic. Only recipients can benefit from the result of the computation, but different parties may receive different kinds of information (*e.g.*, an individual user might only have access to information about how their own metrics stack up against other users, while a service provider might see aggregate metrics across all users). As before, the recipient may not have sophisticated hardware, software, or technical expertise.

- A *broker* or *market* identifies opportunities that can enable other parties to participate, or provides infrastructure that enables or mediates interactions between entities that inhabit the various roles.

- A *medium* for storage and communications enables coordination between parties and stores data (including secret-shared data) in transit or in the midst of computations. Note that the medium can be distinct from the compute parties (*e.g.*, the data contributors could also be compute parties, but might require an intermediate storage medium to enable the exchange of secret shares during the operation of a protocol).

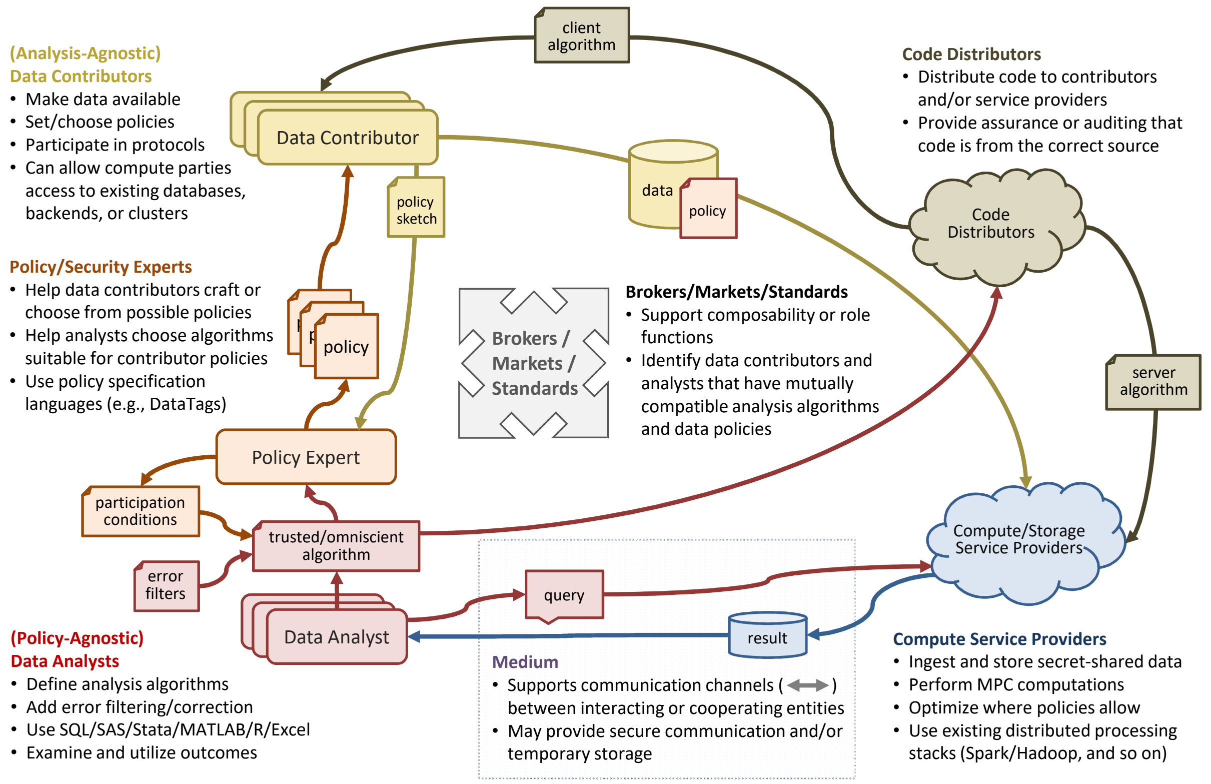
client algorithm
(Analysis-Agnostic) Data Contributors
• Make data available
• Set/choose policies
• Participate in protocols
• Can allow compute parties access to existing databases, backends, or clusters
Data Contributor
policy sketch
data
policy
Code Distributors
• Distribute code to contributors and/or service providers
• Provide assurance or auditing that code is from the correct source
Code Distributors
Policy/Security Experts
• Help data contributors craft or choose from possible policies
• Help analysts choose algorithms suitable for contributor policies
• Use policy specification languages (e.g., DataTags)
policy
Brokers / Markets / Standards
Brokers/Markets/Standards
• Support composability or role functions
• Identify data contributors and analysts that have mutually compatible analysis algorithms and data policies
server algorithm
Policy Expert
participation conditions
trusted/omniscient algorithm
error filters
Compute/Storage Service Providers
query
Data Analyst
result
(Policy-Agnostic) Data Analysts
• Define analysis algorithms
• Add error filtering/correction
• Use SQL/SAS/Stata/MATLAB/R/Excel
• Examine and utilize outcomes
Medium
• Supports communication channels (⟷) between interacting or cooperating entities
• May provide secure communication and/or temporary storage
Compute Service Providers
• Ingest and store secret-shared data
• Perform MPC computations
• Optimize where policies allow
• Use existing distributed processing stacks (Spark/Hadoop, and so on)

Figure 1: Diagram illustrating an instance of the ecosystem, including entities, the roles they inhabit, and the communications that occur between them.

