Privacy-Preserving Machine Learning in TensorFlow with TF Encrypted

Morten Dahl

DropoutLabs

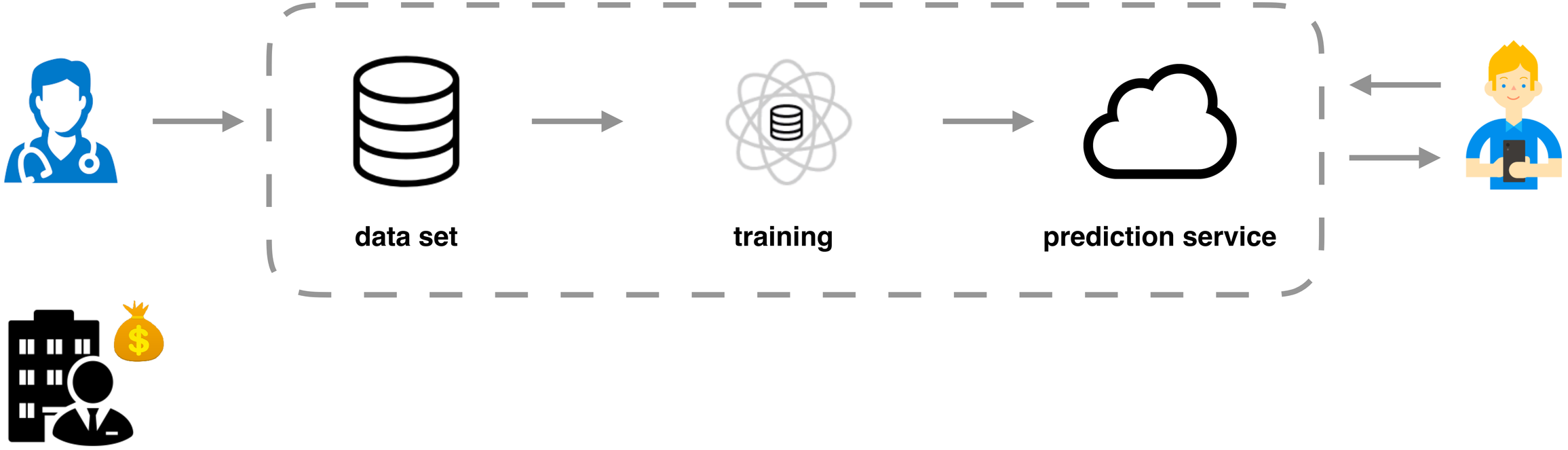
O'Reilly AI Conference, New York, April 2019

Why?

Privacy in machine learning

Machine Learning Process

IMAGENET



DropoutLabs

Skin Cancer Image Classification

Brett Kuprel

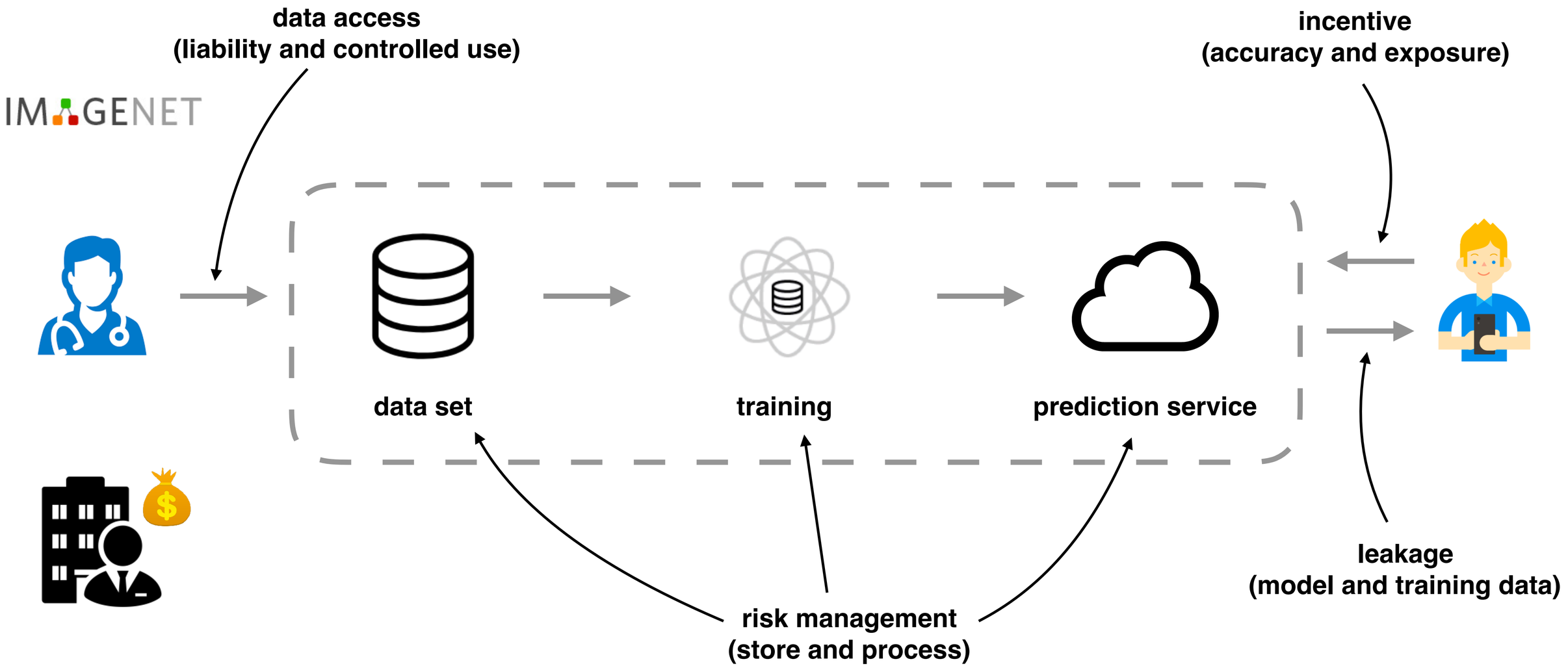
12:30-12:40pm

Join Brett Kuprel, and see how TensorFlow was used by the artificial intelligence lab and medical school of Stanford to classify skin cancer images. He'll describe the project steps: from acquiring a dataset, training a deep network, and evaluating of the results. To wrap up, Brett will give his take on the future of skin cancer image classification.

machine learning positioned to have huge impact on health care

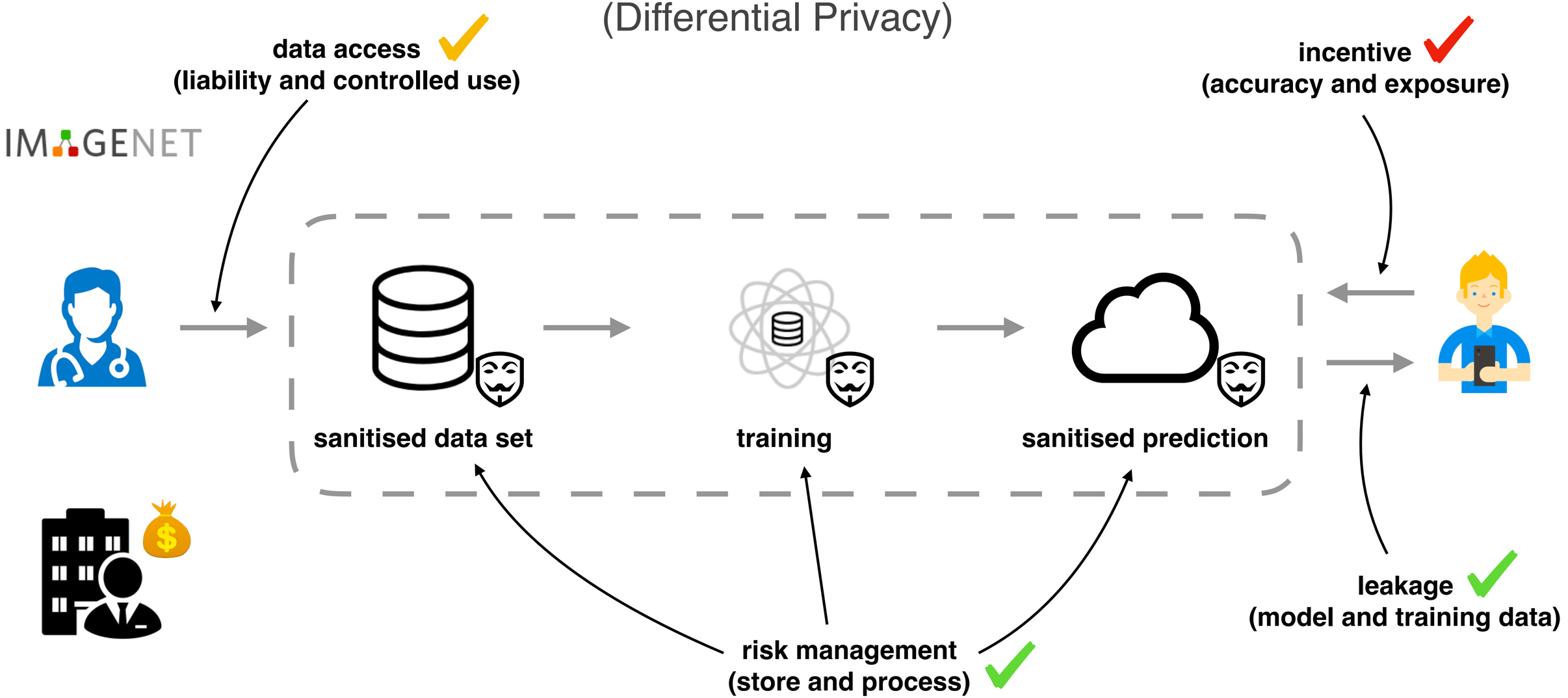
DropoutLabs

Potential Bottlenecks



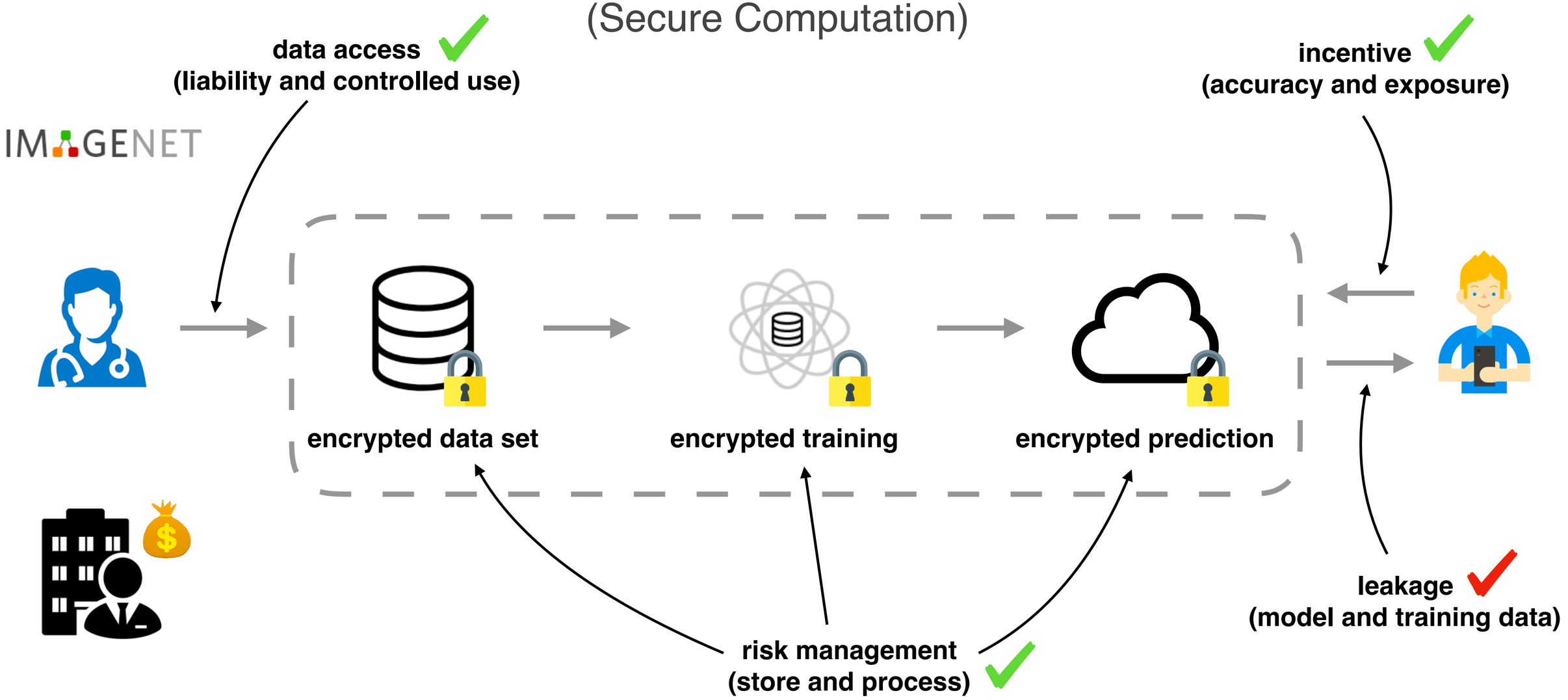
Sanitization

(Differential Privacy)



