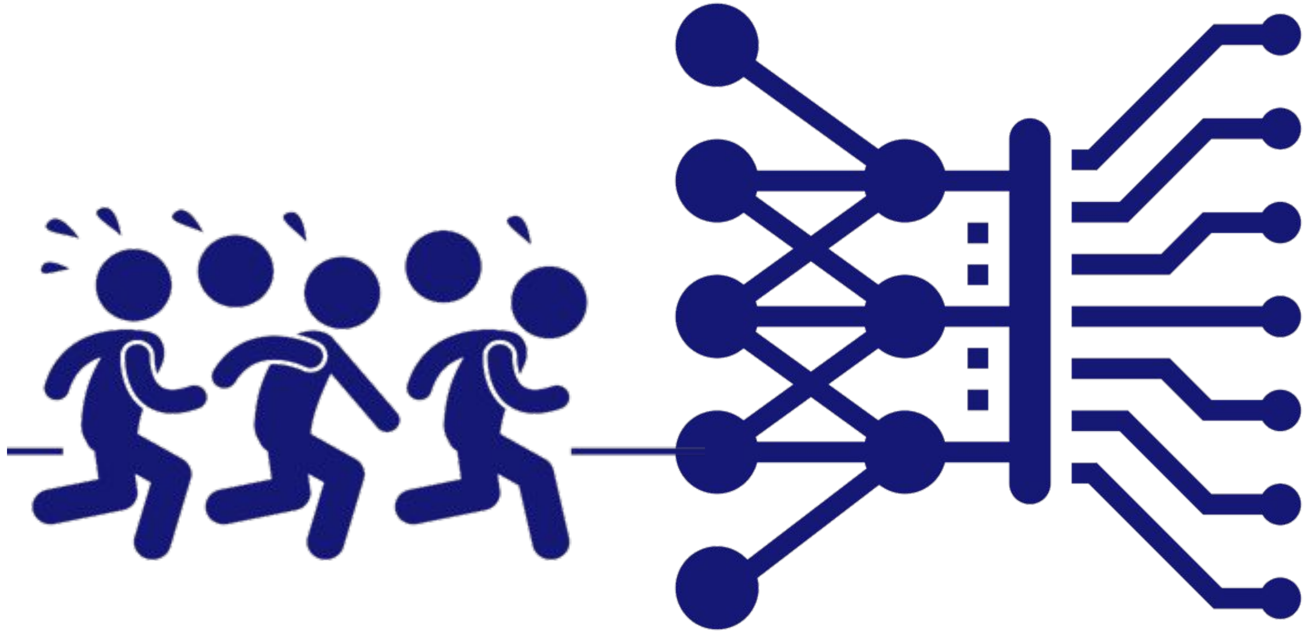
# Deep Learning and Privacy

魏澤人

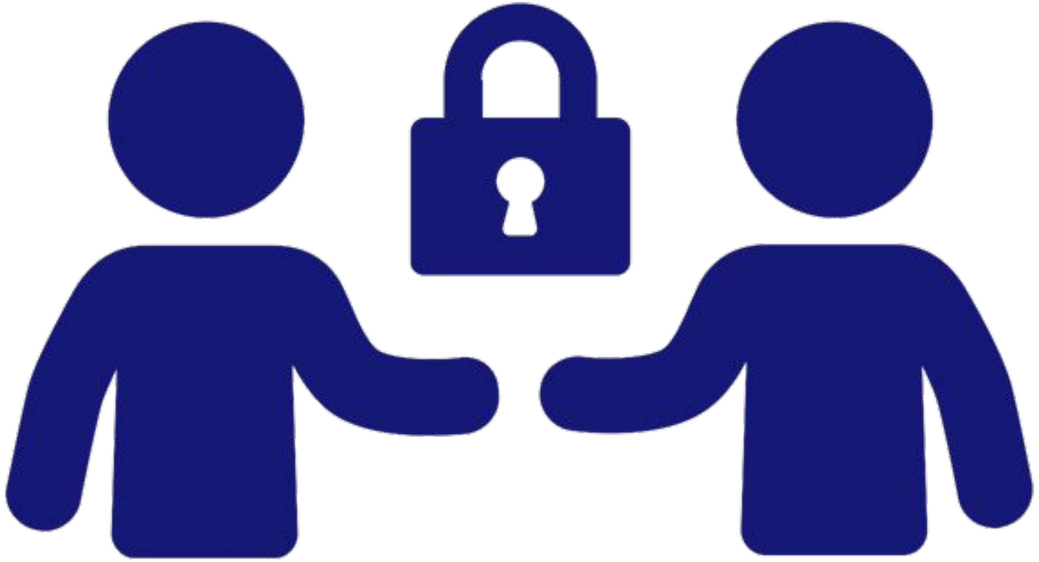
## About Me

- 交大 AI 學院, 之前在東華應數
- GDE (Machine Learning)
- Organizer, GDG Hualien
- Organizer, 花蓮.py
- Chief Mathematician, iiNumbers
- AI and Math Consultant, Ubitus

## Why Deep Learning?



## Why Privacy?



## De-identification

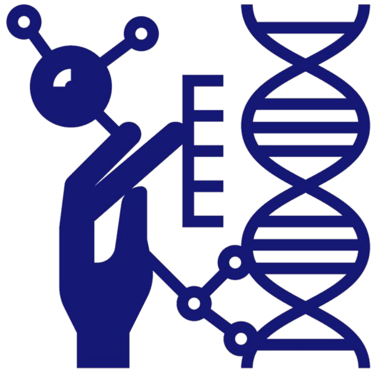
- Pseudonymization
  - replacing real names with a temporary ID
- K-anonymization
  - defines attributes that indirectly points to the individual's identity as quasi-identifiers (QIs) and deal with data by making at least k individuals have same combination of QI values

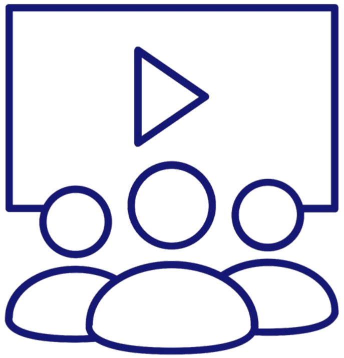
## Re-identification

- AKA De-Anonymization
- 1990 GIC hospital visits data (remove name, addresses, SSN)
    - Still has zip codes, birth data, sex
    - Can be re-identified using voter databases
    - Similar deanonymization method used in 1997, 2001
- 2006 AOL published anonymized user’s search query data
    - Two reporters were able to track down a 62 year old widow named Thelma Arnold
- 2015, Hardesty, Larry. "How hard is it to 'de-anonymize' cellphone data?". MIT news