Figure 1 illustrates one possible instantiation of an ecosystem for secure analytics applications that is organized around roles and the functionalities associated with them.

The explicit separation of the roles of data analyst, data contributor, and policy expert, in particular, emphasizes that decisions related to constraints on data sharing across institutional boundaries are orthogonal to the analytics that may be computed over that data: analysts cannot be expected to know all (current and future) data contributors' security constraints and data contributors do not necessarily have the resources, expertise, or foresight to specify fully exhaustive policies. This can be viewed as a generalization of *policy agnostic programming* [15, 66, 67], an approach in which security constraints and the program itself should be specified separately and combined securely via a compilation process. Furthermore, this separation introduces the possibility of synthesizing potential compatible policies from analysis algorithm definitions [64] or interviews of data contributors [17]. The policy expert could then take on the task of evaluating or curating such policies.

3 Roles, Requirements, and Incentives

In our work developing and deploying MPC solutions there has been a need to satisfy a number of concrete requirements that are derived from the constraints imposed by target users (including their level of technical literacy, their access to appropriate kinds of IT infrastructure, and their logistical constraints). We note that some of these needs may need to be met in order to satisfy more general, overarching goals or to provide essential incentives (such as improving usability to drive adoption by a greater number of participants).

Application Requirements. We enumerate a number of these requirements, explicitly referencing how they relate back to the roles described in Section 2.

- *Comprehensibility*: To drive adoption, MPC protocols must be straightforward enough to explain so that decision-makers and users who might not possess technical expertise can be confident that they understand their operation and guarantees. This is most important for data contributors and policy experts, who are responsible for deciding whether participation in a protocol is appropriate, desirable, and safe.

- *Auditability*: To inspire trust, applications must have complete transparency, with open-source code and/or support for outside auditing. In most cases, those most closely involved with a particular application (analysts, contributors, and policy experts) will not possess the resources to inspect source code or perform auditing over the entire application stack. However, this presents an opportunity for those in the code distributor role (*e.g.*, cloud distribution networks may choose to offer this as a service coupled with application delivery).

- *Accessibility*: To minimize any hurdles that might discourage participation by data contributors and data analysts, solutions must be easily and rapidly deployable, requiring no setup, specialized software, specialized hardware, or public internet addresses for agents behind firewalls. The way that compute service providers, code distributors, brokers, and the communications medium are incorporated can all have an effect on an application's accessibility.

- *Simplicity*: The software must be usable within a relatively narrow time window by non-expert human contributors and data analysts whose technical expertise may only include basic familiarity with common software applications.

- *Asynchronicity*: Agents only need to be online while actively performing relevant tasks (*i.e.*, not throughout the duration of a protocol's operation). This would primarily apply to data analysts and data contributors, and may be enabled through the inclusion of compute service providers.

- *Idempotence*: Contributors must be able to resubmit (*i.e.*, update) their data if they discover the data they submitted was corrupted (either through human error, through a software application failure, or both).

- *Non-commitment*: Contributors may decide not to participate, or may be unable to participate due to technical or logistical issues. The protocol should not require advance knowledge of which parties will participate or their quantity.

- *Robustness*: Incorrect or malformed data from even one contributor destroys the value of aggregate analytics. Hence, interfaces must proactively warn users about spurious data. Furthermore, data analysts should have some assistance in assembling algorithms that are robust to outliers or common errors (one natural source of such expertise might be the security experts, though it could come from a distinct entity or toolchain).

Many of the existing protocols and tools discussed in Section 5 can be wrapped, adapted, adopted, or translated into a common infrastructure (using a shared modern language and platform such as JavaScript and Node.js) that arranges them into role variants that satisfy different combinations of the requirements enumerated above (and that can be inhabited by different agents). However, substantial software engineering efforts may be required to do so; in many cases a framework's authors explicitly discourage direct use of their tools in production [11, 36, 53, 56, 58].

Accessibility and Composability. With regard in particular to accessibility: we believe that there is no "one size fits all" accessibility measurement. Different agents in the ecosystem have different usability concerns. For example, different potential participants "speak" different (programming) languages: distributed systems engineers on the cloud speak Spark, data analysts speak R, and lawyers and privacy experts speak their own less-structured languages. As a result, one of the most important accessibility metrics is the participants' ability not to be burdened by understanding the actions and responsibilities of roles they do not inhabit.

