Microsoft

Masked LARk: The Masked Learning, Aggregation and Reporting worKflow

Joseph Pfeiffer, joelpf@microsoft.com
Denis Charles: cdx@microsoft.com
Mehul Parsana*: mparsana@microsoft.com
Erik Anderson: erikan@microsoft.com
Young Hun Jung: youjung@microsoft.com
Davis Gilton: davisgilton@microsoft.com

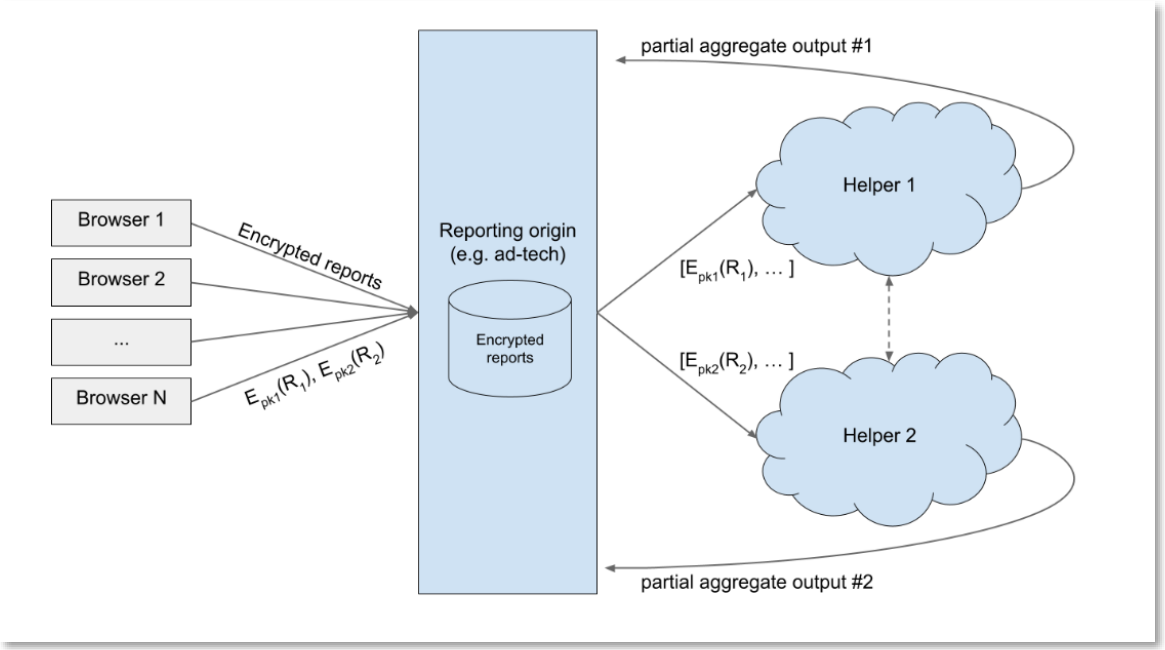
*Now at Google

Agenda

- Related Work:
  Google (Browser) Aggregate Conversion MultiParty Compute Workflow
- Masked LARk: Masked Learning Aggregation and Reporting workflow
  - Workflow Overview
  - Gradient Computation
- Experiments
- Conclusions

Related Work:
Google Proposal for Conversion Reporting

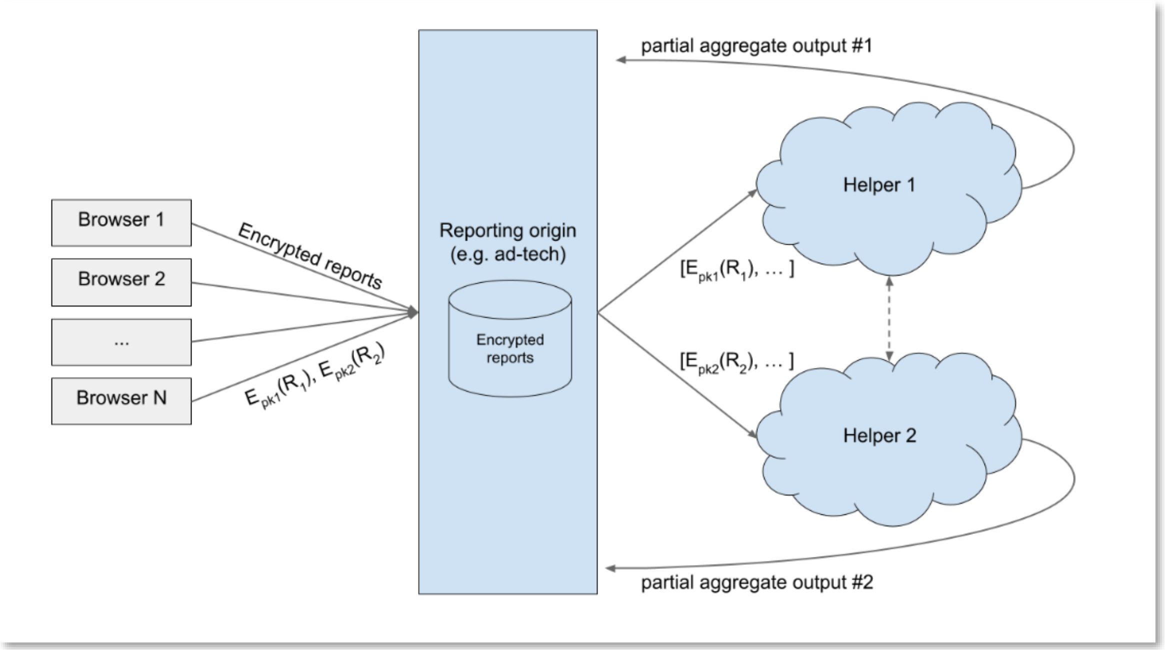
- Recent Google Proposal discusses utilizing secure MultiParty Compute (MPC) for Trusted Mediator*
- MPC performs both aggregation and enforces differential privacy constraints
- Upon leaving browser, no individual entity has a complete picture of an individual record
    - Secret Sharing



*Builds on previous work – Prio by Mozilla.