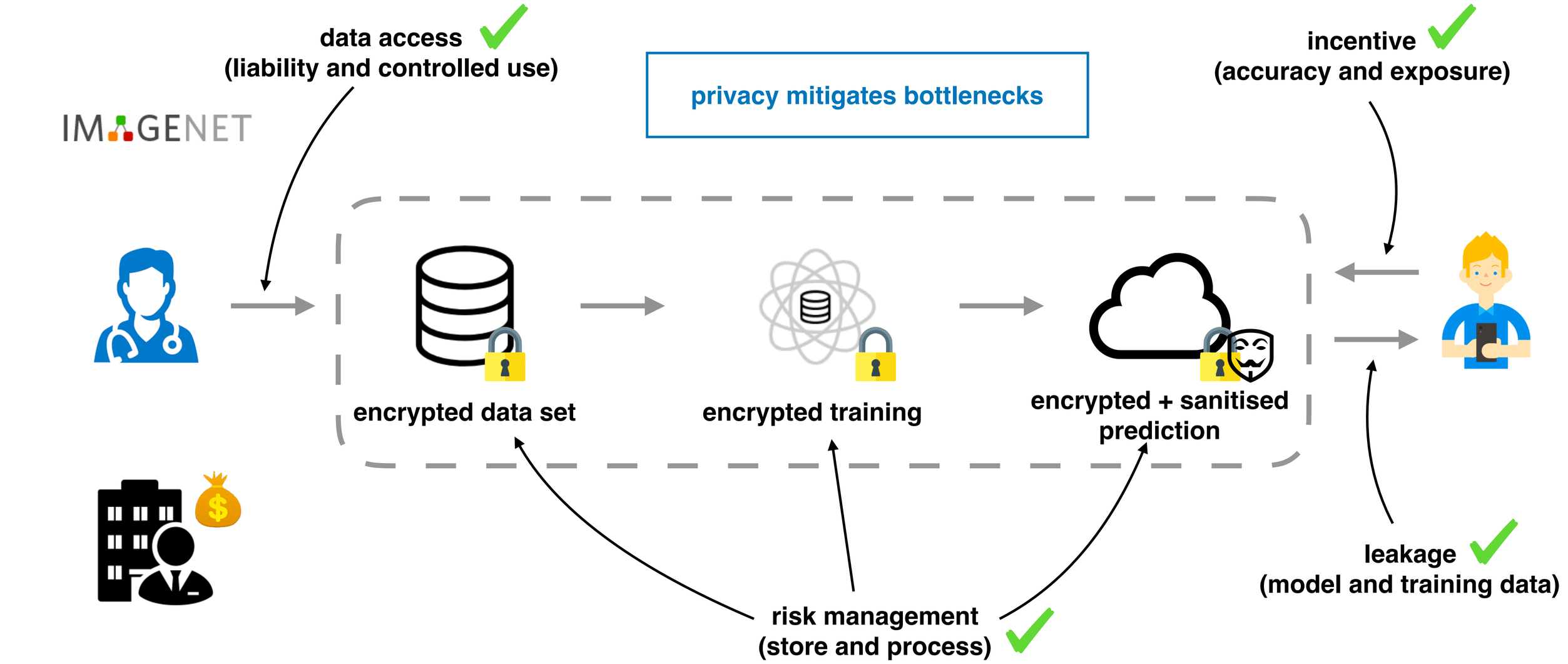
Encryption

(Secure Computation)



DropoutLabs

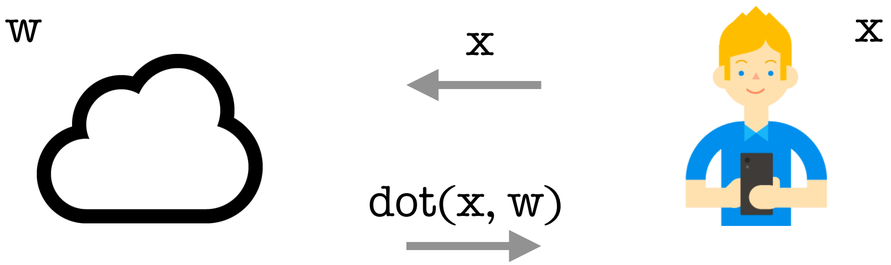
Hybrid


data access
(liability and controlled use)
IMAGENET
privacy mitigates bottlenecks
incentive
(accuracy and exposure)
encrypted data set
encrypted training
encrypted + sanitised
prediction
leakage
(model and training data)
risk management
(store and process)

How?

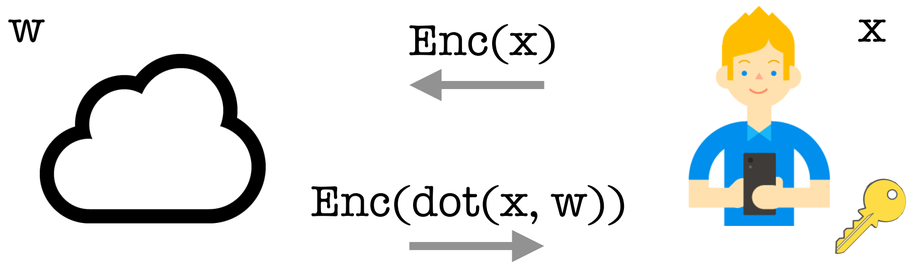
Computing on encrypted data

Prediction with Linear Model



$$\text{dot}\left(\begin{bmatrix} x1 & x2 & x3 \end{bmatrix}, \begin{bmatrix} w1 \\ w2 \\ w3 \end{bmatrix}\right) = x1*w1 + x2*w2 + x3*w3$$

… using Homomorphic Encryption



homomorphic encryption scheme

public multiplication

private addition

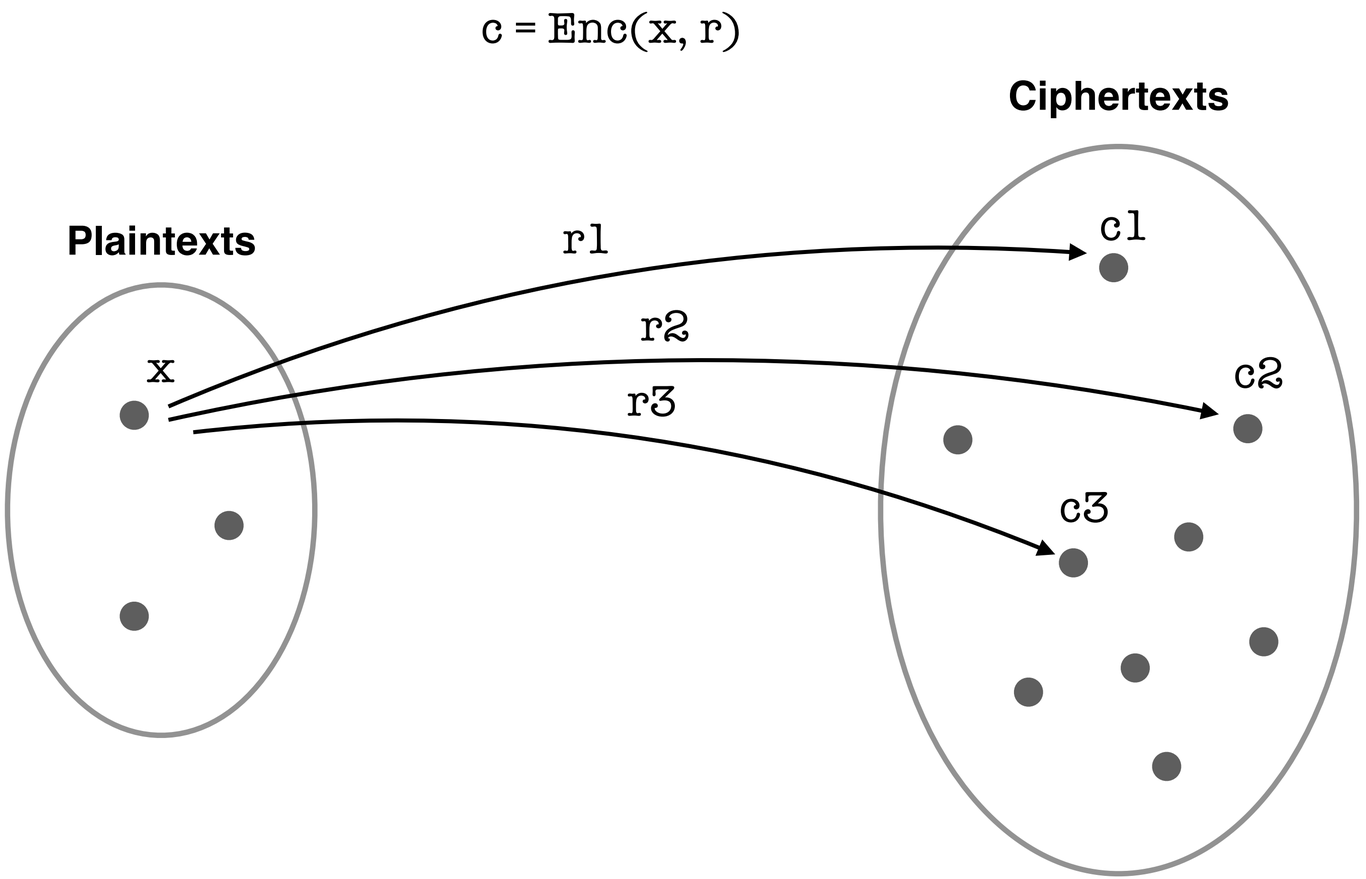
dot(Enc(x1) Enc(x2) Enc(x3) , w1 w2 w3) = Enc(x1)*w1 + Enc(x2)*w2 + Enc(x3)*w3

= Enc(x1*w1) + Enc(x2*w2) + Enc(x3*w3)

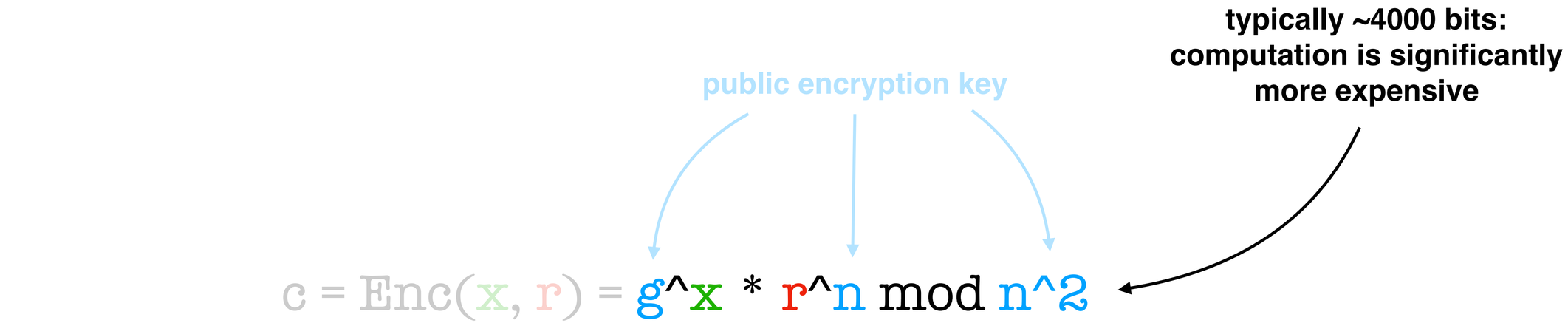
= Enc(x1*w1 + x2*w2 + x3*w3)

DropoutLabs

Probabilistic Encryption



Paillier Homomorphic Encryption



g = 36
n = 35
n^2 = 1225

Enc(5, 2) = 36^5 * 2^35 mod 1225 = 718

Enc(5, 4) = 36^5 * 4^35 mod 1225 = 674

Private Addition in Paillier

$$Enc(x, r) * Enc(y, s)$$

$$= (g^x * r^n \bmod n^2) * (g^y * s^n \bmod n^2)$$

$$= g^{(x+y)} * (r * s)^n \bmod n^2$$

$$= Enc(x+y, r*s)$$

$$Enc(5, 2) * Enc(5, 4)$$
$$= 718 * 674$$
$$= 57$$
$$= 36^{10} * 8^{35}$$
$$= Enc(10, 8)$$