Consequently, the viability and success of such an ecosystem depends on the secure distributed applications being *composable* at the software and algorithmic layers. Composability allows each participant to use and build upon prior contributions while only learning succinct API-level specifications of functionality and security. It also facilitates upgrading: if one component is improved, all derived software packages gain its benefits too.

As we mentioned in Section 2, policy agnostic programming provides composability at the software layer. At the algorithmic layer, we advocate for *universal composability* (UC) [27] to provide separation of responsibilities. UC has been specialized and simplified for the MPC setting [29]. Additionally, the value of UC has been shown throughout the computing stack; it has been used to model the security properties of a filesystem [28], a networking protocol [31], and the OpenStack cloud management framework [30].

Incentives: the Ecosystem as a Marketplace. The population of the role-based ecosystem with appropriate, scalable, and usable MPC systems, applications, tools, and libraries provides an opportunity to explicitly address and leverage the *incentives* that may drive the agents that inhabit various roles. This naturally suggests the notion of a *marketplace*, and such a framing mechanism can further motivate, delineate, and constrain development efforts.

The major challenges of populating the marketplace, in our opinion, lie less with the design of faster MPC algorithmic building blocks but with developing developer- and user-centric software packages that span the range of possible security, usability, and performance trade-offs. Interfacing with existing languages and platforms already in widespread use within the community allows use of existing code distribution infrastructures and enables a broader population of users to participate immediately.

We use the term *marketplace* because a successful ecosystem must provide rational incentives for parties to interact. The multiplier effect of composition is not merely a benefit to software engineers but an incentive to deliver modular code in the first place (a principle that is well-known to all software engineers but is sorely lacking in the security domain). The marketplace must also incentivize data owners to make their data available for secure analyses (*e.g.*, by giving responses quicker [16] or more accurately to those who provide more data). Next, by being open to (and in some cases requiring) multiple compute service providers [20], our vision reduces the current cloud computing incentives toward vendor lock-in and instead favors compute service providers who compete and specialize in areas like trusted client application code delivery, untrusted high-performance computing, data analysis, policy synthesis, and reliable data delivery. Finally, a marketplace allows for new types of actors to emerge who *broker* the exchange of information or offer suggested advice on privacy policies governing this exchange. This has the effect of further reducing barriers to entry: data contributors and analysts may not need to work as hard to understand the details of MPC in order to receive its benefits (as others may be incentivized to help them understand or to shield them from having to do so). Finally, we observe that such a marketplace model naturally points to *mechanism design* [47, 55] as an important area of MPC research–one that can lead to opportunities to exploit the benefits of secure computing for maximal social good.

4 Use Cases and Applications in the Ecosystem Context

The recommendations in this report are in part informed by planning, development, and deployment efforts on a number of real-world applications brought forth by local collaborator organizations with a need for accessible, secure analytics solutions that are critical to their missions. We briefly discuss three of these use cases within the context of the proposed role-based ecosystem.

Accessible MPC for Privacy-Preserving Data Analytics In 2013, the Boston Women's Workforce Council (BWWC) [8] initiated a study of gender and ethnicity wage gaps among employers within Greater Boston. Uniquely, they planned to use employer-provided data to quantify and track progress over time. The BWWC has signed over 200 companies [7] to their 100% Talent Compact [1], in which companies agree to provide wage data corresponding to the EEO-1 form [10] that companies are required to submit annually to the Equal Employment Opportunity Commission. For an organization, participating in the tracking portion of the effort involves aggregating data internally (broken down by by gender, ethnicity, and job category) and then contributing that data to a computation of the overall category totals across all organizations.

The BWWC's efforts were initially stymied by a variety of privacy and legal barriers to sharing of sensitive payroll data. Companies refused to submit data to a "trusted third party", and conversely the BWWC had difficulties recruiting an entity to serve in a trusted data collection role due to the risk of being held liable by Compact signers if the payroll data were accidentally leaked or breached. We designed and deployed a web-based MPC system that the BWWC has used twice (and will deploy for a third time in the fall of 2017) to collect payroll analytics securely, thus sidestepping the legal risks involved with handling payroll data [48, 49]. Thanks to MPC, the BWWC has one of the largest publicly-released metrics about pay disparity: the 2016 payroll disparity metrics are based upon the salary data of 112,600 employees that collectively represent about $11 billion in wages and more than 10% of the Greater Boston workforce [2].

