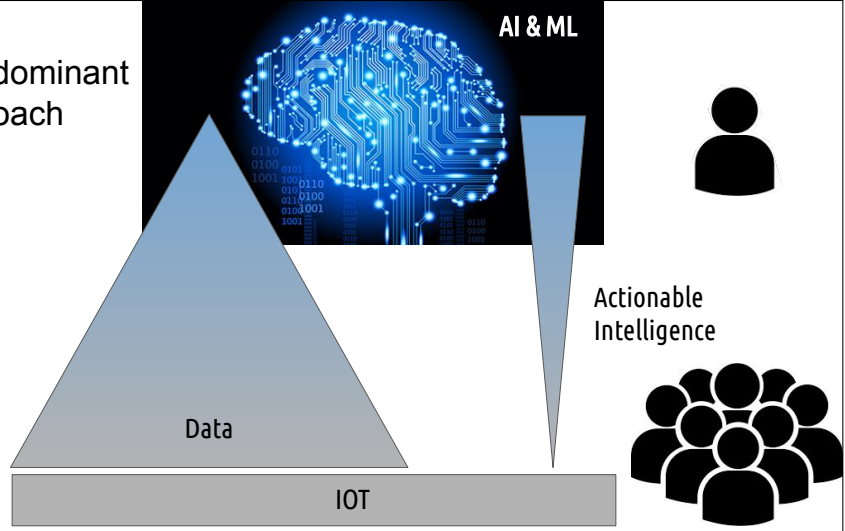
Privacy Preserving Computation
+
Aggregation

Disclaimer: *The purpose of the following slides is to illustrate the high-level concept that underlies some privacy preserving computation techniques — not their mathematical foundations.*

