Privacy-Preserving Machine Learning on Web Browsing for Public Opinion

Sam Buxbaum[1], Lucas M. Tassis[1], Lucas Boschelli[2], Giovanni Comarela[3], Mayank Varia[1], Mark Crovella[1], and Dino P. Christenson[4]

[1] Boston University {sambux,ltassis,varia,crovella}@bu.edu
[2] Maritz boschellil@wustl.edu
[3] Universidade Federal do Espirito Santo gc@inf.ufes.br
[4] Washington University dinopc@wustl.edu

Abstract. We present a real-world deployment of secure multiparty computation to predict political preference from private web browsing data. To estimate aggregate preferences for the 2024 U.S. presidential election candidates, we collect and analyze secret-shared data from nearly 8000 users from August 2024 through February 2025, with over 2000 daily active users sustained throughout the bulk of the survey. The use of MPC allows us to compute over sensitive web browsing data that users would otherwise be more hesitant to provide. We collect data using a custom-built Chrome browser extension and perform our analysis using the CrypTen MPC library. To our knowledge, we provide the first implementation under MPC of a model for the learning from label proportions (LLP) problem in machine learning, which allows us to train on unlabeled web browsing data using publicly available polling and election results as the ground truth.

1 Introduction

Secure multi-party computation (MPC) is a cryptographic protocol that allows several people to contribute their data toward a collective data analysis without ever exposing their personal data to any other party. MPC has been a topic of research for decades [10, 29, 62, 71], and through a variety of algorithmic improvements and software implementations (e.g., [13, 39, 47, 51, 63]), it has seen deployments over the past decade in the commercial and public sectors (e.g., [1,6,11,14,24,44,58]). Recently, there has been a focus on specialized MPC algorithms for machine learning operations like gradient descent and logistic regression (e.g., [4,28,49,50,52]) with corresponding software implementations like CrypTen [40].

In this work, we develop and deploy privacy-preserving machine learning in a real-world application based on political science: namely, the estimation of political preferences in the United States during the months around the 2024 U.S. presidential election. To accomplish this goal, we combine MPC-based privacy-preserving machine learning with the work of Comarela et al. [18], which demonstrates that web browsing patterns can be used to assess aggregate preferences in political candidates.

Background. In more detail, Comarela et al. [18] start from a dataset of web browsing records of a cross-section of people provided by a media measurement company. They design a machine learning algorithm based on Learning from Label Proportions (LLP), shown in Algorithm 1, that uses web browsing data to infer the same type of information that is generally produced by political polls: the fraction of voters in each region (e.g., county or state) that prefer a given candidate at a particular point in time. Hence, this work shows potential to augment traditional methods of political polling, since its lower cost enables more frequent and precise polling—an opinion poll can form the initial "ground truth" in one location at one moment in time, and then LLP on web browsing data can be used to predict political preferences at other times and/or locations.

This LLP algorithm is remarkably effective at estimating political preferences, even when given only browsing visits to the most popular websites rather than the long tail of smaller blogs and personal sites. Nevertheless, data privacy concerns make it challenging and risky to collect and store sensitive data about web browsing activity in practice.

Our contributions. In this work, we design, implement, and deploy a privacy-preserving system that assesses candidate preferences while also providing the cryptographic guarantee that web browsing data never leaves client browsers in the clear. In more detail, we contribute the following.

- An MPC system that comprises a client-side browser extension that calculates a daily histogram of visits to popular websites, and a cloud-based MPC backend built on top of CrypTen [40] that uses these secret-shared histograms to calculate aggregate political preferences. We detail the system architectural design in Section 2 and describe our algorithmic innovations in Section 3.
- A case study of the deployment of our MPC system on a panel of nearly 8000 people during the months around the 2024 U.S. presidential election. We describe our work in Section 4, and emphasize upfront that we focus in this paper more on the computational decisions and security considerations involved in deploying MPC in this setting rather than the political science insights resulting from the data analysis.

Ethics. We conclude this section with a brief but important discussion of the ethics of this type of research. We caution that the use of MPC cryptographic technology does not automatically ensure adherence to social, ethical, or legal obligations to protect privacy [68], and as a result we have taken multiple steps to ensure that our deployment respects personal autonomy and social norms of privacy. First, we received approval from an Institutional Review Board to deploy this system, and we provided clear and simple methods for panelists to learn about their data privacy rights and to opt-out of participation at any time if desired. Second, we made sure to design our privacy-preserving machine learning algorithms to reveal only the overall fraction of people in a region (e.g., a U.S. state) that have a particular political preference, and not to infer or reveal any

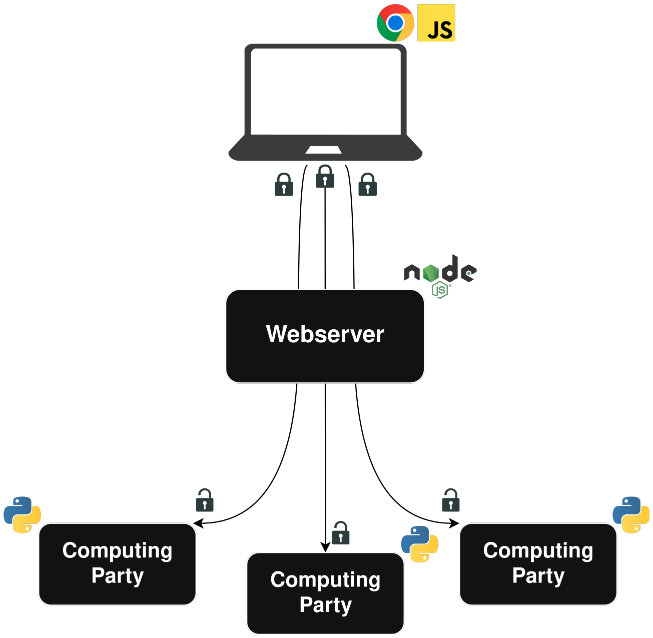

Fig. 1. The OPPS system design. The tools used for each component are included in the upper corner of the component.

information about which people have that preference. Third, we designed the client browser extension so that it becomes “dormant” after the end of the study period: in other words, it automatically stops observing and transmitting secret-shared web browsing data even if the panelist never deletes the extension. Fourth, we provided fair compensation to panelists for their participation (as approved by the IRB), and we partitioned our own research team so that the researchers who connected with the panelists and observed their payment information were unable to access any other part of the system, and vice-versa.