In this scenario, the signing companies are the data contributors (and, indirectly, also analysis recipients). The BWWC is the data analyst and also a recipient, viewing the employee earnings totals aggregated across *all* companies; however, the individual company aggregates remain private. Boston University (BU) facilitates the private aggregation by supplying the personnel resources (thus fulfilling the security expert role) and by installing and running the application backend on an Amazon Web Services (AWS) cloud-based server (thus acting together with AWS as a compute service provider). The latter arrangement also means that AWS acts as the code distributor. The internet is the communications medium, with secure end-to-end communications achieved via TLS.

The choice of MPC protocol for this use case was determined by negotiation and reconciliation of a number of trade-offs and constraints, including which of the roles from Section 2 participants could inhabit and which of the requirements listed in Section 3 had to be satisfied. Ultimately, we developed an asynchronous variant of a secret-sharing protocol that allows multiple parties to collectively compute a sum of their own individual quantities without revealing those quantities to one another [32]. This solution was comprehensible, required only one compute service provider, did not burden the data contributors or data analyst, and had acceptable performance characteristics. However, it was also limited in its expressiveness and in its security guarantees: (1) it only allowed for the computation of linear combinations on the input data and (2) it did not protect against the collusion between the compute party and recipient.

We implemented an accessible web application[1] to allow a group of non-expert participants to execute a session of the protocol. The two main components of the application are (1) the client-facing interactive interfaces to be used by the analyzer and participants and (2) the backend service provider that aids in the computation of the analytic and also delivers the client-facing interfaces. The software application automates all portions of the protocol except the initiation of a session (which can be done with a single manual click), the distribution of the session identifier (which is simply delivered to participants via email by the analyst), and the entry of participant data (participants must use the client-facing interface to paste or enter the data before submitting).

Our experience indicates that, for a number of the roles, computational efficiency of the MPC component is not the primary performance bottleneck. For simple analytics over relatively small datasets, all modern frameworks perform rather well (*i.e.*, seconds to minutes) [13]. However, other seemingly unrelated considerations such as choice of client-side cryptographic library can have a substantial effect when working with small data sets. For example, an initial implementation relied on the JSEncrypt library for web-based cryptography, but (because it is interpreted) it suffered from sub-optimal performance compared to compiled cryptography implementations. We chose to use an alternative (interpreted) library that is 20% slower when encrypting data, but offers forward compatibility with native browser cryptography. In turn, a speedup of 3000% on the decryption side was made possible; this accommodated the particular use case, as the analysis recipient had to decrypt a larger amount of data than the data contributors had to encrypt. Furthermore, human time can dominate any form of computing time when a window spanning multiple days is required to collect data from a large number of human contributors operating according to incompatible schedules. In such cases, MPC solutions should prioritize asynchrony rather than computation time. More generally, performance characteristics can be optimized in a variety of ways for each of the proposed roles. Accommodations for exploring such optimization trade-offs can accompany the modular role-based ecosystem components, allowing application developers to meet the needs of specific MPC adopters.

The constituent functionalities of our solution for this use case can be expanded to be general-purpose by adding support for multiplication of input data (which in principle enables the evaluation of *any* computable function) and extending the architecture to support multiple compute parties, thus improving its robustness to collusion. A variety of existing systems could be utilized to accomplish this; however, if *comprehensibility* based on arithmetic is required, then an intuitive extension of the protocol to enable multiplication is to use an additively homomorphic encryption scheme such as Paillier [57] in place of RSA. Under other circumstances, it may be acceptable to use Yao's garbled circuit protocol within the application [60].

The application provides only a single privacy guarantee: passive (also known as *semi-honest*) security without collusion, which essentially states that parties cannot learn any data besides the analytic as long as they all adhere to the protocol [40]. Passive security suffices in this scenario because incentives provided by existing privacy law are leveraged. We claim that the service provider and analyzer lack any clear incentive to falsify the results of the aggregation or to learn private input data; on the contrary, completing the study successfully is directly beneficial to

[1]This application can be viewed at `https://100talent.org` and its source is available at `https://github.com/multiparty/web-mpc`.

BWWC (as the initiator of the study) as well as to BU (as an institution reliant upon a reputation of integrity). Additionally, obtaining any of the contributors' private data would create a liability risk for the service provider and analyzer (as would any other type of collusion between the two). The passive model is very natural in this case as it protects the service providers from any of the usual legal risks of processing sensitive data so long as the parties follow the protocol.

Another potential improvement is to focus on the issue of trusted code distribution. In the existing application architecture, AWS is the sole code distributor and thus represents a compelling attack surface and a single point of failure. One approach that might mitigate risk would be to federate code delivery across a group of distribution services (such as content distribution networks).