Related Work: Google Proposal for Conversion Reporting

- Pros:
  - Segregated helpers implementing aggregation more palatable than single trusted mediator
- Limitations:
  - Handles aggregation for some reporting needs but does not address modeling



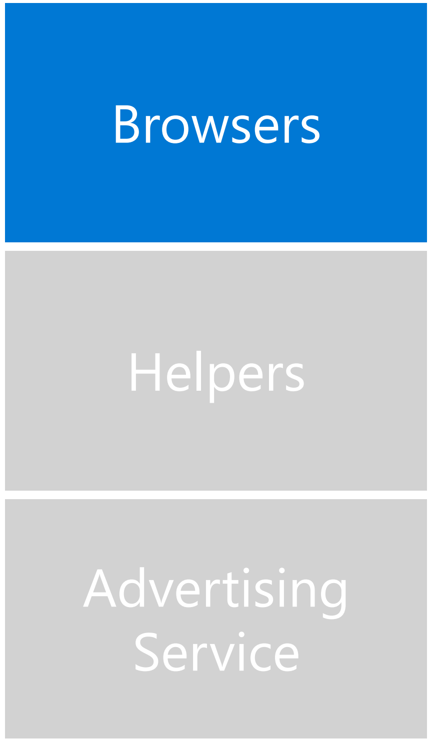
Masked LARk: Masked Learning Aggregation and Reporting workflow

Masked LARk

- **Goal**: Expand the aggregation service towards an abstracted differentially private Map-Reduce framework
  - Browsers implement a secure "Map" function
  - Helpers implement a differentially private "Reduce" operation
  - Advertising Servers are the consumers and data storage
- **Goal**: Expand from single aggregation function, to a platform that enables a variety of differentially private functions (e.g., Aggregation, Model Training)
- Provided
  - Explainer: https://github.com/WICG/privacy-preserving-ads/blob/main/MaskedLARK.md
  - Paper: https://arxiv.org/abs/2110.14794
  - Prototype: https://github.com/microsoft/MaskedLARk

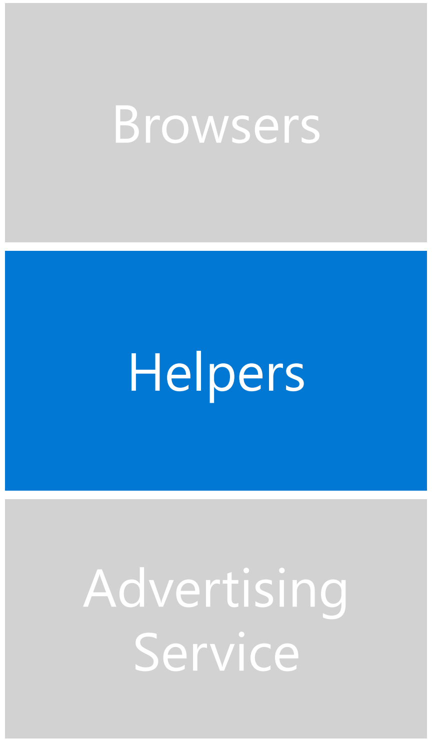
Masked LARk - Browsers

- Browsers represent the user interest
  - Initial viewing / clicking of advertisement
  - The conversion
- Only party with full information about individual users
- Responsibilities
  - Attributing a conversion to a view or click
  - Inserting fake records and masks for true or fake records
  - In certain scenarios, inserting local differential privacy
  - Secret sharing of values for helpers to aggregate / train on
  - Sending reports to the advertising server, which are encrypted so only the helper can read
  - Choosing which helpers to use



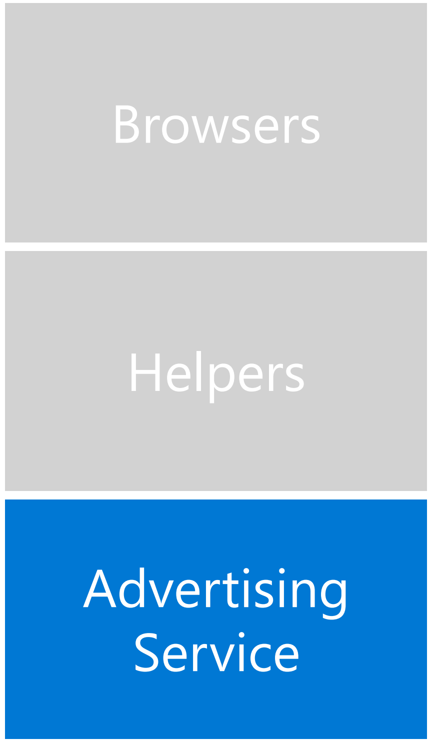
Masked LARk - Helpers

- Receive encrypted reports from advertising servers
- Helpers perform core function implementation, doing most of the heavy lifting within the platform
  - E.g., aggregation or model training
- Enforce community-approved privacy constraints
  - *K-Anonymity:* require a certain minimum number of records to return any results
  - *Global differential privacy* added to outputs of functions
  - *Manage privacy budget*