2 System Design and Software Architecture

We perform our analysis using MPC in the honest-majority three-party setting with semi-honest security. We operate in the outsourced MPC setting, where users upload secret shares to a set of non-colluding servers who carry out the computation. Concretely, our system contains three components: (i) a browser extension used by the clients, (ii) a webserver to handle client interaction, and (iii) a set of three non-colluding compute servers to carry out the MPC.

2.1 Browser Extension

We develop a custom Chrome browser extension to monitor users' web browsing behavior. Emulating the ideas in Comarela et al. [18], the extension tracks (i) how many times each user visits each of the most popular 517 websites and (ii) how many times they were referred to each of the top 517 websites from popular social media websites. See Section 4.1 for details on how we obtained the list of websites.

The visit and referral histograms are secret-shared in the browser on the client side, and the shares are encrypted under the public keys of the three MPC servers. We implement secret-sharing and hybrid public-key encryption (with RSA and AES) using the JavaScript Web Crypto API. Once per day, the plugin secret shares, encrypts, and sends the histograms to the webserver.

2.2 Webserver

We use a continously running webserver to handle all communication with the clients and to eliminate the need to have the more expensive computing servers active at all times. The webserver, written in NodeJS, is lightweight and can process many concurrent connections while running on a small machine in the cloud. When a client uploads their encrypted secret shares, the webserver stores them to be retrieved by the computing servers. We emphasize that the webserver exists to simplify the interaction with the clients, but the secret shares are end-to-end encrypted from the client to the computing parties, so the webserver itself learns nothing.

2.3 Computing Servers

We implement the MPC data analysis on top of the CrypTen Python library [40]. The three MPC servers retrieve and decrypt their individual secret shares of the web browsing data from the webserver, and they collectively (and obliviously) train a model or perform inference with a previously trained model.

In order to use CrypTen out of the box, all three MPC servers must be accessible by a single AWS account, which presents an obvious privacy challenge. To overcome the challenge, we place all three servers in the same AWS account, and the three server operators are collectively locked out of the account. Computation is performed using an automated process that allows server operators to communicate with the servers without giving any of them the ability to see any server or its internal state after the lockout procedure. Specifically, the server operators submit an agreed-upon program to the servers, which evaluate the program and publish the result to the server operators.

3 Secure Computation of an LLP Algorithm

In this section, we describe the LLP problem considered in the work of Comarela et al. [18], and then we describe our process to transform this into a secure multi-party computation protocol with data-oblivious control flow.

3.1 The LLP Problem

Learning from Label Proportions (LLP) is a weakly supervised machine learning problem in which the data is divided into groups or *bags* [57]. For each bag, only the proportion of labels is known, i.e., no individual label is available. The goal of LLP is to train a model that can correctly predict a label given a feature vector as input.

To be more specific, the machine learning task associated with this work can be characterized as follows. The input is composed of a set of pairs $D = \{(\mathbf{x}_i, b_i)\}_{i=1}^{N}$ and a vector $\mathbf{p}$, where: each $\mathbf{x}_i$ is a feature vector, representing the web domains visited by an individual in a given period; b_i denotes the bag, or group, that the individual belongs to (in our work b_i is the individual's state of residence); and $\mathbf{p}$ is a vector of proportions, indicating the proportion of individuals with preference for (without loss of generality) the Democratic Party candidate in each state. From this characterization, it is implied that each individual has a label: label 1 for a Democratic Party preference; and label 0 for a Republican Party preference. However, the individuals' labels are unknown. Hence, the goal of the LLP problem is to learn a model that can predict the label of individuals from D and $\mathbf{p}$. For the interested reader, a more detailed presentation of LLP and its possible variants can be found in the work of Franco et al. [26].

There are many algorithms for training machine learning models in LLP setups. In this work, we follow the approach presented by Comarela et al. [18], mainly for two reasons: first, it has been successfully used in similar contexts; and second, it is suitable for implementation in the employed MPC framework (as discussed in Section 3.2). In summary, the algorithm has three steps. The first step is label initialization, which can be performed randomly or according to proportions in each bag (i.e., U.S. state). The remaining two steps employ an iterative procedure that is repeated until convergence. In the second step, we train a logistic regression model using the previously assigned labels. Finally, in the third step, we refine the logistic regression model obtained in the last iteration and use it to compute updated predictions. More specifically, the default threshold (0.5) used to transform the model's probabilities into labels is changed on a per-bag basis, so that the predicted proportion of 1's in each bag (in the training set) matches the expected proportion (from the problem's input). We repeat the second and third steps until the predictions in consecutive iterations have converged.

The algorithm is formally defined in Algorithm 1, where we have combined Algs. 1 and 2 from Comarela et al. [18] for ease of presentation. As input, the algorithm takes the following:

- a data matrix $\mathbf{U}$, where each row is a user, and each column is a feature,
- a region map R, where $R(u)$ denotes the region to which a user u belongs,
- a proportion map B, where $B(r)$ denotes the fraction of users with label 1 for a region r, and
- a learning algorithm $\mathcal{A}$, which we instantiate with a logistic regression model.