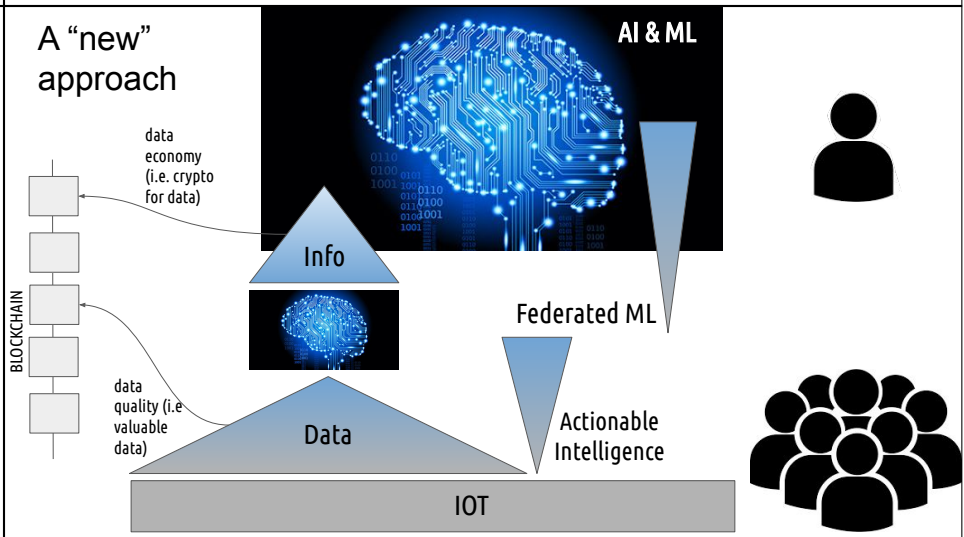
The dominant approach

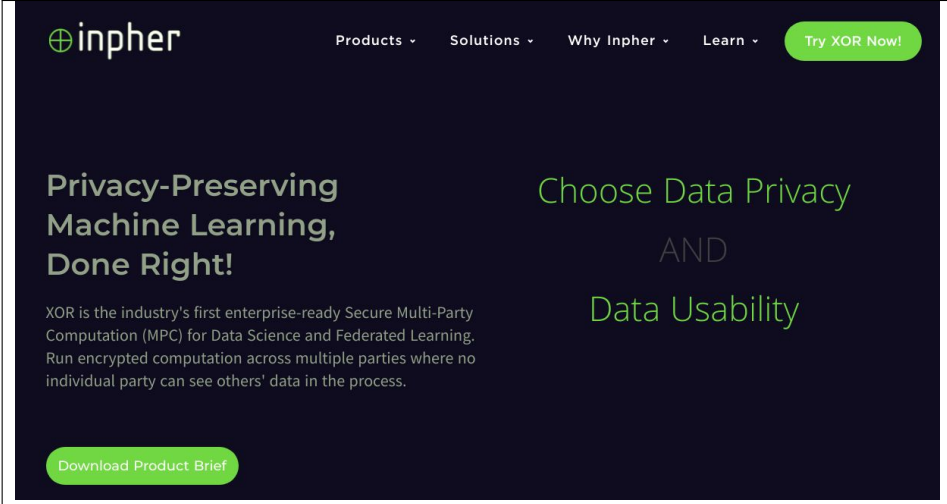


Blockchain, IoT and Privacy Preserving Computation

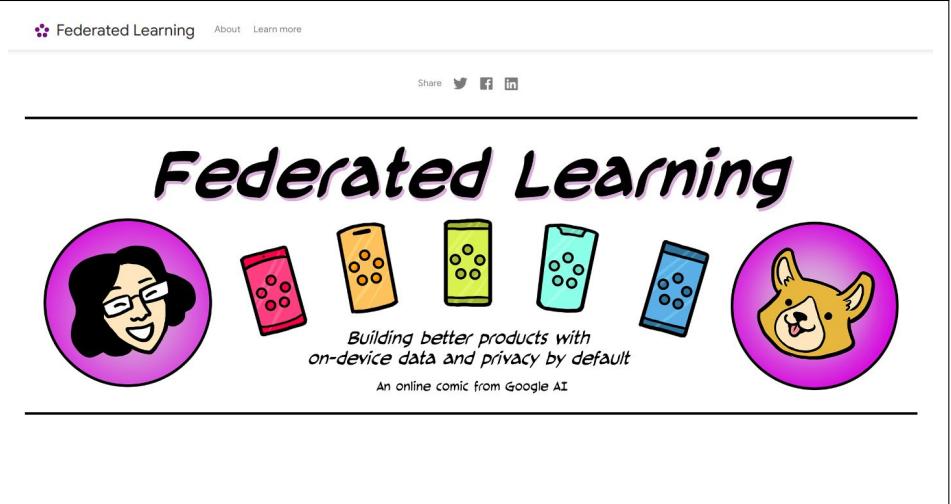
the big picture

A “new” approach



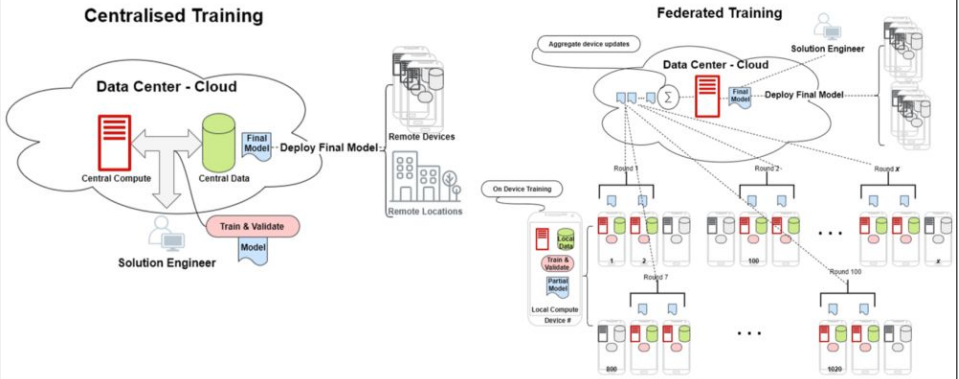


https://inpher.io/xor-secret-computing/

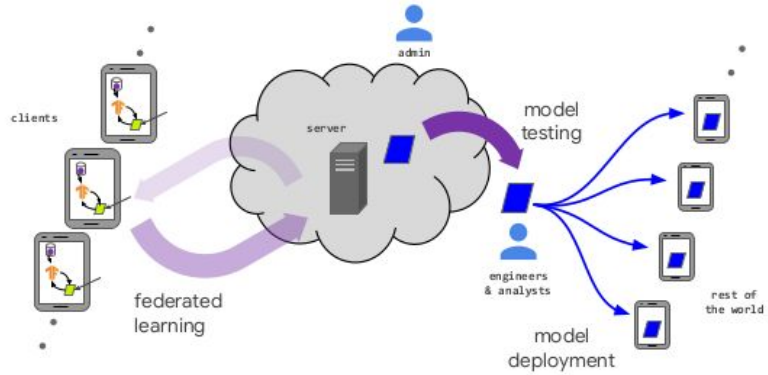


https://federated.withgoogle.com/

https://sites.google.com/view/fl-tutorial/



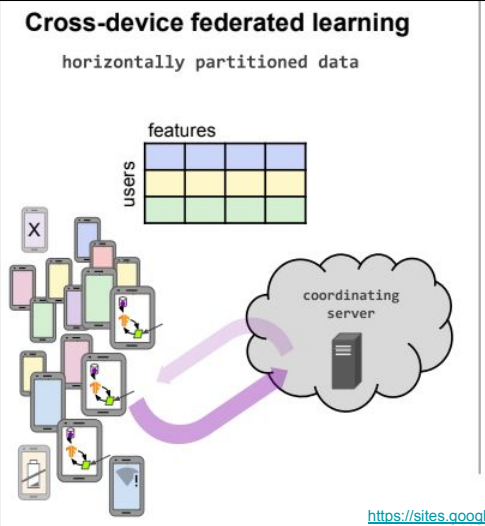
source: https://colinbyrneireland.medium.com/brief-primer-on-federated-learning-part-1-ecb95f36c0df

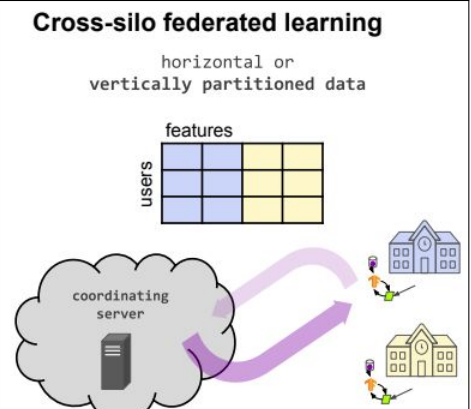


Total population size	10^6–10^{10} devices
Devices selected for one round of training	50 – 5000
Total devices that participate in training one model	10^5–10^7
Number of rounds for model convergence	500 – 10000
Wall-clock training time	1 – 10 days

Table 2: Order-of-magnitude sizes for typical cross-device federated learning applications.

Advances and Open Problems in Federated Learning. Peter Kairouz et al. Arxiv (2019)





https://sites.google.com/view/fl-tutorial/

In Machine Learning [edit]

In machine learning theory, i.i.d. assumption is often made for training datasets to imply that all samples stem from the same generative process and that the generative process is assumed to have no memory of past generated samples.

https://sites.google.com/view/fl-tutorial/

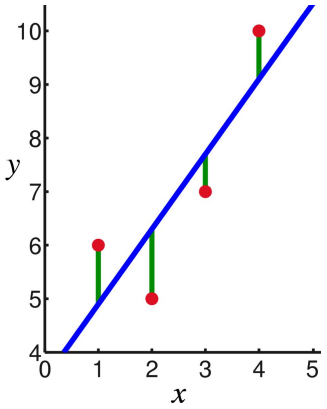
	Datacenter distributed learning	Cross-silo federated learning	Cross-device federated learning
Setting	Training a model on a large but "flat" dataset. Clients are compute nodes in a single cluster or datacenter.	Training a model on siloed data. Clients are different organizations (e.g. medical or financial) or geo-distributed datacenters.	The clients are a very large number of mobile or IoT devices.
Data distribution	Data is centrally stored and can be shuffled and balanced across clients. Any client can read any part of the dataset.	**Data is generated locally and remains decentralized.** Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	**A central orchestration server/service organizes the training**, but never sees raw data.	
Wide-area communication	None (fully connected clients in one datacenter/cluster).	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	
Data availability	——— All clients are almost always available. ———		Only a fraction of clients are available at any one time, often with diurnal or other variations.
Distribution scale	Typically 1 - 1000 clients.	Typically 2 - 100 clients.	Massively parallel, up to 10^{10} clients.
Primary bottleneck	Computation is more often the bottleneck in the datacenter, where very fast networks can be assumed.	Might be computation or communication.	Communication is often the primary bottleneck, though it depends on the task. Generally, cross-device federated computations use wi-fi or slower connections.
Addressability	Each client has an identity or name that allows the system to access it specifically.		Clients cannot be indexed directly (i.e., no use of client identifiers).
Client statefulness	Stateful — each client may participate in each round of the computation, carrying state from round to round.		Stateless — each client will likely participate only once in a task, so generally a fresh sample of never-before-seen clients in each round of computation is assumed.
Client reliability	——— Relatively few failures. ———		Highly unreliable — 5% or more of the clients participating in a round of computation are expected to fail or drop out (e.g. because the device becomes ineligible when battery, network, or idleness requirements are violated).
Data partition axis	Data can be partitioned / repartitioned arbitrarily across clients.	Partition is fixed. Could be example-partitioned (horizontal) or feature-partitioned (vertical).	Fixed partitioning by example (horizontal).