DropoutLabs

Public Multiplication in Paillier

$$\text{Enc}(x, r) \hat{} w$$

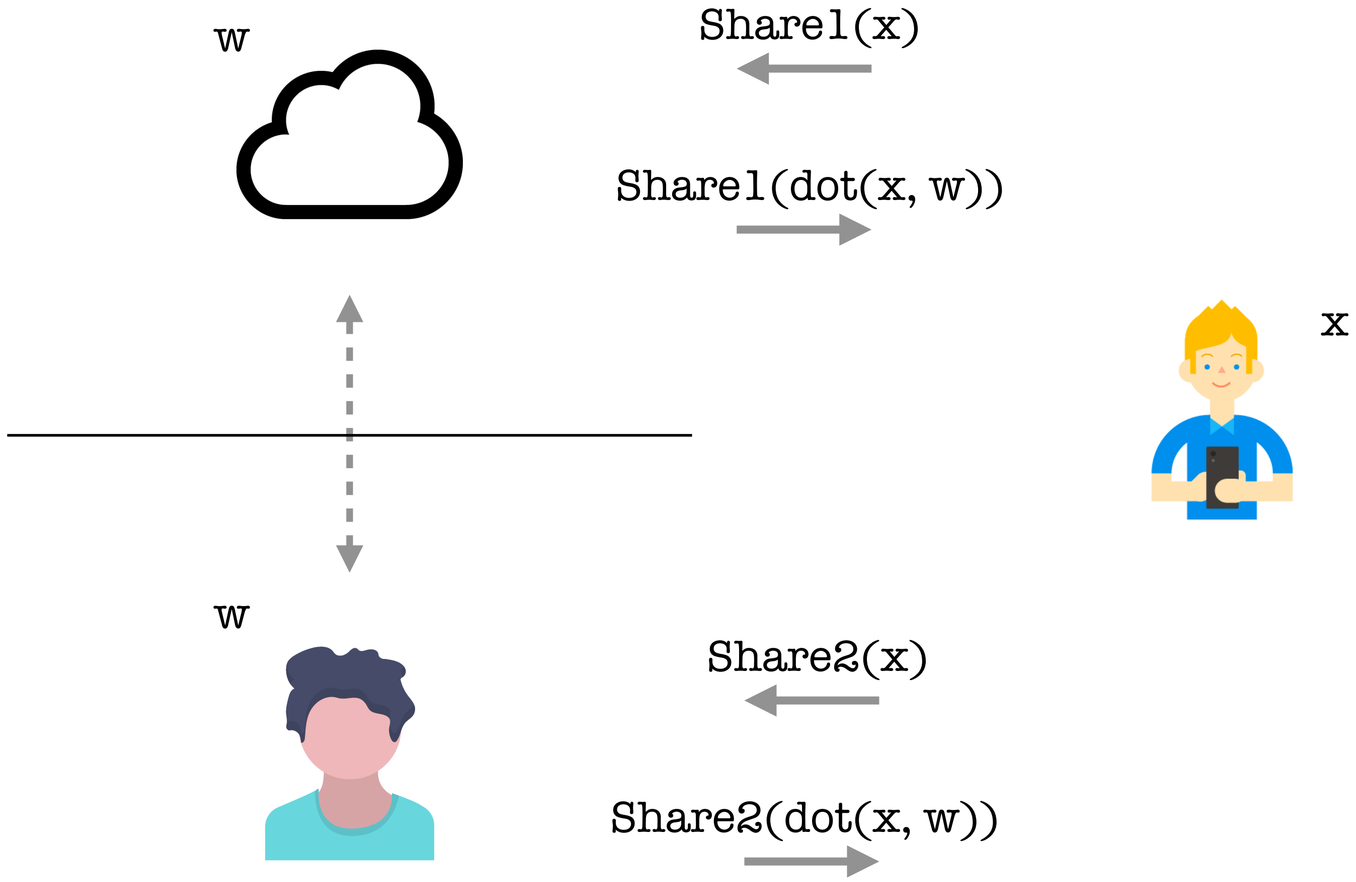
$$= (g \hat{} x * r \hat{} n \bmod n \hat{} 2) \hat{} w$$

$$= g \hat{} (x * w) * (r \hat{} w) \hat{} n \bmod n \hat{} 2$$

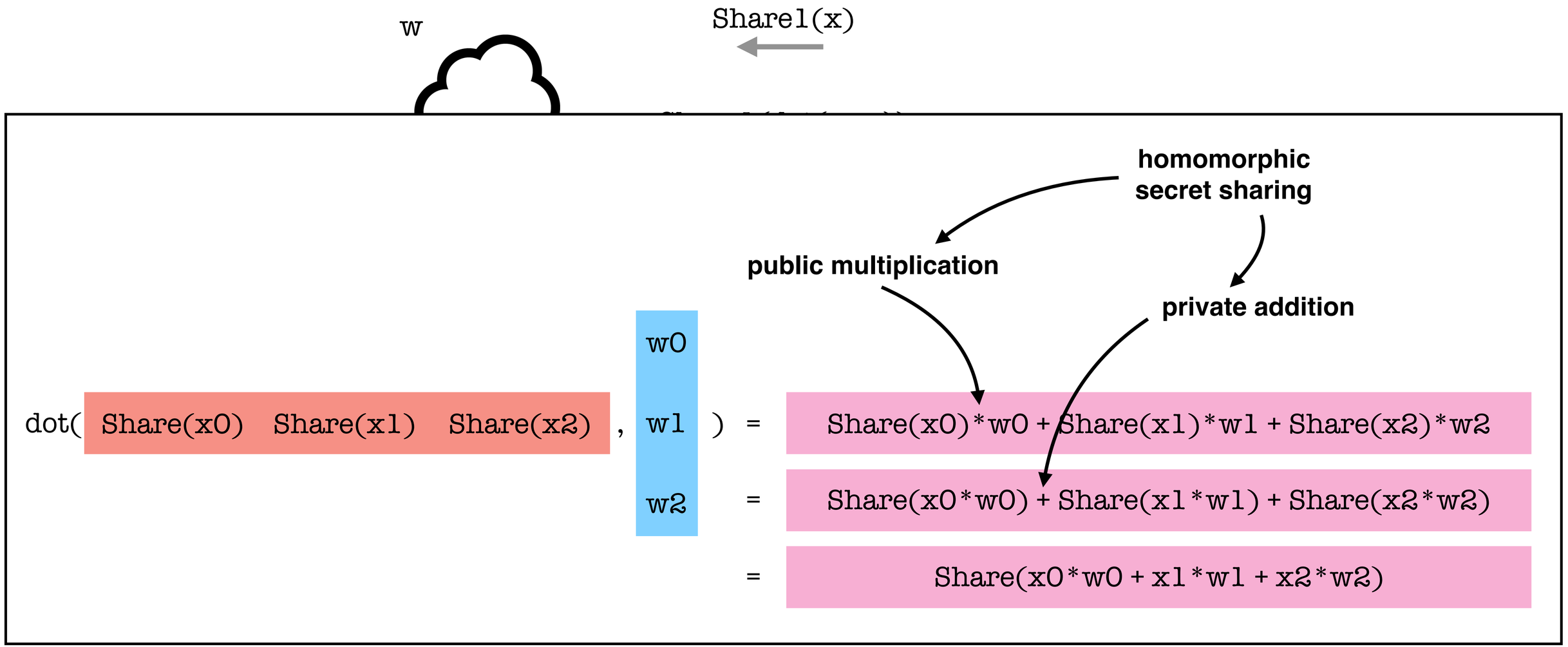
$$= \text{Enc}(x * w, r \hat{} w)$$

$$\text{Enc}(5, 2) \hat{} 2$$
$$= 718 * 718$$
$$= 1024$$
$$= 36 \hat{} 10 * 4 \hat{} 35$$
$$= \text{Enc}(10, 4)$$

… using Secret Sharing



… using Secret Sharing



Secret Sharing

$$\text{Share1}(x, r) = r \bmod m$$

$$\text{Share2}(x, r) = x - r \bmod m$$

$$x = \text{Share1}(x, r) + \text{Share2}(x, r) \bmod m$$

$m = 10$

$$\text{Share1}(5, 7) = 7 \bmod 10 = 7$$

$$\text{Share2}(5, 7) = 5 - 7 \bmod 10 = 8$$

$$7 + 8 = 15 = 5 \bmod 10$$

Private Addition with Secret Sharing

x_1 y_1 $z_1 = x_1 + y_1$

x_2 y_2 $z_2 = x_2 + y_2$

$x = x_1 + x_2$ $y = y_1 + y_2$

$$
\begin{aligned}
x + y &= (x_1 + x_2) + (y_1 + y_2) \\
&= (x_1 + y_1) + (x_2 + y_2) \\
&= z_1 + z_2
\end{aligned}
$$