As output, the algorithm produces a trained model $\mathbf{\Theta}$. The algorithm uses two labeling functions: $L(u)$ denotes the true label of a user u, while $L'(u)$ denotes our current prediction of the user's label, and $P(L(u) = 1 \mid \mathbf{\Theta})$ is the predicted probability that a user has label 1 according to the model $\mathbf{\Theta}$.

Algorithm 1: Learning Algorithm (from Algorithms 1 and 2 of [18])

```
   Input: $\mathbf{U}, R, B, \mathcal{A}$
   Output: $\mathbf{\Theta}$
1  $L' \leftarrow \{\}$
2  foreach row $u \in U$ do
3      $L'(u) = \begin{cases} 1, & \text{if } B(R(u)) \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$
4  repeat
5      $L'' \leftarrow L'$
6      $\mathbf{\Theta} \leftarrow$ train $\mathcal{A}$ on $(\mathbf{U}, L')$
7      foreach region $r \in R$ do
8          $t_r \leftarrow$ percentile $(1 - B(r))$ of $P(L(u) = 1 \mid \mathbf{\Theta})\ \forall u \in r$
9      foreach row $u \in \mathbf{U}$ do
10         $L'(u) = \begin{cases} 1, & \text{if } P(L(u) = 1 \mid \mathbf{\Theta}) \geq t_{R(u)} \\ 0, & \text{otherwise} \end{cases}$
11 until $L' \approx L''$;
12 return $\mathbf{\Theta}$
```

3.2 Implementation Under MPC

Here we present a method of realizing Algorithm 1 under MPC, which is, to our knowledge, the first instantiaton of a model for the LLP problem under MPC. We use and build upon the CrypTen [40] library. To simplify the following presentation, we freely and implicitly convert between an arithmetic secret sharing and a boolean secret sharing (in both directions) as needed. In practice, this is done using arithmetic-to-boolean and boolean-to-arithmetic conversions under MPC (see e.g., [22, 49, 55]).

Initialization (Lines 1-3). Initialization can be done in plaintext, as both the state proportions and the users' reported states are known in the clear. One could imagine secret sharing the users' states as well. In that case, initialization would entail obliviously selecting the correct state proportion and obliviously comparing it with 0.5.

Model Training (Line 6). Training the underlying model is one of the two cruxes of the computation. We instantiate the underlying learning algorithm $\mathcal{A}$ with a logistic regression model. Logistic regression is supported out-of-the-box by CrypTen, allowing us to use standard features of the library to train $\mathbf{\Theta}$ at each iteration.

Computing Thresholds (Lines 7-8). The other crux of the computation is determining the threshold for each state, because it requires oblivious sorting. Oblivious sorting is a complex and expensive operation under MPC. More importantly, it is not natively supported by CrypTen. We implement a vectorized Bitonic sort [7] on top of CrypTen that uses the library's efficient primitives for batched operations to minimize round complexity. We sort each state's predictions separately, meaning we never sort more than several hundred values in a single call to the sorting algorithm. As a result, we prioritize ease of implementation over maximizing performance.

Given an oblivious sorting implementation, computing thresholds becomes rather straightforward. We compute $P(L(u) = 1 \mid \mathbf{\Theta})$ for each row $u \in \mathbf{U}$ and sort the results. Since the state proportions are known in the clear, we can compute the target index in plaintext and retrieve the (secret-shared) result at that index in the list. We remark that it would certainly be possible to compute thresholds even if the state proportions were not known in the clear, but in this work we assume the proportions are public information and do not need to be protected.

Computing the New Predictions (Lines 9-10). Updating the predictions with the newly-computed model and thresholds is straightforward. For all rows $u \in \mathbf{U}$, we compare the secret-shared inference result $P(L(u) = 1 \mid \mathbf{\Theta})$ with the secret-shared threshold. We can perform the comparisons in a single batch using a vectorized oblivious comparison. We store the secret-shared prediction bit for each user in L'.

Checking Convergence (Line 11). The last step in the algorithm is to check for convergence. We have two lists of secret-shared predictions, L' and L'', and we need to determine their similarity. We first compute an elementwise (vectorized) oblivious equality to obtain a vector of secret-shared bits. The value we wish to compute is the number of those bits that are 1's, and we wish to compare it to some convergence threshold. We sum the vector to obtain a secret sharing of the number of 1's, and we obliviously compare it with the convergence threshold. We open the comparison result, and if the value is greater than the threshold, the opened value is 1, so we terminate the algorithm. If it is 0, we continue to another iteration. The only information revealed by the opening is whether the model has converged or not.

In summary, we implement Algorithm 1 step-by-step while ensuring that each step follows a data-oblivious control flow. We use CrypTen out-of-the-box where possible and implement new functionality where needed.

4 Data and Results

In this section, we describe our case study application of the MPC-for-LLP technique to analyze web browsing data of several thousand participants during the months before and after the November 2024 U.S. presidential election.

4.1 Panelist Data and Metadata

We recruit users through an advertisement on Amazon Mechanical Turk, with an emphasis on recruiting panelists who live in "swing states": that is, states that are likely to have close outcomes in the 2024 U.S. presidential election. Throughout the study, which lasted from August 2024 to February 2025, we collected data from 7926 total unique users, with over 2000 users contributing data daily for the majority of the study.

We collect both web browsing behavior and basic demographic information.

- In terms of web browsing data, the client browser extension contains a list of 517 popular websites, which we obtained by taking the union of the Alexa Top 500 list and the Semrush Top 100 list [61], removing advertising websites. The Alexa Top 500 list is no longer published, so we use the same version as Comarela et al. [18]. The extension records the number of visits that the panelist makes to the website, as well as the number of visits that resulted from referrals from one of the nine most popular social media websites. For all website visits, the extension records only the website's domain, not a complete URL. On a daily basis, the browser uploads the histogram vector of web visits to the web server. Each data upload consists of the three encrypted secret shares of a vector of 1034 elements, along with plaintext metadata described next.
- In terms of demographic metadata, we collect the panelist's Mechanical Turk ID, their self-reported state and ZIP code, the number of total visits to websites in the list, and the number of total referrals to websites in the list.