Advances and Open Problems in Federated Learning. Peter Kairouz et al. Arxiv (2019)

	Datacenter distributed learning	Cross-silo federated learning	Cross-device federated learning
Setting	Training a model on a large but "flat" dataset. Clients are compute nodes in a single cluster or datacenter.	Training a model on siloed data. Clients are different organizations (e.g. medical or financial) or geo-distributed datacenters.	The clients are a very large number of mobile or IoT devices.
Data distribution	Data is centrally stored and can be shuffled and balanced across clients. Any client can read any part of the dataset.	**Data is generated locally and remains decentralized.** Each client stores its own data and cannot read the data of other clients. Data is not independently or identically distributed.	
Orchestration	Centrally orchestrated.	**A central orchestration server/service organizes the training**, but never sees raw data.	
Wide-area communication	None (fully connected clients in one datacenter/cluster).	Typically a hub-and-spoke topology, with the hub representing a coordinating service provider (typically without data) and the spokes connecting to clients.	
Data availability	All clients are almost always available.		Only a fraction of clients are available at any one time, often with diurnal or other variations.
Distribution scale	Typically 1 - 1000 clients.	Typically 2 - 100 clients.	Massively parallel, up to 10^{10} clients.

	Datacenter distributed learning	Cross-silo federated learning	Cross-device federated learning
Primary bottleneck	Computation is more often the bottleneck in the datacenter, where very fast networks can be assumed.	Might be computation or communication.	Communication is often the primary bottleneck, though it depends on the task. Generally, cross-device federated computations use wi-fi or slower connections.
Addressability	Each client has an identity or name that allows the system to access it specifically.		Clients cannot be indexed directly (i.e., no use of client identifiers).
Client statefulness	Stateful — each client may participate in each round of the computation, carrying state from round to round.		Stateless — each client will likely participate only once in a task, so generally a fresh sample of never-before-seen clients in each round of computation is assumed.
Client reliability	Relatively few failures.		Highly unreliable — 5% or more of the clients participating in a round of computation are expected to fail or drop out (e.g. because the device becomes ineligible when battery, network, or idleness requirements are violated).
Data partition axis	Data can be partitioned / re-partitioned arbitrarily across clients.	Partition is fixed. Could be example-partitioned (horizontal) or feature-partitioned (vertical).	Fixed partitioning by example (horizontal).

A simple example: Linear Regression



$$Y = mX + c$$

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)^2 = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

Mean Squared Error Equation

LOSS FUNCTION

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

$$D_m = \frac{1}{n}\sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \bar{y}_i)$$

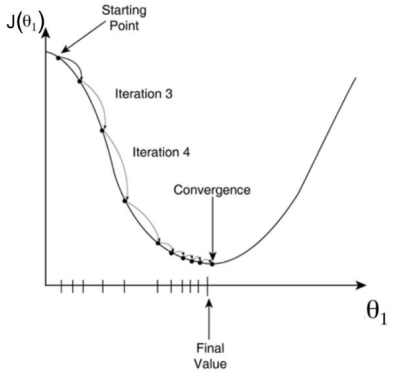
Derivative with respect to m

$$D_c = \frac{-2}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)$$

Derivative with respect to c

$$m = m - L \times D_m$$

$$c = c - L \times D_c$$



Cost Function – "One Half Mean Squared Error":

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Objective:

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

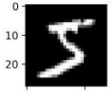
Derivatives:

$$\frac{\partial}{\partial\theta_0} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}$$

Given one input sample pair (x_0, y_0), the goal of deep learning model training is to find a set of parameters w, to maximize the probability of outputting y_0 given x_0.

Given input: x_0



Maximize: $p(5|x_0, w)$

Minimize LOSS = (yi-xi,w)

https://inst.eecs.berkeley.edu/~cs294-163/fa19/slides/federated-learning.pdf

For a training dataset containing n samples $(x_i, y_i), 1 \le i \le n$, the training objective is:

$$\min_{w\in\mathbb{R}^d} f(w) \qquad \text{where } f(w) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

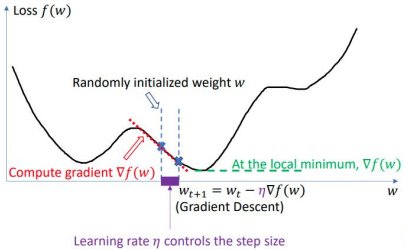
$f_i(w) = l(x_i, y_i, w)$ is the loss of the prediction on example (x_i, y_i)

No closed-form solution: in a typical deep learning model, w may contain millions of parameters.
Non-convex: multiple local minima exist.

https://inst.eecs.berkeley.edu/~cs294-163/fa19/slides/federated-learning.pdf

Solution: Gradient Descent

$$\min_{w\in\mathbb{R}^d} f(w)$$



How to stop? – when the update is small enough – converge.

$$\| w_{t+1} - w_t \| \le \epsilon$$

or $$\| \nabla f(w_t) \| \le \epsilon$$

Problem: *Usually the number of training samples n is large – slow convergence*

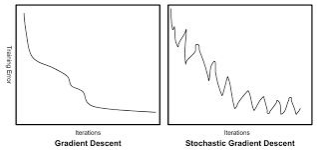
https://inst.eecs.berkeley.edu/~cs294-163/fa19/slides/federated-learning.pdf

Solution: Stochastic Gradient Descent (SGD)

- At each step of gradient descent, instead of compute for all training samples, randomly pick a small subset (mini-batch) of training samples (x_k, y_k).

$$w_{t+1} \leftarrow w_t - \eta \nabla f(w_t; x_k, y_k)$$

- Compared to gradient descent, SGD takes more steps to converge, but each step is much faster.



https://inst.eecs.berkeley.edu/~cs294-163/fa19/slides/federated-learning.pdf

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

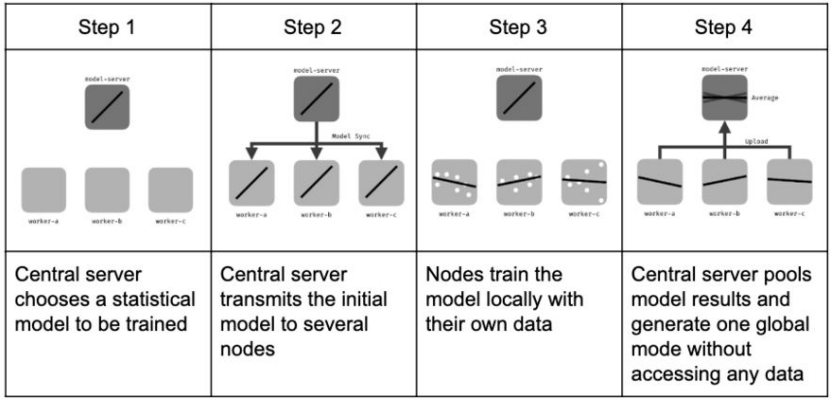
Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

