Multi-party Computation and Trustless Wage Sharing

Recent studies have shown that the disclosure of employee salaries has an effect on mitigating unconscious bias in determining salaries (Miller, 2020). The results suggest that salary transparency can help close the gender wage gap (Bennedsen, et al., 2019). In partnership with Boston University (BU), the Boston Women's Workforce Council (BWWC) has developed an effective solution that promotes salary transparency in the gender wage gap. They have leveraged the power of **Multi-Party Computation** (MPC) to allow employers to securely provide wage information on their employees without the fear that their data will fall into the hands of the government, journalists or competitors.

When the BWWC took on the challenge of the gender wage gap, they understood the high competition, low-trust environment your companies operate in. This led them to lean on cryptography to develop a secure system for your companies to contribute to this greater mission of wage-sharing, which also speaks to the business concerns you face by guaranteeing the utmost security of your data.

We believe that this solution opens the door to furthering improvements on the wage gap, directly impact the lives of millions, and benefit the companies that choose to partner with us on this mission. In this document we present you with a comprehensive explanation of MPC, its implementation for this project, an outline of the role your company would play in its execution, and the opportunity to take part in closing the wage gap.

1. An Introduction to MPC

1.1 A short thought experiment

Computations like wage-sharing would be trivial in an *ideal world* where a **universally trusted party** (UTP) exists. In this world, all participants would communicate with the UTP directly, trusting that their data would not be disclosed (knowingly or unknowingly) to any other parties.

As a concrete example of how the UTC could be used, we can imagine a situation where two hospitals would like to know if the same patient visited them both; however, due to privacy concerns, each hospital is unable to directly confirm or deny information about a specific patient, even including if that patient visited. Here, both hospitals can turn to the UTP; each hospital provides the UTP with their list of patients, the UTP compares the lists in secret, and returns with an answer to whether the patient visited both hospitals, and nothing more. To an outside observer or malicious actor, this computation communicates no information about the patients in the hospital, except if patient X visited both hospitals. Through this system the

hospitals were able to share information with each other without accessing each other's information. In this manner, a UTP would make sharing confidential information easy and secure.

UTPs are baked into everyday interactions, and everyday we interact with them dozens of times. For example, when you wish to pay Petco with your credit card, you trust your bank to remove the price of dog food from your bank account, and deliver it to Petco's bank account; at no point does petco need to know your personal information or how much money is really in your account, both parties simply trust the bank.

However, in many instances a UTP is not available or potentially not even possible, and for such situations cryptographers have developed tools like **multi-party computation (MPC)**, which allows us to replace the trust required to share information privately and securely and resolve the tension between the costs of and benefits of sharing information (Barak, 2021).

1.2 Defining MPC

MPC is a cryptographic tool to create a protocol for parties to follow in order to compute a function over their data while keeping their data private from each other and outside adversaries. In the concrete example of BWWC, the goal is to compute statistics (like average and standard deviation) about multiple parties' data without disclosing any wage data to any other party, and without needing a single trusted party.

A (simple) Formal Definition of MPC:

While some complications will need to be added, we will first provide a slightly technical (and simplified) cryptographic definition of what cryptographers seek to solve.

In MPC, we seek to define a protocol that allows us to compute some function F using data from K number of parties (e.g. what is the average salary of all software engineers in America). We assume that each party will follow the protocol exactly, with the exception of an **adversary** A, which will be able to completely control some subset of the K parties in any fashion.

A protocol is **secure** if it ensures that for any subset T of the K parties and any adversary *A,* there exists an "ideal adversary" *S* such that for any inputs the following two outputs are indistinguishable:

a. The outputs of all the parties, where adversary A controls the inputs to the parties in T, and the rest of the parties behave honestly according to their inputs.

b. The outputs of all the parties, where S chooses the inputs to F, and calculates $y = F(x_0, x_1, \ldots)$. Then for any party in T, S replaces the output with y_i, and for any party not in T, S can choose the output.

More colloquially, the protocol is secure if whatever an adversary can gain from taking complete control over the parties in T, would be gained by only having control of the inputs to those parties in T (Barak, 2021).

A (simple) systems understanding of MPC:

Most simply, an adversary cannot gain any more information than what can already be gained from running the protocol honestly and observing the output of the function. This definition captures the essence of MPC, other definitions also account for scenarios in which we could distinguish between the output in a real setting and the output in an ideal setting, in the case that one party, controlled by an adversary, aborts the protocol, and outputs something different from the ideal adversary *S* (Barak, 2021).

There are three types of roles in an MPC protocol. There are the **contributors** who contribute private data for computation. The number of possible contributors is unbounded and may be kept public. Next, there is an automated, publicly available, **service provider**, who connects all the participants in the MPC and partially computes the analytic. The service provider does not require any participant to maintain their own server, nor synchronously perform any action. Finally, there are one or more **analyzers** who receive the output and may also play a role in its computation (Lapets, 2016).

