DropoutLabs

Privacy-Preserving Machine Learning in TensorFlow

Jason Mancuso
Yann Dupis

TFEncrypted

PySyft

DropoutLabs

OpenMined

Overview

Privacy-preserving machine learning and why we may need it

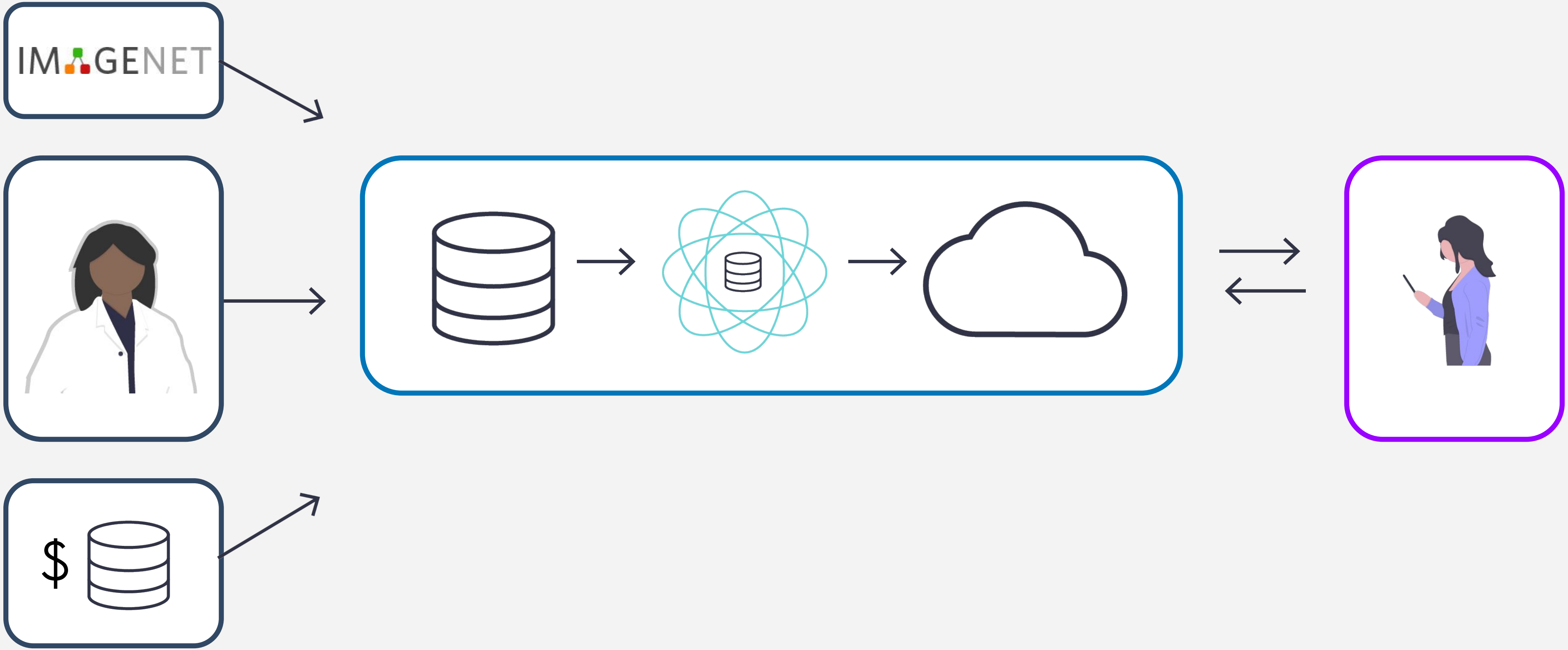
- Training **Remote execution** using PySyft-TensorFlow
- Training with **Differential Privacy**
- **Federated Learning** with secure aggregation using TF Encrypted
- **Encrypted Predictions** using TF Encrypted

Why?