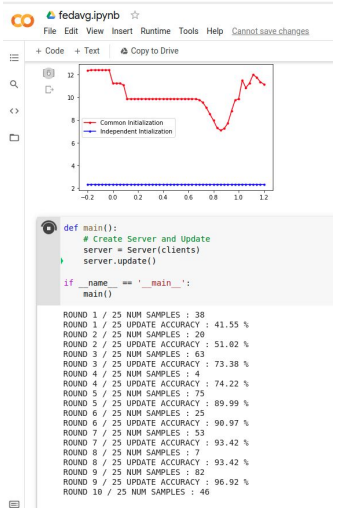
ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

https://arxiv.org/pdf/1602.05629.pdf

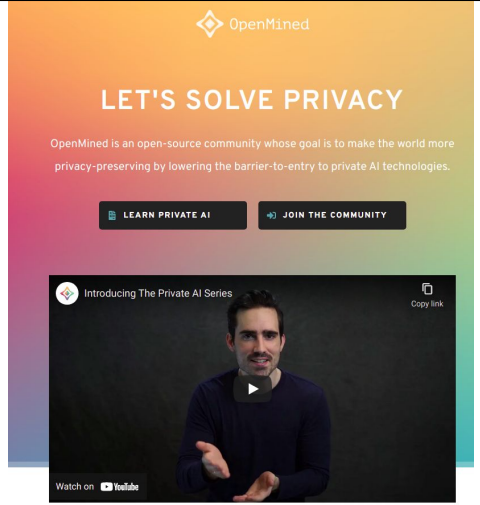
1. At first, a model is randomly initialized on the central server.

2. For each round t:
   i. A random set of clients are chosen;
   ii. Each client performs local gradient descent steps;
   iii. The server aggregates model parameters submitted by the clients.

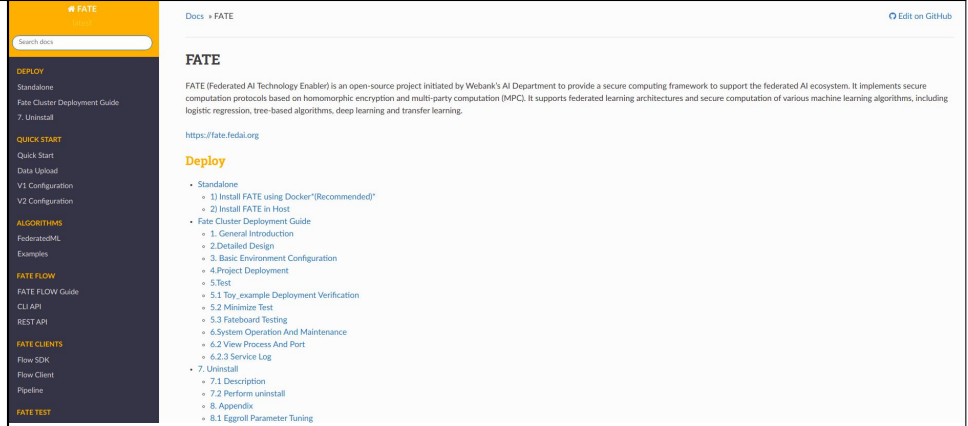




https://colab.research.google.com/drive/1p98m12lD-czEL2WyJSN1YI2tTExz71H3



https://www.openmined.org/



https://fate.readthedocs.io/en/latest/index.html

POSSIBLE LEAK OF INFORMATION

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

A number of evidences in the literature

- Exploiting unintended feature leakage in collaborative learning (in SP2019),
- Deep leakage from gradients (in NeurIPS2019),
- Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning (in INFOCOM2019).

POSSIBLE LEAK OF INFORMATION

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Secure Multiparty Computaion (MCP)

UNBOUND
(MATH OVER MATTER)

A Primer in Secure Multiparty Computation

Prof. Yehuda Lindell
Co-Founder & Chief Scientist of Unbound Tech

https://www.unboundtech.com/wp-content/uploads/2020/09/Unbound_Tech_A_Primer_in_Secure_Multiparty_Computation_MPC.pdf

Secure multiparty computation (MPC)

Jointly computing a function among a set of mutually distrusting parties.

BASIC SCENARIO:

A group of parties wish to compute a given function on their private inputs, while still keeping their inputs private from each other

IoT example

Who is consuming less water?

Alice

Bob

Joe

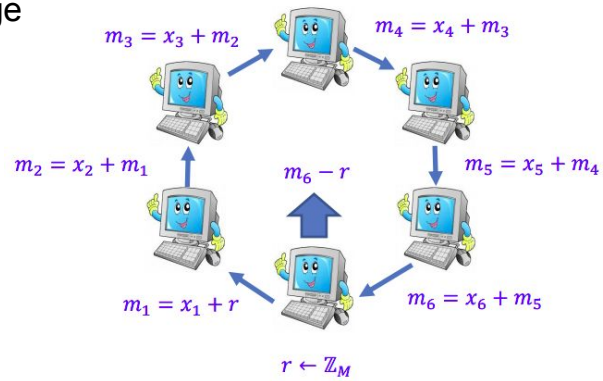
OUTPUT: Alice … they do not know anything about Alice consumption or if Bob consumption is more or less than Joe one

Properties

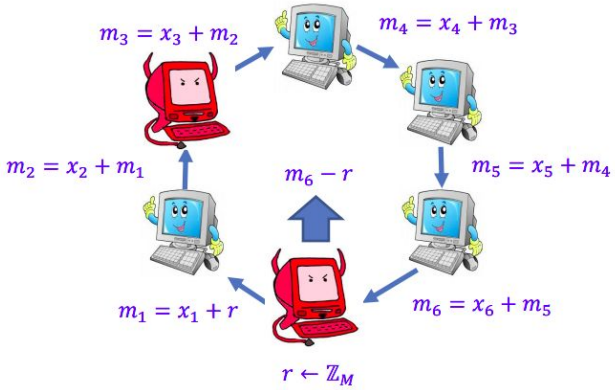
Input privacy: The information derived from the execution of the protocol should not allow any inference of the private data held by the parties, except for what is revealed by the prescribed output of the function.

Correctness: Adversarially colluding parties willing to share information or deviate from the instructions during the protocol execution should not be able to force honest parties to output an incorrect result.

Average



source: https://www.cs.tau.ac.il/~iftachh/Courses/Seminars/MPC/Intro.pdf



source: https://www.cs.tau.ac.il/~iftachh/Courses/Seminars/MPC/Intro.pdf

Adversaries

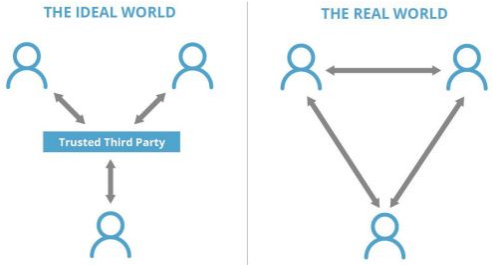
- **Semi-honest (passive) security:** In this case, it is assumed that corrupted parties merely cooperate to gather information out of the protocol, but do not deviate from the protocol specification. This is a rather naive adversary model, that may yield weak security in real situations. However, protocols achieving this level of security prevent inadvertent leakage of information between parties, and are thus useful if this is the only concern. In addition, protocols in the semi-honest model can suffice in cases where it is possible to guarantee that the code being run cannot be replaced.
- **Malicious (active) security:** In this case, the adversary may arbitrarily deviate from the protocol execution in its attempt to cheat. Protocols that achieve security in this model provide a very high security guarantee. The only thing that an adversary can do in the case of dishonest majority is to cause the honest parties to “abort” having detected cheating. If the honest parties do obtain output, then they are guaranteed that it is correct. Of course, their privacy is always preserved.