# Re-identification

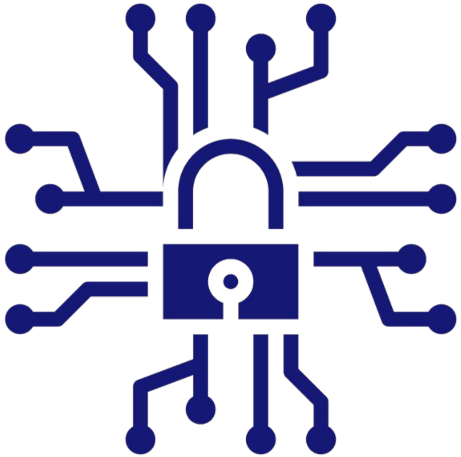
- 2013, Gymrek et al.  "Identifying personal genomes by surname inference", Science
- 2007,  $1 million Netflix Prize was reversed by researchers
  - Both user and movie titles are de-identified before release
  - Narayanan, Shmatikov,  Robust De-anonymization of Large Sparse Datasets





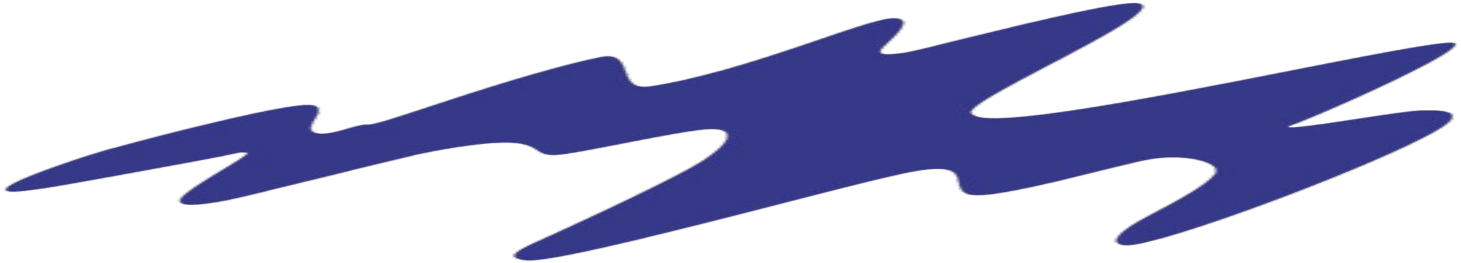
## Towards Theoretical Definition

- Cryptography definitions?
  - Zero-Knowledge
  - Indistinguishability
- Early works
  - 1977 Dalenius
  - 1979 Query Privacy
- 2003 Nissim, Dinur demonstrated some impossible results.



# Differential Privacy

- Fundamental Law of Information Recovery
  - in the most general case, privacy cannot be protected without injecting some amount of noise
- 2006 Cynthia Dwork et al, formalized the definition
  - 2016 TCC Test-of-Time Award, 2017 Godel Prize
- **ε-differential privacy**:  a person's privacy cannot be compromised by a statistical release if their data are not in the database.

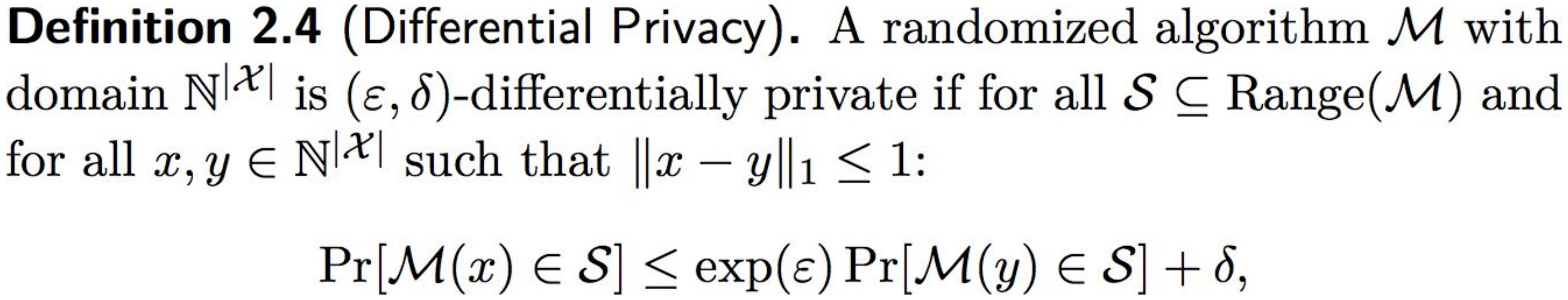


## Formal Definition

$$\boxed{\Pr[\mathcal{A}(D_1) \in S] \leq e^{\epsilon} \times \Pr[\mathcal{A}(D_2) \in S]}$$

- D1 and D2 are datasets that differ on a single element
- A is a randomized algorithm
- S is a subset of the image of A
- Composability, robustness to post-processing

# Formal Definition



**Definition 2.4** (Differential Privacy). A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is $(\varepsilon, \delta)$-differentially private if for all $\mathcal{S} \subseteq \mathrm{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta,$$

## Sensitivity

$$\boxed{\Delta f = \max \|f(D_1) - f(D_2)\|_1}$$

- Example Function: mean of 1,0,1,1,...,0,1,0
- Example Query: sum > 5
- Compare full dataset to every partial dataset(that remove single element)

## Add Noise

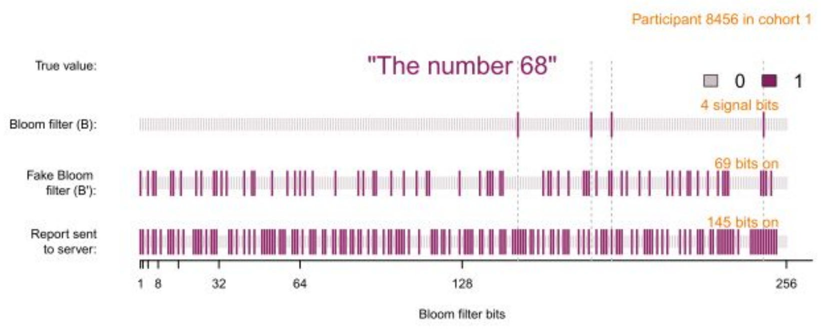
- The Laplace mechanism
  - $\mathcal{T}_{\mathcal{A}}(x) = f(x) + Y$
- Randomized response
  - Throw a coin
  - If head, then throw the coin again (ignoring the outcome), and answer the question honestly.
  - If tail, then throw the coin again and answer "Yes" if head, "No" if tail.