Privacy in machine learning

Privacy leakage creates bottlenecks

Machine Learning Workflow

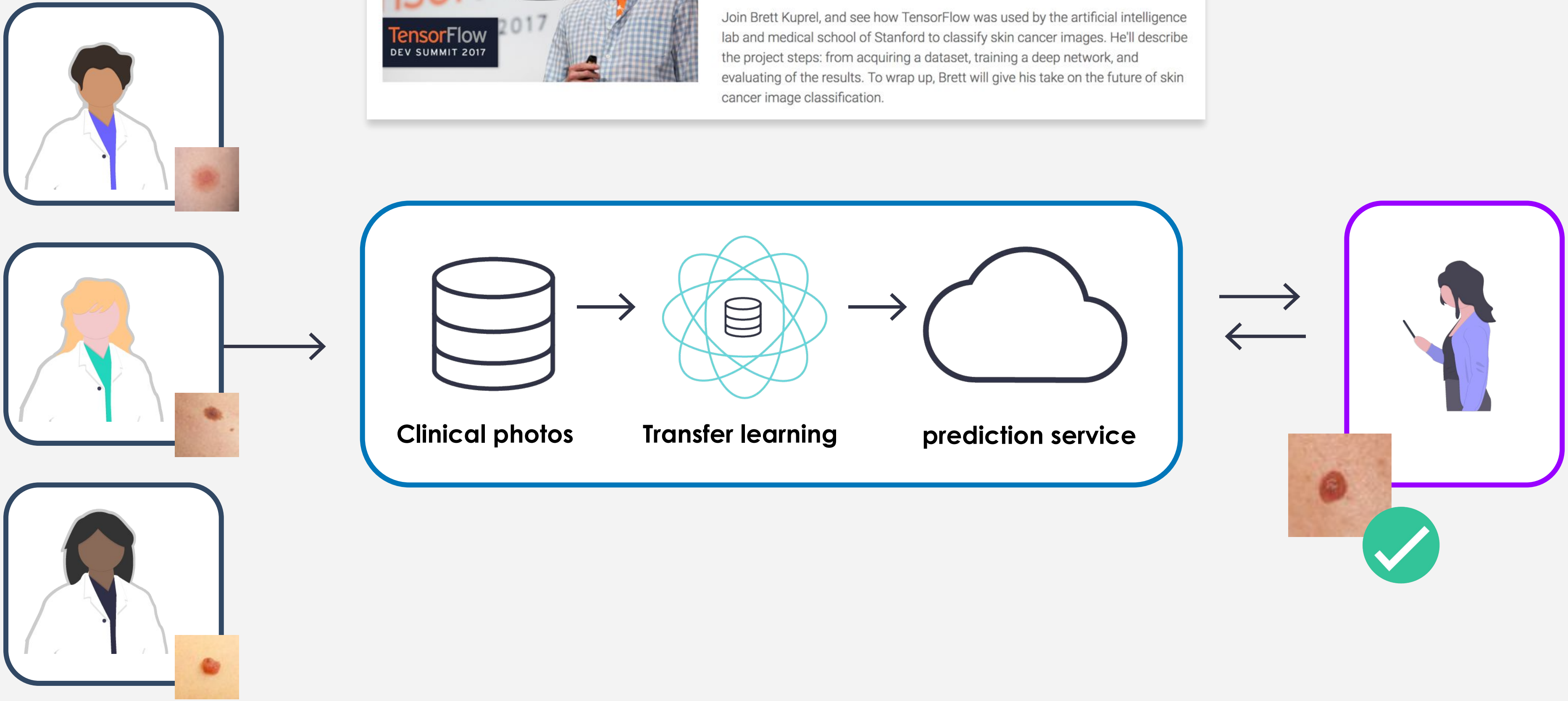


Skin Cancer Image Classification

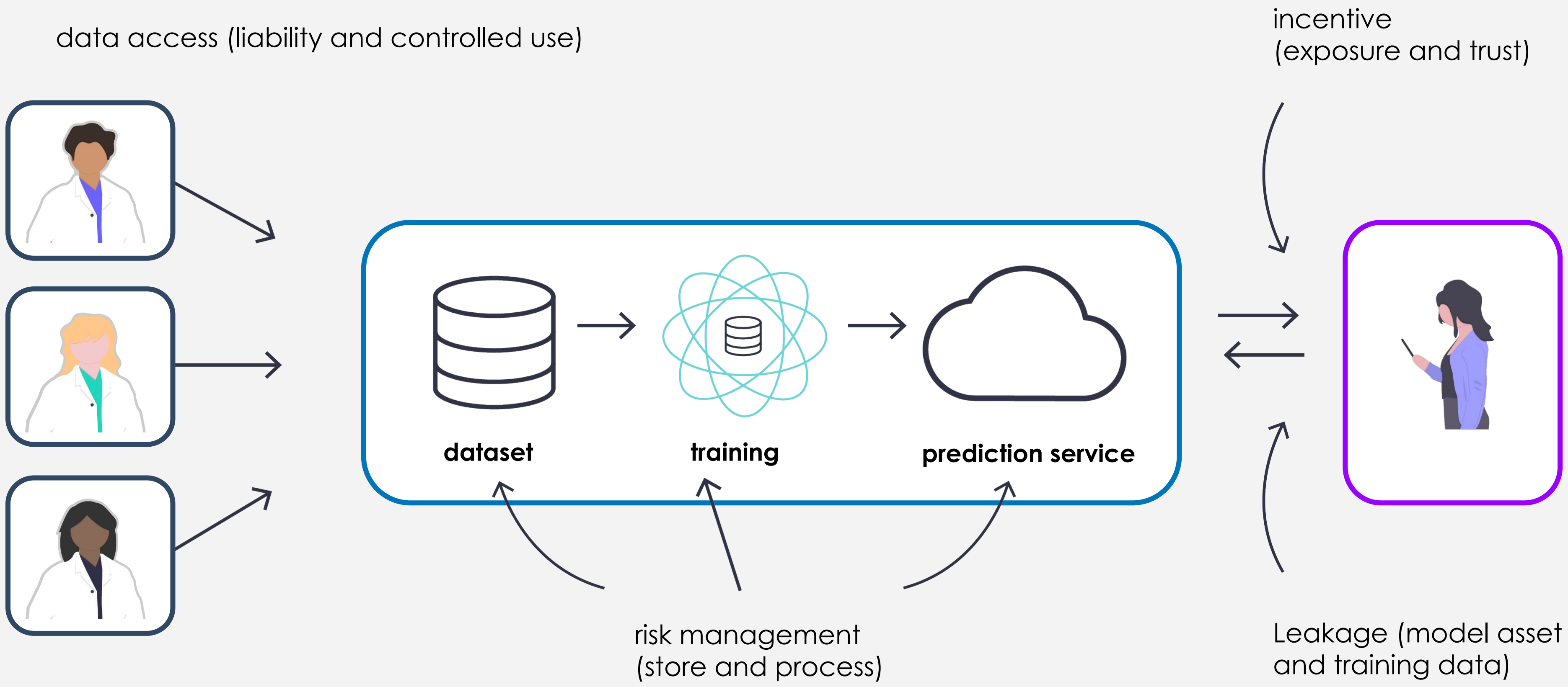
Brett Kuprel

12:30-12:40pm

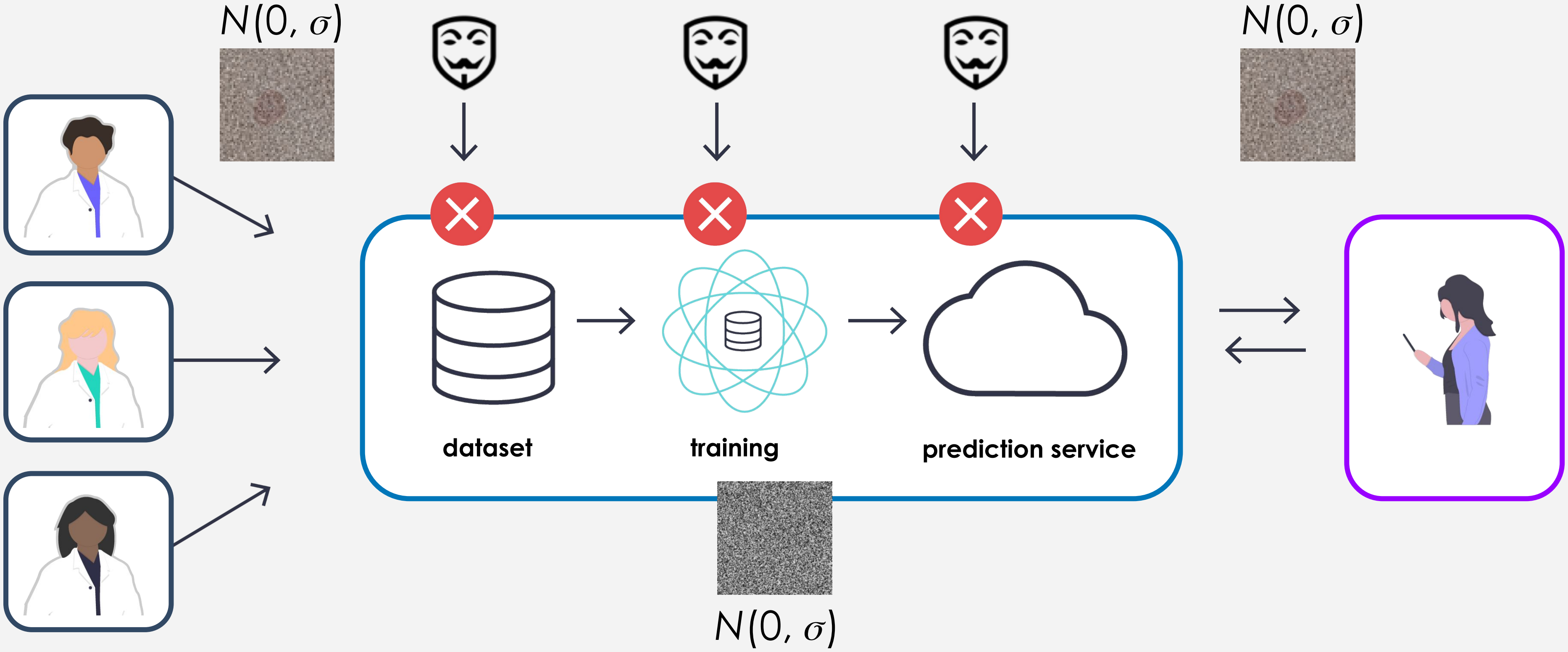
Join Brett Kuprel, and see how TensorFlow was used by the artificial intelligence lab and medical school of Stanford to classify skin cancer images. He'll describe the project steps: from acquiring a dataset, training a deep network, and evaluating of the results. To wrap up, Brett will give his take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