Public Multiplication with Secret Sharing

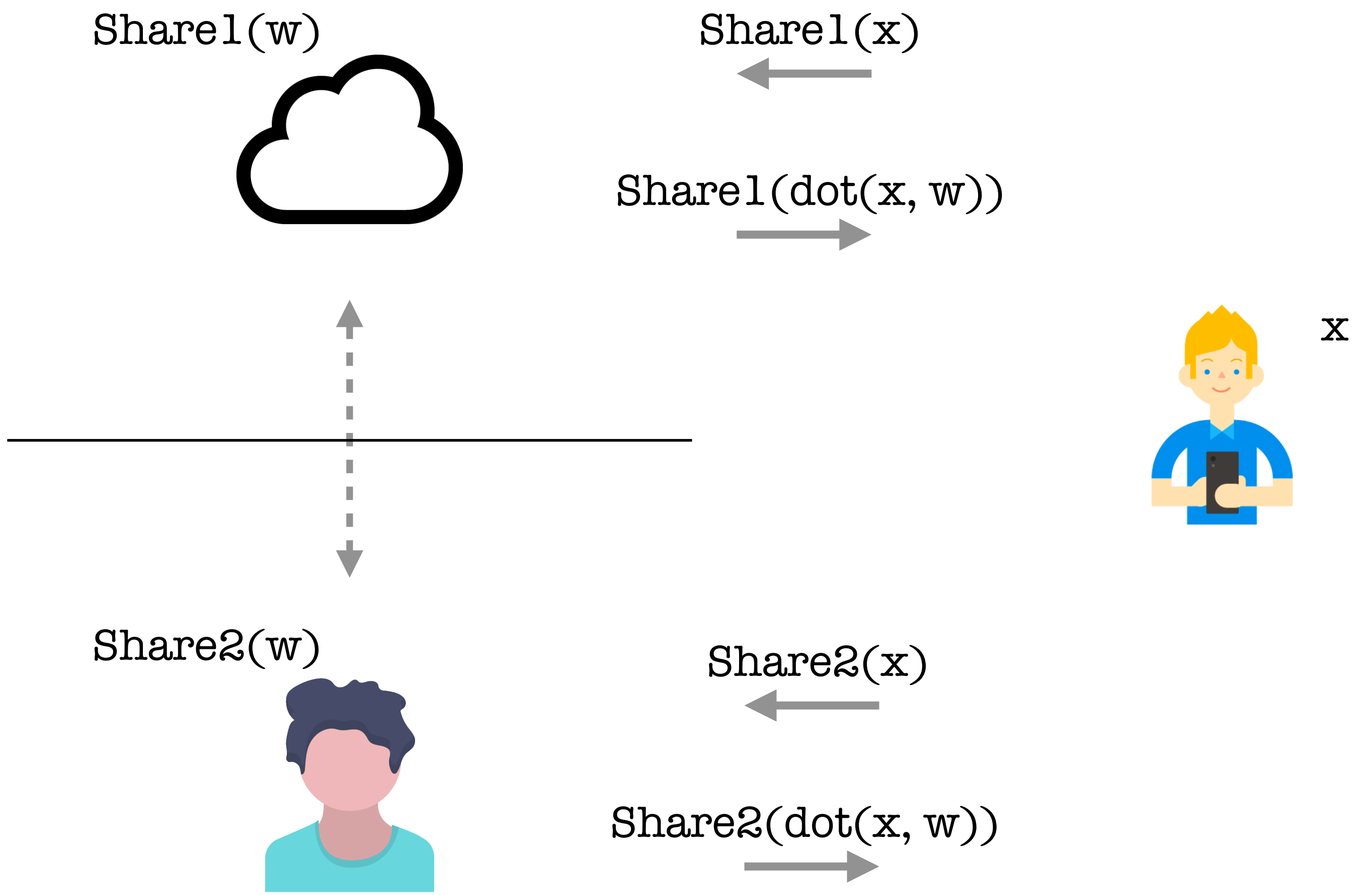
x_1 w $z_1 = x_1 * w$

x_2 w $z_2 = x_2 * w$

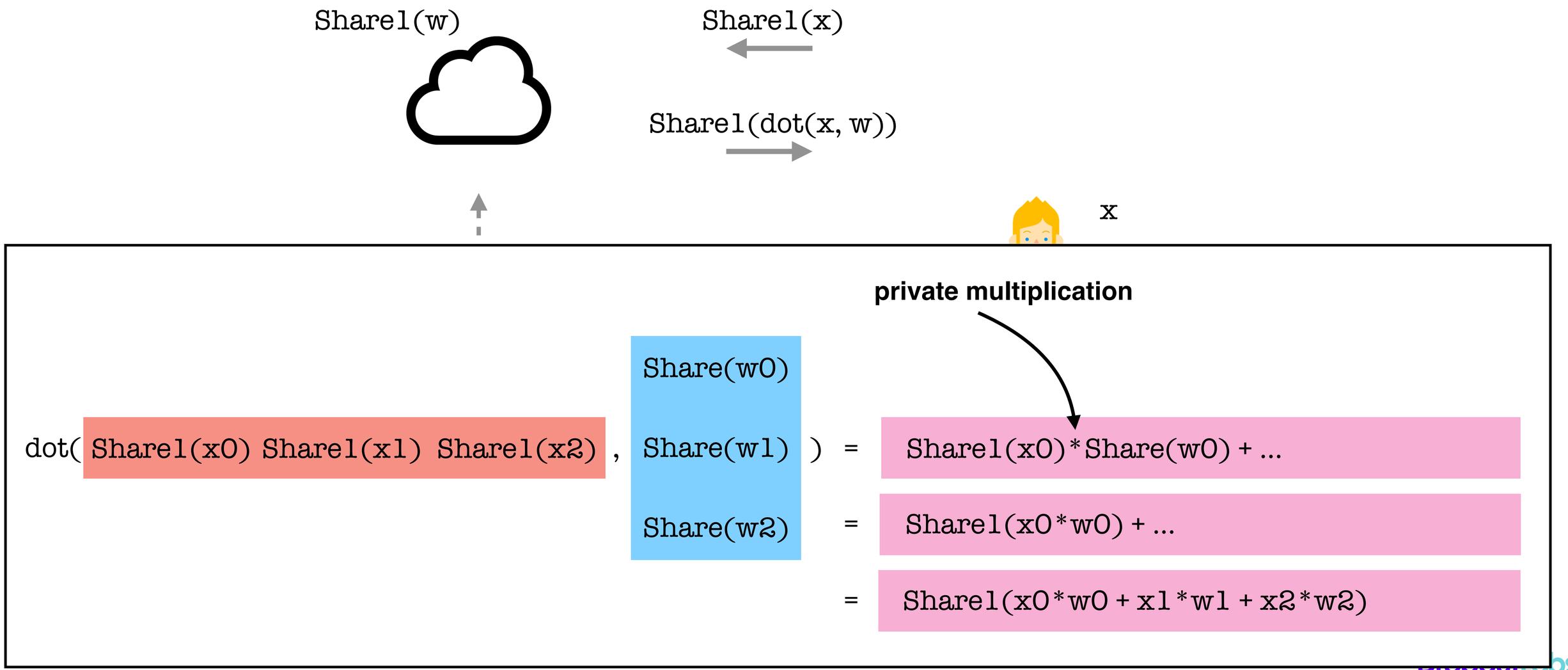
$x = x_1 + x_2$

$$
\begin{aligned}
x * w &= (x_1 + x_2) * w \\
&= (x_1 * w) + (x_2 * w) \\
&= z_1 + z_2
\end{aligned}
$$

… using Secret Sharing, with Private Model



… using Secret Sharing, with Private Model



Private Multiplication with Secret Sharing

(a1 , b1 , c1) x1 y1 alpha beta

z1 = alpha*beta
+ alpha*b1
+ beta*a1
+ c1

(a2 , b2 , c2) x2 y2 alpha beta

z2 = alpha*b2
+ beta*a2
+ c2

a = a1 + a2
b = b1 + b2
c = a*b = c1 + c2

x = x1 + x2

y = y1 + y2

alpha = x - a

beta = y - b

x * y
= ...
= z1 + z2

Multidisciplinary Challenge

Data science
(use-cases, workflow, monitoring)

Cryptography
(techniques, protocols, trust)

Machine learning
(models, approx, precision)

Engineering
(distributed, multi-core, readability)

need common language

DropoutLabs

TF Encrypted

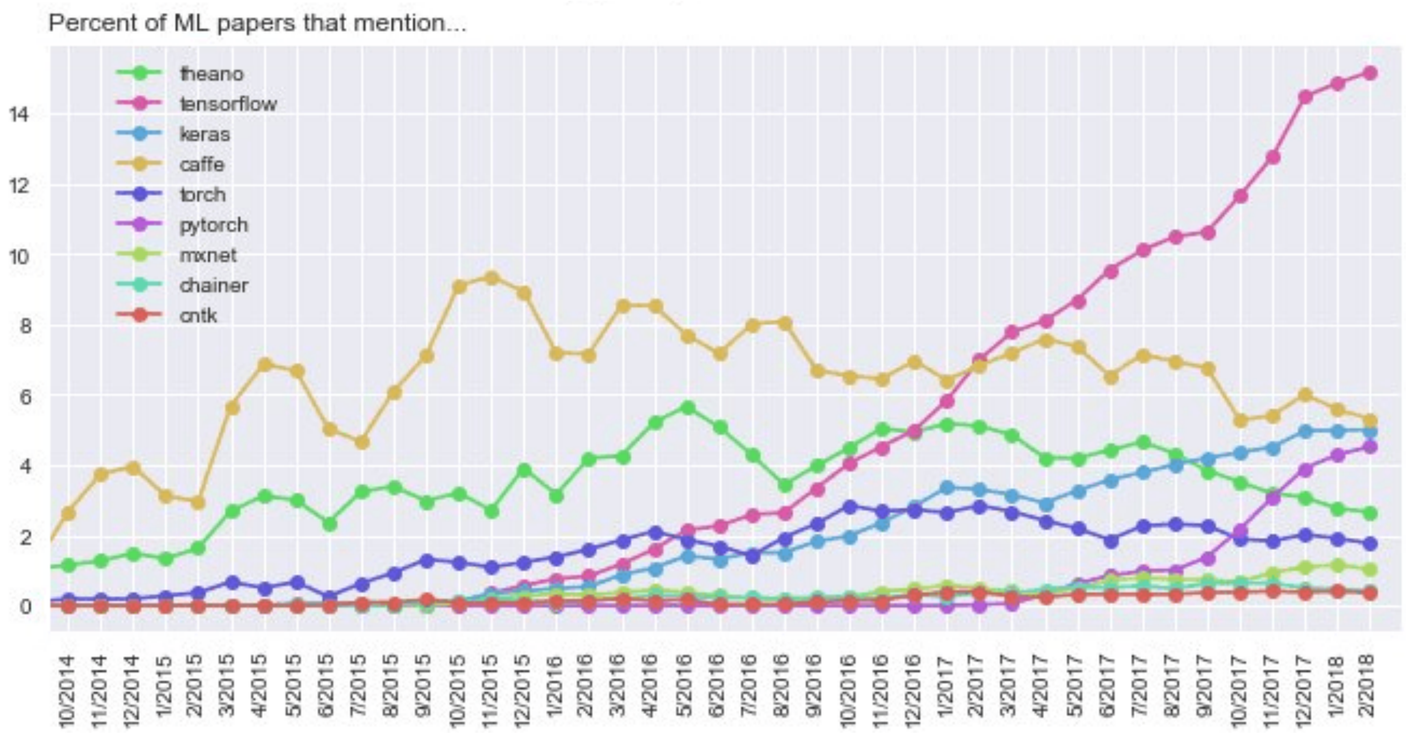
Making it accessible

DropoutLabs

TensorFlow

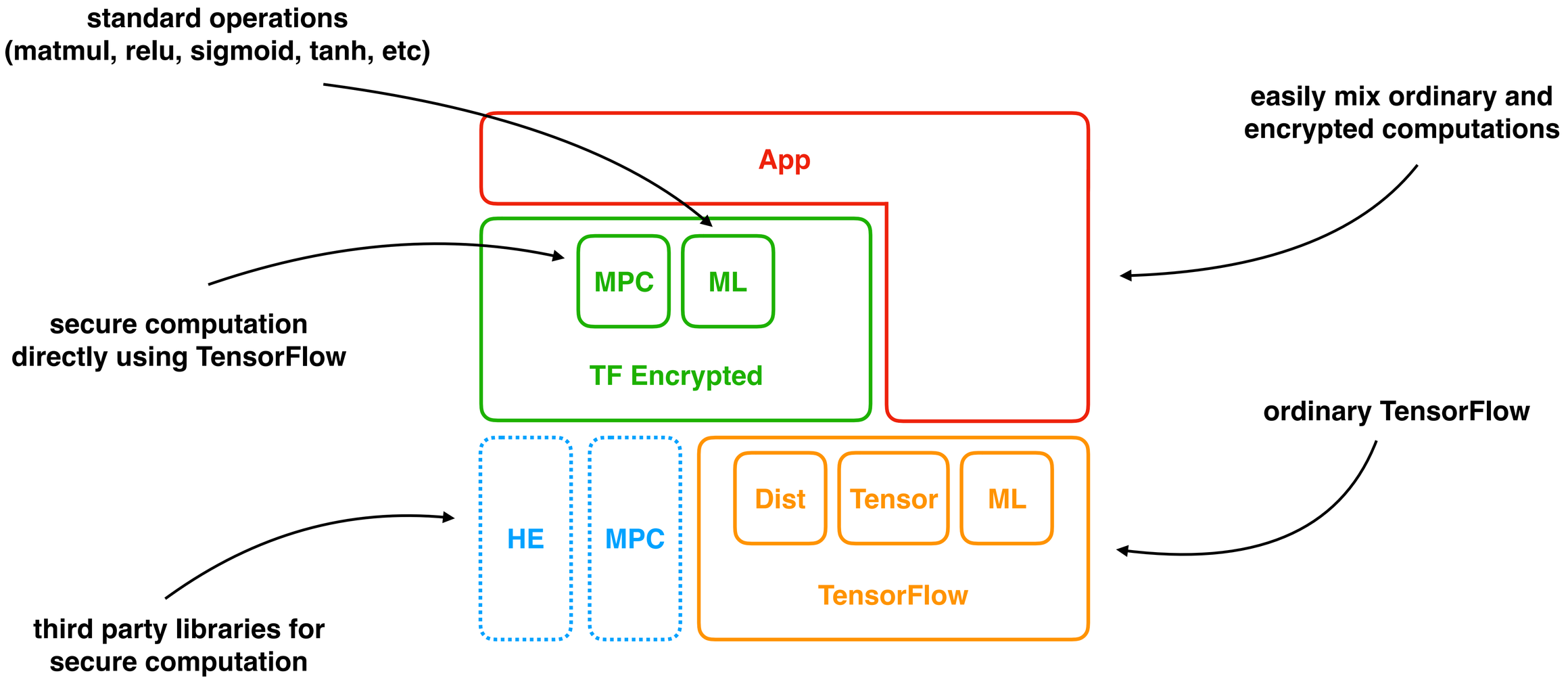
platform for research and production-level training and deployment