# In Practice (according to wikipedia)

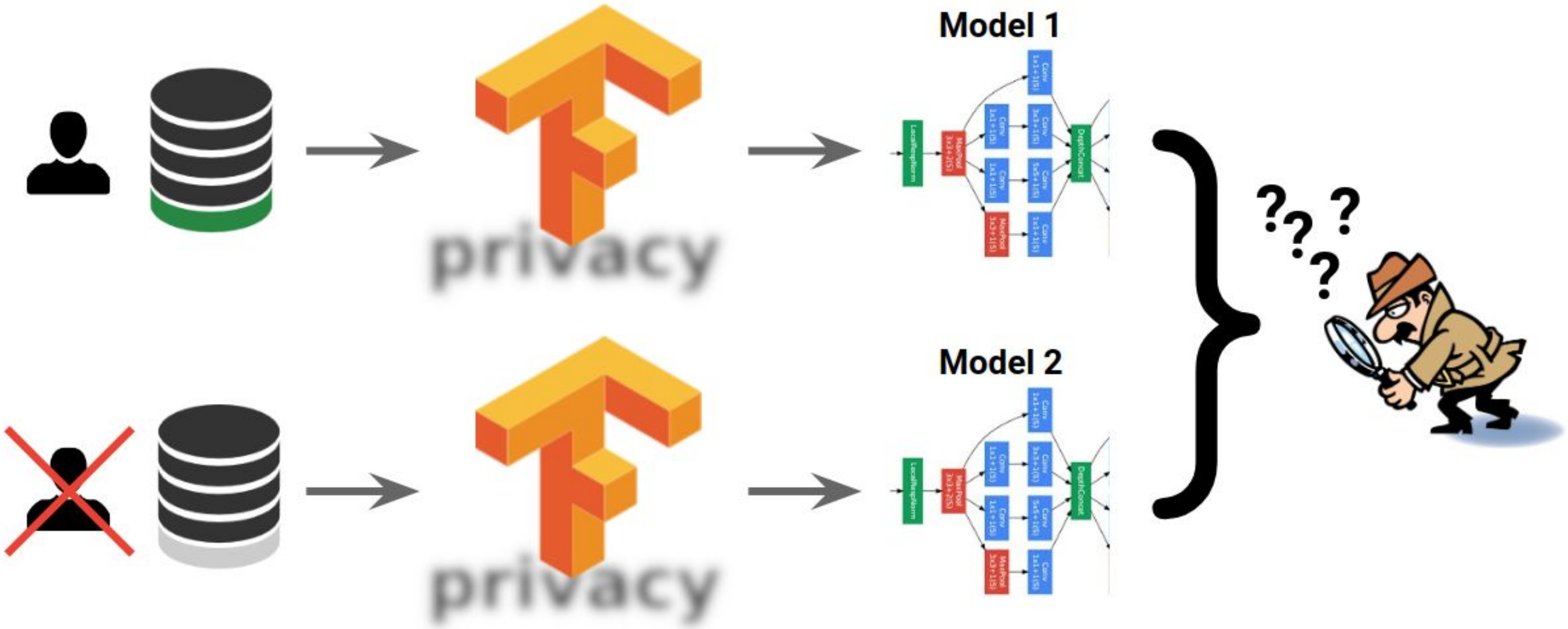
- 2008: U.S. Census Bureau, for showing commuting patterns.[20]
- 2014: Google's RAPPOR, for telemetry such as learning statistics about unwanted software hijacking users' settings [21] (RAPPOR's open-source implementation).
- 2015: Google, for sharing historical traffic statistics.[22]
- 2016: Apple announced its intention to use differential privacy in iOS 10 to improve its Intelligent personal assistant technology.[23]
- 2019: Privitar Lens is an API using differential privacy.[24]

# Google related

- Responsible AI Practices: https://ai.google/responsibilities/responsible-ai-practices/?category=privacy
- RAPPOR: Learning Statistics with Privacy, aided by the Flip of a Coin
- Prochlo: Strong Privacy for Analytics in the Crowd
- Tensorflow Privacy: https://github.com/tensorflow/privacy



# Tensorflow Privacy



## Make SGD Differentially Private

- Limit how much each individual training point sampled in a minibatch can influence the resulting gradient computation.
- Randomize the algorithm's behavior to make it statistically impossible to know whether or not a particular point was included in the training set by comparing the updates stochastic gradient descent applies when it operates with or without this particular point in the training set.

# DP-SGD Algorithm

1. Sample a minibatch of training points `(x, y)`
2. Compute loss `L(theta, x, y).`
3. Compute gradient of the loss `L(theta, x, y)` with respect to the model parameters `theta`.
4. Clip gradients, per training example included in the minibatch, to ensure each gradient has a known maximum Euclidean norm.
5. Add random noise to the clipped gradients.
6. Multiply these clipped and noised gradients by the learning rate and apply the product to update model parameters `theta`.

## DP-SGD with TF Privacy

```
optimizer =
optimizers.dp_optimizer.DPGradientDescentGaussianOptimizer(
    l2_norm_clip=FLAGS.l2_norm_clip,
    noise_multiplier=FLAGS.noise_multiplier,
    num_microbatches=FLAGS.microbatches,
    learning_rate=FLAGS.learning_rate,
    population_size=60000)
 train_op = optimizer.minimize(loss=vector_loss)
```

# MNIST Example

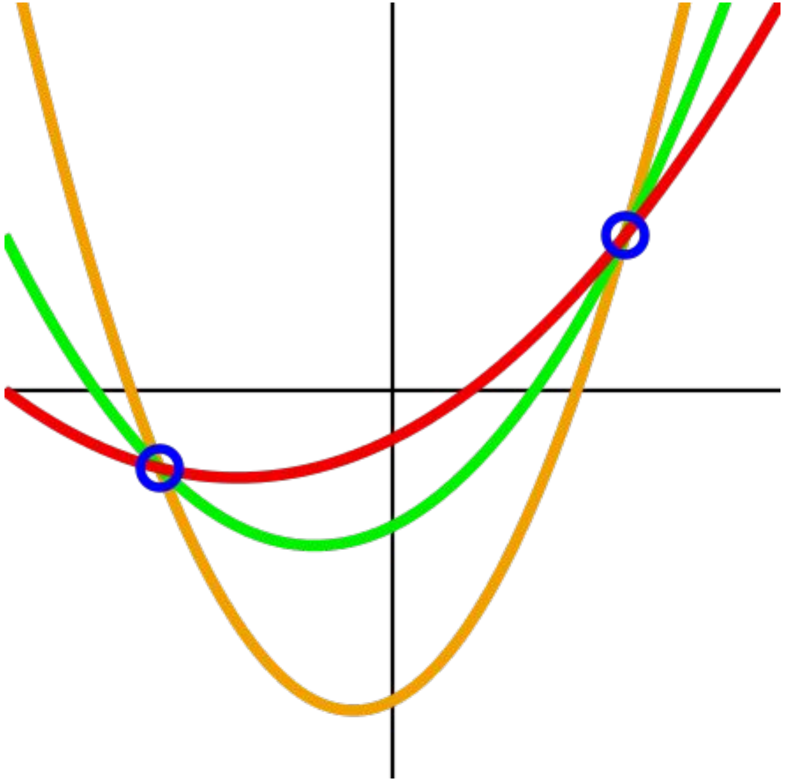
A set of unique outlier digits in MNIST training data. With TensorFlow Privacy, models can learn from such outliers, without memorizing them in any way.