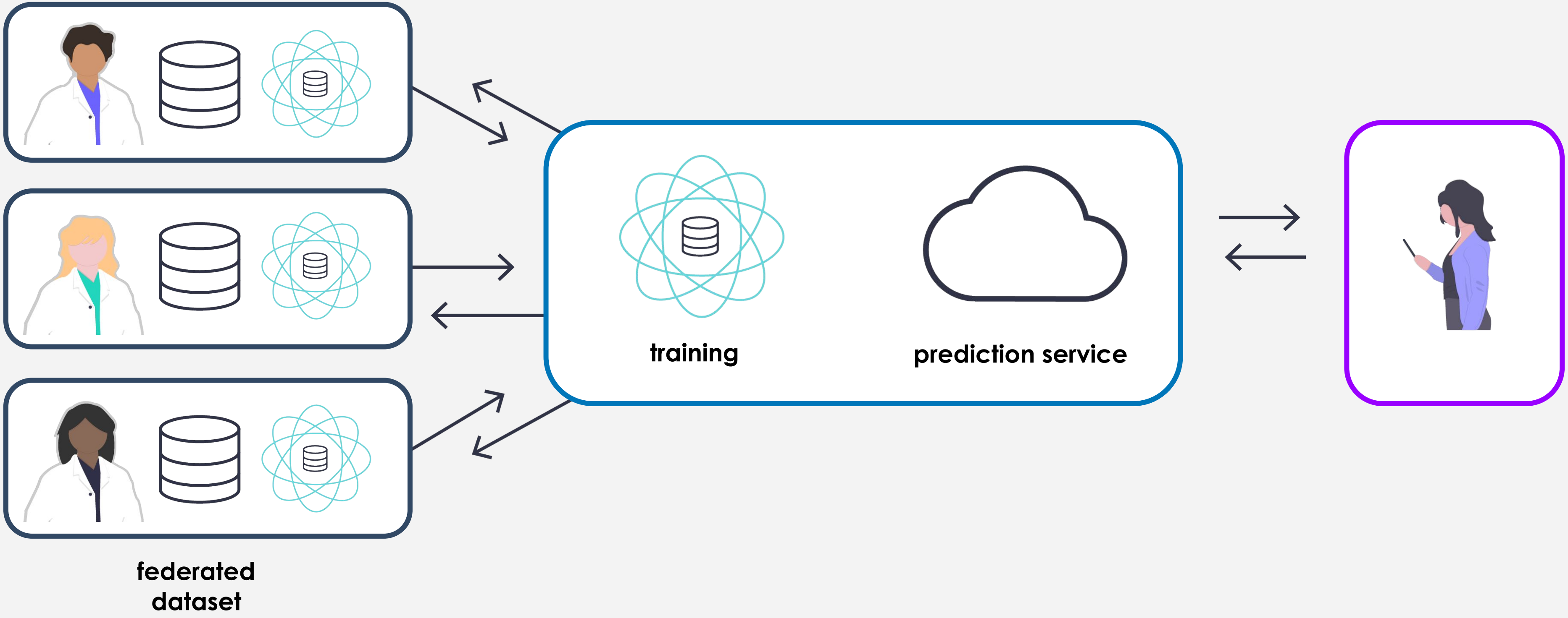
Bottlenecks



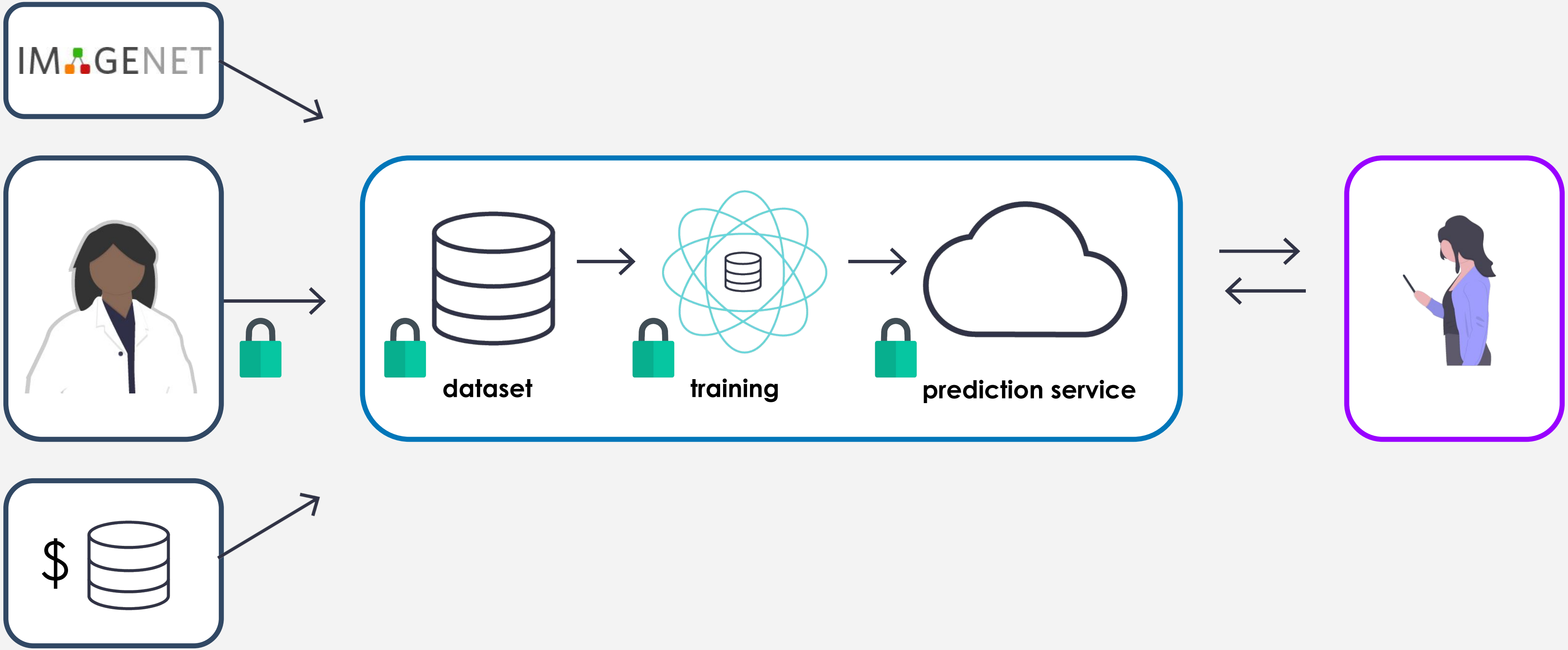
Sanitization (differential privacy)



On-Device (federated learning)



Encryption



Privacy Preserving Machine Learning Technologies

Remote Execution

Federated Learning

Secure Computation

Differential Privacy

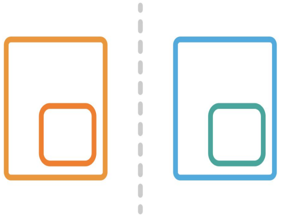
Use Case

Opportunity:

- You want to train a model on sensitive patients' data.

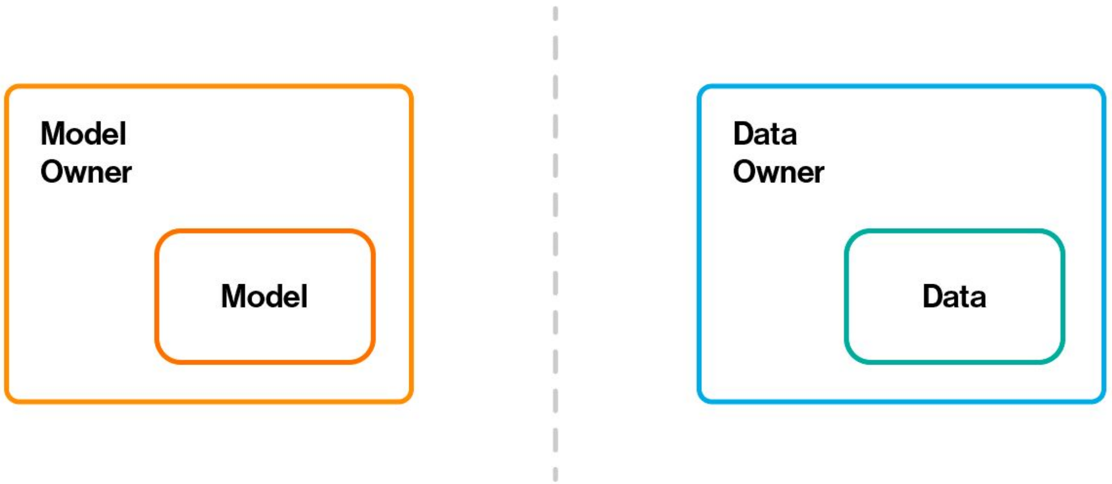
Problem:

- The data is **owned by one hospital**
- It **can't be accessed** directly for liability reasons