What it means for a protocol to be secure?

Hard to formalize:

- The parties should “learn nothing”, **but** they need to learn the output and this depends on the inputs.
- The output must be “correct”, **but** correct output depends on the parties’ inputs, and we do not know what inputs corrupted parties will use.

Intuition: ideal-real-world paradigm

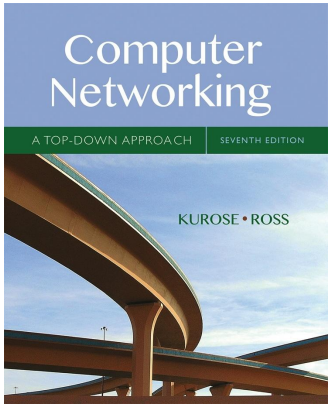


MPC protocol is secure if a real-world protocol "behaves" like an ideal-world one, meaning that the only information revealed by the real-world protocol is that which is revealed by the ideal-world one, and that the output distribution is the same in both (guaranteeing correctness since this holds by definition in the ideal world).

DETOUR on RSA BASICS

Slides from Chapter 8 Security

of Computer Networking: A Top Down Approach
7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016



Two-Party Secure Computation of RSA

Joint computation: decrypt an RSA encrypted text

Know in advance: the public key, the encrypted text

Know after: the decrypted text … nothing more and in particular not the secret key

Two-Party Secure Computation of RSA (some simplification please see the original paper)

Choose $d1$ and $d2$ uniformly at random under the constraint that d= $d1$ + $d2$
- Alice has ($d1$, N)
- Bob has ($d2$, N)

Neither Alice nor Bob can carry out the decryption operation by themselves, since they only hold a random share of the secret exponent d.

y is the encrypted text
- Alice can compute $x1 = y^{d1} \bmod N$ in isolation
- Bob can compute $x2 = y^{d2} \bmod N$ in isolation

Then, the product $x1 \cdot x2 \bmod N$ is the required result, since $x1 \cdot x2 = y^{d1} \cdot y^{d2} = y^{(d1+d2)} = y^{d}$ (where all operations are modulo N).

$x1 \cdot x2 \bmod N$ is the required result, since

$x1 \cdot x2 = y^{d1} \cdot y^{d2} = y^{(d1+d2)} = y^{d}$ (where all operations are modulo N).

This operation was computed without d ever being revealed.

Refreshing keys

A very important feature of this method is that it is actually possible for Alice and Bob to continually modify the value of their share of the key, without modifying the key itself.
Specifically, the parties can run a secure coin-tossing protocol to obtain a random value r of the same length as N.

- Then, Alice can set $d1' = d1 + r$ and Bob can set $d2' = d2 - r$.
- The result is that $d1' + d2' = d1 + d2 = d$ and so everything still works as above.

However, if an attacker successfully attacks Alice and steals $d1$, and then later attacks Bob and steals $d2'$, then it will actually learn nothing about d (because $d1 + d2' = d - r$ and so the secret d is masked by the random r).

This means that the attacker has to successfully attack Alice and Bob at the same time in order to steal the private key. In the literature, this type of security is known as “proactive security”.

Is it general?

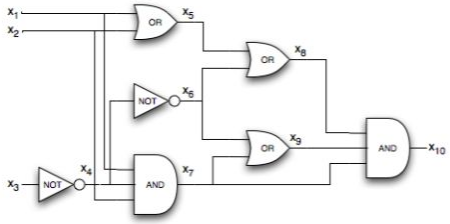
The example of how the RSA function can be securely computed shows that it is indeed possible to compute with shared inputs. However, one may be tempted to conclude that this is only possible because of the specific algebraic structure of the function. In particular, how is it possible to securely compute AES or HMAC-SHA256 on shared keys, since these functions have no such clean structure by design? We will therefore now describe a method that can be used to securely compute any function whatsoever; this method is called Yao’s protocol.

In the academic literature, it was shown in the late 1980s that any function can be securely computed. However, the solutions proposed were not efficient enough to actually be used in practice. For the first 20 or so years, MPC research focused mainly on theoretical aspects.

In recent years, significant algorithmic improvements have been made to MPC protocols, as a result of a major research effort to make MPC practical. Great progress has been made and secure computation can now be used to solve a wide range of problems with practical response times.

Boolean Circuits

Yao's basic protocol is secure against semi-honest adversaries and is extremely efficient in terms of number of rounds, which is constant, and independent of the target function being evaluated. The starting point of the protocol is the translation of the function to be computed into a Boolean



It is well known that any function can be represented as a Boolean circuit. For example, the AES encryption function can be converted into a Boolean circuit with approximately 32,000 gates

Garbled Circuits and Yao's Protocol

Yao explained how to garble (or "encrypt") a circuit so that it can be evaluated without revealing anything but the output of the circuit.

At a high level, Alice prepares the garbled circuit and sends it to Bob, who obliviously evaluates the circuit, learning the output corresponding to both his and Alice's input.

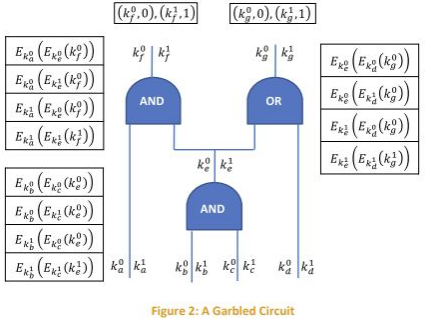


Figure 2: A Garbled Circuit

POSSIBLE LEAK OF INFORMATION

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

SMCP

Methods	Security level	Efficiency	Suitable number of participants	Suitable model for ML training
Secure Multi-party Computation	Provable secure ★★★★★	Low	Small (Cross-organization collaboration)	Simpler model
Federated Learning	Leak intermediate information ★★★☆☆	Medium	Big (Edge computing)	All

https://medium.com/applied-mpc/federated-learning-and-mpc-93ab84c7909a

MPyC: Secure Multiparty Computation in Python

From VIFF, via TUeVIFF, to MPyC, launched on Wednesday May 30, 2018 at the Theory and Practice of Multiparty Computation (TPMPC) 2018 workshop in Aarhus, Denmark.

See MPyC--Python Package for Secure Multiparty Computation (PDF Slides) for some background information.

Use `pip install mpyc` for version 0.7 (October 31, 2020) of MPyC on PyPI. The documentation of this version is available at MPyC on github.io.

Use `pip install git+https://github.com/lschoe/mpyc` to get the latest version from GitHub.

See github.com/lschoe/mpyc for source code and demos (Python scripts as well as Jupyter notebooks).