For the system to work, each role has its own assumptions of trust; MPC is a powerful tool, because there is no single point of failure that requires significant amounts of trust. The analyzers trust that the contributors submit data that is valid. The contributors trust that both the analyzers protect the aggregated output and that no individual analyzer is conspiring with anyone else. If there are multiple analyzers involved then their trust is federated rather than added, meaning that the contributors only need to trust that one analyzer is honest. Most importantly, the service provider requires zero trust with the data from anyone since it never learns the data, and only computes it. The participants only need to believe that the service provider will act on their behalf. In other words, the service provider will not deny service (Lapets, 2016).

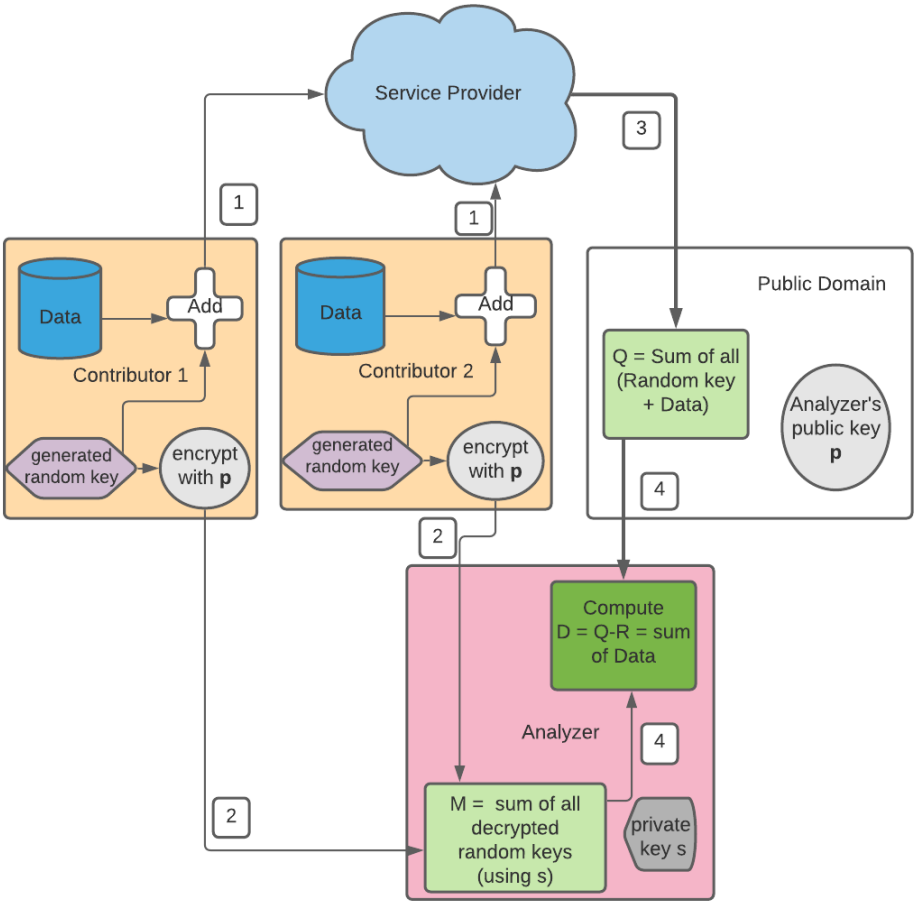
2. One Specific MPC Implementation

2.1 Description of the Algorithm

While there are many forms of MPC and many ways to implement it, the same basic dynamics between contributors, service provider, and analyzers exist. In the case of the BWWC wage-sharing, BWWC used a protocol very similar to what is described below was used (which has been slightly simplified):

1. The analyzer generates a private key “s” and corresponding public key “p”, which they then share publicly.
2. Each contributor (company) converts their data into a number (which is trivial as data is stored in bits), then they generate an unbounded length random key m_i and add it to their data d_i. Let $r_i = d_i + m_i$.
3. Each contributor sends r_i to the service provider.
4. Each contributor encrypts m_i with the analyzer’s public key, then sends the encrypted m_i to the analyzer.
5. The service provider computes the sum $R = \sum r_i$, and publicly shares it.
6. The analyzer decrypts each mask m_i , and computes the sum $M = \sum m_i$.
7. The analyzer computes D= R - M, which is equal to $\sum d_i$, through associativity (the rule that order of operations doesn’t matter for addition: (a+ b)-c -d = (a-c) + (b-d)).

See diagram below for a visual representation:



2.2 Extensions: Higher moments

While we have so far discussed how to compute a sum, it is trivial to see how this would allow us to compute an average (divide the sum by n, where n is the number of parties). It is also straightforward to see how higher-moments (like for variance) could be computed by modifying the computation that the individual contributors make from [data + random_key] to [data * probability(data)^j + random_key], in order to compute the jth moment of the data.

2.3 What claims can we make about security and trust

We now will show that no data from an individual contributor is at risk of being identified by an adversary, and we will dig deeper into what minimal trust we are placing in which parties (and why).