Masked LARk – Advertising Server

- At impression time, passes relevant features to Browser for later processing
- Acts as a data storage unit
  - Records received from Browsers are encrypted with keys held by the helpers
  - Advertising Service holds these for later processing, bearing this cost
- Can only retrieve aggregation information utilizing the associated helpers
- Applies aggregated information to future tasks
  - Reporting conversion data to advertisers
  - Model training and later inference



Implemented Helper Functions: Gradient Computation

Model Training

- Want to allow Masked LARk to handle complex scenarios, specifically model training, where the features are observed yet the labels are secrets
- Core issue:
  - Most differentiable models (e.g., neural networks) optimize some variation of SGD

$$\theta_{j+1} = \theta_j - \eta \sum_i \nabla Loss(features = x_i, label = y_i, Model = \theta_j)$$

  - $\nabla L(x_i, y_i, \theta_j)$ is computed per sample, requiring the features and labels together
    - As is, this would reveal features and labels to the helper
  - Will show, can utilize MPC with masking to compute the gradient step

Model Training – Key Idea

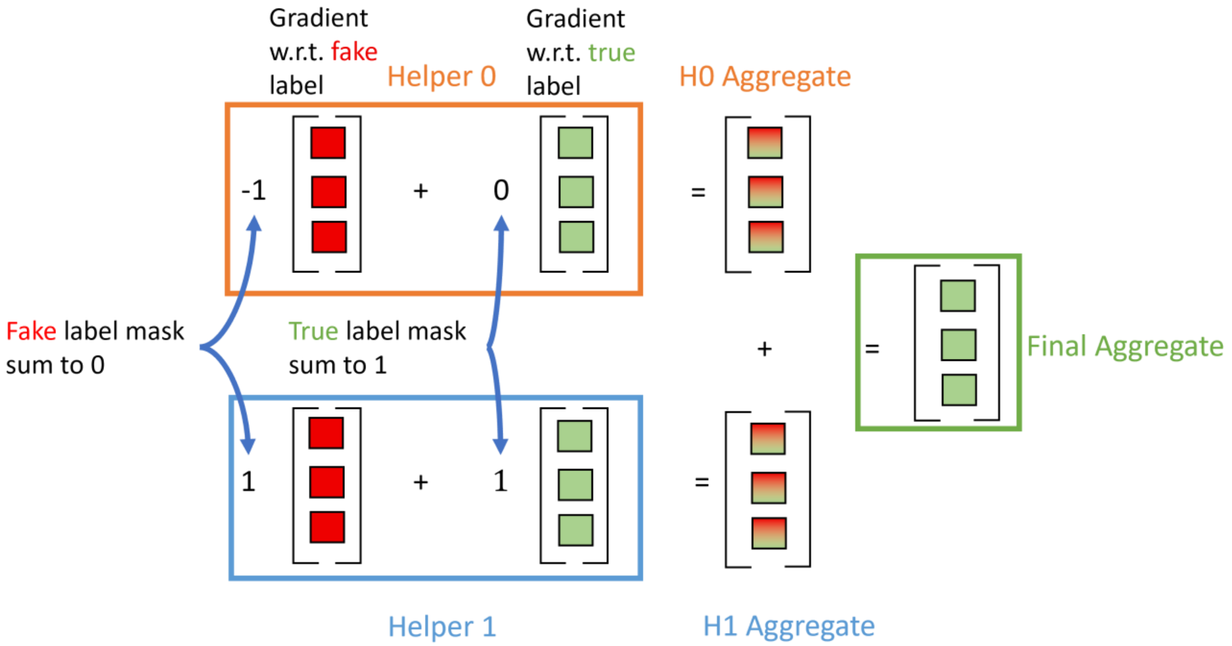
- Browsers will generate many possible labels / features, some which may be true but will contain some that are "fake" (i.e., not the correct label / feature)
- Both helpers will compute gradients for both the true / fake examples
- Browser will additionally send masks α to the helpers to include in the summation

$$\sum_i \alpha_i \nabla Loss(x_i, y_i, \theta_j)$$

- α are constructed in such a way that fake examples are removed from the final MPC summation

Example: Single Sample Mask (Proof Sketch)

- Assume
  - One Sample, Two Helpers
  - One True Label (e.g., conversion occurred) and One Fake Label (e.g., no conversion occurred)
- Each helper computes gradient for both True and Fake Labels
- Each Helper multiplies each gradient by the matching mask
- Each Helper sums the vectors
- Advertising server sums the partials returned by the Helpers