Remote Execution

Remote Execution



Why Not Use TensorFlow with Cluster Configuration?

Remote Execution Requirements:

- Access control
- Logging
- Policy
- Audit
- Environment agnostic

Challenges:

- Difficult to control protocol/communication
- Difficult to adapt TensorFlow distributed protocol for other environments (e.g internet)

PySyft

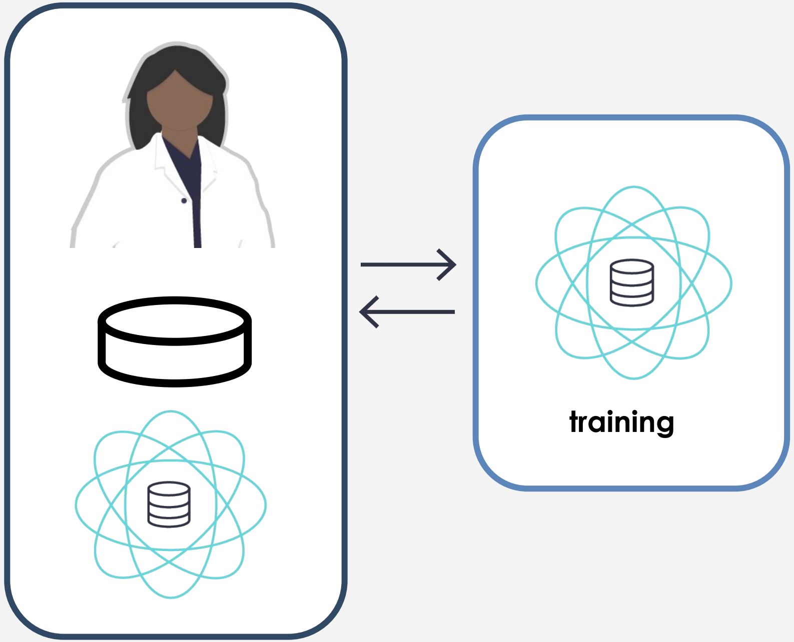
Secure, private Deep Learning library - decouples private data from model training

- Remote execution
- Federated learning
- Differential Privacy
- Multi Party Computation