The Mechanical Turk ID is used to compensate the panelist for their participation. In each data upload, this ID is encrypted using message-locked encryption [9] that is only accessible to a member of our research team who processes the payments; this team member did not have access to any of the MPC compute servers. The MTurk ID is never provided as input to the machine learning algorithms discussed in Section 3, although the team member responsible for payment processing used the MTurk IDs to attach an anonymous ID to each upload for the purpose of anonymously aggregating data from the same user over several days. We discuss anonymous payments in Section 6.2 as a potential avenue for minimizing the trust required in the payment process.

The remaining metadata is used within the machine learning operation: the self-reported state is used to partition the data into bags for each state, and the total visits are used in order to convert counts into fractions. In principle, these operations could be performed under MPC as well (albeit with increased runtime). The state and ZIP code are also used within the data integrity check, described in Section 4.3.

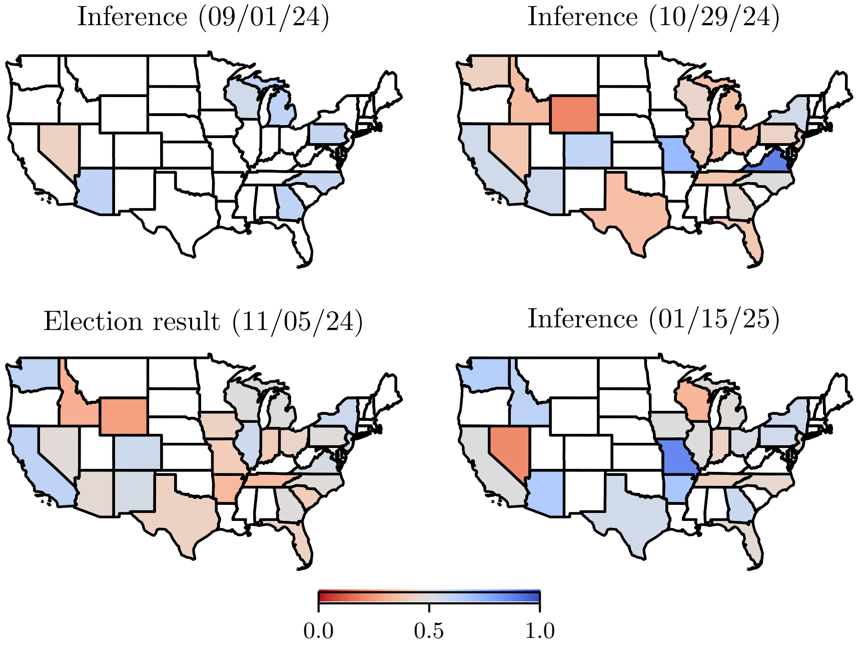

Fig. 2. State-level inference results at three points in time. One point is well before the election (top left), another is the week before the election (top right), and the last is after the election (bottom right). For comparison purposes, we also show the ground-truth election results (bottom left).

4.2 Empirical Results

We use the panelist data together with the actual vote tally in the U.S. presidential election on November 5, 2024 to estimate political preferences at different points in time. As described in Section 3, our machine learning algorithm trains on the ground truth election results and corresponding web visits on November 5 and the surrounding days in order to produce model weights for the influence of each web domain on political preferences and state-by-state 'cutoff' thresholds to assess candidate preferences. From this model, we use the web visit data to infer aggregate political preferences on other days, both backwards and forwards in time.

Findings. An example of our state-level inference results is shown in Figure 2. Here we aggregate up the individual level candidate support scores for a set of states with ample respondents at each of three weekly periods: early in the campaign, election day, and the presidential inauguration. We have colored each state according to the degree of support for each party's candidate, with red indicating more Republican and blue more Democratic.

While a detailed analysis of the results from a political science perspective is outside the scope of this work, we observe that our data show a general swing toward the Republican candidate before the election and a swing in the opposite direction after the election. Indeed, our indicators suggest a substantial movement towards the Republican candidate over the course of the campaign, with strong support in many of the swing states by the week of the election. In addition, our results suggest that the so-called honeymoon period of new presidents—where they typically enjoy high approval for the first few months of their presidency—was either totally missed or extremely short-lived in many states for the new president. The results suggest a number of avenues of future research for political science, which typically relies on polls that are limited in number and locales in these periods.

Performance. This case study also serves to demonstrate the viability of our methods for performing meaningful data analysis privately. An end-to-end training run including data for the week before and the week after the election takes approximately 70 minutes. Additionally, running inference for a single day's data takes between 1 and 5 minutes, depending on the number of users in the sample on that day. All analysis took place on moderately powerful `m5n.2xlarge` AWS instances with 8 vCPUs and 32GB of memory. We emphasize that we did not seek to maximize performance, but only to make all computation run in a reasonable time frame of several hours. If we were to process, say, an order of magnitude more data, the present performance may become impractical, and more aggressive tuning and optimization would be necessary.

4.3 Data Integrity

For our inference of aggregate political preference to be accurate, it is critical that we correctly partition panelists into the bags for each state. For this reason, we ask panelists to self-report their state. However, we found that this self-reporting is not always accurate; for instance, our first Mechanical Turk advertisement was only open to people who live in swing states, which may have created an incentive for people to misrepresent their location. Although subsequent advertisements rewarded participants equally regardless of their reported state, the potential for dishonest reporting remains a concern. Future deployments should be careful to avoid incentive structures that encourage dishonest reporting. To examine the integrity of self-reported location information, we use geolocation data that allows us to draw conclusions about the validity of the sample.