## Secret Sharing

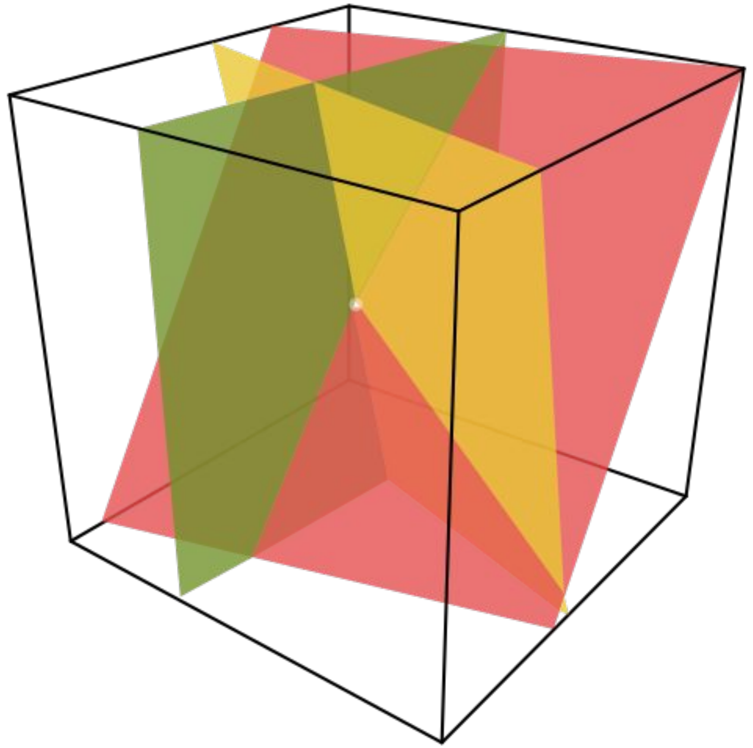
Encode the secret as an arbitrary l

($s$ XOR $p_1$ XOR $p_2$ XOR ... XOR $p_{n-1}$)

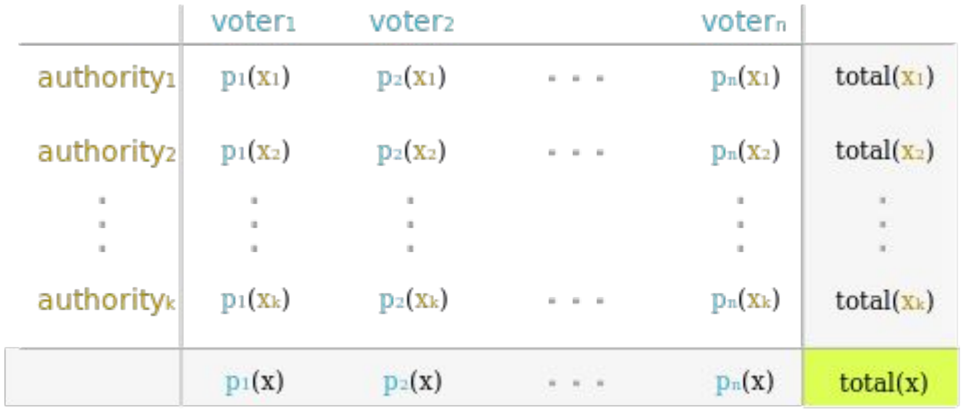
# Shamir’s Secret Sharing



## Blakley's scheme



# Homomorphic secret sharing

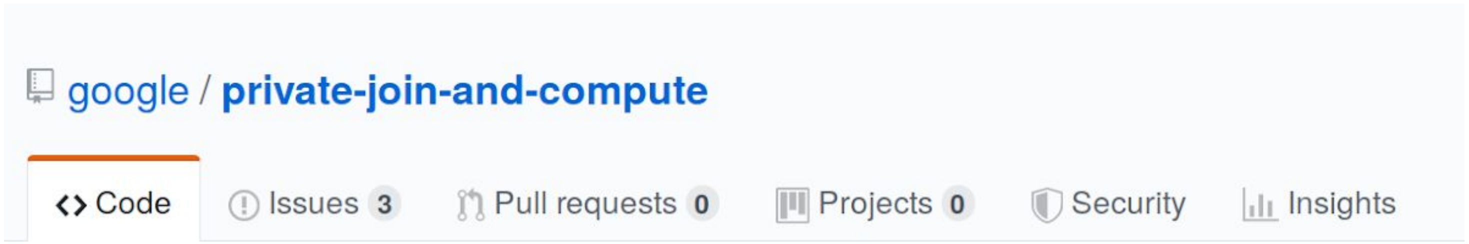


| | $voter_1$ | $voter_2$ | | $voter_n$ | |
|---|---|---|---|---|---|
| $authority_1$ | $p_1(x_1)$ | $p_2(x_1)$ | $\cdots$ | $p_n(x_1)$ | $total(x_1)$ |
| $authority_2$ | $p_1(x_2)$ | $p_2(x_2)$ | $\cdots$ | $p_n(x_2)$ | $total(x_2)$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | $\vdots$ |
| $authority_k$ | $p_1(x_k)$ | $p_2(x_k)$ | $\cdots$ | $p_n(x_k)$ | $total(x_k)$ |
| | $p_1(x)$ | $p_2(x)$ | $\cdots$ | $p_n(x)$ | $total(x)$ |

Add up each row to get a total.

Each column of numbers determines a polynomial.

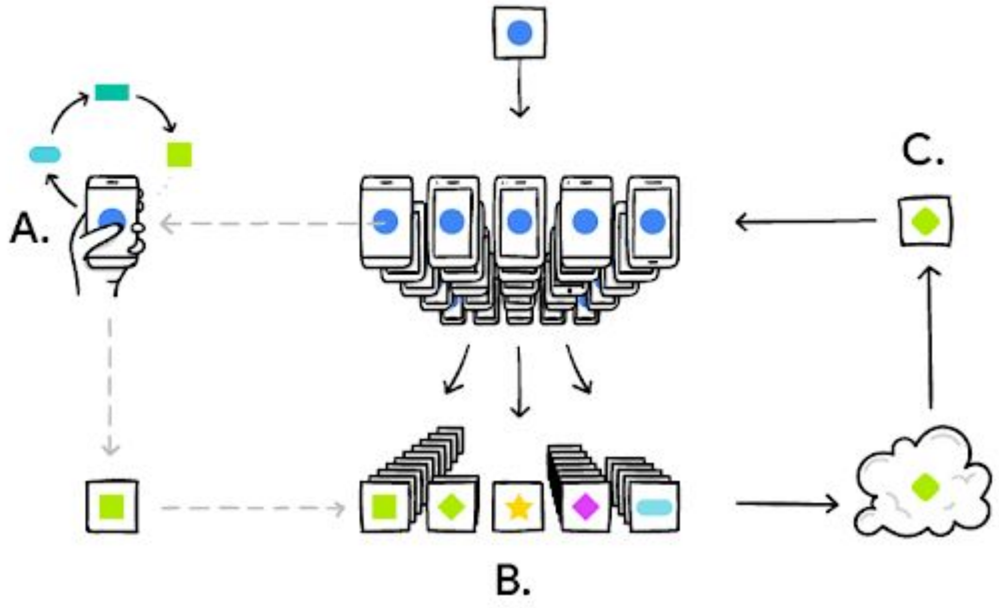
This voting procedure works because you get the same answer whether you (1) Use each column of numbers to determine a polynomial, then add up the polynomials, or (2) Add up the numbers in each row, then use the totals to determine a polynomial.