Mobile Applications for Digital Health. Hey,Charlie[2] is a digital behavioral intervention platform designed to help individuals recovering from opioid addiction avoid triggers that might lead to relapse and connect with their support network when they need it most. It additionally provides tools to enhance strategy development between patients and clinicians. The platform consists of a mobile application that can track progress and collect data, and a backend web service that stores encrypted data and makes it available for analysis to clinicians. Among its features is an assessment that can track the communication patterns of consenting users with individuals on their contact list (*i.e.*, phone calls and text messages), as well as their travel patterns in daily life. It is desirable for the app to (1) upload tracking data so that users may switch devices and clinicians can get better feedback and (2) aggregate communication patterns across users to provide a better model for all participants. MPC ensures that the aggregation process does not violate users' privacy and confidentiality, or cause the service itself to become a target.

The roles and requirements in this scenario overlap with those of the wage data use case: it is preferable that only one party act as the compute service provider (Hey,Charlie) while users act as data contributors and clinicians act as data analysts and analysis recipients. Depending on the kind of benefits users might experience thanks to insights that can be derived from aggregate data (and automatically incorporated into the app's behavior), the users may also act as analysis recipients. Comprehensibility is not as important in this scenario; this is fortunate, as the particular features of the app necessitate the implementation of privacy-preserving set intersections. This likely requires more sophisticated MPC techniques [44].

Scalable MPC Supporting Heterogeneous Data Stacks. The goal of the *Conclave* framework [65] is to enable MPC deployment in scenarios that involve multiple organizations, each with their own distinct data storage and processing stacks. The framework's design addresses three real-world challenges associated with using MPC at scale: poor integration with existing analytics workflows and data processing systems, a requirement of expert knowledge to implement analytics in an MPC framework, and poor scalability to large data sets. Conclave addresses these challenges by adding secure MPC support to the Musketeer big data workflow manager [39], which automatically generates code for a variety of widely-used data processing frameworks such as Hadoop and Spark. Conclave extends Musketeer to generate Python code automatically for MPC framework backends such as VIFF [11] and Sharemind [24]. Conclave accepts input programs specified in a relational language inspired by LINQ [54]. Experiments [65] using Conclave to compute the Herfindahl-Hirschman Index (HHI) [42] over a large data set [59] showed that running a partion of the computation using MPC in this framework had only negligible overhead compared to an insecure (non-MPC) solution.

[2]More information is available at `http://heycharlie.org/`.

Conclave's design assumes that data contributors are also willing to act as compute parties. It seeks to eliminate the burden on data contributors of tasking their in-house software developers and IT administrators with adopting brand new data storage and computation stacks. The design also seeks to allow data analysts to use a familiar data analysis language rather than learning a new domain-specific language, and to avoid dealing with the issue of data sharing policies (which should be the domain of the data contributors or policy experts). Ultimately, the goal of the framework is to support the policy-agnostic programming approach described in Section 2 in which responsibilities are divided between data contributors, data analysts, and policy experts.

5 Related Work

This report draws on prior work to propose a model for the further development of real-world MPC libraries, systems, and applications that can operate in practical settings. We briefly review related efforts in developing MPC frameworks, deploying MPC applications in the real world, and addressing usability of applications that utilize MPC.

The past few years have seen several successful deployments of MPC [23, 25, 35]. Further, an array of software frameworks is available. These range from proprietary implementations [24, 50] to open-source, proof-of-concept work [11, 19, 26, 36, 37, 53, 56, 58]. Available frameworks vary not only in software maturity, security guarantees, and programming APIs but also in the roles that parties must inhabit to participate, the requirements that can be satisfied, and the communication topology of the participating parties.

Most frameworks do not clearly distinguish between data analysts, data contributors, and policy experts. However, two predominant communication models can be identified: (1) *peer-to-peer* frameworks in which all participating parties are connected to each other and are required to install the MPC framework locally on their own computing infrastructure to participate, and (2) *client-server* frameworks in which there is a central server running an installation of the MPC framework while the other parties are mutually unreachable, web-based clients. This distinction most closely relates to the features of our proposed ecosystem and is of particular interest with regard to accessibility of deployed MPC solutions. We examine some of the frameworks available in each category and highlight some feature-enhancing opportunities and adaptation challenges to be addressed in incorporating these frameworks into the proposed ecosystem.

Peer-to-peer. The majority of existing MPC frameworks follow the peer-to-peer communication model. Among these, most frameworks are open-source research prototypes [11, 19, 26, 36, 37, 53, 56, 58]. In many cases, the framework's authors explicitly discourage use in production [11, 36, 53, 56, 58]; furthermore, in these frameworks all participating parties, including contributors, must deploy the software framework on mutually available servers that remain online for the duration of the protocol execution [19, 26, 37, 46]. The proprietary frameworks [24, 50] are closed-source, which does not meet the auditability criterion. It is important to note that there is a variation of the peer-to-peer model [24] that enhances usability by distinguishing between the role of a contributor and service provider. In this model contributors secret-share their input data among multiple service providers via the browser. The service providers then jointly compute the required analytics over the secret-shared data. We were able to extend VIFF [11] with this functionality during our prototyping phase. In this approach, the analyzer acts as a second service provider and uses VIFF's built-in output mechanism to retrieve the results. Note that this solutions does not meet the criteria of asynchronicity (both service providers must be simultaneously online for the duration of the computation) and accessibility (users lacking the technical expertise to configure a web server, obtain a web certificate, and install our VIFF prototype and its dependencies cannot be accommodated).