Privacy considerations. We exercised care to balance the data integrity goal with our overall privacy commitment. We design a validation process with the overarching objective of never having long-term storage of IP addresses or of location data beyond the ZIP code level. When the browser extension connects to the webserver to upload data, we use the connecting IP address and the IP locating service `ipgeolocation.io` to infer the panelist's state and ZIP code and compare to the self-reported data. By performing this check at the time of upload, we do not need to store IP addresses.

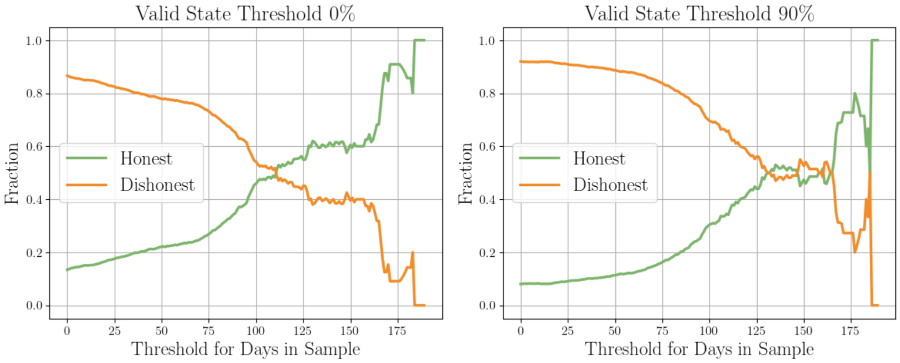

Fig. 3. Fractions of users who are honest and dishonest about their state of residence when filtering by a minimum number of days that the user is in the sample. The left and right plots consider a user honest if they are in the correct state for more than 0% or 90% of their total days in the sample, respectively.

Findings. Our primary takeaway is that while there are significant reasons to be concerned about the honesty of the users, the concerns ultimately are unlikely to substantially harm the ability to extract meaningful patterns from the data.

Approximately 98% of the uploads come from users located in the United States, indicating that the vast majority of web browsing data corresponds to our target demographic at the highest level. However, the state-level integrity is much lower. Only 15% of uploads come from users located in their self-reported state. Furthermore, 15% of uploads contain a self-reported ZIP code that is not a valid ZIP code for the self-reported state. Taken together, the state-level integrity and ZIP code validity data paint a picture of a noisy dataset.

However, perhaps surprisingly, the honest panelists provide far more valuable data. To justify the data sample, we show that not all panelists and uploads are equal in terms of our ability to learn from them.

First, Figure 3 shows that as we select for panelists who were in the sample for an increasingly large threshold of days, the panelists become more honest. We consider two ways of categorizing panelists into honest and dishonest groups: either we consider any panelist who has *ever* had their inferred state match their reported state to be honest, or we consider panelists to be honest if their inferred state matched their reported state for more than 90% of their uploads. In both cases, we see that the sample contains a large number of short-lived dishonest panelists, while consistent panelists tend to be more honest.

More importantly, Figure 4 shows that honest panelists contribute visit and referral histograms that are significantly more dense than their dishonest counterparts. Using the same method of splitting panelists into honest and dishonest groups, we see that in either case, honest panelists and dishonest panelists contribute similar numbers of total visits, while honest panelists far outpace dishonest panelists in their total referrals. This observation is important due to the

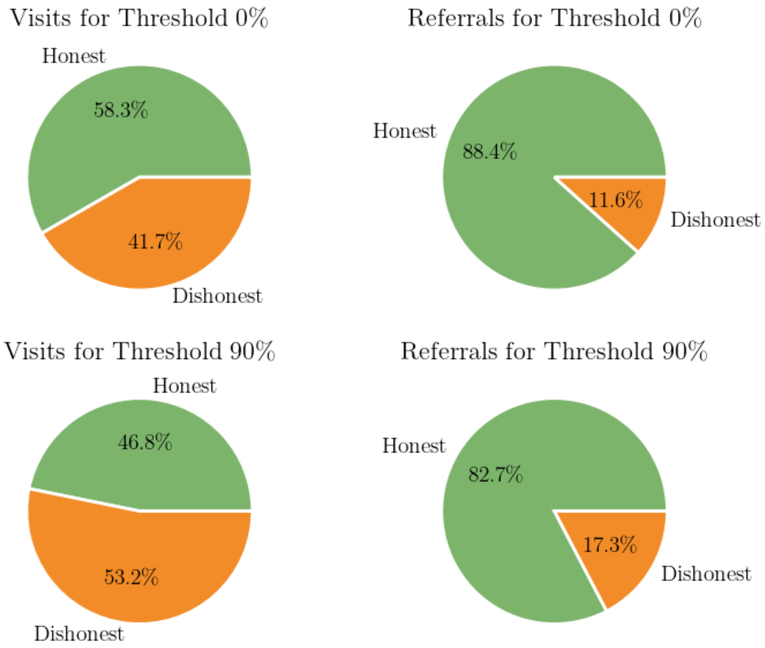

Fig. 4. The fractions of total visits and referrals reported by honest and dishonest users. The upper and lower plots consider a user honest if they are in the correct state for more than 0% or 90% of their total days in the sample, respectively.

finding by Comarela et al. [18] that referral data provides a far better signal for political preference than visits alone do.

We can draw two conclusions from the data and discussion above. First, detecting lying users and, if possible, obtaining some kind of proof or verification that a user is honest is of critical importance for ensuring data integrity, and this is an important direction for future work, as we discuss in Section 6. Second, while we should aim to maximize the integrity of the data, the fact that the heaviest users tend to be honest means our learning process is surprisingly robust to dishonest users.

5 Related Work

This work presents a deployment of privacy-preserving machine learning (PPML). In this section, we describe and compare with a few different categories of related work into PPML.