More general formulation

- Let $X \subset \mathcal{X}$, $\mathrm{Y} \subset \mathcal{Y}$, $\theta \in \Theta$ indicate features, labels and a model
- Let $f: \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}^d$ be a mapping from features, labels and models to some vector space
  - Focus on gradient computation for f, but other functions could potentially apply
- Let g indicate a bi-linear aggregation function over a set of samples after applying f, e.g.
  - $g(\langle f(x_1, y_1, \theta), \dots, f(x_n, y_n, \theta)\rangle) \coloneqq \sum_{i \in n} f(x_i, y_i, \theta)$
  - g satisfies bi-linearity, i.e., for $\alpha_i, \beta_i \in \mathbb{R}$:
    - $g(\langle(\alpha_i + \beta_i) f(x_i, y_i, \theta)\rangle) = g(\langle \alpha_i f(x_i, y_i, \theta), \beta_i f(x_i, y_i, \theta)\rangle)$
  - Focus on summation for g, but other functions could potentially apply

Summation over Masked Gradients equals summation over true gradients

- Section 4.2, Corollary 4.1.1
  - For each helper h
    - $\Psi^h := \sum_i \alpha_i^h \nabla Loss(x_i, y_i, \theta_j)$
  - $\sum_i \nabla Loss(x_i, y_i, \theta_j) = \sum_h \Psi^h$

- Generalizes to arbitrary mappings f and all bilinear functions g

Fake Labels and Masks

- Let $\overline{y_i} \in \mathcal{Y} \backslash \{y_i\}$ be a fake label additionally sent from the browser
  - Cannot be y_i
  - Both fake and real can be sent from the browser, but the order is randomized
    - The corresponding masks (to be defined) must match the randomized order
  - For real-valued labels, the values are quantized
  - For non-integer values, the values are randomly rounded
- Without loss of generality, define masks for the real/fake labels for two helpers:
  - $\alpha_i^0 + \alpha_i^1 = 1$ [True label]
  - $\overline{\alpha_i^0} + \overline{\alpha_i^1} = 0$ [Fake label]
  - $\alpha_i^0, \overline{\alpha_i^0}$ are both random variables on a finite ring, solve for $\alpha_i^1, \overline{\alpha_i^1}$ as appropriate

Threat Models - Attacks

- High cardinality label space (e.g., floats) – ad network could send ID as label (lower order bits, etc), then collude with helper to recover features and label
  - Randomly Quantized labels to a desired cardinality
  - In expectation, minimizes same loss
- Poisoning by the Browser (sending invalid label masks)
  - Some overhead for a validity function (e.g., SNIPs)
- Requirement of default (0) label when expired
  - Randomized ending times
- One Helper colludes w/ advertising service
  - Features exposed: In some situations, features could be considered sensitive
  - Add random noise to feature vector (local DP)
  - Extra cost: Can secret share $x = x_0 + x_1$, as $W \cdot x = W \cdot x_0 + W \cdot x_1$

Efficiency-Privacy Tradeoffs

Efficiency/Utility

Privacy

Event Conversion API

Plausible Deniability

Limited Information

Masked Event Conversion API

Rather than limit bits, return multiple possible (quantized) labels.

Keys / features derived only from first party information

Also provide secret shared masks for both true and false labels.

Requires minimal MPC, can do various partitioning / slicing / any model training as needed

Heavy computation (gradient) can be performed on ad side

Obvious timing attacks

Masked LARk

Advertising Server cannot recover features w/o collusion

Helpers cannot recover labels w/o collusion with each other

Fairly efficient (communication)

Insert local DP

Buckets/iDPF

Secret Share Feature Vector

Aggregation scope

SS Masked LARk

Secret Share Feature Vector

Much more compute

Does not necessarily extend to all bilinear functions

Experiments

Additional Privacy Requirements and Implementation Details

- Tested Local Differential Privacy for the feature vector
- Malicious advertising server and helper could reverse engineer label from a gradient and a feature vector (4.2.2)
  - k-anonymity
  - Global differential privacy (Gradient Clipping & Laplace noise)
- Feature vector assumed small and dense (e.g., 100 dimensional vector)
  - Feature vector assumed bytes to save space