popular and backed by Google



DropoutLabs

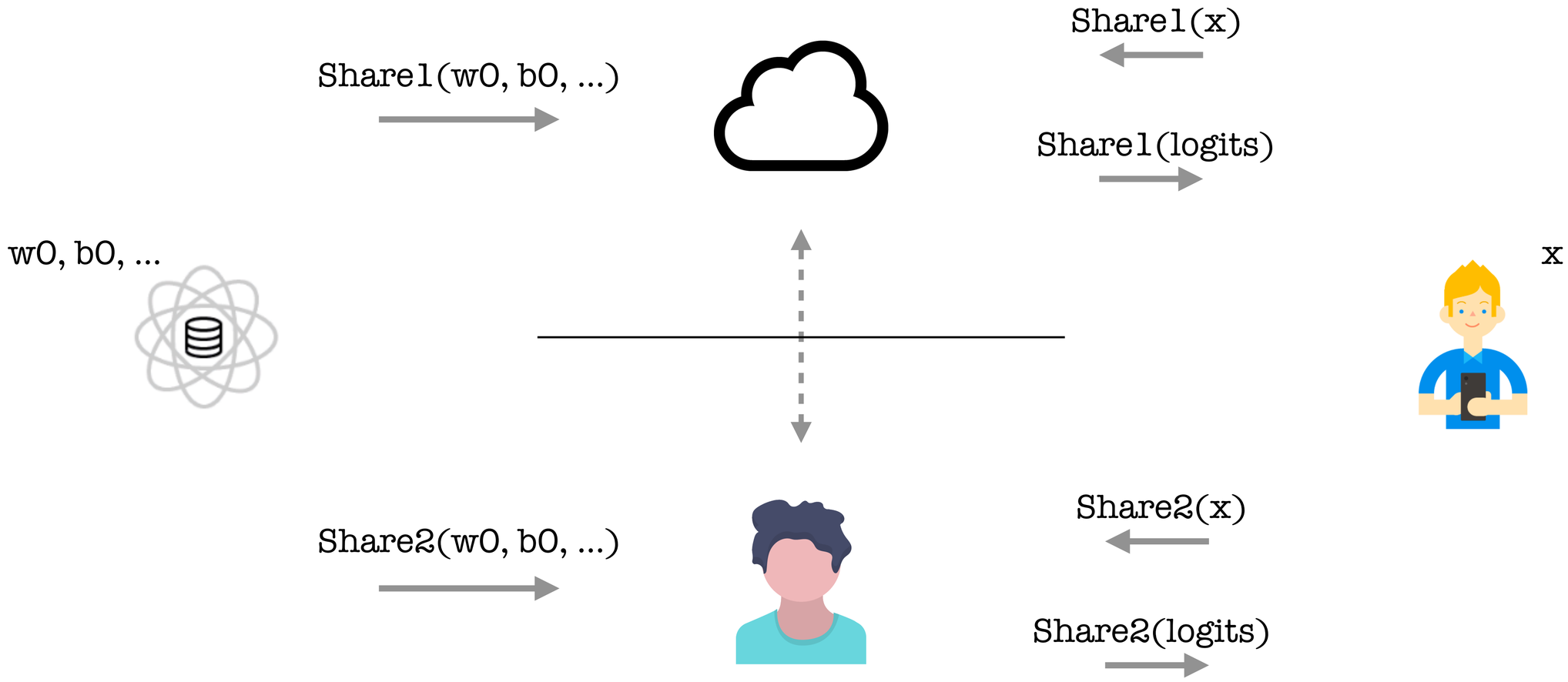
TF Encrypted Architecture



Prediction

Encouraging use

Participants



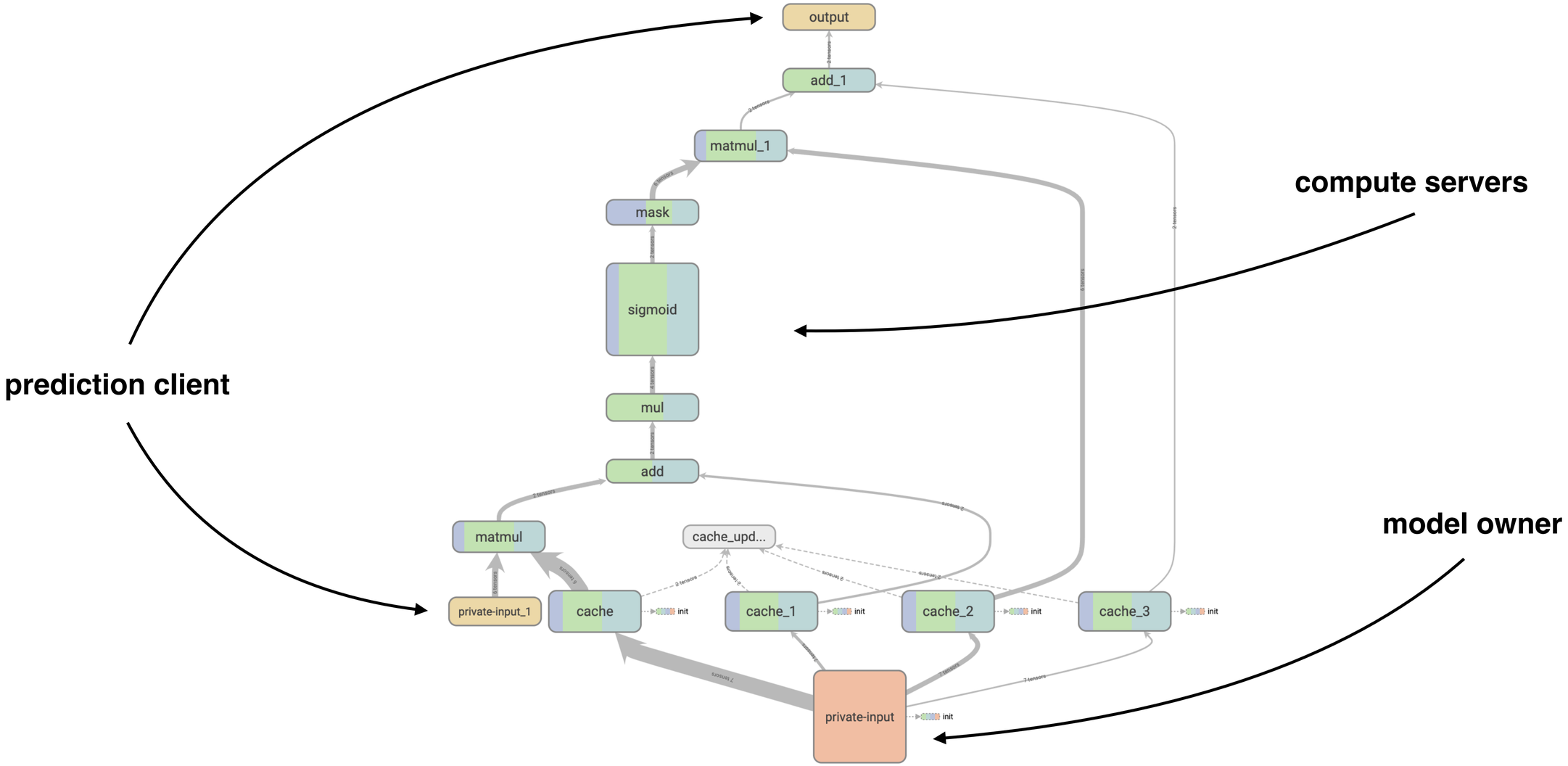
Private Prediction with TF Encrypted

```python
import tensorflow as tf

def provide_weights():""" Load from disk """
def provide_input(): """ Pre-process """
def receive_output(logits): return tf.Print([], [tf.argmax(logits)])

# get model weights
w0, b0, w1, b1, w2, b2 = provide_weights()

# get prediction input
x = provide_input()

# compute prediction
layer0 = tf.nn.relu((tf.matmul(x, w0) + b0))
layer1 = tf.nn.relu((tf.matmul(layer0, w1) + b1))
logits = tf.matmul(layer2, w2) + b2

# process result of prediction
prediction_op = receive_output(logits)

# run graph execution in a tf.Session
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(prediction_op)
```

```python
import tensorflow as tf
import tf_encrypted as tfe

def provide_weights():""" Load from disk """
def provide_input(): """ Pre-process """
def receive_output(logits): return tf.Print([], [tf.argmax(logits)])

# get model weights as private tensors from owner
w0, b0, w1, b1, w2, b2 = tfe.define_private_input("model-owner", provide_weights)

# get prediction input as private tensors from client
x = tfe.define_private_input("prediction-client", provide_input)

# compute private prediction on servers
layer0 = tfe.relu((tfe.matmul(x, w0) + b0))
layer1 = tfe.relu((tfe.matmul(layer0, w1) + b1))
logits = tfe.matmul(layer1, w2) + b2

# process result of prediction on client
prediction_op = tfe.define_output("prediction-client", logits, receive_output)

# run secure graph execution in a tf.Session
with tfe.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(prediction_op)
```