Algorithmic designs. Building upon a long line of research into MPC in general, there have been significant advances in the past decade in the design of privacy-preserving machine learning algorithms. Some initial work, including SecureML [50], ABY^3 [49], and SecureNN [66] built broad support for machine learning operations under MPC. Next, several works contributed new protocols for privacy-preserving ML training and inference that collectively offer a variety of options for the number of parties, adversarial threshold, active vs. passive threat model, and more [15, 17, 19, 20, 37, 41, 42, 55, 56, 67]. A more recent line of work has explored privacy-preserving algorithms for deep learning operators like transformers [3, 23, 30, 31, 45, 48, 53, 72]. Our work only requires a relatively simple machine learning algorithm based on gradient descent (see Algorithm 1), which can be built even from the earlier line of PPML algorithms.

Software and hardware frameworks. There are several software frameworks for privacy-preserving machine learning. ABY^3 [49] provides protocols for many specific fixed-point and vector operations used in machine learning. EzPC [16, 43, 59] provides a compiler from high-level machine learning operators to low-level crypto protocols. MP-SPDZ [39] is an extensive repository that contains several machine learning-friendly MPC protocols on its own right, and it is also used within the cloud-native Carbyne Stack framework [8].

Additionally, there are several frameworks that take advantage of dedicated hardware. Some works leverage dedicated GPUs [27, 33, 35, 36, 64, 69] or FPGAs [25, 34, 54] to improve performance. Other works combine MPC with trusted execution enclaves in order to provide defense-in-depth [21, 46, 70].

We use CrypTen [40] in this work due to its ease of use in developing Algorithm 1. That said, there are a number of frameworks listed above that we could have used instead, several of which also support the three-party setting that we use in this work.

Deployments of MPC. To the best of our knowledge, this work is the first use of MPC in the domain of political science. More generally, it contributes toward the ongoing efforts to increase adoption of MPC in scientific research settings to derive publicly useful findings from private data.

We provide several examples of prior work on MPC for public benefit. First, national statistics organizations have used MPC to conduct surveys and to aggregate data across agencies, as documented in a list compiled by the United Nations global working group on big data [65]. Second, policymakers have used MPC to gather statistical evidence for data-driven policymaking [2] in domains like education [12] and health services [32]. Third, healthcare organizations have used MPC to measure the aggregate spread of COVID-19 [5] and to conduct clinical research [60]. Finally, civic non-profit groups have used MPC to measure workplace culture [38] and empower survivors of sexual assault on college campuses [58].

6 Conclusion and Future Work

In this work, we have built and deployed a system for private analysis of political sentiment from web browsing data. We use the remainder of the paper to discuss some areas for possible improvement and future work based on lessons learned from the deployment.

6.1 Balancing Integrity Verification with Privacy

As with any data system, the quality of the data is of critical importance. Section 4.3 details our experience with measuring the honesty of users in reporting their residence, an important variable in our ability to make accurate state-level predictions. Our approach involved using the user's IP address (which we will inevitably know by operating a standard webserver) to infer their current location. We could consider stronger methods of identity verification, but any strengthening would certainly lead to more invasive data collection, seemingly contradicting our core privacy goal.

In addition to the privacy element of the data integrity question, there is the fact that it is not immediately clear what information could be used to prove the location or residence of a user, even in a plaintext setting. Many companies and organizations use know-your-customer (KYC) systems for knowing the residence of their users. Perhaps one could design a private KYC service. We leave the problem of private residence verification as a challenging open problem.

6.2 Strengthening the Threat Model

The other major direction for future work is broadly to strengthen the threat model. To begin, the most immediate concern with the current system is the use of AWS as a single point of failure; this is an unfortunate but immediate consequence of our out-of-the-box use of CrypTen. While CrypTen provides a simple interface for running multiparty computation, future work could seek to build a similarly friendly interface that simultaneously can work across multiple cloud providers and multiple trust domains.

Another possible avenue towards a stronger threat model is to minimize the trust required from the user in the core machine learning computation. We can explore different system architectures and/or cryptographic primitives for the core computation. In the case of inference, the problem is comparatively easy, as each user can be treated independently, except for the aggregation of results. Given a model (either public or private), the user and a server could partake in a two-party computation to reveal only the prediction of the model on the user's private input. Such a two-party computation would be lightweight enough to be run every day to obtain live-updated polling results. For training, the problem is more difficult, as individual users' data can no longer be treated independently. One potential solution is to strengthen the MPC by incorporating multiple independent organizations to distribute the trust more widely. Alternatively, another option is to employ fully homomorphic encryption (FHE), involving limited client-side work to perform a threshold decryption to obtain the final results.

A final possible method for strengthening the threat model is the incorporation of anonymous payments. The payment method employed in this work requires knowledge of which Mechanical Turk IDs were active in the sample and for how many days. An anonymous payment mechanism would add an additional layer of protection for users beyond protecting their browsing data.

Acknowledgments

This research was supported by an Impact Program Grant from the Weidenbaum Center on the Economy, Government and Public Policy at Washington University; by the National Science Foundation under grants CNS-2209194, CNS-2312711, and CNS-2319369; by the DARPA SIEVE program under Agreement No. HR00112020021; and by a gift from Robert Bosch GmbH. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or DARPA.