Code: https://github.com/microsoft/MaskedLARk

Implementation: PyTorch-friendly Training Interface

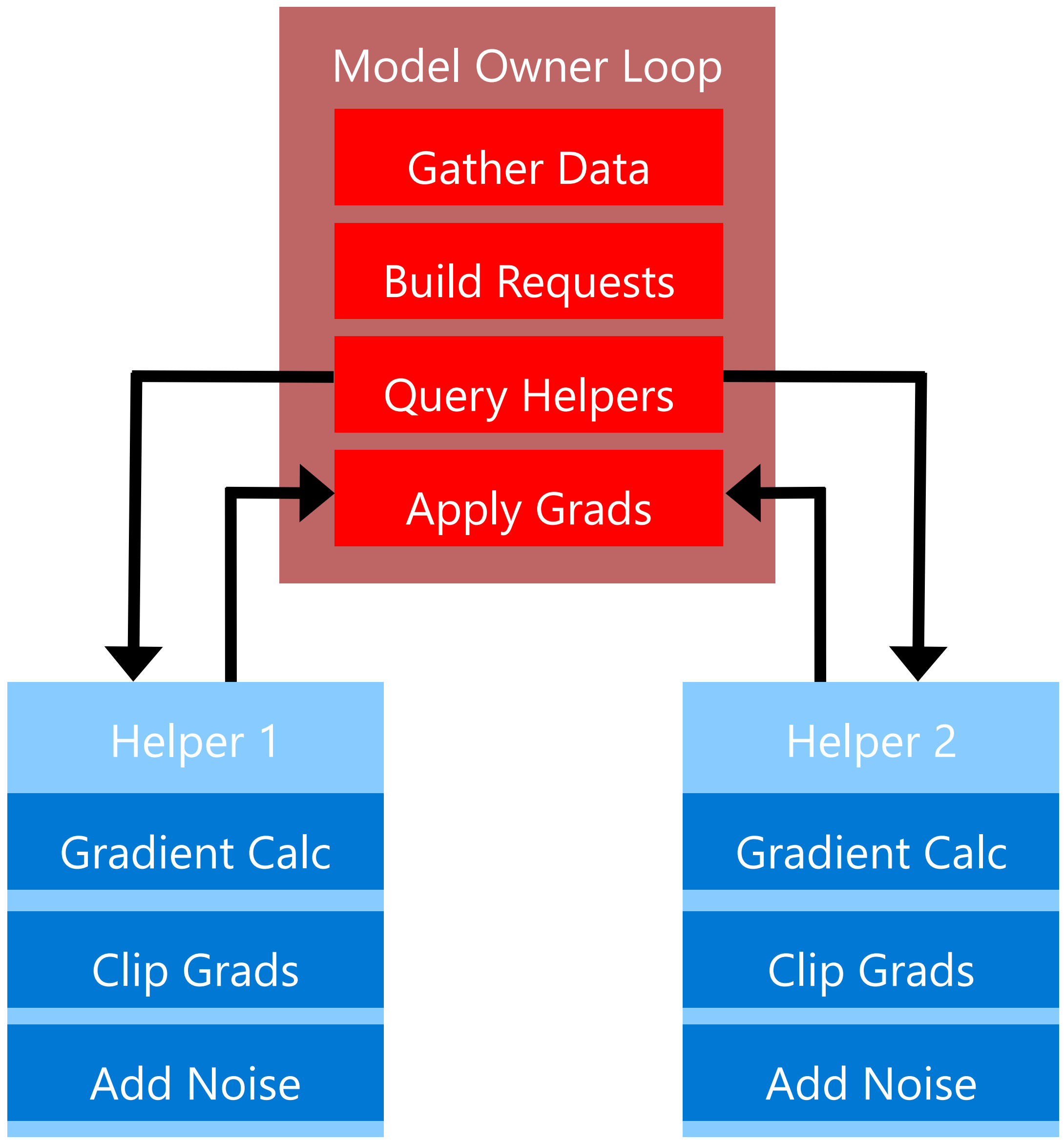
```
import MaskedLark as mlark

# Acts as an interface to Azure-based helper services
helper = mlark.Helper(helper_config)
for ii, sample_batch in enumerate(dataloader):
    helper.fetch_gradients(model)
    helper.backward()
```