Check out this cool YouTube video on MPC by TNO to see where the MPyC logo comes from!

Try it out in the cloud!

Run MPyC without installing anything by running a Jupyter notebook in your browser with Binder:

- Click launch binder to try the notebook `SecureSortingNetsExplained.ipynb`.
- Click launch binder to try the notebook `SecureSantaExplained.ipynb`, including example runs with multiple parties.
- Click launch binder to try the notebook `OneWayHashChainsExplained.ipynb`.
- Click launch binder to try the notebook `KaplanMeierSurvivalExplained.ipynb`.

Last updated Saturday, 31-Oct-2020 20:02:31 CET by Berry Schoenmakers.

https://www.win.tue.nl/~berry/mpyc/

POSSIBLE LEAK OF INFORMATION

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Differential Privacy

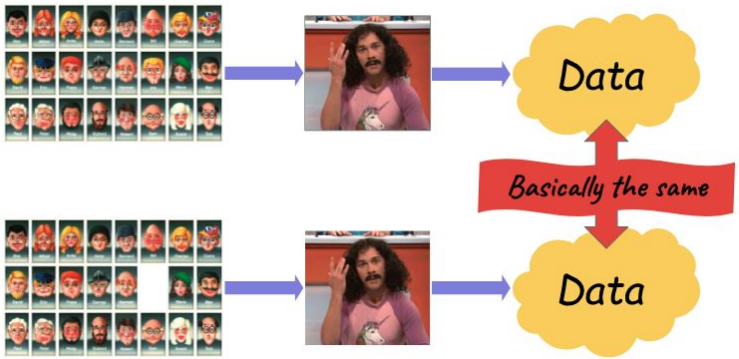
Why differential privacy is awesome

2018-07-30 — updated 2019-02-20

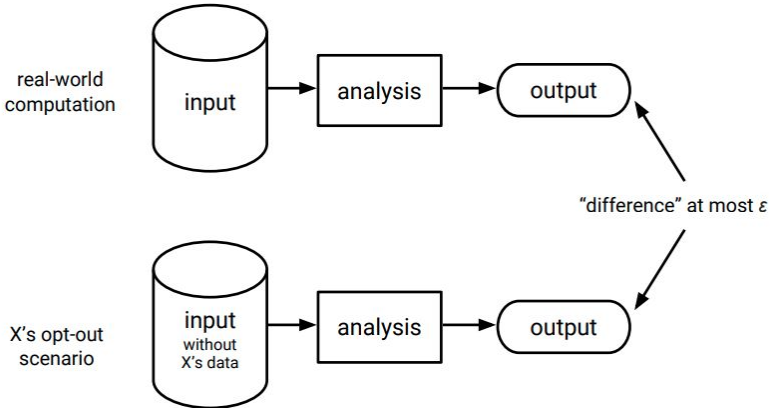
Part of a series on differential privacy. In case you need reading material once you finished this post!

1. **Why differential privacy is awesome** (this article) presents a non-technical explanation of the definition.
2. Differential privacy in (a bit) more detail introduces the formal definition, with very little math.
3. Differential privacy in practice (easy version) explains how to make simple statistics differentially private.
4. Almost differential privacy describes how to publish private histograms without knowing the categories in advance.
5. Local vs. global differential privacy presents the two main *models* of differential privacy, depending on who the attacker is.
6. The privacy loss random variable explains the real meaning of (ε, δ)-differential privacy.
7. The magic of Gaussian noise introduces Gaussian noise and its shiny properties.
8. Why not differential privacy? explores what it means for an algorithm to *not* be differentially private.

https://desfontain.es/privacy/differential-privacy-awesomeness.html



https://desfontain.es/privacy/differential-privacy-awesomeness.html



https://privacytools.seas.harvard.edu/files/privacytools/files/pedagogical-document-dp_0.pdf

A process A is ε-differentially private if for all databases D_1 and D_2 which differ in only one individual:

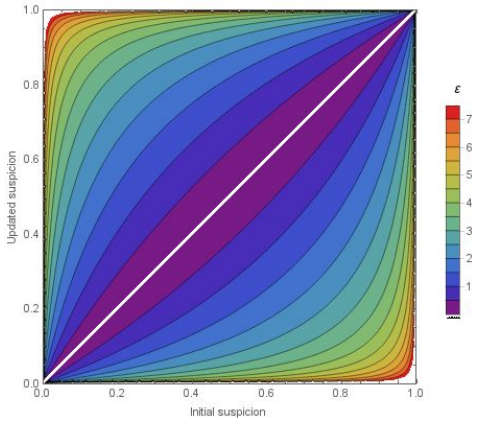
$$\mathbb{P}[A(D_1) = O] \leq e^{\varepsilon} \cdot \mathbb{P}[A(D_2) = O]$$

... and this must be true for all possible outputs O. Let's unpack this.

$\mathbb{P}[A(D_1) = O]$ is the probability that when you run the process A on the database D_1, the output is O. This process is probabilistic: if you run it several times, it might give you different answers. A typical process might be: "count the people with blue eyes, add some random number to this count, and return this sum". Since the random number changes every time you run the process, the results will vary.

e^{ε} is the exponential function applied to the parameter $\varepsilon > 0$. If ε is very close to 0, then e^{ε} is very close to 1, so the probabilities are very similar. The bigger ε is, the more the probabilities can differ.

https://desfontain.es/privacy/differential-privacy-awesomeness.html



We have a mechanism A which is ε-differentially private. We run it on some database D, and release the output A(D) to an attacker. Then, the attacker tries to figure out whether someone (their *target*) is in D.

Let's take the stronger attacker we can think of: they know *all the database, except their target*. This attacker has to determine which database is the real one, between two options: one with their target in it (let's call it D_{in}), the other without (D_{out})

Attacker sees $A(D)=O$ updated suspicion $\mathbb{P}[D=D_{in} \mid A(D)=O]$

https://desfontain.es/privacy/differential-privacy-awesomeness.html

Gertrude, a 65-year-old woman, is considering whether to participate in a medical research study. While she can envision many potential personal and societal benefits that could result in part from her participation, she is concerned that the personal information she discloses in the course of the study could lead to an increase in her life insurance premium in the future.

For example, Gertrude is concerned that the tests she would undergo as part of the research study would reveal that she is predisposed to suffer a stroke and is significantly more likely to die in the coming year than the average person of her age and gender. If such information related to Gertrude's increased risk of morbidity and mortality is discovered by her life insurance company, it will likely increase her premium substantially.

Before she opts to participate in the study, Gertrude wishes to be assured that privacy measures are in place to ensure that her participation will have, at most, a limited effect on her life insurance premium.

https://privacytools.seas.harvard.edu/files/privacytools/files/pedagogical-document-dp_0.pdf

Gertrude holds a $\$100,000$ life insurance policy. Her life insurance company has set her annual premium at $\$1,000$, i.e., 1% of $\$100,000$, based on actuarial tables that show that someone of Gertrude's age and gender has a 1% chance of dying in the next year.