google / private-join-and-compute

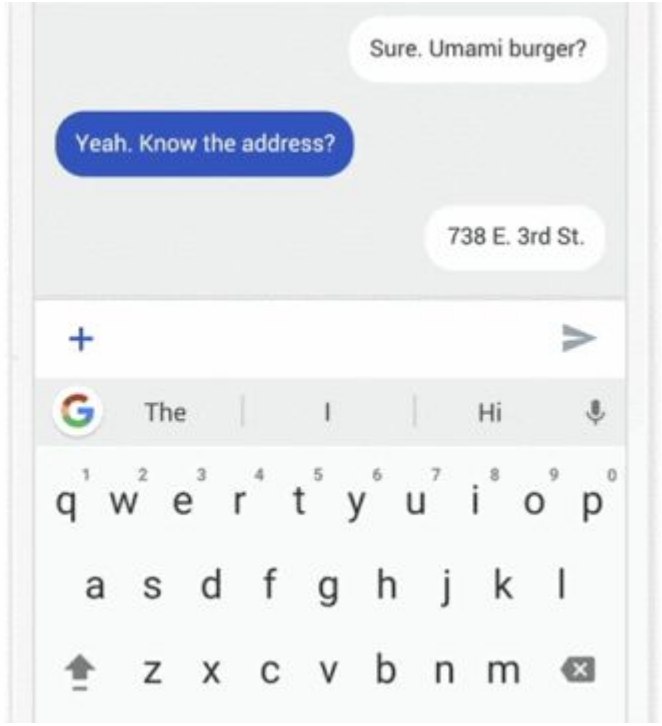
| Identifiers |
|---|
| Sam |
| Ada |
| Ruby |
| Brendan |

| Identifiers | Associated Values |
|---|---|
| Ruby | 10 |
| Ada | 30 |
| Alexander | 5 |
| Mika | 35 |

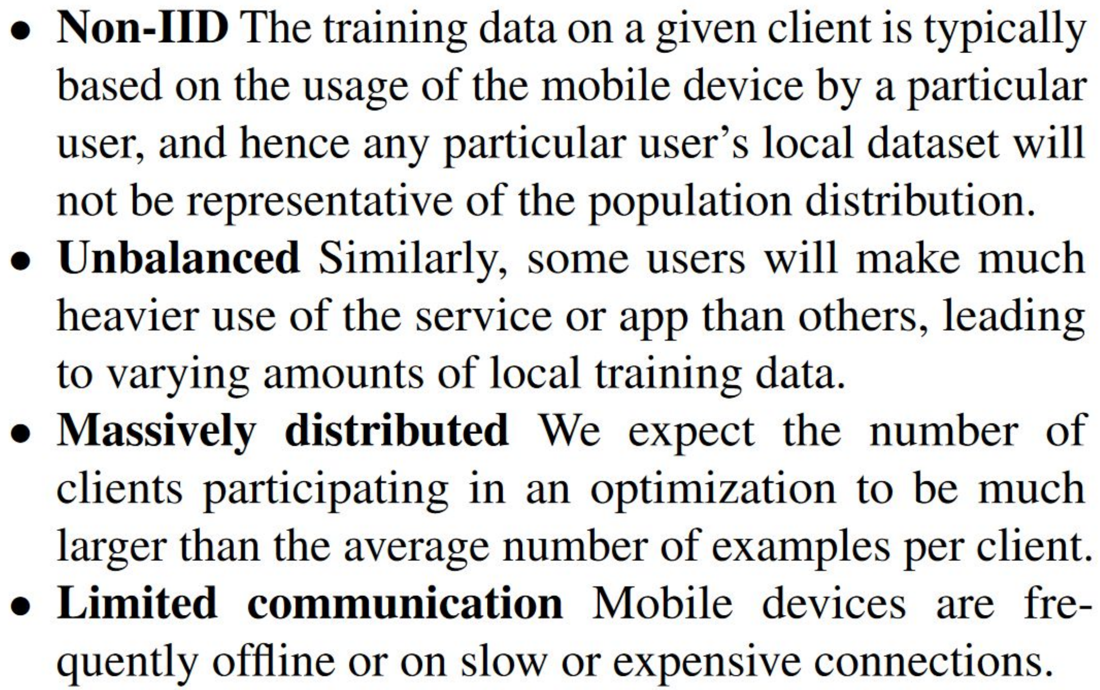
## Federated Learning



# Input Method



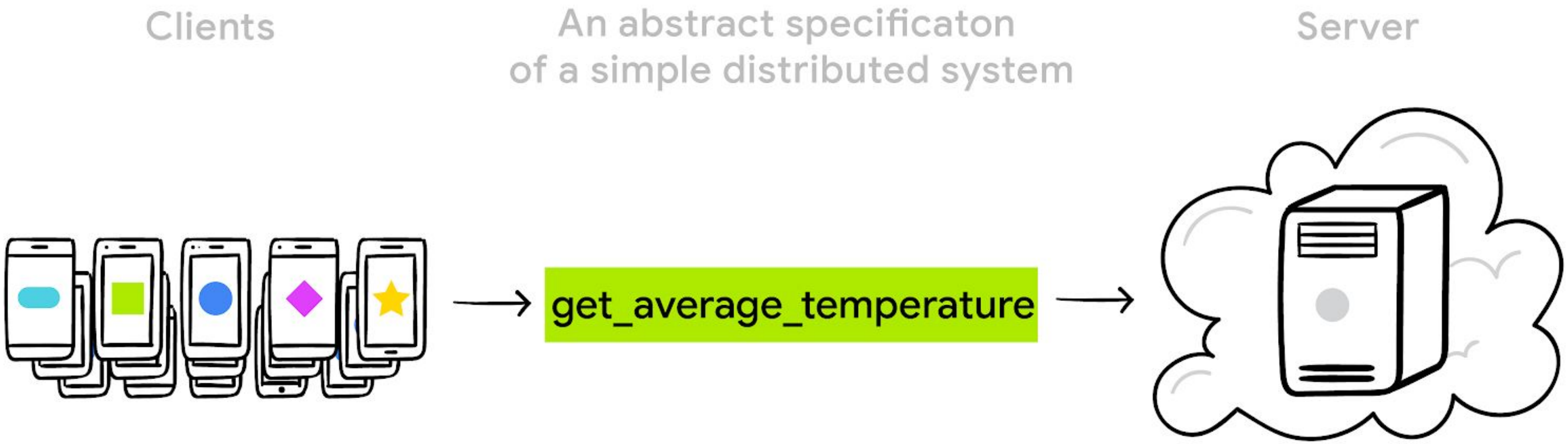
# Federated Optimization



- **Non-IID** The training data on a given client is typically based on the usage of the mobile device by a particular user, and hence any particular user's local dataset will not be representative of the population distribution.
- **Unbalanced** Similarly, some users will make much heavier use of the service or app than others, leading to varying amounts of local training data.
- **Massively distributed** We expect the number of clients participating in an optimization to be much larger than the average number of examples per client.
- **Limited communication** Mobile devices are frequently offline or on slow or expensive connections.