Client-server. In the client-server model one entity effectively acts as a service provider while the analyzer and contributors can be viewed as resource-constrained clients. Unfortunately, this setting has received far less attention in the community and available frameworks are sparse [43, 61]. The work by Schröpfer et al. [61] is a web-based implementation of Yao's garbled circuit scheme that allows two browser clients to perform a secure two-party computation leveraging a webserver for communication and code delivery. The framework is closed-source and restricted to two parties. Canon-MPC [43] offers a web-based system that supports MPC with symmetric binary functions. Each participant registers an account on the system, can start a session by choosing a symmetric binary function, and can indicate which other registered users are contributors. The contributors evaluate the agreed-upon function with the aid of a service provider. Each contributor interacts with the service provider exactly once, simultaneously submitting data and performing part of the computation. At the end of the computation the contributors receive the outcome of the function. While Canon-MPC addresses many of the shortcomings of the traditional peer-to-peer model, it falls short of meeting several usability requirements: auditability (the cryptographic library is delivered as a compiled binary and runs in Google's proprietary NativeClient), accessibility (only Google Chrome is supported because of the reliance on NativeClient; Google has also indicated it ceased development on NativeClient [5]), asynchronicity (while participants do not have to enter data in any particular order, no two participants can access the system at the same time; this does not scale past a few participants), and idempotence (data cannot be corrected after submission). Furthermore, we remark that if not all contributors submit their data, a second pass is required to finish the session. Importantly, Canon-MPC does not distinguish between the contributor and service provider roles: any party that contributes data must also actively participate in the computation. In terms of usability this is, in a sense, a step backwards from the model adopted in peer-to-peer frameworks such as Sharemind which do distinguish between contributors and service providers. Minimizing the participation effort for contributors and increasing the robustness of the platform to contributors failing or neglecting to participate is crucial as it addresses the fact that, in some cases, contributors only have weak incentive to contribute data. At the same time it is natural to expect more active and reliable involvement from the analyzers since these parties directly benefit from a successful completion of the computation.

References

[1] 100% Talent: The Boston Women's Compact. https://www.boston.gov/departments/womens-advancement/womens-workforce-council. [Accessed: August 24, 2017].

[2] Boston Women's WOrkforce Council Report 2016. https://www.boston.gov/sites/default/files/bwwc_report_final_january_4_2017.pdf. [Accessed: August 24, 2017].

[3] NCI Cloud Resources. https://cbiit.nci.nih.gov/ncip/cloudresources. [Accessed: August 24, 2017].

[4] Open Science Data Cloud. https://www.opensciencedatacloud.org/. [Accessed: August 24, 2017].

[5] Refactor NaCl integration to eliminate the trusted, in-process plugin. https://bugs.chromium.org/p/chromium/issues/detail?id=239656#c160. [Accessed: August 24, 2017].

[6] SCOPE: A Smart-city Cloud-based Open Platform and Ecosystem. `https://www.bu.edu/hic/research/scope/`. [Accessed: August 24, 2017].

[7] Signers of 100% Talent: The Boston Women's Compact. `https://www.bostonwomensworkforcecouncil.com/compact-signers/`. [Accessed: August 24, 2017].

[8] The Boston Women's Workforce Council. `https://www.boston.gov/departments/womens-advancement/womens-workforce-council`. [Accessed: August 24, 2017].

[9] ThreatExchange. `https://developers.facebook.com/products/threat-exchange`. [Accessed: August 24, 2017].

[10] U.S. Equal Employment Opportunity Commission EEO-1 Survey. `https://www.eeoc.gov/employers/eeo1survey/index.cfm`. [Accessed: August 24, 2017].

[11] VIFF, the Virtual Ideal Functionality Framework. `http://viff.dk/`. [Accessed: August 24, 2017].

[12] A. Abidin, A. Aly, S. Cleemput, and M. A. Mustafa. *An MPC-Based Privacy-Preserving Protocol for a Local Electricity Trading Market*, pages 615–625. Springer International Publishing, Cham, 2016.

[13] D. W. Archer, D. Bogdanov, B. Pinkas, and P. Pullonen. Maturity and Performance of Programmable Secure Computation. *IACR Cryptology ePrint Archive*, 2015(1039).