Code: https://github.com/microsoft/MaskedLARk

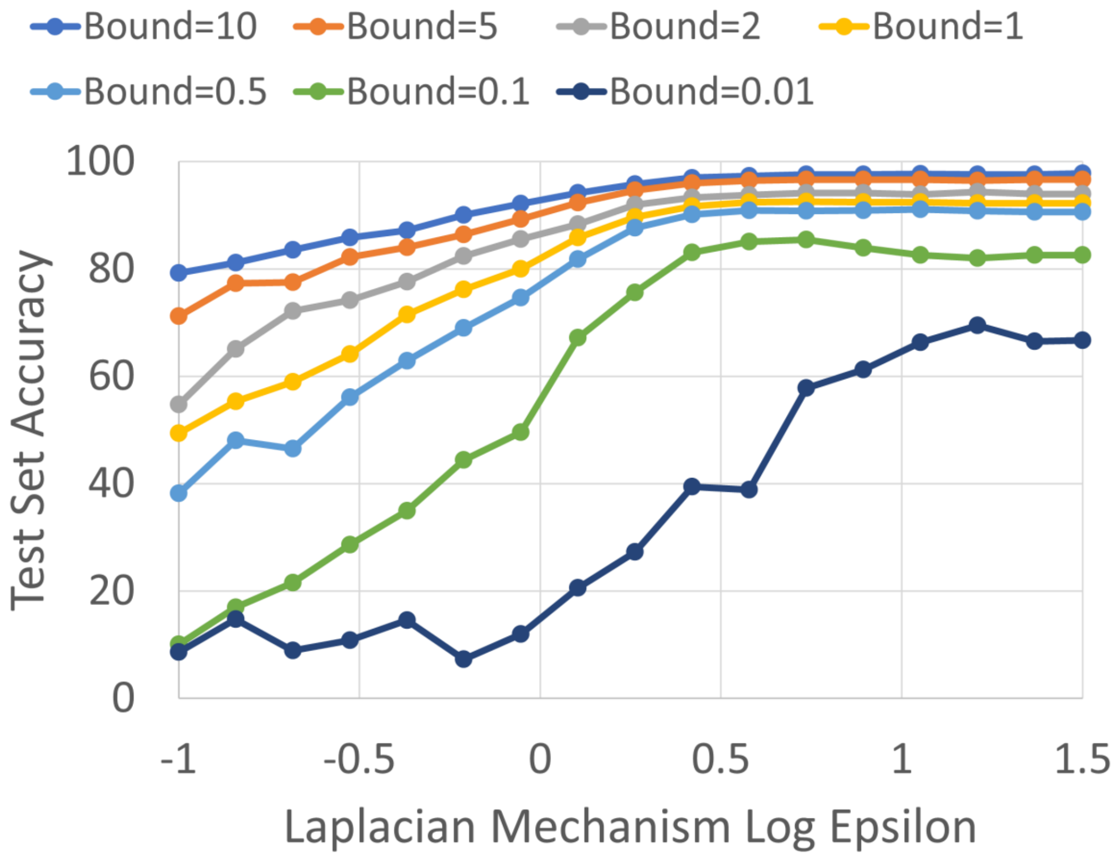
Experiment Setup

- Test scenario: model training
  - Helpers are simulated with Azure Functions and called via our API
  - Helpers use PyTorch's autograd to calculate gradients
- Models are passed as serialized ONNX models.
- We test on MNIST (784 dimensions) and WBCD (30 dimensions)
  - MNIST model: one hidden layer with 500 nodes. WBCD model: two hidden layers with 50 nodes.

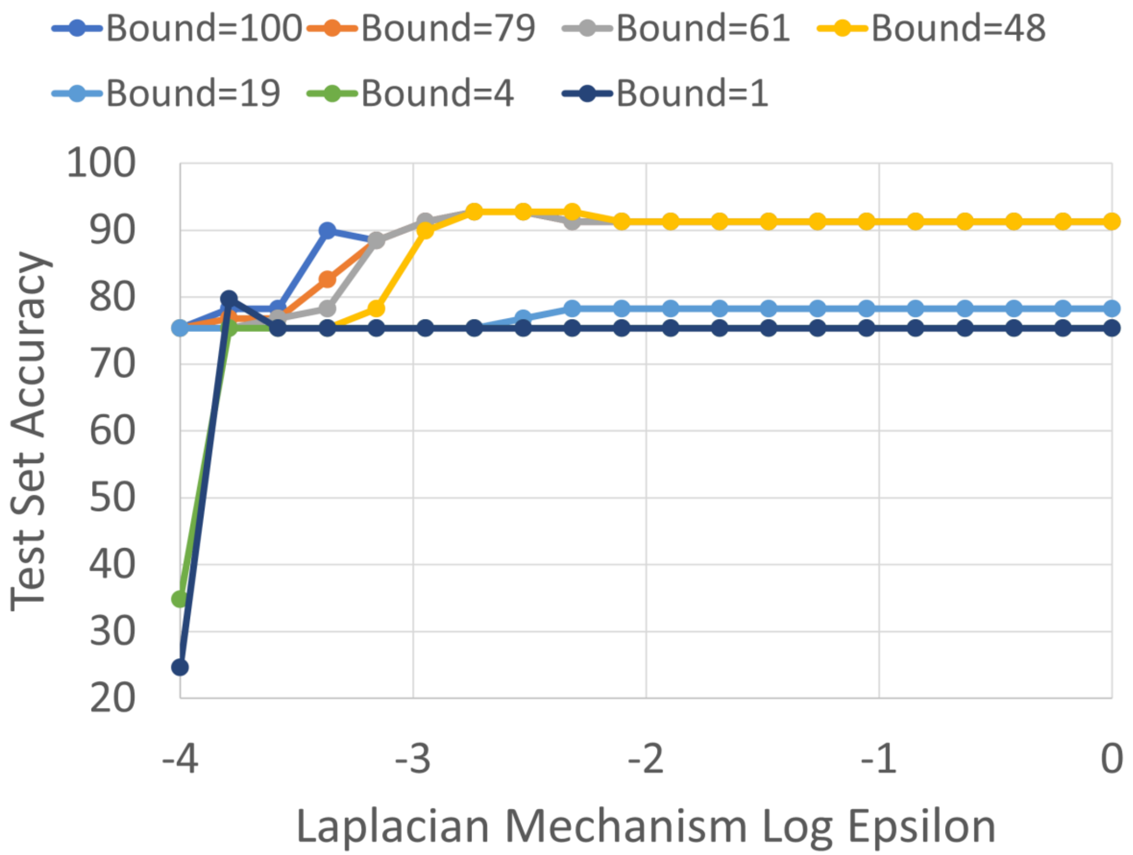


Lesson: Privacy Mechanisms Affect Model Performance

Aggressive gradient clipping hurts performance, but networks are resilient to small amounts of noise.