Suppose Gertrude opts out of participating in the medical research study. Regardless, the study reveals that coffee drinkers are more likely to suffer a stroke than non-coffee drinkers. Gertrude's life insurance company may update its assessment and conclude that, as a 65-year-old woman who drinks coffee, Gertrude has a 2% chance of dying in the next year. The company decides to increase Gertrude's annual premium from $\$1,000$ to $\$2,000$ based on the findings of the study.

https://privacytools.seas.harvard.edu/files/privacytools/files/pedagogical-document-dp_0.pdf

Suppose Gertrude decides to participate in the medical research study. Based on the results of medical tests performed on Gertrude over the course of the study, the researchers conclude that Gertrude has a 50% chance of dying from a stroke in the next year. If the data from the study were to be made available to Gertrude's insurance company, it might decide to increase her insurance premium from $2,000 to more than $50,000 in light of this discovery.

Fortunately for Gertrude, this does not happen. Rather than releasing the full dataset from the study, the researchers release only a differentially private summary of the data they collected. If the researchers use a value of $\epsilon = 0.01$, then the insurance company's estimate of the probability that Gertrude will die in the next year can increase from 2% to at most

$$2\% \cdot (1 + 0.01) = 2.02\%.$$

Thus Gertrude's insurance premium can increase from $2,000 to, at most, $2,020. Gertrude's first-year cost of participating in the research study, in terms of a potential increase in her insurance premium, is at most $20.

https://privacytools.seas.harvard.edu/files/privacytools/files/pedagogical-document-dp_0.pdf

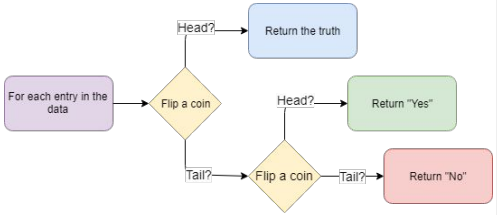
Example: Avg age of a group of peple

Instead of having each person send you their true age, you have them send you their true age + random number between -100 and 100. So, if someone was 42, they might send you 42 + (-50) = -8.
we can generate random numbers so that if you average over enough of them, they cancel each other out. Thus, if 10,000 people all add a random number (pulled from a distribution with a mean of 0) to their age before reporting it, the average age reported will still be similar to the underlying raw data despite the fact that nobody revealed their true age.

The bigger the random numbers (on average), the more privacy protection we give people, but the larger the group of people we need to average over before we can get aggregate statistics

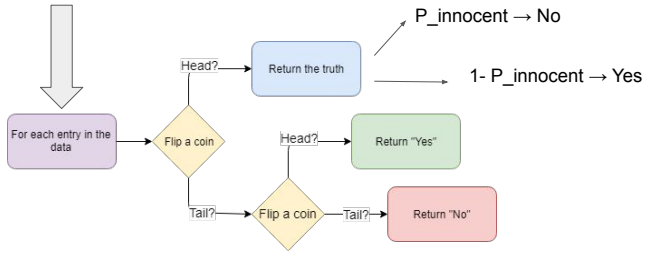
This approach is useful for allowing app users to transform their local data in a way that protects it so that the central server can collect useful statistics without the central server being able to reverse engineer any specific person's personal data.

Example



https://towardsdatascience.com/understanding-differential-privacy-85ce191e198a

Are you guilty?



P_Yes = P_Head(1-P_innocent)+(1-P_Head)P_Head

P_innocent = 1- (P_Yes-P_Head(1-P_Head))/P_Head

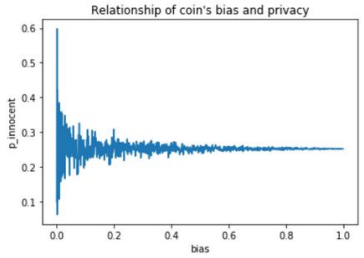


Figure 4: Compute on a simulated survey data of 20,000 individuals. Low bias coins add more noise to the data. A consequence of adding noise is the decrease in privacy loss.

- What does DP tell us?

  As you can see in Figure 4, the variance of *p_innocent* increases dramatically and approaches infinity when *p_head* approaches 0, lead to a rapid decrease in privacy loss. DP also gives us the same conclusion. Thus, when *p_head* is 0, the distribution of returned result is identical, no matter an individual is an innocent or not (the distance of 2 distributions is P("yes"|not innocent)-P("yes"|innocent)=*p_head*, the bias). If the number of innocents participates in the data changed, it does not lead to any changes in information in the noisy returned data. It means that there is no private information in the noisy returned data.

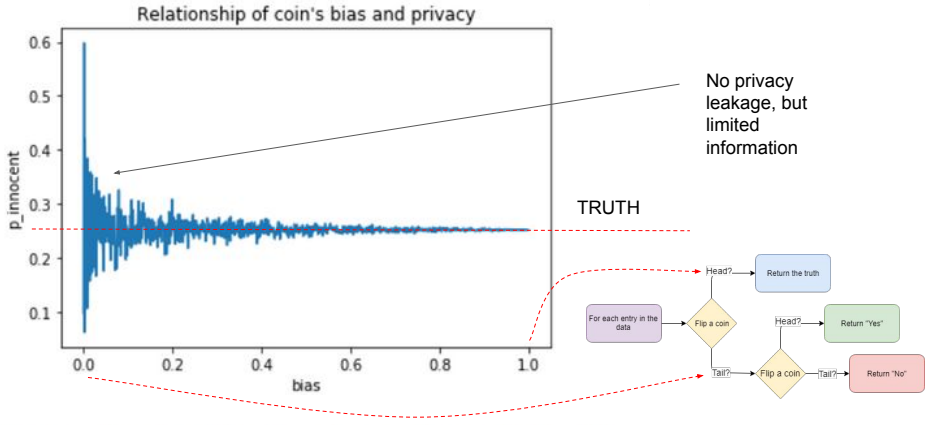


Figure 4: Compute on a simulated survey data of 20,000 individuals. Low bias coins add more noise to the data. A consequence of adding noise is the decrease in privacy loss.

Differential privacy in practice (easy version)

2018-11-22 — updated 2019-02-20

- **Counting unique users**
- **Counting things**
- **Summing or averaging numbers**
- **Releasing many things at once**
- **Traps to avoid**

https://desfontain.es/privacy/differential-privacy-awesomeness.html

Counting unique users

Suppose you have a database, and you want to publish how many people in there satisfy a given condition. Say, how many have green eyes? Even if you have many people in your database, you can't just publish the true answer.