First we observe that an unbounded length, one-time, random key added to a string of data is perfectly secure, and it is impossible to get any information about the data if no information about the key is ever shared and the key is never used again.

Intuitively, if X = A + B, and B can be any real number, knowing X does not allow you to determine any information about A. For any value of A, there is exactly 1 value of B that satisfies X = A + B, and since B is completely uniformly random, all values of B are equally likely, therefore any A, all values of X are equally likely. Here we see that no information is shared with the service provider by each contributor sharing [(data) + (random key)]. The service provider computes the sum of all the values and shares it publicly; since each piece of data contains no information, their sum also contains no distinguishable data.

We assume that the reader is familiar with the concept of public-key encryption, and so won't discuss this here. We also assume that public-key encryption is infeasible to break, so the contributors are able to share their random keys in an encrypted format with the analyzer without any risk of someone else learning it. In the end, the analyzer is able to privately compute the sum of all the unencrypted random values, and to subtract it from the public sum the service provider published.

In the end we see that there are only 2 points of trust : 1. Public-key encryption isn't broken (which we believe is an incredibly weak assumption), and 2. The service provider does not collude with the analyzer, to store the values ($r_i = d_i + m_i$), then get the value of m_i from the analyzer, and then discover d_i, for a contributor.

This second assumption is actually a non-trivial amount of trust, but worries on this front can be assuaged by requiring the data stored by the service provider to be deleted immediately. This can be confirmed by making the code public or by subjecting the service-provider (or analyzer) to third party audits, we can minimize the amount of trust required.

Only the analyzer gets the final computation F, and no individual party's data is shared during the process.

2.4 Caveats: What can be learned?

While it is true that no information can be learned about an individual contributor from what they share, it is not true that no information can be learned about an individual contributor at all, as that is not the goal of the computation. For example, if the total number of female managers paid more than 100k across all companies is equal to 5, then we know for a fact that *no company* has more than 5 female managers paid more than 100k.

If the goal is to minimize the amount that can be learned about an individual's contribution, we recommend that you look into the use of differential privacy, which seeks to accomplish exactly that.

3. Company Specifics:

3.1 Why is this beneficial to the company, Economic Analysis:

In this MPC, you, the companies, would take on the role of contributors, as described in part 1.2. As contributors your role is to submit to the service provider, in this case, Boston University, cumulative employee earnings aggregated by job category and gender. Once the data is submitted and the aggregates are computed, The analyzer, the BWWC, will be able to see earnings totals aggregated across all the companies who participate. The individual company data and their aggregates will remain private, and no one will be given access to them.

Your company's participation will not only be beneficial to the BWWC's research but the results of the aggregates can benefit your company as well, by providing you with a clear picture of the earnings landscape across companies in your area or industry, without the need for releasing sensitive data to specific agents that are required your trust, and in the process expose your data to security risks while at rest (Lapets, 2016). From these results your companies may encounter insights that allow you to benchmark yourselves across your industry, ensure you are developing the right talent, better understand your workforce, to name a few (Galvin, 2017). Ultimately these insights may help you create new opportunities for yourselves and your employees, address issues, and foster strategies for greater diversity, equity and inclusion.

4. Conclusion

We have now presented you with a novel method for computing computations on large sets of data with minimal trust in any party, and have discussed how information remains private, while enabling a group computation. This document presented a general technique used for computing basic statistics (like summations and moments), which can be applied to any similar situation, not just wage-data. Some other ideas for where MPC is a natural

application is : doing multi-departmental censuses across a company, sharing metadata about medical information between hospitals.

References:

Barak, Boaz. “Multiparty Secure Computation.” *An Intensive Introduction to Cryptography, Lectures 16, 17*, 2021, https://intensecrypto.org/public/lec_17_SFE.html.

Bennedsen, Morten, et al. “Research: Gender Pay Gaps Shrink When Companies Are Required to Disclose Them.” *Harvard Business Review*, 23 Jan. 2019, https://hbr.org/2019/01/research-gender-pay-gaps-shrink-when-companies-are-required-to-disclose-them.

Galvin, Jane. “Five Ways the Gender Pay Gap Reporting Can Benefit Business.” *Reba*, 2017, https://reba.global/content/five-ways-the-gender-pay-gap-reporting-can-benefit-business.

Lapets, A., et al. “Secure Multi-Party Computation for Analytics Deployed as a Lightweight Web Application: Semantic Scholar.” *Boston University*, 2016, https://www.semanticscholar.org/paper/Secure-multi-party-computation-for-analytics-as-a-Lapets-Volgushev/f69d9844bcc6dfb59607ab966fa33db02ce80bd4 .

Miller, Stephen. “Transparency Shrinks Gender Pay Gap.” *SHRM*, SHRM, 31 Jan. 2020, https://www.shrm.org/resourcesandtools/hr-topics/compensation/pages/transparency-shrinks-gender-pay-gap.aspx.