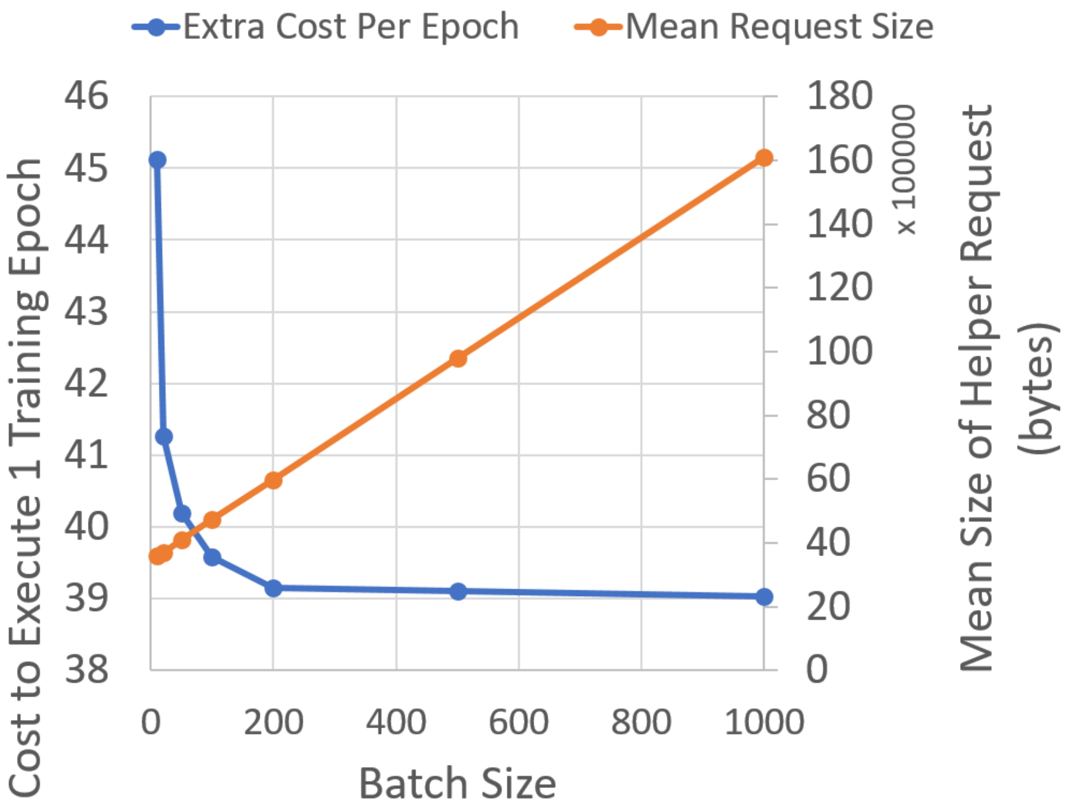
MNIST



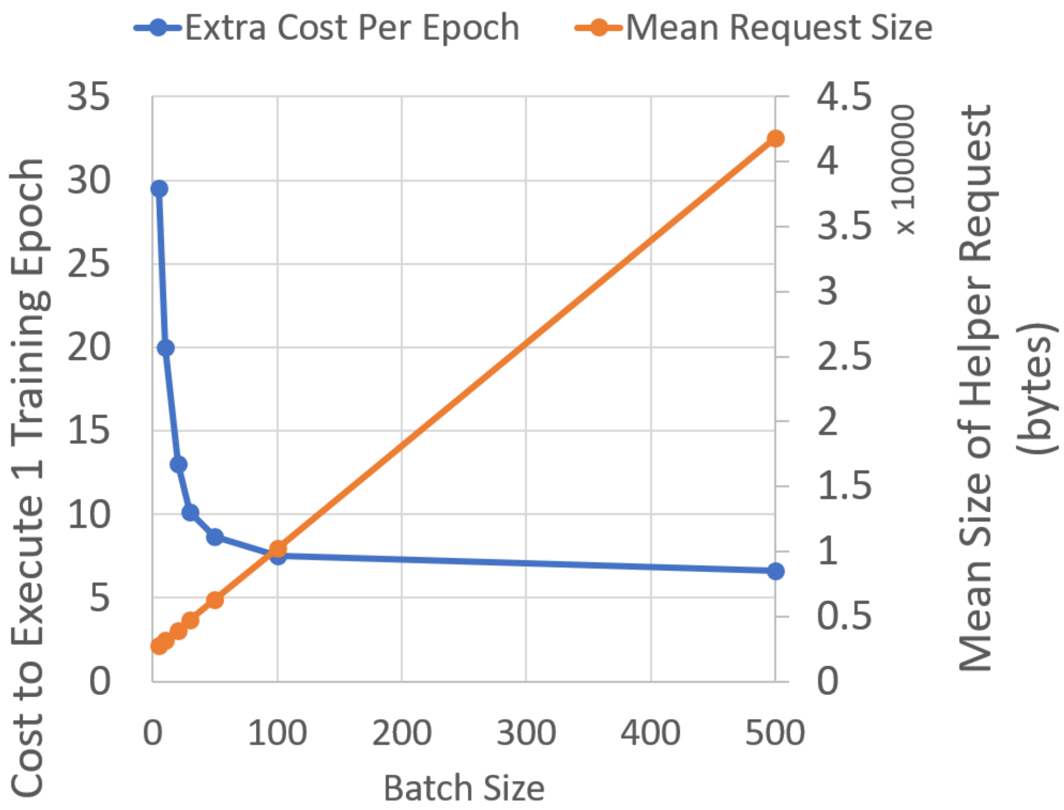
WBCD

Lesson: When Training Remotely, Use Large Batch Sizes

Training using helper services incentivizes sending large amounts of data at one time.