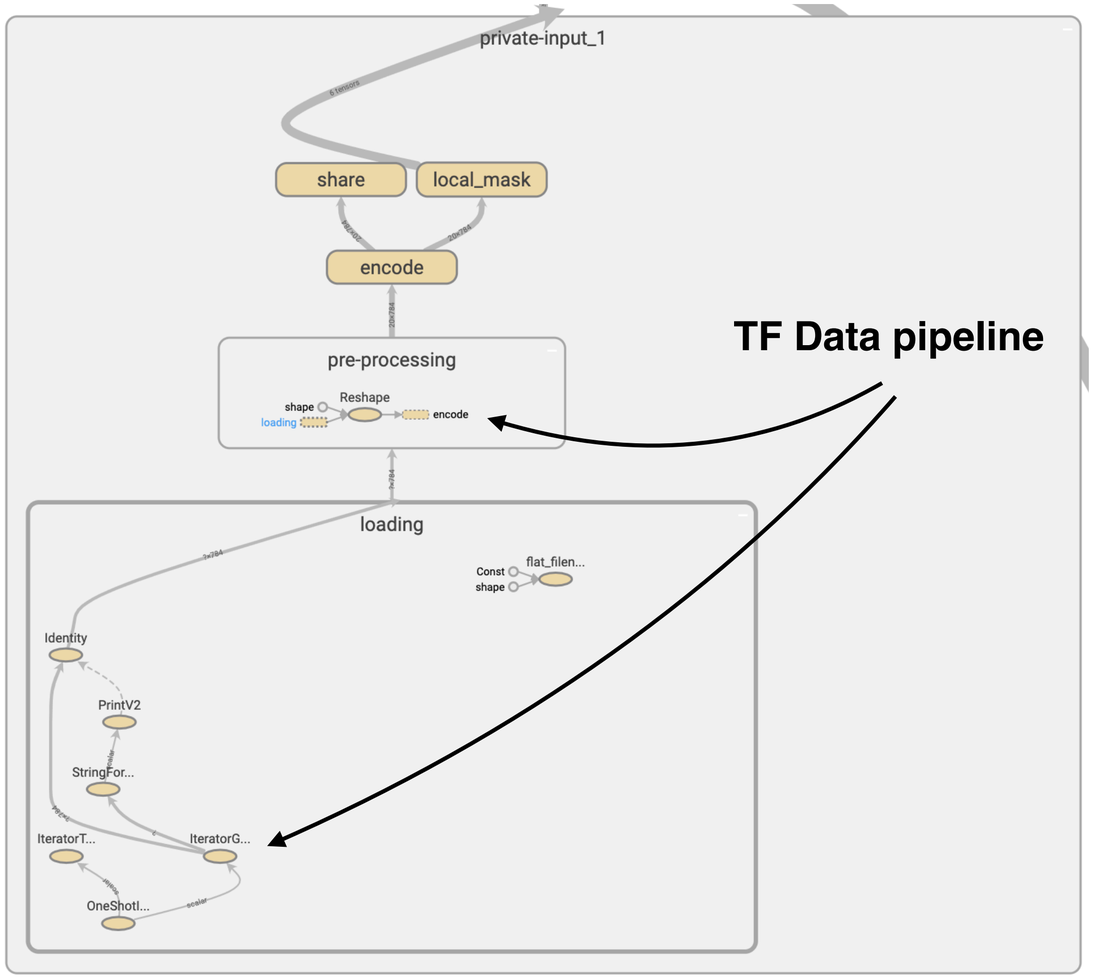
DropoutLabs

Overall Computation

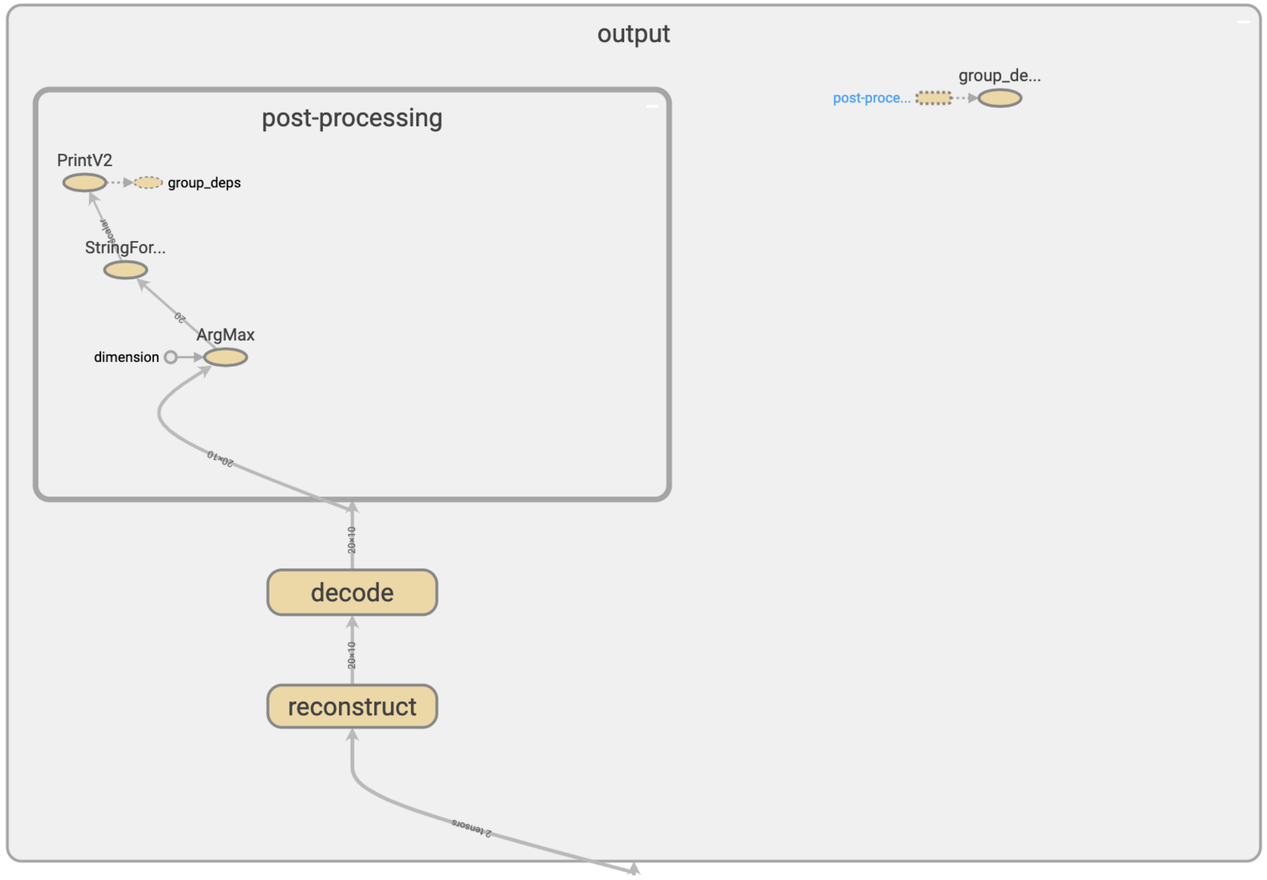


DropoutLabs

Local Processing



TF Data pipeline

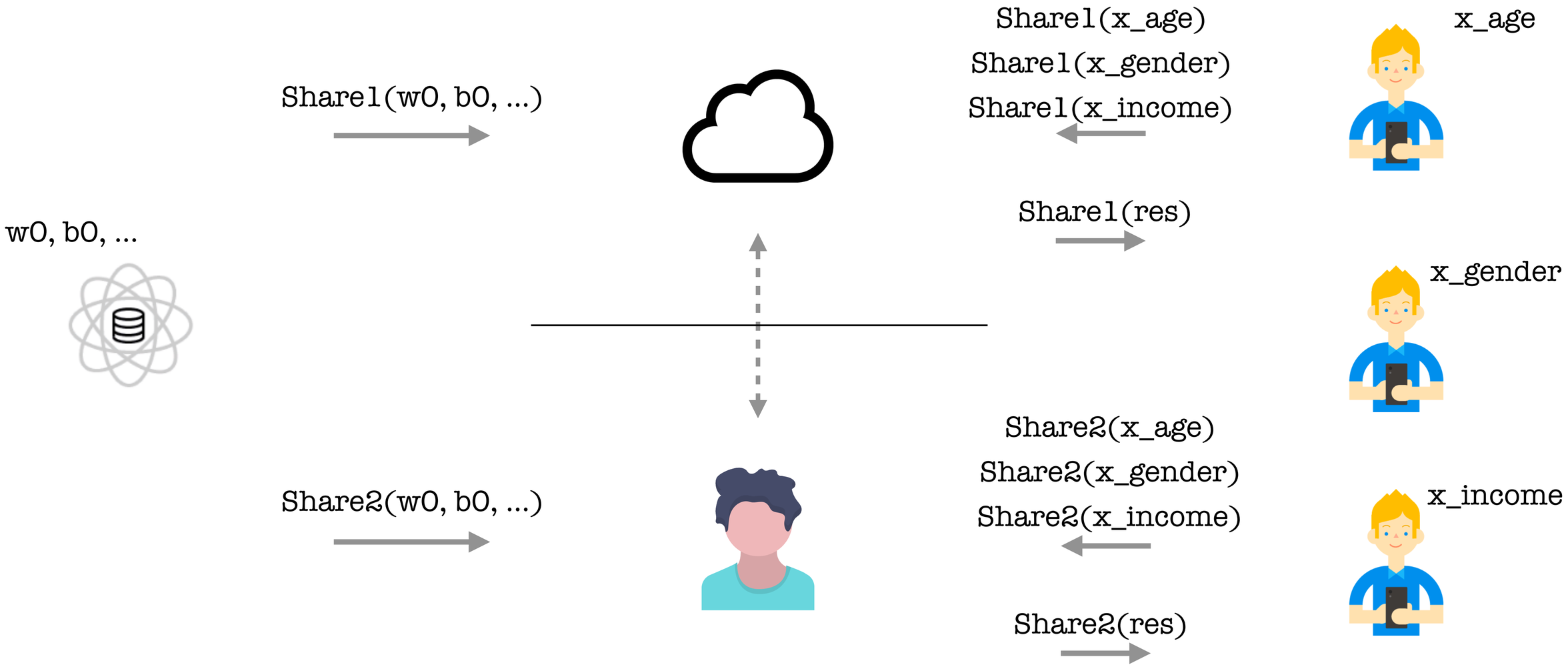


DropoutLabs

Joint Prediction

Combining knowledge for nuance

Participants



Private Joint Prediction with TF Encrypted

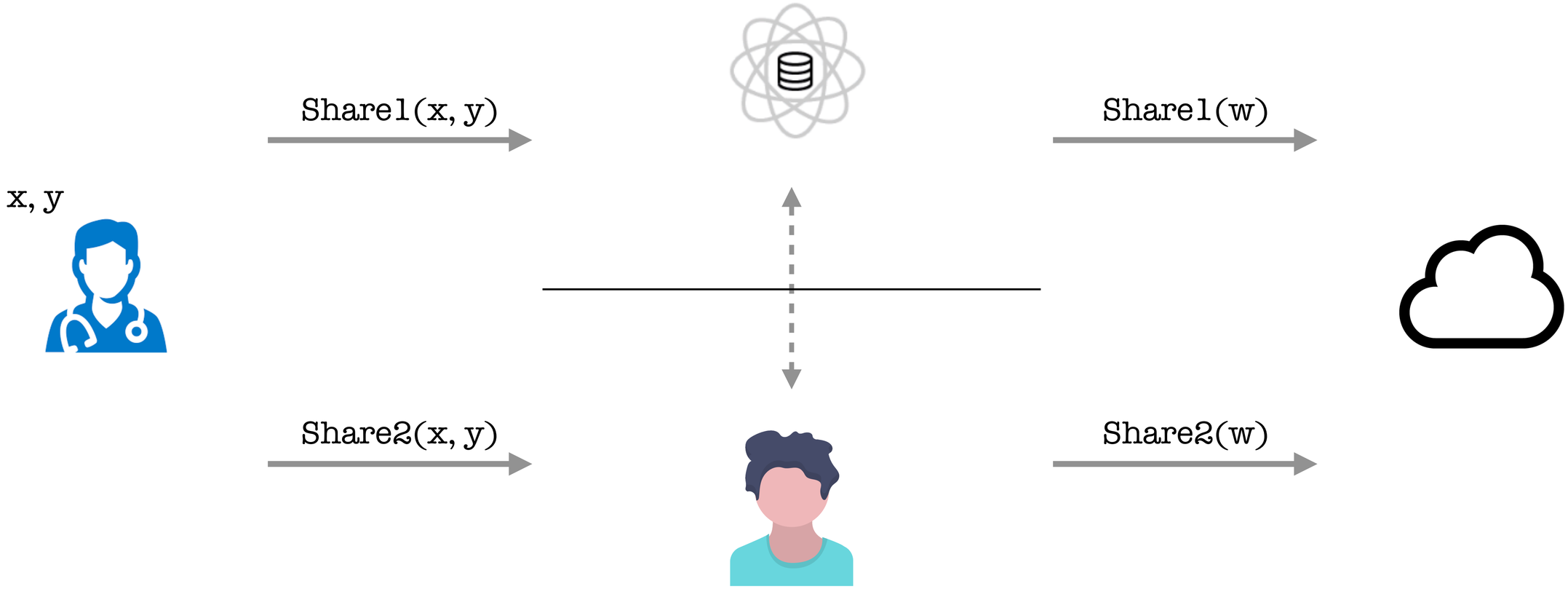
```
1   w, b = tfe.define_private_input("model-owner", provide_weights)
2
3   x_age = tfe.define_private_input("age-provider", provide_age)
4   x_gender = tfe.define_private_input("gender-provider", provide_gender)
5   x_income = tfe.define_private_input("income-provider", provide_income)
6
7   x = tfe.concat([x_age, x_gender, x_income], axis=1)
8
9   y = tfe.matmul(x, w) + b
10
11  prediction_op = tfe.define_output("result-receiver", y, receive_output)
12
13  with tfe.Session() as sess:
14      sess.run(prediction_op)
```