PySyft - Remote Execution

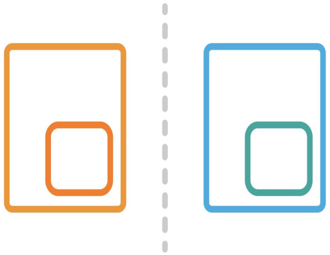
PySyft

```python
import tensorflow as tf
import syft as sy

hook = sy.TensorFlowHook(tf)
remote = sy.VirtualWorker(hook, id="remote")

x = tf.constant([1, 2, 3, 4])
y = tf.constant([5, 6, 7, 8])

x_ptr = x.send(remote)
y_ptr = y.send(remote)

z_ptr = x_ptr + y_ptr

z_ptr.get()
```

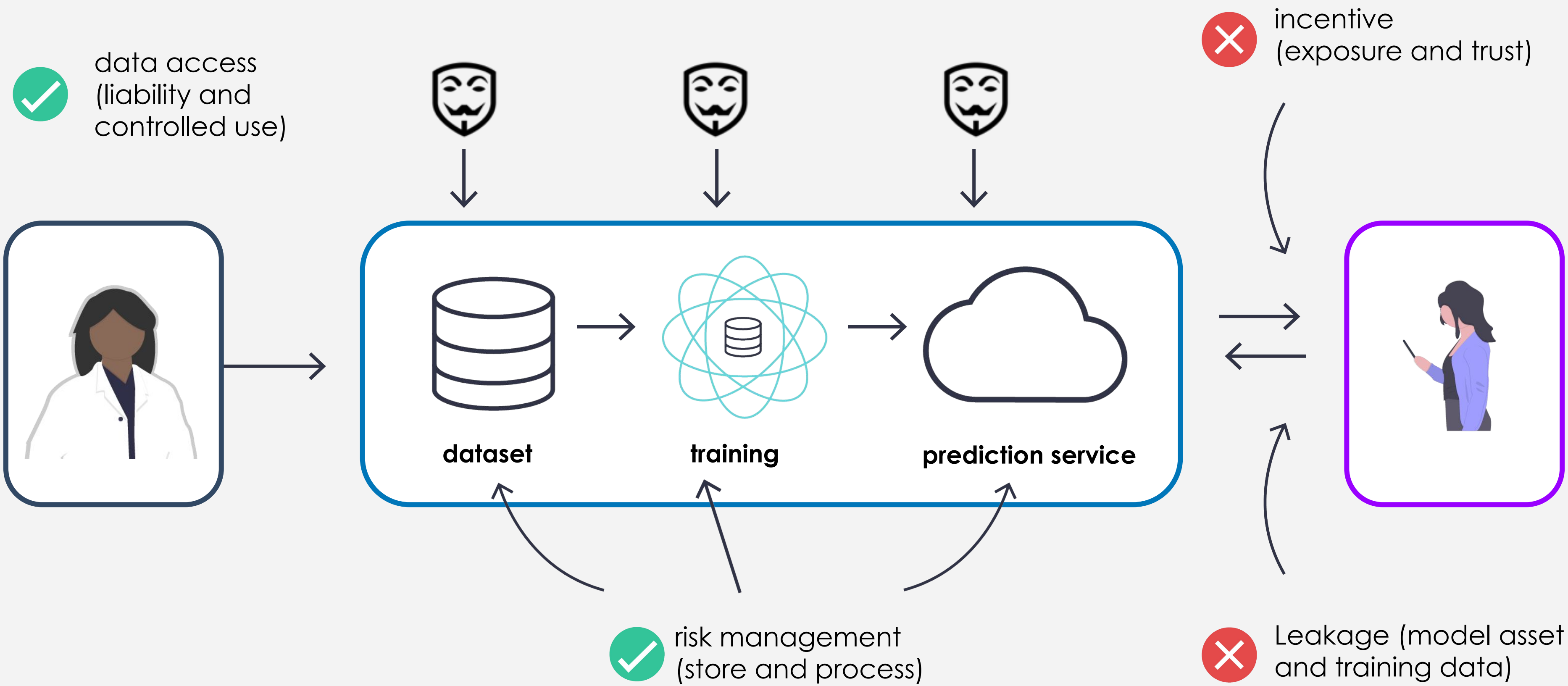


Remote Execution Tutorial

https://github.com/dropoutlabs/tf-world-tutorial/tree/master/remote-execution



Remote Execution



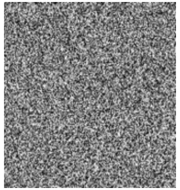
Privacy Leakage With Remote Execution

Problem:

- The model can memorize sensitive data from the training dataset
- Exposed to membership attack and model inversion

Solution:

- Train deep learning model with **Differential Privacy**



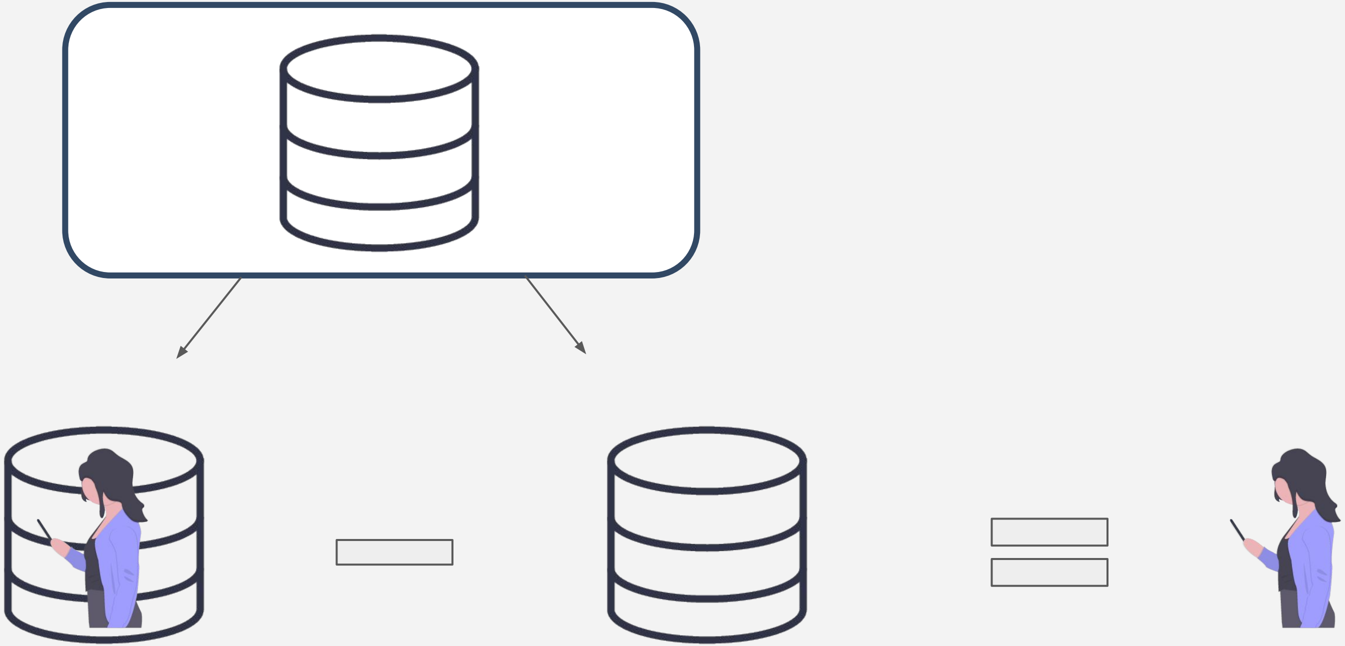
Differential Privacy

Differential Privacy

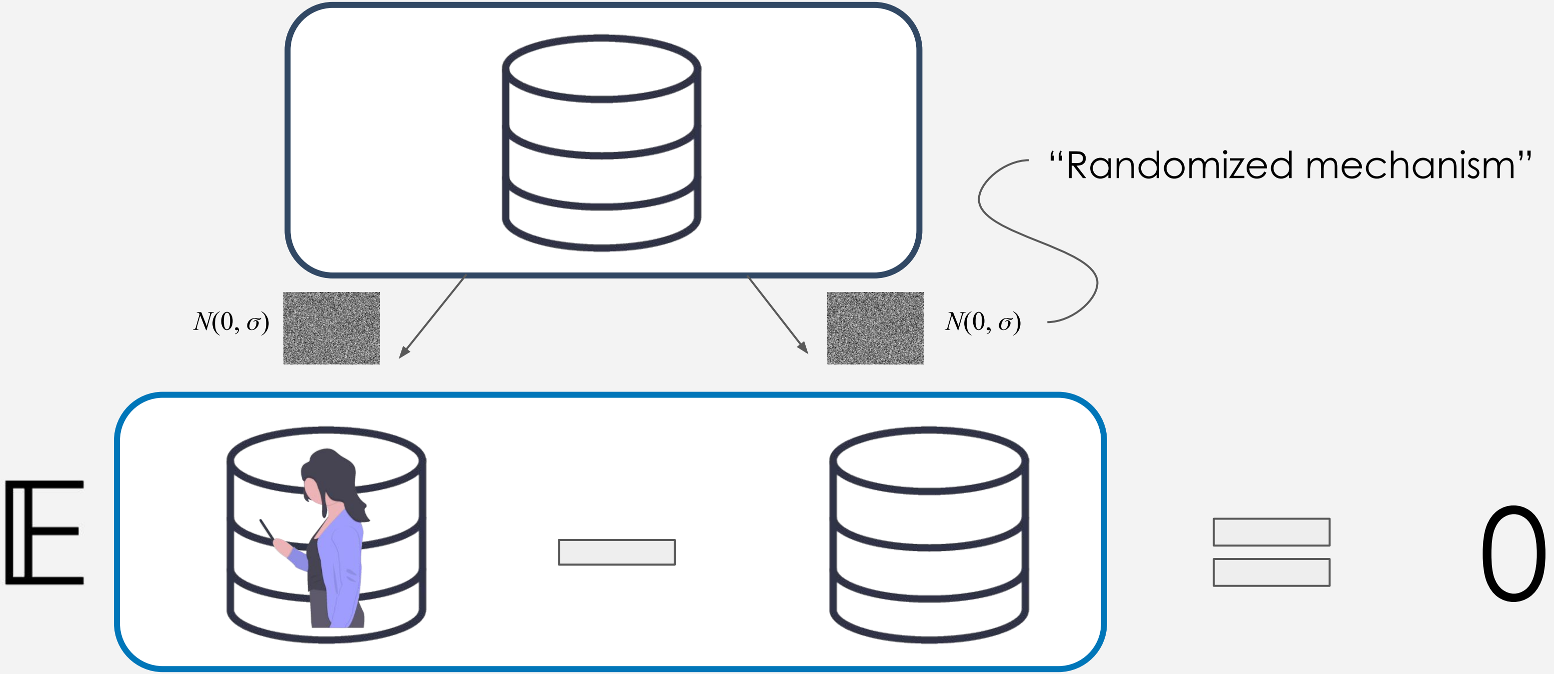
Definition 1. A randomized mechanism $\mathcal{M}\colon \mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ε, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it holds that