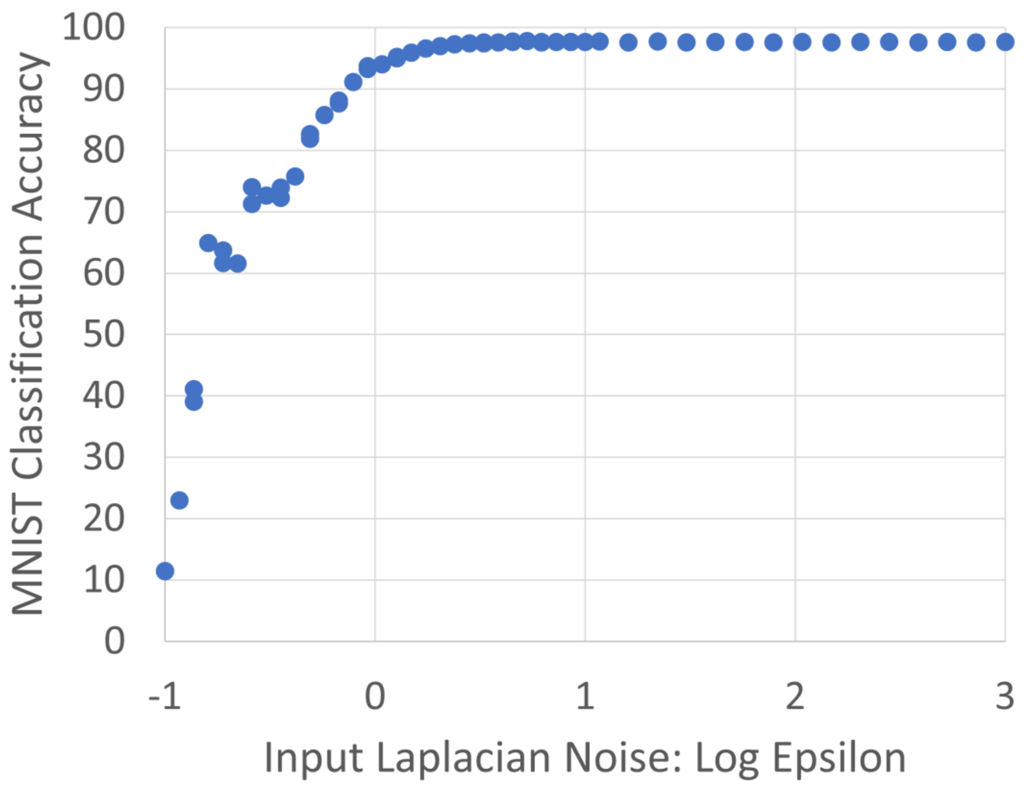
MNIST



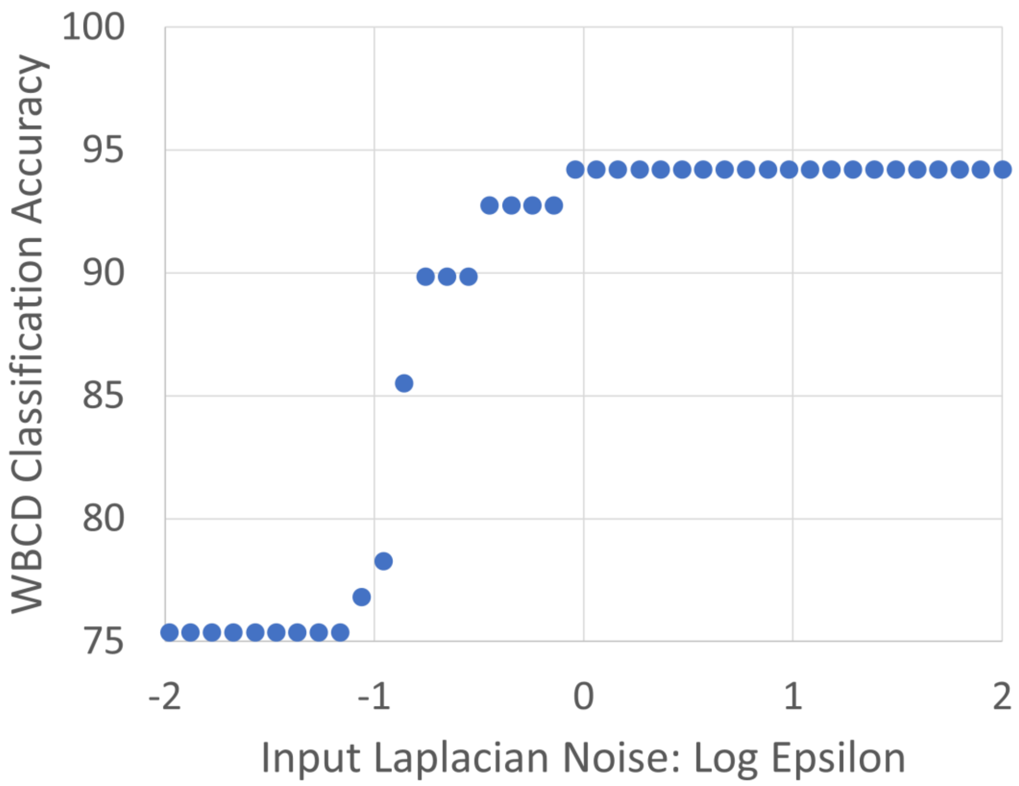
WBCD

Alternate Option: Local Differential Privacy

Gradient clipping can be costly: we can try adding DP noise to the inputs. Networks are surprisingly resilient to this type of noise!



MNIST



WBCD

Conclusions

Conclusions

- Designed platform, Masked LARk, for secure MPC and differentially private aggregation and model training
- Outlined the core functionality and algorithms, along with extensions for additional user or model privacy
- Implemented helper services within Azure, notably the gradient computation for model training
- Analysis on two publicly available datasets
- Released at https://github.com/microsoft/maskedlark
- arXiv version of paper - link.

Q&A