References

1. Aysajan Abidin, Abdelrahaman Aly, Sara Cleemput, and Mustafa A. Mustafa. An MPC-based privacy-preserving protocol for a local electricity trading market. In Sara Foresti and Giuseppe Persiano, editors, *CANS 16*, volume 10052 of *LNCS*, pages 615–625. Springer, Cham, November 2016.
2. Advisory Committee on Data for Evidence Building. Year 2 report. https://www.bea.gov/sites/default/files/2022-10/acdeb-year-2-report.pdf, 2022.
3. Yoshimasa Akimoto, Kazuto Fukuchi, Youhei Akimoto, and Jun Sakuma. Privformer: Privacy-preserving transformer with MPC. In *2023 IEEE European Symposium on Security and Privacy*, pages 392–410. IEEE Computer Society Press, July 2023.
4. Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In Elisa Bertino, Ravi Sandhu, and Alexander Pretschner, editors, *Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016*, pages 142–144. ACM, 2016.
5. Apple and Google. Exposure notification privacy-preserving analytics (ENPA) white paper. https://covid19-static.cdn-apple.com/-applications/covid19/current/static/contact-tracing/pdf/ENPA_White_Paper.pdf, 2021.
6. David W. Archer, Dan Bogdanov, Y. Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P. Smart, and Rebecca N. Wright. From keys to databases – real-world applications of secure multi-party computation. Cryptology ePrint Archive, Report 2018/450, 2018.
7. K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), page 307–314, New York, NY, USA, 1968. Association for Computing Machinery.
8. Sebastian Becker, Paul Duplys, Johannes Graf, Kalman Graffi, Alessandro Grassi, David Greven, Julian Grewe, Shalabh Jain, Timo Klenk, Nikolay Matyunin, Hanna Modica, Vadim Raskin, Petra Scherer, Volker Suschke, Sven Trieflinger, Veselin Vlasakiev, and Jared Weinfurtner. Carbyne Stack. https://github.com/carbynestack/carbynestack, 2021.

9. Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 296–312, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
10. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.
11. Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 227–234. Springer, Berlin, Heidelberg, January 2015.
12. Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and taxes: a privacy-preserving study using secure computation. *PoPETs*, 2016(3):117–135, July 2016.
13. Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *ESORICS 2008*, volume 5283 of *LNCS*, pages 192–206. Springer, Berlin, Heidelberg, October 2008.
14. Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 325–343. Springer, Berlin, Heidelberg, February 2009.
15. Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. FLASH: Fast and robust framework for privacy-preserving machine learning. *PoPETs*, 2020(2):459–480, April 2020.
16. Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. EzPC: Programmable and efficient secure two-party computation for machine learning. In *2019 IEEE European Symposium on Security and Privacy*, pages 496–511. IEEE Computer Society Press, June 2019.
17. Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. Trident: Efficient 4PC framework for privacy preserving machine learning. In *NDSS 2020*. The Internet Society, February 2020.
18. Giovanni Comarela, Ramakrishnan Durairajan, Paul Barford, Dino Christenson, and Mark Crovella. Assessing candidate preference through web browsing history. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 158–167, 2018.
19. Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. Secure evaluation of quantized neural networks. *PoPETs*, 2020(4):355–375, October 2020.
20. Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 2183–2200. USENIX Association, August 2021.
21. Romain de Laage, Peterson Yuhala, François-Xavier Wicht, Pascal Felber, Christian Cachin, and Valerio Schiavoni. Practical secure aggregation by combining cryptography and trusted execution environments. *CoRR*, abs/2504.08325, 2025.
22. Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *NDSS 2015*. The Internet Society, February 2015.

23. Ye Dong, Wen jie Lu, Yancheng Zheng, Haoqi Wu, Derun Zhao, Jin Tan, Zhicong Huang, Cheng Hong, Tao Wei, and Wenguang Chen. Puma: Secure inference of llama-7b in five minutes, 2023.
24. Khaled El Emam, Jun Hu, Jay Mercer, Liam Peyton, Murat Kantarcioglu, Bradley Malin, David Buckeridge, Saeed Samet, and Craig 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.
25. Xin Fang, Stratis Ioannidis, and Miriam Leeser. Secure function evaluation using an FPGA overlay architecture. In *FPGA*, pages 257–266. ACM, 2017.
26. Gabriel Franco, Mark Crovella, and Giovanni Comarela. Dependence and model selection in LLP: The problem of variants. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 470–481, 2023.
27. Tore Kasper Frederiksen, Thomas P. Jakobsen, and Jesper Buus Nielsen. Faster maliciously secure two-party computation using the GPU. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 358–379. Springer, Cham, September 2014.
28. Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *PoPETs*, 2017(4):345–364, October 2017.
29. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
30. Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. SIGMA: Secure GPT inference with function secret sharing. *PoPETs*, 2024(4):61–79, October 2024.
31. Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. In *NeurIPS*, 2022.
32. Nicholas R. Hart, David W. Archer, and Erin Dalton. Privacy-preserved data sharing for evidence-based policy decisions: A demonstration project using human services administrative records for evidence-building activities. https://bipartisanpolicy.org/download/?file=/wp-content/uploads/2019/06/Privacy-Preserved-Data-Sharing-for-Evidence-Based-Policy-Decisions.pdf, 2019.
33. Christopher Harth-Kitzerow, Yongqin Wang, Rachit Rajat, Georg Carle, and Murali Annavaram. PIGEON: A high throughput framework for private inference of neural networks using secure multiparty computation. Cryptology ePrint Archive, Paper 2024/1371, 2024.
34. Kai Huang, Mehmet Güngör, Xin Fang, Stratis Ioannidis, and Miriam Leeser. Garbled circuits in the cloud using FPGA enabled nodes. In *HPEC*, pages 1–6. IEEE, 2019.
35. Nathaniel Husted, Steven A. Myers, Abhi Shelat, and Paul Grubbs. GPU and CPU parallelization of honest-but-curious secure two-party computation. In *ACSAC*, pages 169–178. ACM, 2013.
36. Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. Orca: FSS-based secure training and inference with GPUs. In *2024 IEEE Symposium on Security and Privacy*, pages 597–616. IEEE Computer Society Press, May 2024.
37. Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In William Enck and

Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 1651–1669. USENIX Association, August 2018.