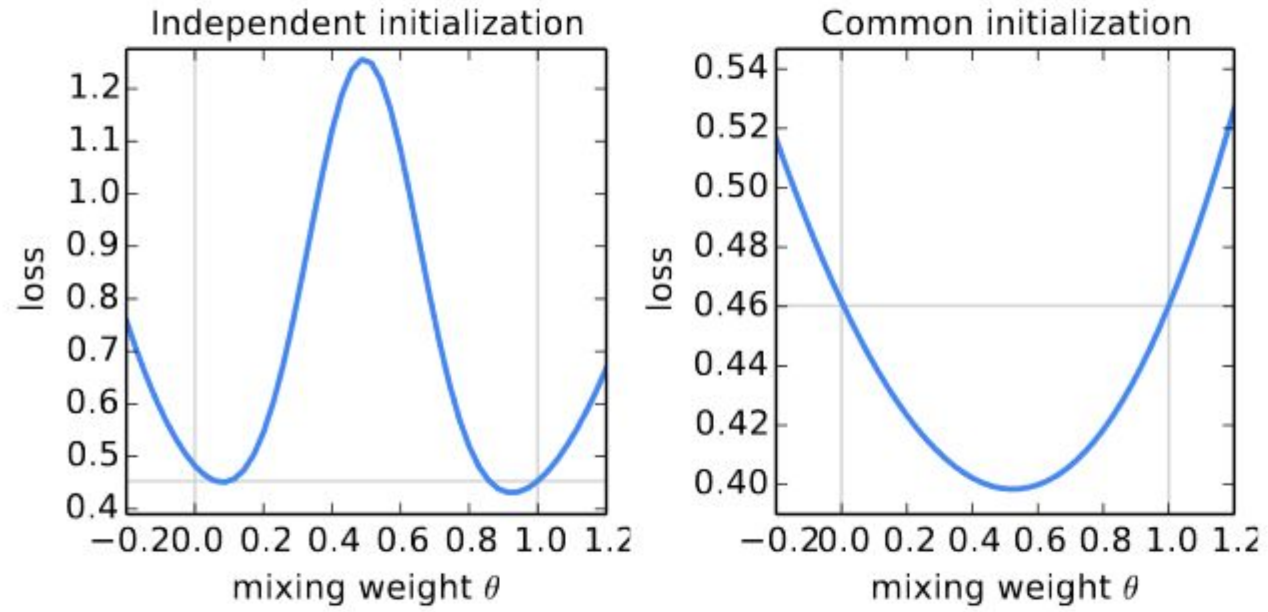
Federated Learning
Building better products with on-device data and privacy by default
An online comic from Google AI

Boss! I'm back from the conference!
*And* I know how we can win back our users!
About time! Our brand is in *shambles.*
Don't worry. I learned all about a new approach that can handle our privacy concerns *and* improve functionality. It's called *federated learning...*
Federated *whatnow?*
It lets us do machine learning while keeping data on-device.
It's resilient, low-impact, secure—
Whatever, *I'M SOLD!*
I'll give you a team of our very best—

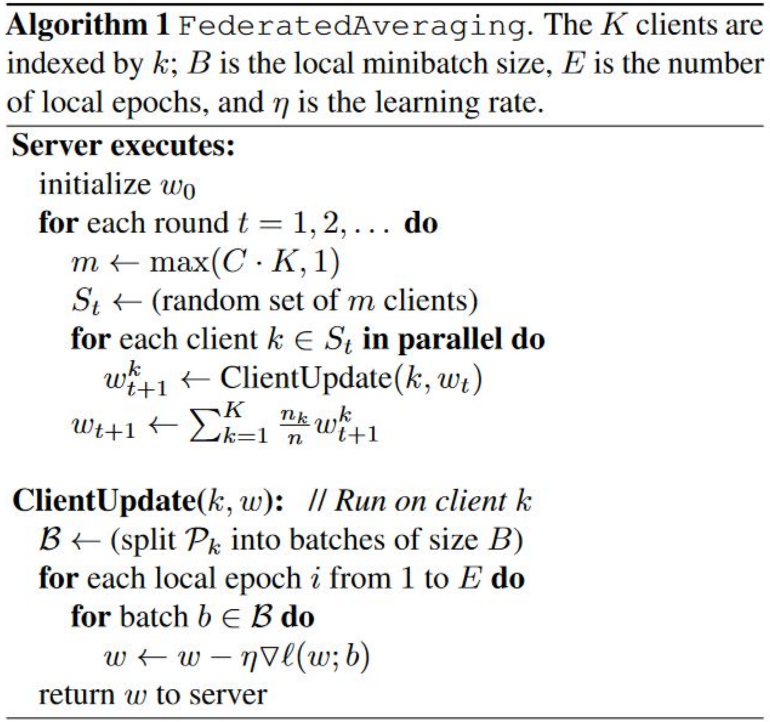


```
{float32}@CLIENTS -> float32@SERVER
```