DropoutLabs

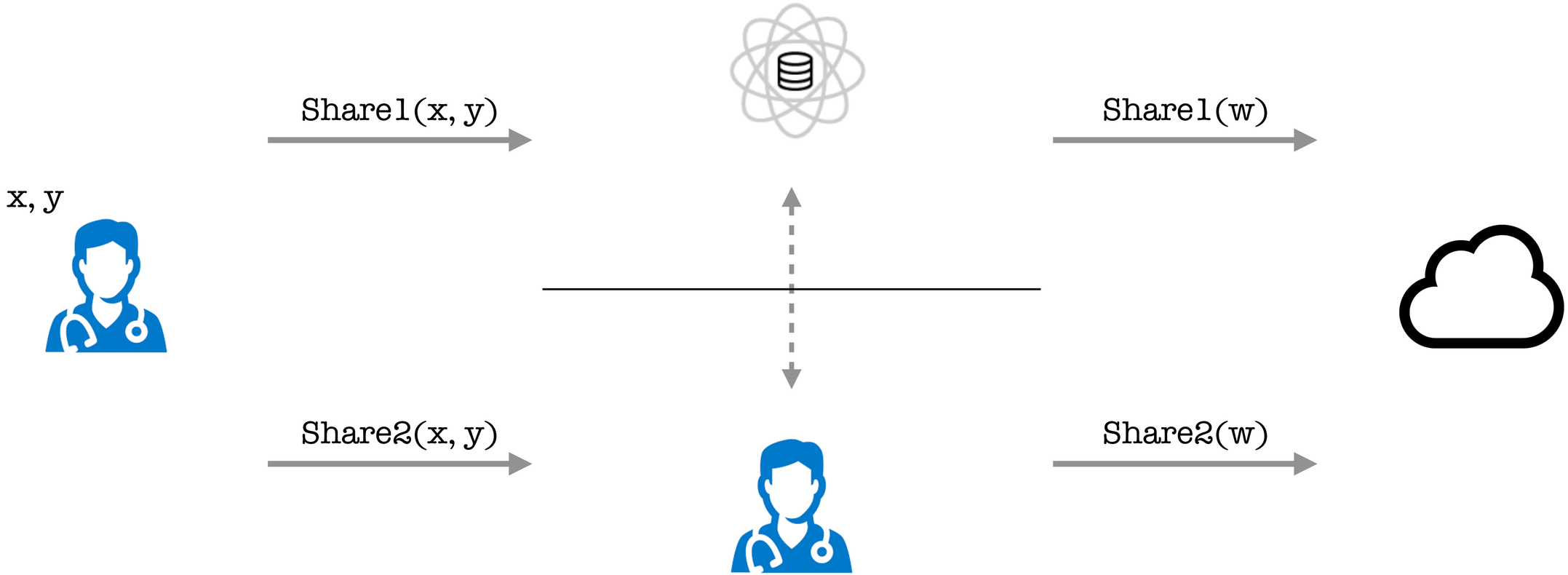
Training

Learning without seeing

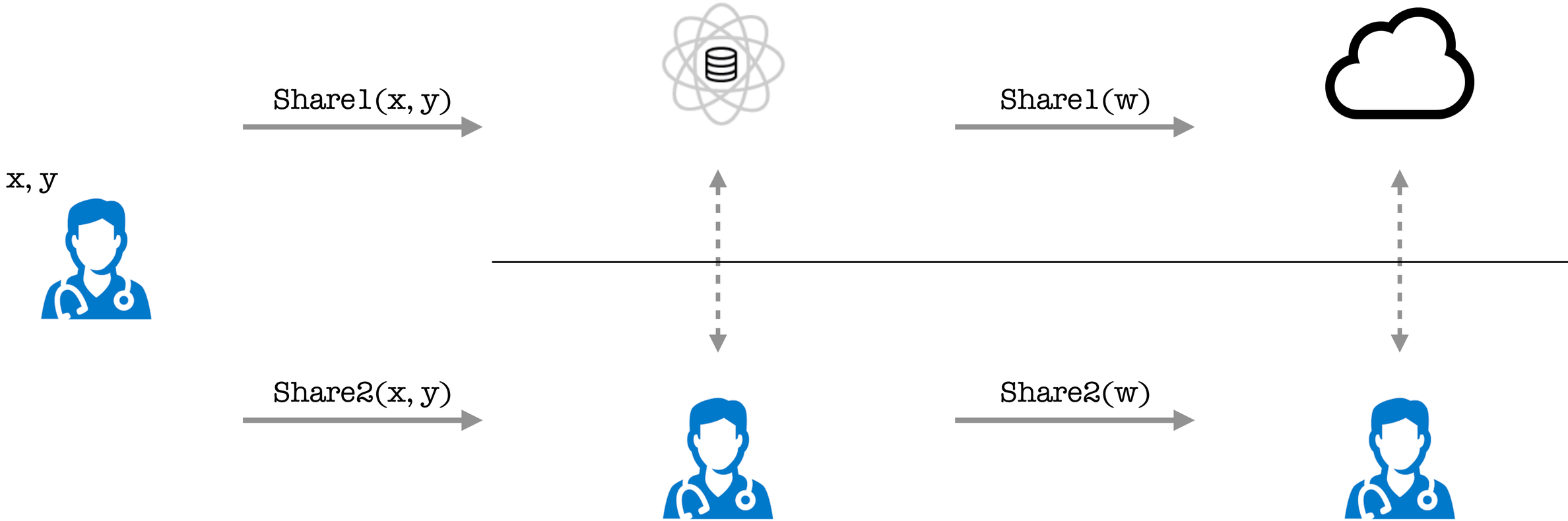
Participants



Participants



Participants

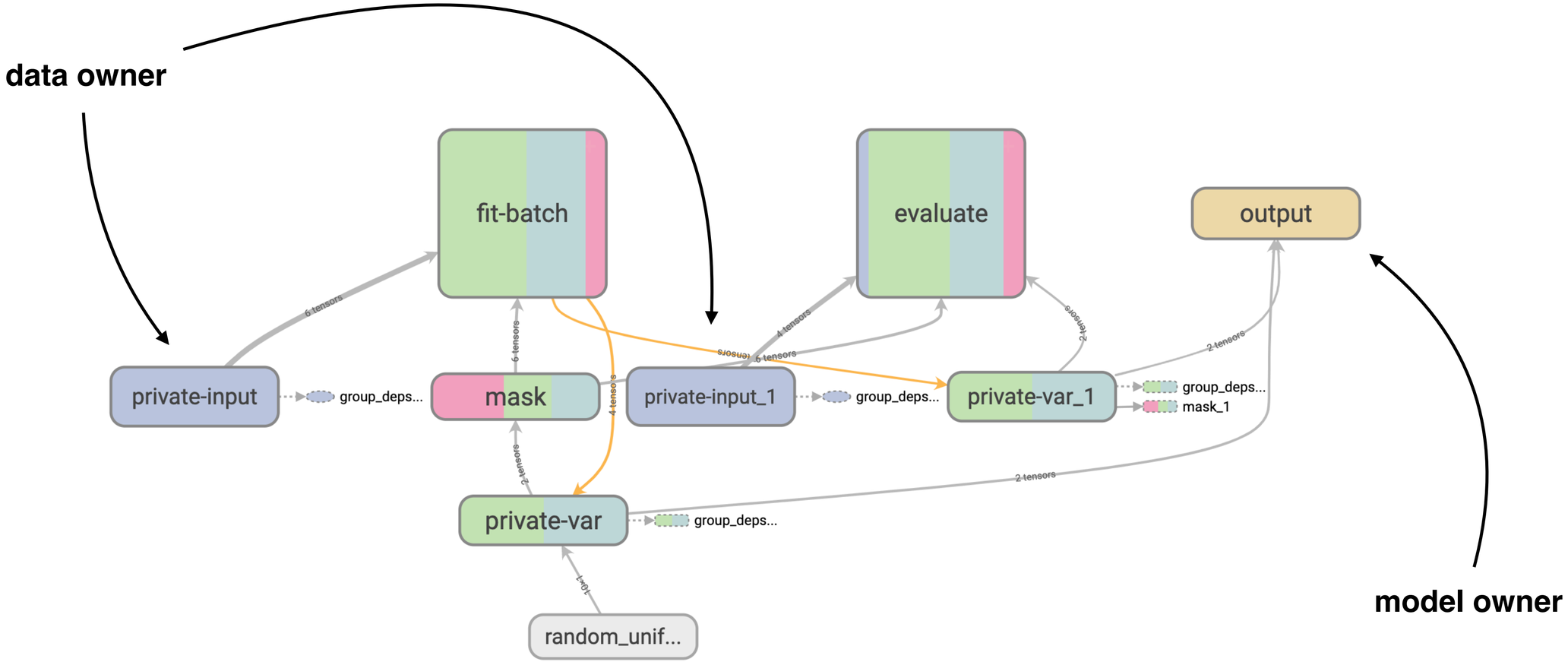


Private Training with TF Encrypted

```
class LogisticRegression(Model):

    @property
    def weights(self):
        # ...

    def forward(self, x):
        # ...

    def backward(self, x, dy, learning_rate=0.01):
        # ...

    def loss_grad(self, y, y_hat):
        # ...

model = LogisticRegression()

data_owner = DataOwner('data-owner')
model_owner = ModelOwner('model-owner')

x_train, y_train = tfe.define_private_input(data_owner.player_name, data_owner.provide_training_data)
x_test, y_test = tfe.define_private_input(data_owner.player_name, data_owner.provide_testing_data)

reveal_weights_op = tfe.define_output(model_owner.player_name, model.weights, model_owner.receive_weights)

with tfe.Session() as sess:
    sess.run([tfe.global_variables_initializer(), data_owner.initializer])

    model.fit(sess, x_train, y_train, epochs=10)
    model.evaluate(sess, x_test, y_test, data_owner)

    sess.run(reveal_weights_op)
```

DropoutLabs

Overall Computation

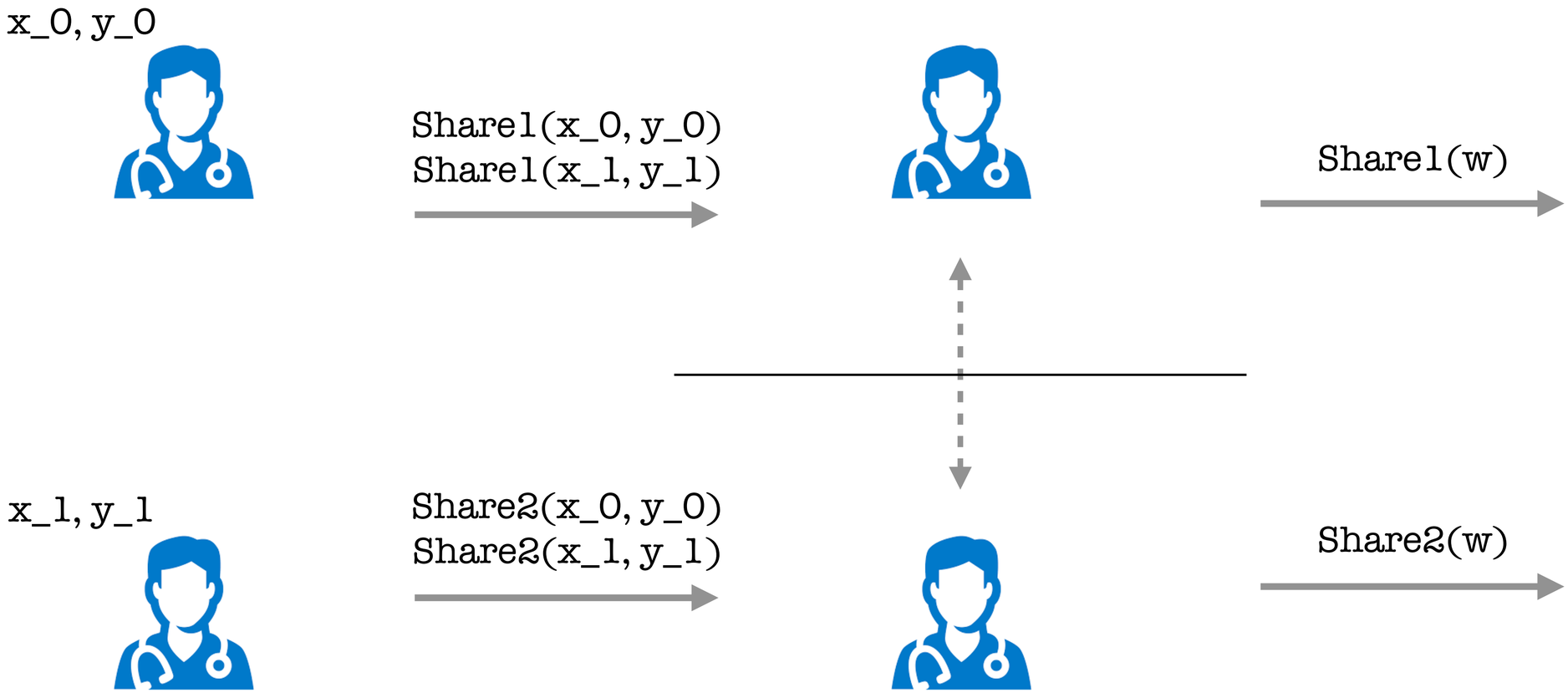


DropoutLabs

Joint Training

Combining insights for better models

Participants



Private Joint Training with TF Encrypted

```
data_owner_0 = DataOwner('data-owner-0')
data_owner_1 = DataOwner('data-owner-1')

tfe.set_protocol(tfe.protocol.Pond(data_owner_0.player_name, data_owner_1.player_name))

x_train_0, y_train_0 = tfe.define_private_input(data_owner_0.player_name, data_owner_0.provide_training_data)
x_train_1, y_train_1 = tfe.define_private_input(data_owner_1.player_name, data_owner_1.provide_training_data)

x_train = tfe.concat([x_train_0, x_train_1], axis=0)
y_train = tfe.concat([y_train_0, y_train_1], axis=0)
```