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta.$$

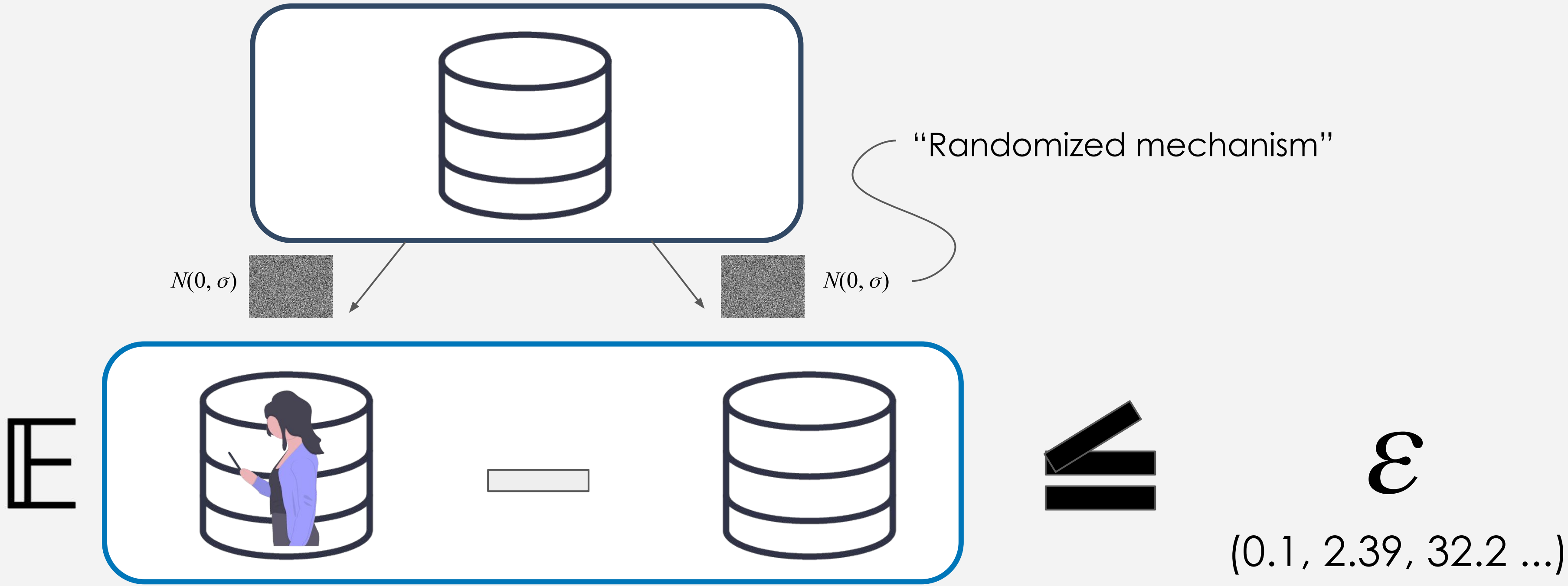
Without Differential Privacy



With Differential Privacy



With Differential Privacy



Privacy vs. Utility

noise

$$z \sim N(0, \sigma)$$

$$D = \{1, 2, 3, 3, 4, n\}$$
$$D' = \{1, 2, 3, 3, 4\}$$

"Randomized mechanism" that is (ε,δ)-differentially private.

$$f(n) = \text{Avg}(D) - \text{Avg}(D') < \varepsilon$$

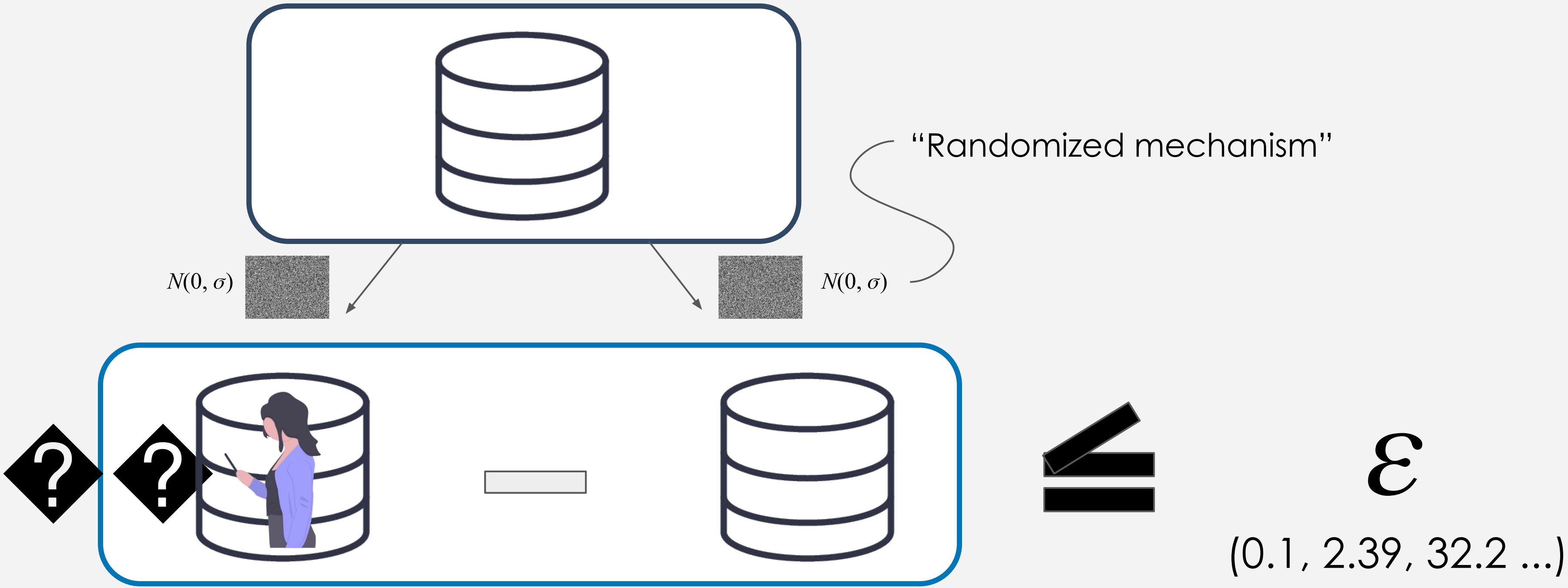
$$n \longrightarrow \infty\ ?$$

$$f(n) + z \longrightarrow \infty$$

$$f(n) \longrightarrow \infty - z < \varepsilon$$

$$z \longrightarrow \infty$$

With Differential Privacy



Privacy vs. Utility

- Provide optimizers for training models with differential privacy (e.g DP-SGD)
- Ensures that no single such example has any influence, by itself, due to the added noise