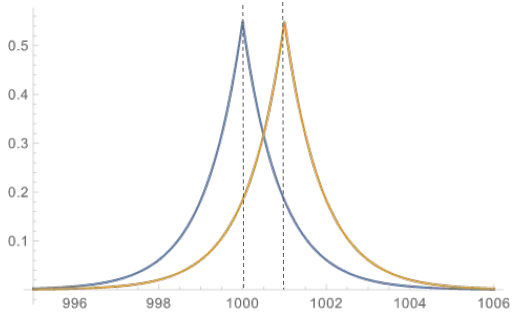
With differential privacy, we assume that the attacker knows *almost all elements*. They only have uncertainty about their target. Say they want to know whether their target has green eyes. If you output the real number *k*, they can compare it with the number of people with green eyes among the people they know. If it's *k*−1, then the target has green eyes. If it's k, then the target does not.

https://desfontain.es/privacy/differential-privacy-awesomeness.html

So, to get ε-differential privacy, we pick a random value according to Laplace(1/ε) and we add this noise to the real value.

Why does it work?

Let's look at the distribution of the number we return, depending on whether the true count is *k*=1000 (blue line, the target doesn't have green eyes) or k=1001 (yellow line, the target has green eyes).

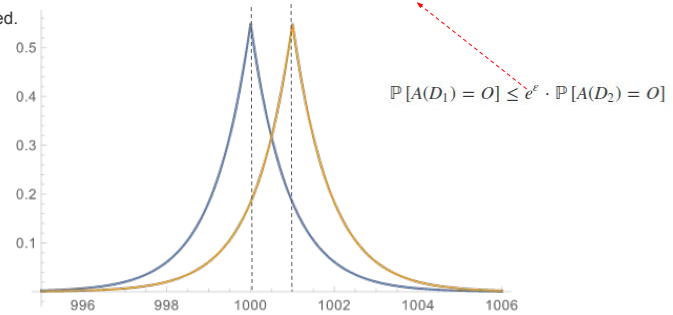


https://desfontain.es/privacy/differential-privacy-awesomeness.html

Let's say the real number is *k*=1001, and after adding noise, we published 1003.
Let's put ourselves in the attacker's shoes.
What's the likelihood that the original number was 1001 vs. 1000?

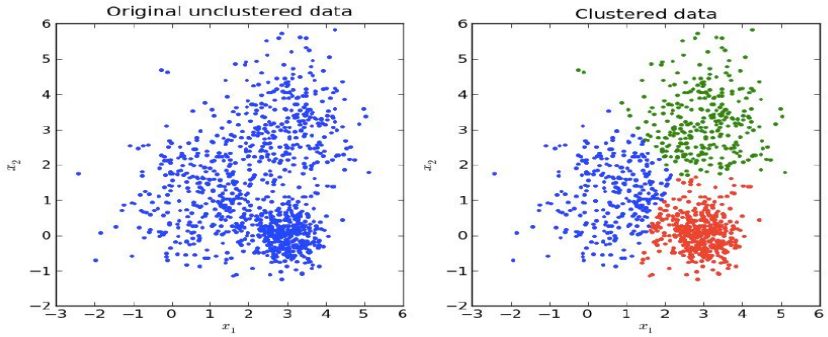
The hypothesis "*k*=1001" is a bit more likely: generating a noise of 2 is more likely than a noise of 3

How much more likely? It turns out that the *ratio* between these likelihoods is e^{ε} → The ratio of probabilities of differential privacy is satisfied.



https://desfontain.es/privacy/differential-privacy-awesomeness.html

Case Study: K-means Clustering



K-means Clustering

- Partition a set of points x1, x2, …, xn into k clusters S1, S2, …, Sk such that the following is minimized:

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|_2^2$$

Mean of the cluster S_i

K-means Clustering

Algorithm (Lloyd):

- Initialize a set of k centers
- Repeat
  - Assign each point to its nearest center
  - Recompute the set of centers

  Until convergence …
- Output final set of k centers

Differentially Private K-means

- Suppose we fix the number of iterations to T
- In each iteration (given a set of centers):
  1. Assign the points to the new center to form clusters
  2. Noisily compute the size of each cluster
  3. Compute noisy sums of points in each cluster

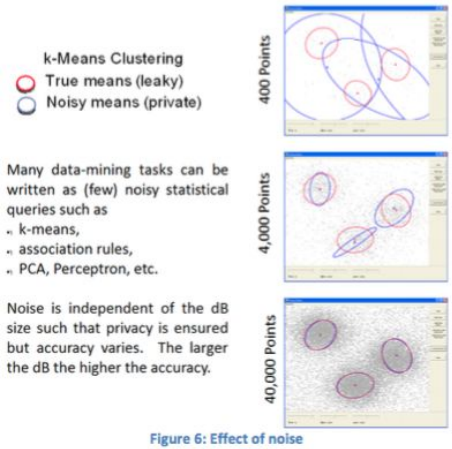


Figure 6: Effect of noise

https://dennyglee.com/2007/09/24/analyzing-data-while-protecting-privacy-a-case-study/

Differentially Private Kmeans

- Suppose we fix the number of iterations to T
- In each iteration (given a set of centers):
  1. Assign the points to the new center to form clusters
  2. Noisily compute the size of each cluster
  3. Compute noisy sums of points in each cluster

Each iteration uses ε/T privacy budget, total privacy loss is ε

POSSIBLE LEAK OF INFORMATION

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Homomorphic Encryption

Homomorphic Encryption Standardization

An Open Industry / Government / Academic Consortium to Advance Secure Computation

Home Introduction Standard Participants Standards Meetings Affiliated Workshops Mailing Lists Contact

Homomorphic Encryption

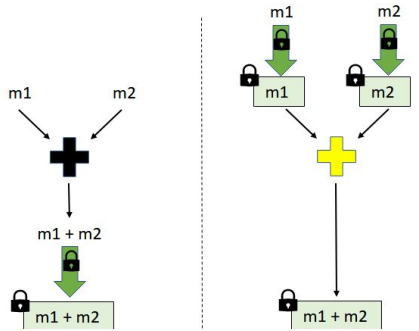
Homomorphic Encryption provides the ability to compute on data while the data is encrypted. This ground-breaking technology has enabled industry and government to provide never-before enabled capabilities for outsourced computation securely.

HomomorphicEncryption.org is an open consortium of industry, government and academia to standardize homomorphic encryption.

Please join our mailing list and participate in our standardization efforts.

https://homomorphicencryption.org/

- *Partially homomorphic encryption* encompasses schemes that support the evaluation of circuits consisting of only one type of gate, e.g., addition or multiplication.
- *Somewhat homomorphic encryption* schemes can evaluate two types of gates, but only for a subset of circuits.
- *Leveled fully homomorphic encryption* supports the evaluation of arbitrary circuits composed of multiple types of gates of bounded (pre-determined) depth.
- *Fully homomorphic encryption* (FHE) allows the evaluation of arbitrary circuits composed of multiple types of gates of unbounded depth, and is the strongest notion of homomorphic encryption.



Unpadded RSA

If the RSA public key has modulus n and encryption exponent e, then the encryption of a message m is given by $\mathcal{E}(m) = m^e \mod n$.
The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= m_1^e m_2^e \mod n \\ &= (m_1 m_2)^e \mod n \\ &= \mathcal{E}(m_1 \cdot m_2)\end{aligned}$$