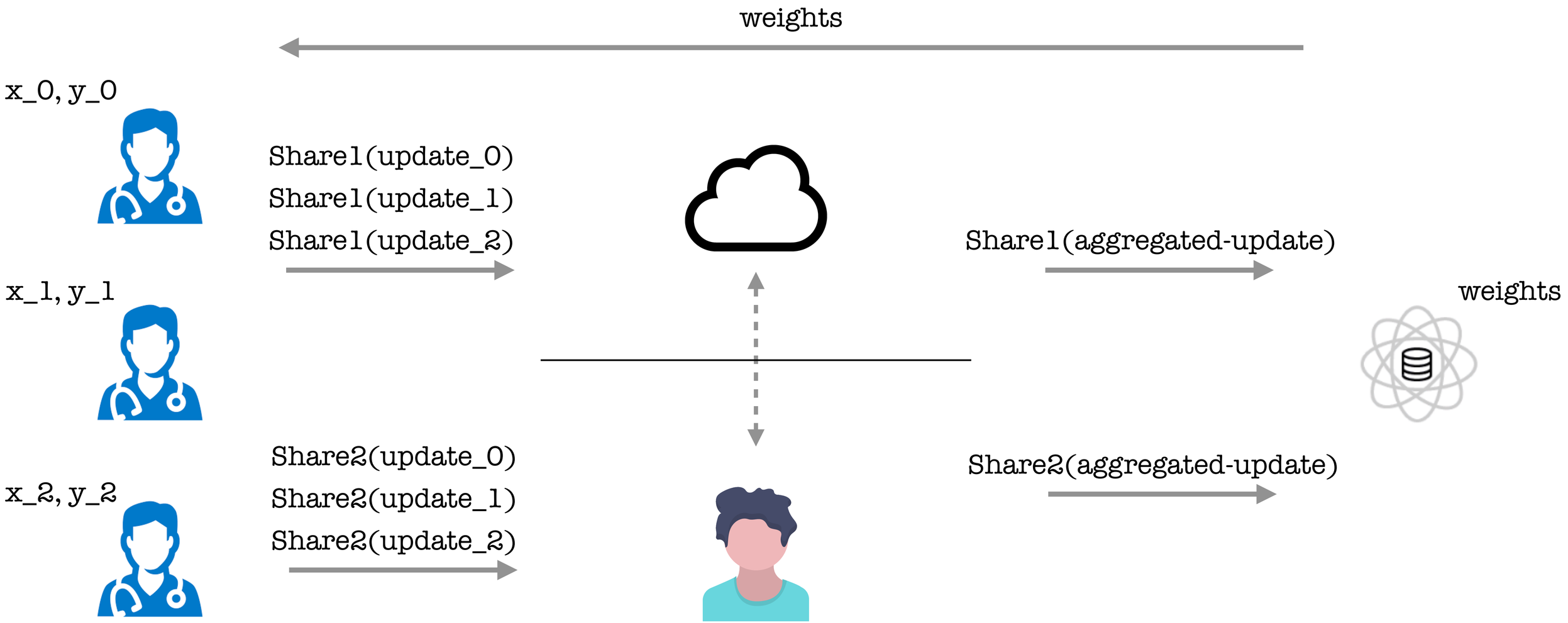
DropoutLabs

Federated Learning

Keeping data decentralized

DropoutLabs

Participants

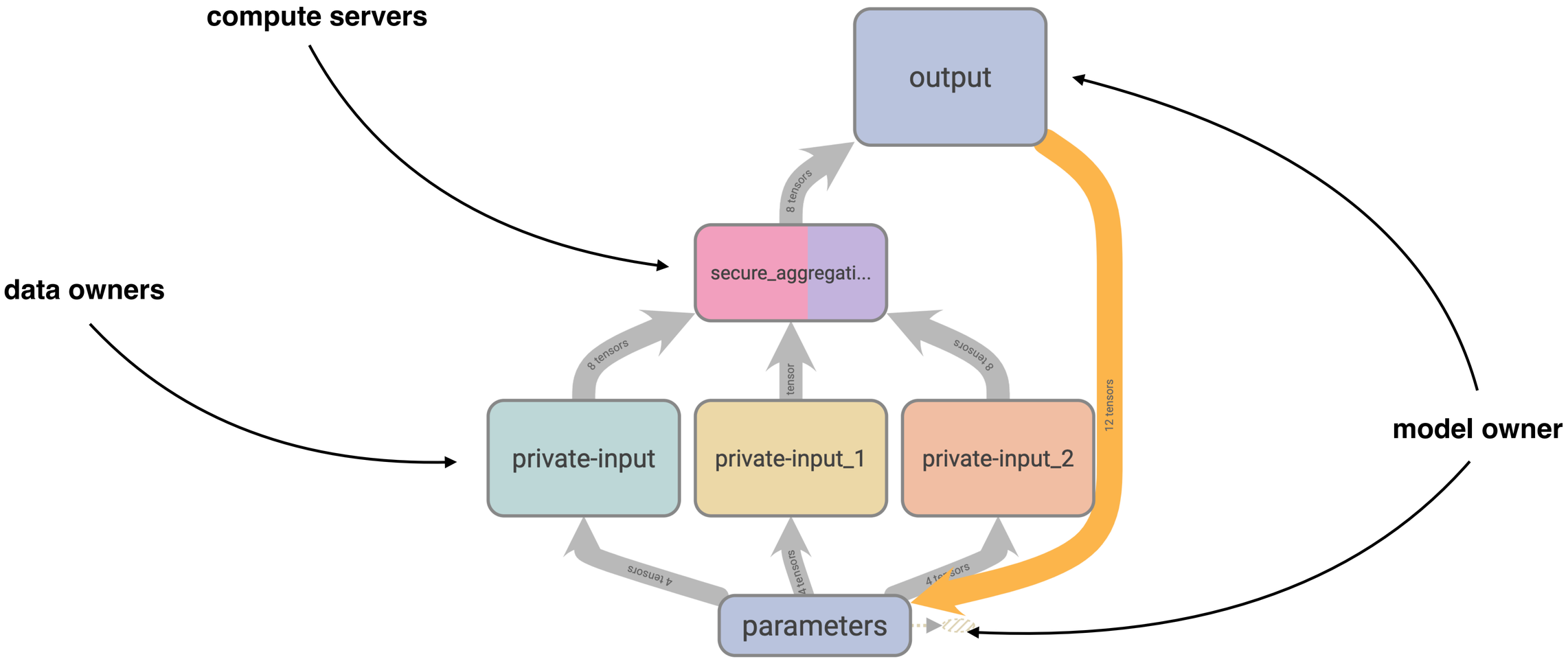


Secure Federated Learning in TF Encrypted

```
model_owner = ModelOwner('model-owner')
data_owners = [
    DataOwner('data-owner-0', model_owner.build_training_model),
    DataOwner('data-owner-1', model_owner.build_training_model),
    DataOwner('data-owner-2', model_owner.build_training_model),
]

model_grads = zip(*(
    tfe.define_private_input(data_owner.player_name, data_owner.compute_gradient)
    for data_owner in data_owners
))

aggregated_model_grads = [
    tfe.add_n(grads) / len(grads)
    for grads in model_grads
]

iteration_op = tfe.define_output(model_owner.player_name, aggregated_model_grads, model_owner.update_model)

with tfe.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for i in range(model_owner.ITERATIONS):
        sess.run(iteration_op)
```

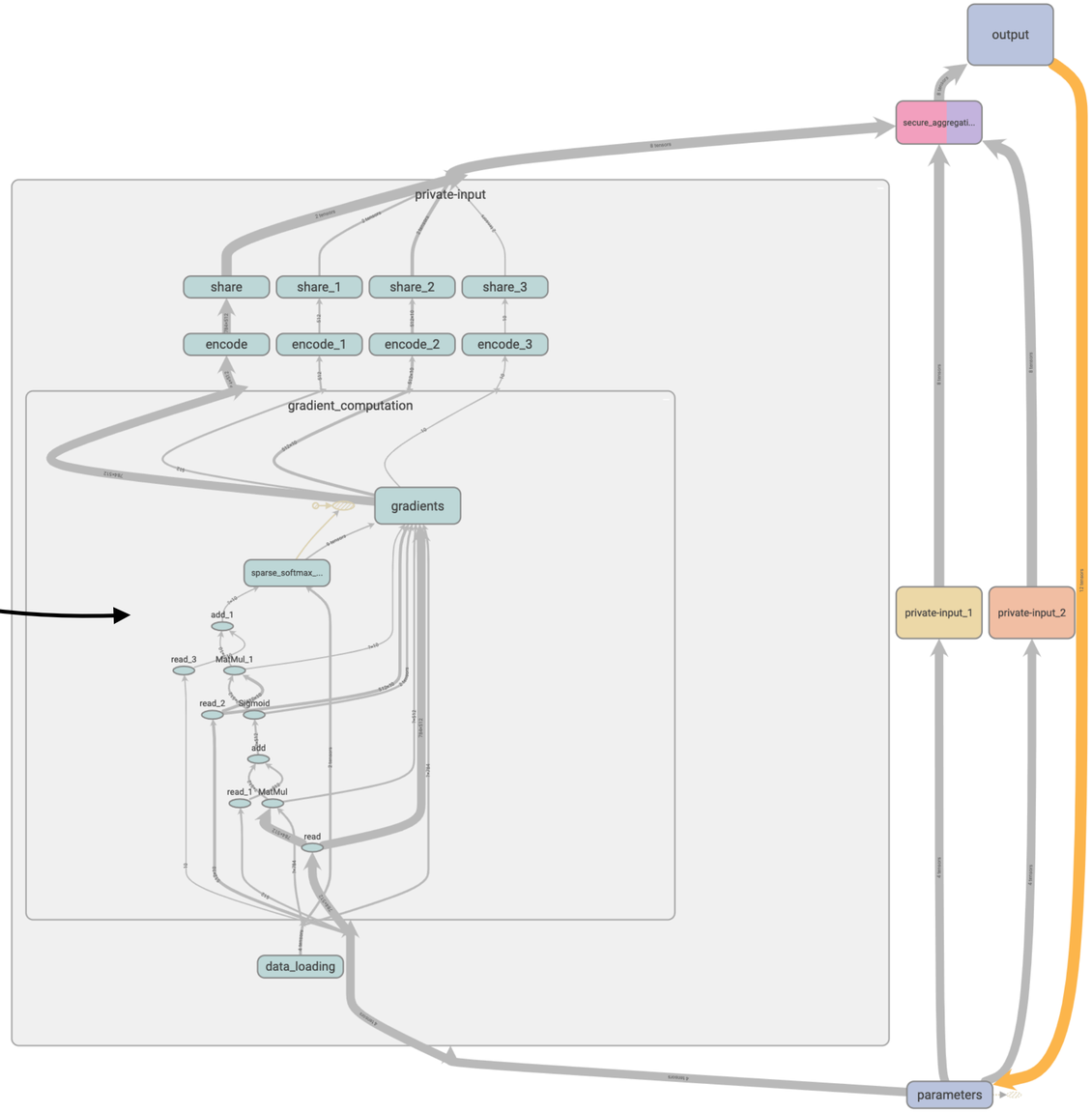
DropoutLabs

Overall Computation



Local Optimization

TF optimization



DropoutLabs

Roadmap

High-level API (Private Keras, Pre-trained Models, Owned Data)

Tighter integration (TF Data, TF 2.0, TF Privacy, TF Federated)

Third-party cryptographic libraries (HE, MPC)

Improved performance

Wrap-Up

You can **compute on encrypted data**,
without the ability to decrypt

Privacy-preserving ML mitigate **bottlenecks** and
enable access to sensitive information

Secure computation **distributes trust and control**,
and is complementary to e.g. differential privacy

Privacy-preserving ML is a multidisciplinary field
benefitting from **adaptations** on both sides

TF Encrypted focuses on
usability and **integration**

Thank you!

github.com/tf-encrypted/

@mortendahlcs

@dropoutlabsai

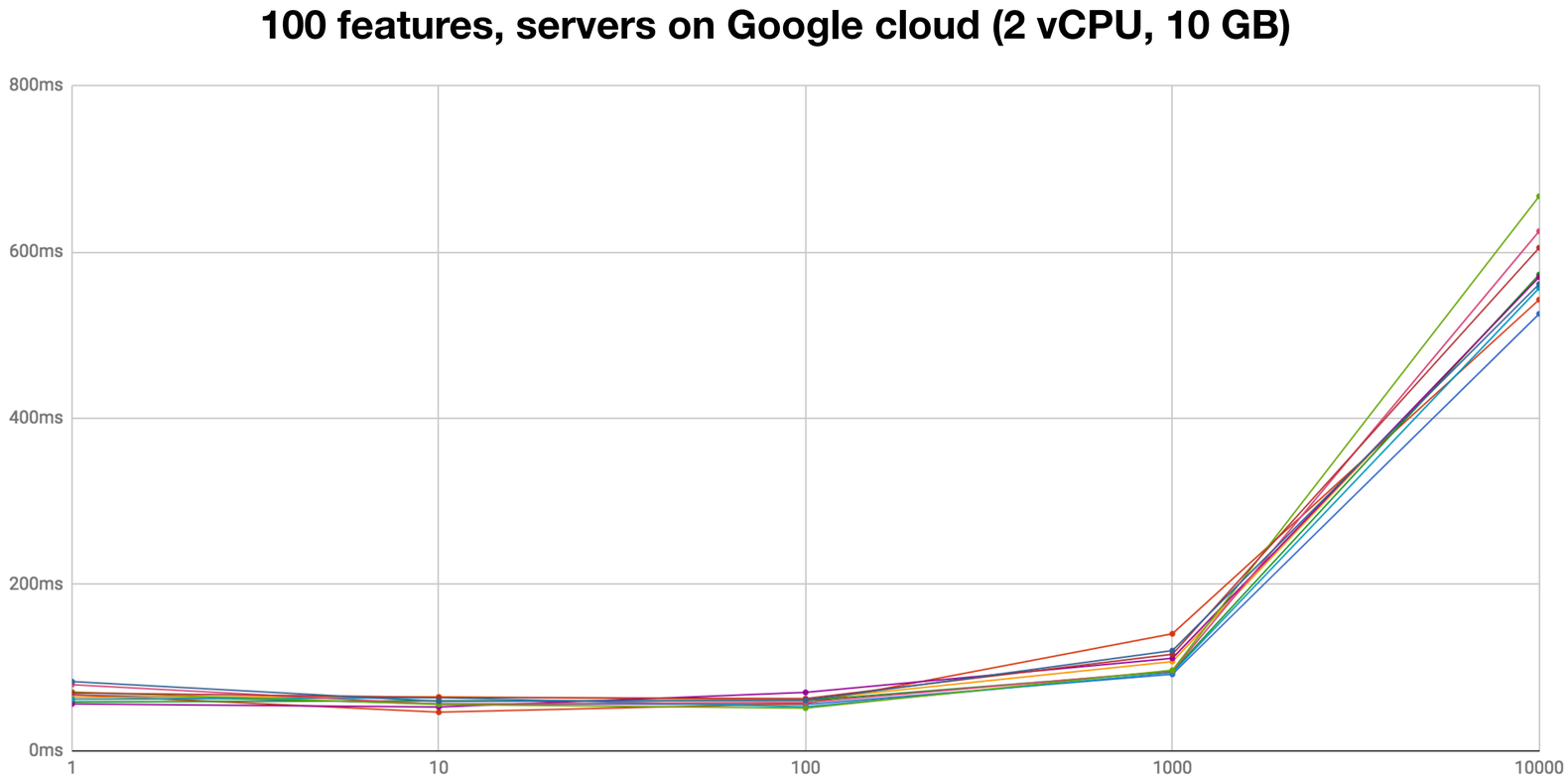
DropoutLabs

Appendix

Configuration

```
{
    "server0": "10.0.0.1:4000",
    "server1": "10.0.0.2:4000",
    "model-owner": "10.0.0.3:4000",
    "prediction-client": "10.0.0.4:4000",
}
```

Performance



More at *https://arxiv.org/abs/1810.08130*

DropoutLabs

Resources

Recent research papers using secure computation

***CryptoNets**: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy*, Dowlin et al.

***SecureML**: A System for Scalable Privacy-Preserving Machine Learning*, Mohassel and Zhang

***DeepSecure**: Scalable Provably-Secure Deep Learning*, Rouhani et al.

***Gazelle**: A Low Latency Framework for Secure Neural Network Inference*, Juvekar et al.

***ABY3**: A Mixed Protocol Framework for Machine Learning*, Mohassel and Rindal

***SecureNN**: Efficient and Private Neural Network Training*, Wagh et al.

***Blind Justice**: Fairness with Encrypted Sensitive Attributes*, Kilbertus et al.

(also great summary in https://eprint.iacr.org/2017/1190)

Specialised projects

***TF Encrypted** (https://github.com/tf-encrypted/)*

***PySyft** (https://github.com/OpenMined/PySyft)*

Secure computation frameworks

***SCALE-MAMBA** (https://homes.esat.kuleuven.be/~nsmart/SCALE/)*

***MP-SPDZ** (https://github.com/n1analytics/MP-SPDZ)*

***ABY** (https://github.com/encryptogroup/ABY)*

***OblivC** (http://oblivc.org/)*

(much more at https://github.com/rdragos/awesome-mpc)

DropoutLabs