38. Gabriel Kaptchuk, Jen Benoit-Bryan, Kinan Dak Albab, Mia Locks, and Mayank Varia. The good, the bad, and the ugly — lessons from an mpc for social good deployment. In *Real World Crypto Symposium*, 2024.
39. Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1575–1590. ACM Press, November 2020.
40. B. Knott, S. Venkataraman, A.Y. Hannun, S. Sengupta, M. Ibrahim, and L.J.P. van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning*, 2020.
41. Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. SWIFT: Super-fast and robust privacy-preserving machine learning. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 2651–2668. USENIX Association, August 2021.
42. Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. Tetrad: Actively secure 4PC for secure training and inference. In *NDSS 2022*. The Internet Society, April 2022.
43. Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow: Secure TensorFlow inference. In *2020 IEEE Symposium on Security and Privacy*, pages 336–353. IEEE Computer Society Press, May 2020.
44. Andrei Lapets, Frederick Jansen, Kinan Dak Albab, Rawane Issa, Lucy Qin, Mayank Varia, and Azer Bestavros. Accessible privacy-preserving web-based data analysis for assessing and addressing economic inequalities. In *COMPASS*, pages 48:1–48:5. ACM, 2018.
45. Dacheng Li, Hongyi Wang, Rulin Shao, Han Guo, Eric Xing, and Hao Zhang. Mpcformer: fast, performant and private transformer inference with mpc. In *The Eleventh International Conference on Learning Representations*, 2023.
46. Xiaoguo Li, Bowen Zhao, Guomin Yang, Tao Xiang, Jian Weng, and Robert H. Deng. A survey of secure computation using trusted execution environments. *CoRR*, abs/2302.12150, 2023.
47. Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. ObliVM: A programming framework for secure computation. In *2015 IEEE Symposium on Security and Privacy*, pages 359–376. IEEE Computer Society Press, May 2015.
48. Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and Wenguang Chen. BumbleBee: Secure Two-party Inference Framework for Large Transformers. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025*. The Internet Society, February 23-28, 2025.
49. Payman Mohassel and Peter Rindal. ABY^3: A mixed protocol framework for machine learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 35–52. ACM Press, October 2018.
50. Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy*, pages 19–38. IEEE Computer Society Press, May 2017.
51. Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. GraphSC: Parallel secure computation made easy. In *2015 IEEE Symposium on Security and Privacy*, pages 377–394. IEEE Computer Society Press, May 2015.

52. Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE Computer Society Press, May 2013.
53. Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. BOLT: Privacy-preserving, accurate and efficient inference for transformers. In *2024 IEEE Symposium on Security and Privacy*, pages 4753–4771. IEEE Computer Society Press, May 2024.
54. Rushi Patel, Pierre-Francois Wolfe, Robert Munafo, Mayank Varia, and Martin C. Herbordt. Arithmetic and boolean secret sharing MPC on fpgas in the data center. In *HPEC*, pages 1–8. IEEE, 2020.
55. Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: Improved mixed-protocol secure two-party computation. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 2165–2182. USENIX Association, August 2021.
56. Arpita Patra and Ajith Suresh. BLAZE: Blazing fast privacy-preserving machine learning. In *NDSS 2020*. The Internet Society, February 2020.
57. Novi Quadrianto, Alex J. Smola, Tiberio S. Caetano, and Quoc V. Le. Estimating Labels from Label Proportions. In *International Conference on Machine Learning*, pages 776–783, 2008.
58. Anjana Rajan, Lucy Qin, David W. Archer, Dan Boneh, Tancrède Lepoint, and Mayank Varia. Callisto: A cryptographic approach to detecting serial perpetrators of sexual misconduct. In *COMPASS*, pages 49:1–49:4. ACM, 2018.
59. Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. CrypTFlow2: Practical 2-party secure inference. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 325–342. ACM Press, November 2020.
60. Jennie Rogers, Elizabeth Adetoro, Johes Bater, Talia Canter, Dong Fu, Andrew Hamilton, Amro Hassan, Ashley Martinez, Erick Michalski, Vesna Mitrovic, Fred Rachman, Raj Shah, Matt Sterling, Kyra VanDoren, Theresa L. Walunas, Xiao Wang, and Abel Kho. VaultDB: A real-world pilot of secure multi-party computation within a clinical research network. https://arxiv.org/pdf/2203.00146.
61. Semrush. Top 100: The most visited websites in the us. https://www.semrush.com/blog/most-visited-websites/.
62. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.
63. Ebrahim M. Songhori, Siam U. Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. TinyGarble: Highly compressed and scalable sequential garbled circuits. In *2015 IEEE Symposium on Security and Privacy*, pages 411–428. IEEE Computer Society Press, May 2015.
64. Sijun Tan, Brian Knott, Yuan Tian, and David J. Wu. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *2021 IEEE Symposium on Security and Privacy*, pages 1021–1038. IEEE Computer Society Press, May 2021.
65. UN Global Working Group Task Team on Privacy Preserving Techniques. Case study repository. https://unstats.un.org/wiki/spaces/UGTTOPPT/pages/150012018/Case+study+repository, 2024.
66. Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *PoPETs*, 2019(3):26–49, July 2019.
67. Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *PoPETs*, 2021(1):188–208, January 2021.

68. Julissa Milligan Walsh, Mayank Varia, Aloni Cohen, Andrew Sellars, and Azer Bestavros. Multi-regulation computing: Examining the legal and policy questions that arise from secure multiparty computation. In *CSLAW*, pages 53–65. ACM, 2022.
69. Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. Piranha: A GPU platform for secure computation. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 827–844. USENIX Association, August 2022.
70. Pengfei Wu, Jianting Ning, Jiamin Shen, Hongbing Wang, and Ee-Chien Chang. Hybrid trust multi-party computation with trusted execution environment. In *NDSS 2022*. The Internet Society, April 2022.
71. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.
72. Chenkai Zeng, Debiao He, Qi Feng, Xiaolin Yang, and Qingcai Luo. Securegpt: A framework for multi-party privacy-preserving transformer inference in gpt. *IEEE Transactions on Information Forensics and Security*, 19:9480–9493, 2024.