[14] D. W. Archer, D. Bogdanov, B. Pinkas, and P. Pullonen. Maturity and performance of programmable secure computation. 2015.

[15] T. H. Austin, J. Yang, C. Flanagan, and A. Solar-Lezama. Faceted execution of policy-agnostic programs. pages 15–26, 2013.

[16] P. D. Azar, S. Goldwasser, and S. Park. How to incentivize data-driven collaboration among competing parties. In M. Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 213–225. ACM, 2016.

[17] M. Bar-Sinai, L. Sweeney, and M. Crosas. Datatags, data handling policy spaces and the tags language. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 1–8, May 2016.

[18] R. Barlow. Computational Thinking Breaks a Logjam. `http://www.bu.edu/today/2015/computational-thinking-breaks-a-logjam/`. [Accessed: August 24, 2017].

[19] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.

[20] A. Bestavros and O. Krieger. Toward an open cloud marketplace: Vision and first steps. *IEEE Internet Computing*, 18(1):72–77, 2014.

[21] A. Bestavros, A. Lapets, and M. Varia. User-centric distributed solutions for privacy-preserving analytics. *Communications of the ACM*, 60(2):37–39, February 2017.

[22] D. Bogdanov, M. Jõemets, S. Siim, and M. Vaht. *How the Estonian Tax and Customs Board Evaluated a Tax Fraud Detection System Based on Secure Multi-party Computation*, pages 227–234. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[23] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, and R. Talviste. Students and Taxes: a Privacy-Preserving Study Using Secure Computation. *PoPETs*, 2016(3):117?135, 2016.

[24] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A Framework for Fast Privacy-Preserving Computations. In S. Jajodia and J. Lopez, editors, *Proceedings of the 13th European Symposium on Research in Computer Security - ESORICS'08*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer Berlin / Heidelberg, 2008.

[25] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Financial cryptography and data security. chapter Secure Multiparty Computation Goes Live, pages 325–343. Springer-Verlag, Berlin, Heidelberg, 2009.

[26] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 15–15, Berkeley, CA, USA, 2010. USENIX Association.

[27] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2013 version), 2013. `https://eprint.iacr.org/2000/067/20130717:020004`.

[28] R. Canetti, S. Chari, S. Halevi, B. Pfitzmann, A. Roy, M. Steiner, and W. Venema. Composable security analysis of OS services. *IACR Cryptology ePrint Archive*, 2010:213, 2010.

[29] R. Canetti, A. Cohen, and Y. Lindell. A simpler variant of universally composable security for standard multiparty computation. Cryptology ePrint Archive, Report 2014/553, 2014. `http://eprint.iacr.org/2014/553`.

[30] R. Canetti, J. Hennessey, M. van Dijk, K. Hogan, H. Maleki, R. Rahaeimehr, M. Varia, and H. Zhang. A modular, user-centric security analysis of openstack. In *Computer and Communications Security (in submission)*, 2017.

[31] R. Canetti, K. Hogan, A. Malhotra, and M. Varia. Universally composable treatment of network time protocol. In *Computer Security Foundations*, 2017.

[32] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, Dec. 2002.

[33] L. Constantin. IBM opens up its threat data as part of new security intelligence sharing platform. `http://www.infoworld.com/article/2911154/security/ibm-opens-up-its-threat-data-as-part-of-new-security-intelligence-sharing-platform.html`. [Accessed: August 24, 2017].

[34] M. Dahl, V. Pastro, and M. Poumeyrol. Private data aggregation on a budget. Cryptology ePrint Archive, Report 2017/643, 2017. `http://eprint.iacr.org/2017/643`.

[35] I. Damgård, K. Damgård, K. Nielsen, P. S. Nordholt, and T. Toft. Confidential benchmarking based on multiparty computation. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, pages 169–187, 2016.

[36] D. Demmler, T. Schneider, and M. Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[37] Y. Ejgenberg, M. Farbstein, M. Levy, and Y. Lindell. Scapi: The secure computation application programming interface. iacr cryptology eprint archive, 2012.

[38] K. El Emam, J. Hu, J. Mercer, L. Peyton, M. Kantarcioglu, B. Malin, D. Buckeridge, S. Samet, and C. Earle. A secure protocol for protecting the identity of providers when disclosing data for disease surveillance. *Journal of the American Medical Informatics Association : JAMIA*, 18(3):212?217, May 2011.

[39] I. Gog, M. Schwarzkopf, N. Crooks, M. P. Grosvenor, A. Clement, and S. Hand. Musketeer: all for one, one for all in data processing systems. In *Proceedings of the 10th ACM European Conference on Computer Systems (EuroSys)*, Apr. 2015.