# Average the Model



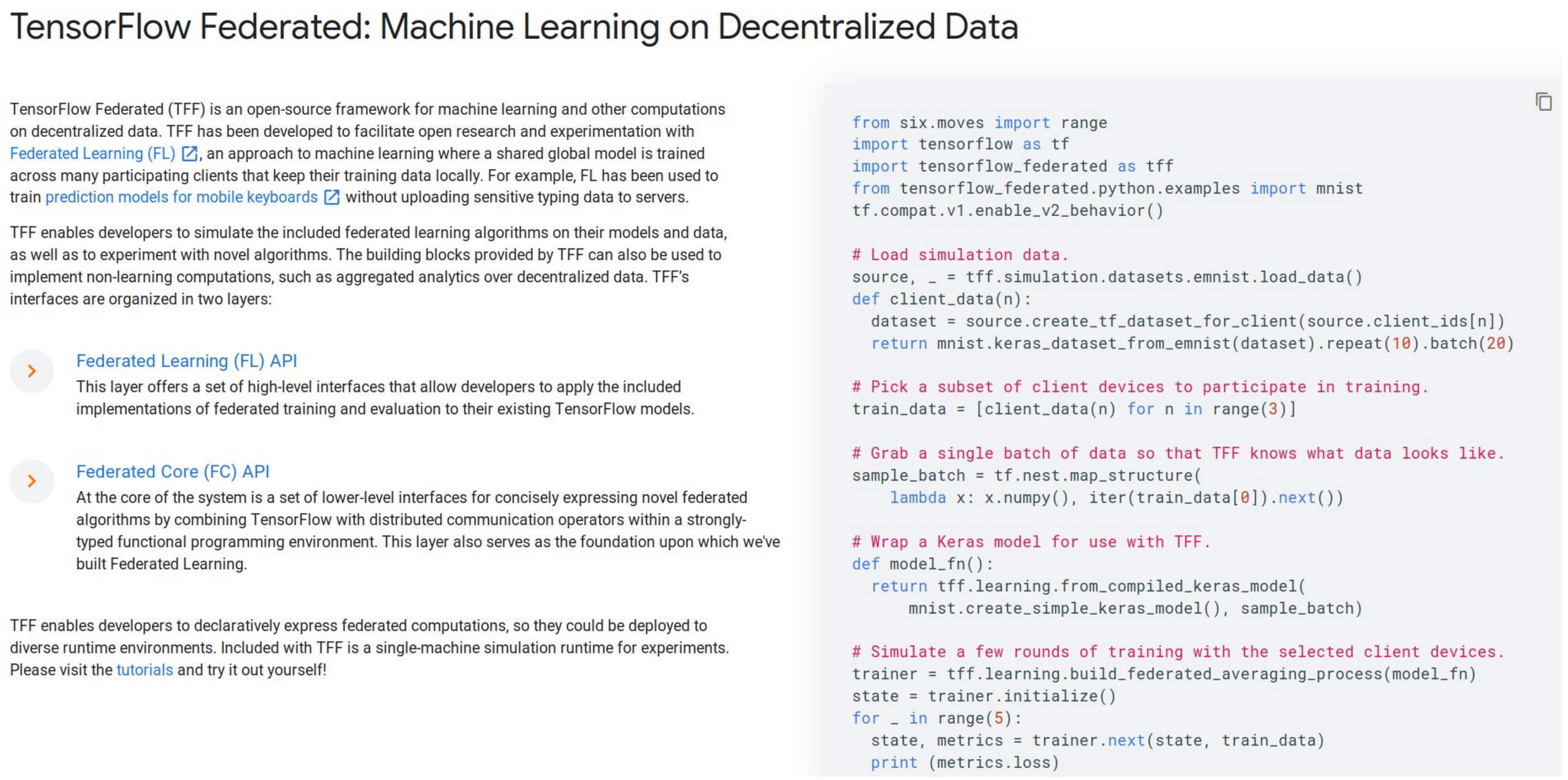
# FedAVG



**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ 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
```



# TensorFlow Federated: Machine Learning on Decentralized Data

TensorFlow Federated (TFF) is an open-source framework for machine learning and other computations on decentralized data. TFF has been developed to facilitate open research and experimentation with Federated Learning (FL), an approach to machine learning where a shared global model is trained across many participating clients that keep their training data locally. For example, FL has been used to train prediction models for mobile keyboards without uploading sensitive typing data to servers.

TFF enables developers to simulate the included federated learning algorithms on their models and data, as well as to experiment with novel algorithms. The building blocks provided by TFF can also be used to implement non-learning computations, such as aggregated analytics over decentralized data. TFF's interfaces are organized in two layers:

### Federated Learning (FL) API

This layer offers a set of high-level interfaces that allow developers to apply the included implementations of federated training and evaluation to their existing TensorFlow models.

### Federated Core (FC) API

At the core of the system is a set of lower-level interfaces for concisely expressing novel federated algorithms by combining TensorFlow with distributed communication operators within a strongly-typed functional programming environment. This layer also serves as the foundation upon which we've built Federated Learning.

TFF enables developers to declaratively express federated computations, so they could be deployed to diverse runtime environments. Included with TFF is a single-machine simulation runtime for experiments. Please visit the tutorials and try it out yourself!

```
from six.moves import range
import tensorflow as tf
import tensorflow_federated as tff
from tensorflow_federated.python.examples import mnist
tf.compat.v1.enable_v2_behavior()

# Load simulation data.
source, _ = tff.simulation.datasets.emnist.load_data()
def client_data(n):
  dataset = source.create_tf_dataset_for_client(source.client_ids[n])
  return mnist.keras_dataset_from_emnist(dataset).repeat(10).batch(20)

# Pick a subset of client devices to participate in training.
train_data = [client_data(n) for n in range(3)]

# Grab a single batch of data so that TFF knows what data looks like.
sample_batch = tf.nest.map_structure(
    lambda x: x.numpy(), iter(train_data[0]).next())

# Wrap a Keras model for use with TFF.
def model_fn():
  return tff.learning.from_compiled_keras_model(
      mnist.create_simple_keras_model(), sample_batch)