[40] O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[41] A. Hamlin, N. Schear, E. Shen, M. Varia, S. Yakoubov, and A. Yerukhimovich. Cryptography for Big Data Security. In F. Hu, editor, *Chapter in* Big Data: Storage, Sharing, and Security. CRC Press, May 2016.

[42] A. O. Hirschman. The paternity of an index. *The American Economic Review*, 54(5):761–762, 1964.

[43] A. Jarrous and B. Pinkas. Canon-mpc, a system for casual non-interactive secure multi-party computation using native client. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 155–166, New York, NY, USA, 2013. ACM.

[44] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. *Scaling Private Set Intersection to Billion-Element Sets*, pages 195–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[45] L. Kamm and J. Willemson. Secure floating point arithmetic and private satellite collision analysis. *Int. J. Inf. Secur.*, 14(6):531–548, Nov. 2015.

[46] M. Keller, P. Scholl, and N. P. Smart. An architecture for practical actively secure mpc with dishonest majority. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 549–560, New York, NY, USA, 2013. ACM.

[47] V. Krishna. Chapter five - mechanism design. In V. Krishna, editor, *Auction Theory (Second Edition)*, pages 61 – 83. Academic Press, San Diego, second edition edition, 2010.

[48] A. Lapets, N. Volgushev, A. Bestavros, F. Jansen, and M. Varia. Secure Multi-Party Computation for Analytics Deployed as a Lightweight Web Application. Presented at IEEE SecDev 2016: Security Innovation in Design and Development, Boston, MA, USA, November 2016.

[49] A. Lapets, N. Volgushev, A. Bestavros, F. Jansen, and M. Varia. Secure Multi-Party Computation for Analytics Deployed as a Lightweight Web Application. Technical Report BUCS-TR-2016-008, CS Dept., Boston University, July 2016.

[50] J. Launchbury, I. S. Diatchki, T. DuBuisson, and A. Adams-Moran. Efficient lookup-table protocol in secure multiparty computation. In *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming*, ICFP '12, pages 189–200, New York, NY, USA, 2012. ACM.

[51] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.

[52] J. Lipman. Let's Expose the Gender Pay Gap. `http://www.nytimes.com/2015/08/13/opinion/lets-expose-the-gender-pay-gap.html`. [Accessed: August 24, 2017].

[53] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi. Oblivm: A programming framework for secure computation. In *IEEE S & P*, 2015.

[54] E. Meijer, B. Beckman, and G. Bierman. LINQ: Reconciling Object, Relations and XML in the .NET Framework. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 706–706, 2006.

[55] Y. Narahari. *Game Theory and Mechanism Design.* World Scientific Publishing Company Pte. Limited, 2014.

[56] K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, and E. Shi. Graphsc: Parallel secure computation made easy. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 377–394. IEEE Computer Society, 2015.

[57] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[58] A. Rastogi, M. A. Hammer, and M. Hicks. Wysteria: A programming language for generic, mixed-mode multiparty computations. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 655–670, Washington, DC, USA, 2014. IEEE Computer Society.

[59] T. W. Schneider. NYC taxi trip data. `https://github.com/toddwschneider/nyc-taxi-data`. [Accessed: August 24, 2017].

[60] A. Schroepfer and F. Kerschbaum. Demo: Secure computation in javascript. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 849–852, New York, NY, USA, 2011. ACM.

[61] A. Schroepfer and F. Kerschbaum. Demo: Secure computation in javascript. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 849–852, New York, NY, USA, 2011. ACM.

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

[63] E. Shen, M. Varia, R. K. Cunningham, and W. K. Vesey. Cryptographically secure computation. *IEEE Computer*, 48(4):78–81, 2015.

[64] N. Volgushev, A. Lapets, and A. Bestavros. Programming Support for an Integrated Multi-Party Computation and MapReduce Infrastructure. In *Proceedings of HotWeb 2015: The Third IEEE Workshop on Hot Topics in Web Systems and Technologies*, Washington, D.C., USA, November 2015.

[65] N. Volgushev, M. Schwarzkopf, A. Lapets, M. Varia, and A. Bestavros. DEMO: Integrating MPC in Big Data Workflows. In *Proceedings of CCS 2016: The 23rd ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, October 2016.

[66] J. Yang, T. Hance, T. H. Austin, A. Solar-Lezama, C. Flanagan, and S. Chong. End-to-end policy-agnostic security for database-backed applications. *CoRR*, abs/1507.03513, 2015.

[67] J. Yang, K. Yessenov, and A. Solar-Lezama. A language for automatically enforcing privacy policies. pages 85–96, 2012.

[68] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[69] M. Yung. From mental poker to core business: Why and how to deploy secure computation protocols? In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1–2, New York, NY, USA, 2015. ACM.