# Simulate a few rounds of training with the selected client devices.
trainer = tff.learning.build_federated_averaging_process(model_fn)
state = trainer.initialize()
for _ in range(5):
  state, metrics = trainer.next(state, train_data)
  print (metrics.loss)
```

## Notes on Secure Computing and Protocols

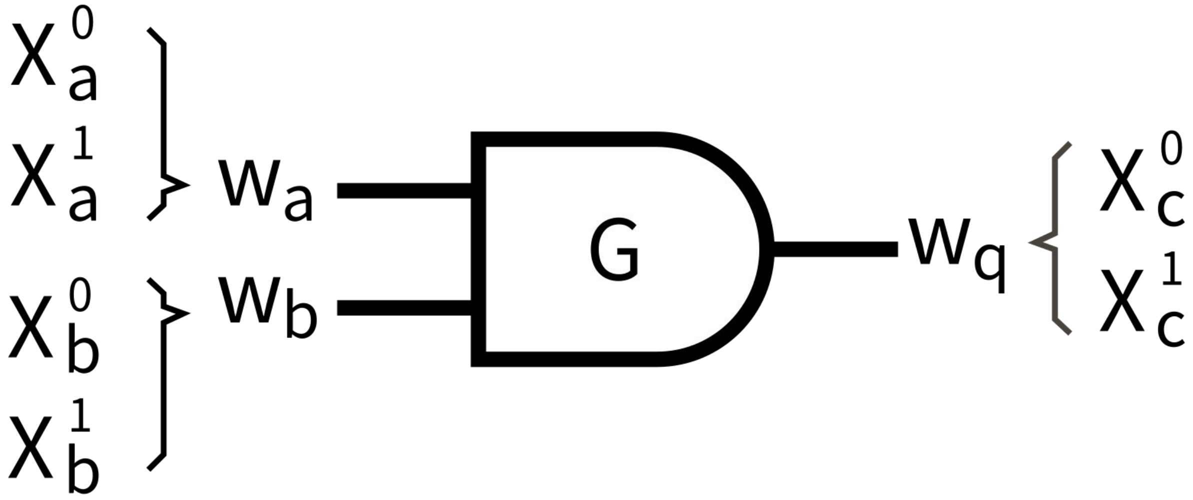
## Yao's Millionaires' Problem

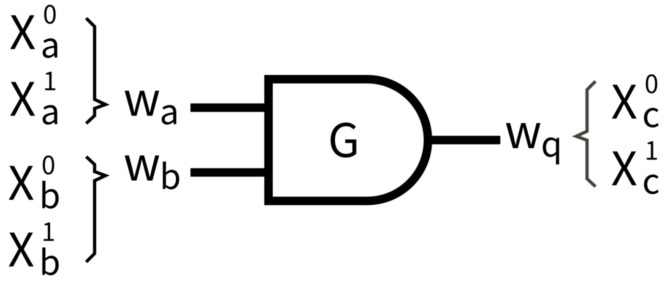
- Introduced in 1982 by Andrew Yao
- Two millionaires, Alice and Bob, who are interested in knowing which of them is richer without revealing their actual wealth.
- Socialist millionaire, a variant in which the millionaires wish to determine whether their fortunes are equal.
- Original protocol works like this:
  - Bob send E(x)-j+1 to Alice
  - Alice compute y_u = D(E(x)-j+u) +(u>=i) for u=1..10
  - Bob checks whether y_j = x
- http://twistedoakstudios.com/blog/Post3724_explain-it-like-im-five-the-socialist-millionaire-problem-and-secure-multi-party-computation

## Oblivious Transfer

- Pick one out of two, without revealing which
- A protocol
  - Random x0, x1
  - Bob choose x from x0, x1 and send x+E(k) to Alice
  - Alice send M0+D(x+E(k)-x0), M1+D(x+E(k)-x1)  to Bob
  - Bob can recover one of M0, M1

## Garbled circuit protocol





| b \ a | **0** | **1** |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |

$\rightarrow$

| b \ a | $X_a^0$ | $X_a^1$ |
|---|---|---|
| $X_b^0$ | $X_c^0$ | $X_c^0$ |
| $X_b^1$ | $X_c^0$ | $X_c^1$ |

$\rightarrow$

| b \ a | $X_a^0$ | $X_a^1$ |
|---|---|---|
| $X_b^0$ | $\mathbf{E}_{X_a^0,X_b^0}(X_c^0)$ | $\mathbf{E}_{X_a^1,X_b^0}(X_c^0)$ |
| $X_b^1$ | $\mathbf{E}_{X_a^0,X_b^1}(X_c^0)$ | $\mathbf{E}_{X_a^1,X_b^1}(X_c^1)$ |

- Semi-Honest (Passive) Security
- Malicious (Active) Security

例如打牌作弊：
Passive：聯手合作
Active：藏牌

SPDZ: A malicious adversary will get caught with probability 1-1/c

前面Sharing secret就是依據這個

## Zero-Knowledge Proof

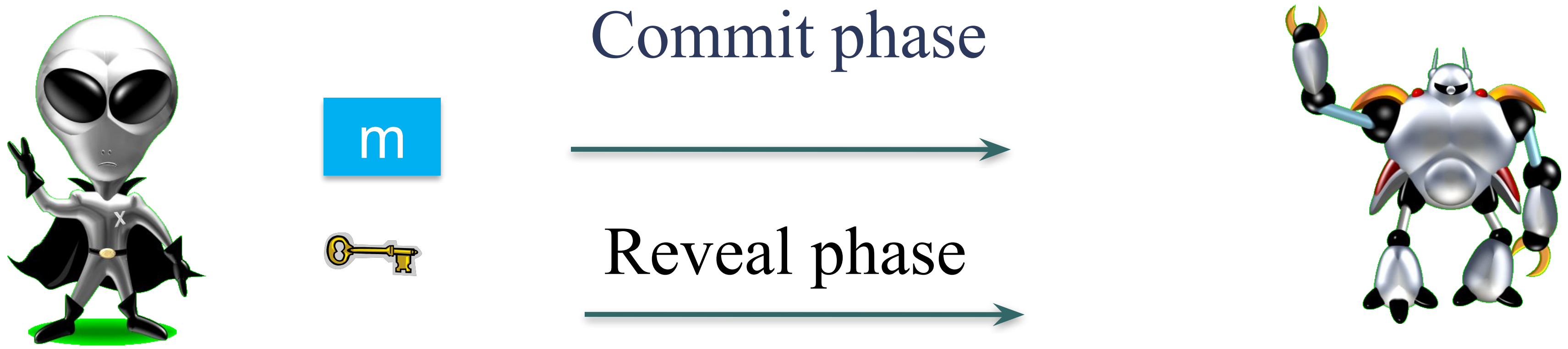
區塊鏈、密碼學會用到這個概念

bitrhday Problem : Hash collision問題

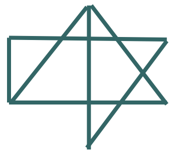
[https://docs.google.com/presentation/d/1y-1EKAUDKNF2il-a32r1OOyaSnbXf6Hb/present?token=AC4w5Vg3kB_T_QqAjWgfJT1EVzct2Dn4rA%3A1569422134484&includes_info_params=1&eisi=CPeA0YaZ7OQCFZVFJAodNw8Jeg#slide=id.p23](https://docs.google.com/presentation/d/1y-1EKAUDKNF2il-a32r1OOyaSnbXf6Hb/present?token=AC4w5Vg3kB_T_QqAjWgfJT1EVzct2Dn4rA%3A1569422134484&includes_info_params=1&eisi=CPeA0YaZ7OQCFZVFJAodNw8Jeg#slide=id.p23)

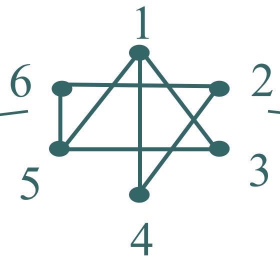
# How?

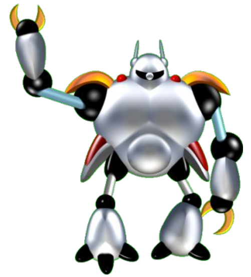
## Commitment Scheme

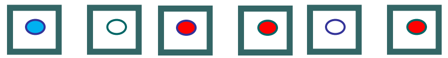


# ZKP for GRAPH 3-COLORING [GMW86]









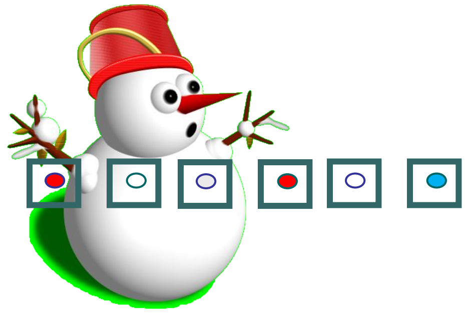
1. Permute colors & commit them. →

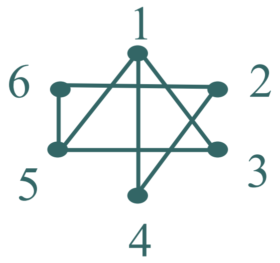
← (1,4) 2. Pick random edge.

3. Send keys for endpoints. ( $K_1$, $K_4$) →

4. Accept if colors different. ✔

# Simulation





1. Commit a random coloring.

$$\xrightarrow{\hspace{6cm}}$$

2. If the edge is bad, go to 1. $\xleftarrow{\hspace{6cm}}$ ( i, j)

3. Send keys for endpoints. $\xrightarrow{\hspace{6cm}}$ ✔

( $K_i$, $K_j$)