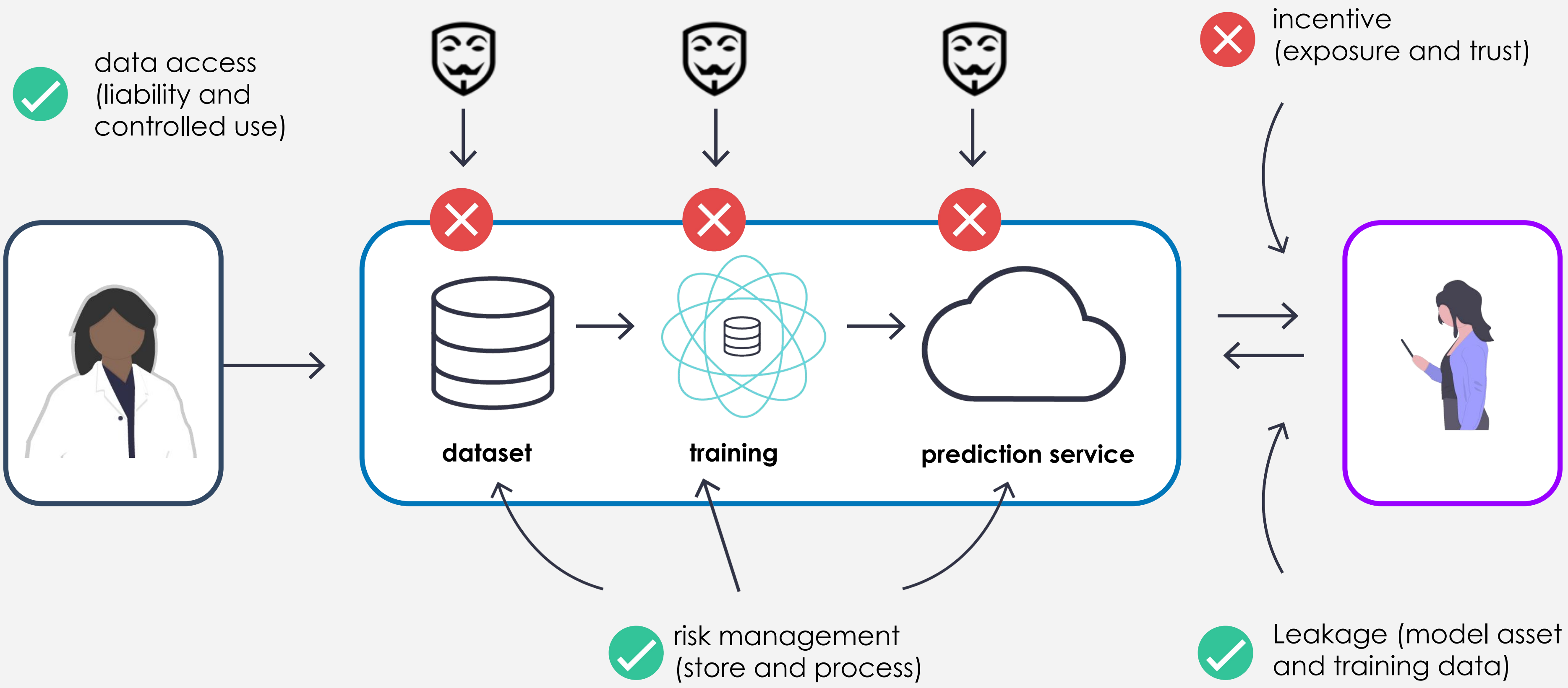
```
optimizer = DPGradientDescentGaussianOptimizer(
        l2_norm_clip,
        noise_multiplier,
        FLAGS.microbatches,
        FLAGS.learning_rate)

loss = tf.keras.losses.CategoricalCrossentropy(
        from_logits=True, reduction=tf.losses.Reduction.NONE)

model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

- TF Privacy is on PySyft-TensorFlow's roadmap

Further Reading About TensorFlow Privacy

- Introducing TensorFlow Privacy: Learning with Differential Privacy for Training Data
- Cleverhans-blog: Privacy and machine learning: two unexpected allies?
- DPSGD: “Deep Learning with Differential Privacy” Abadi et al.
- PATE: “Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data” Papernot et al.

Remote Execution + Differential Privacy



Open Questions

Differential privacy does not protect from inferential privacy loss.

If other people give up their data to train models, data about you can be easily/cheaply predicted as long as it belongs to the same distribution -- the privacy loss is contagious.

We currently don’t have any good technical solutions to this problem.

PySyft TensorFlow Roadmap

- Federated Learning Support
- Encrypted deep learning: integration with TF Encrypted
- Differential Privacy: integration with TF Privacy
- Add tf.data support

PySyft

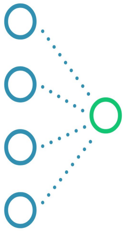
Use Case

Opportunity:

- You want to train a model on sensitive patients' data.

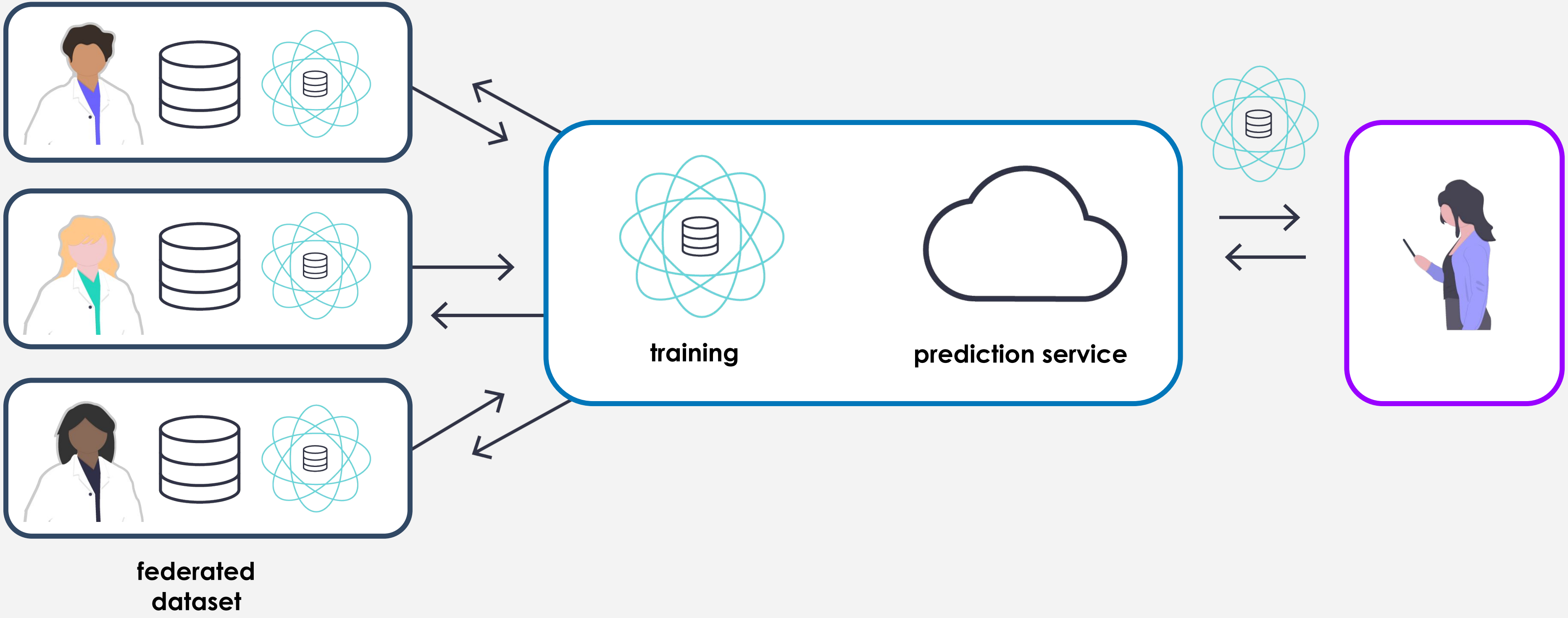
Problems:

- The data is **siloed** across **several hospitals.**
- **The data can't be accessed** directly for liability reasons.



Federated Learning

Federated Learning



Towards Federated Learning at Scale: System Design

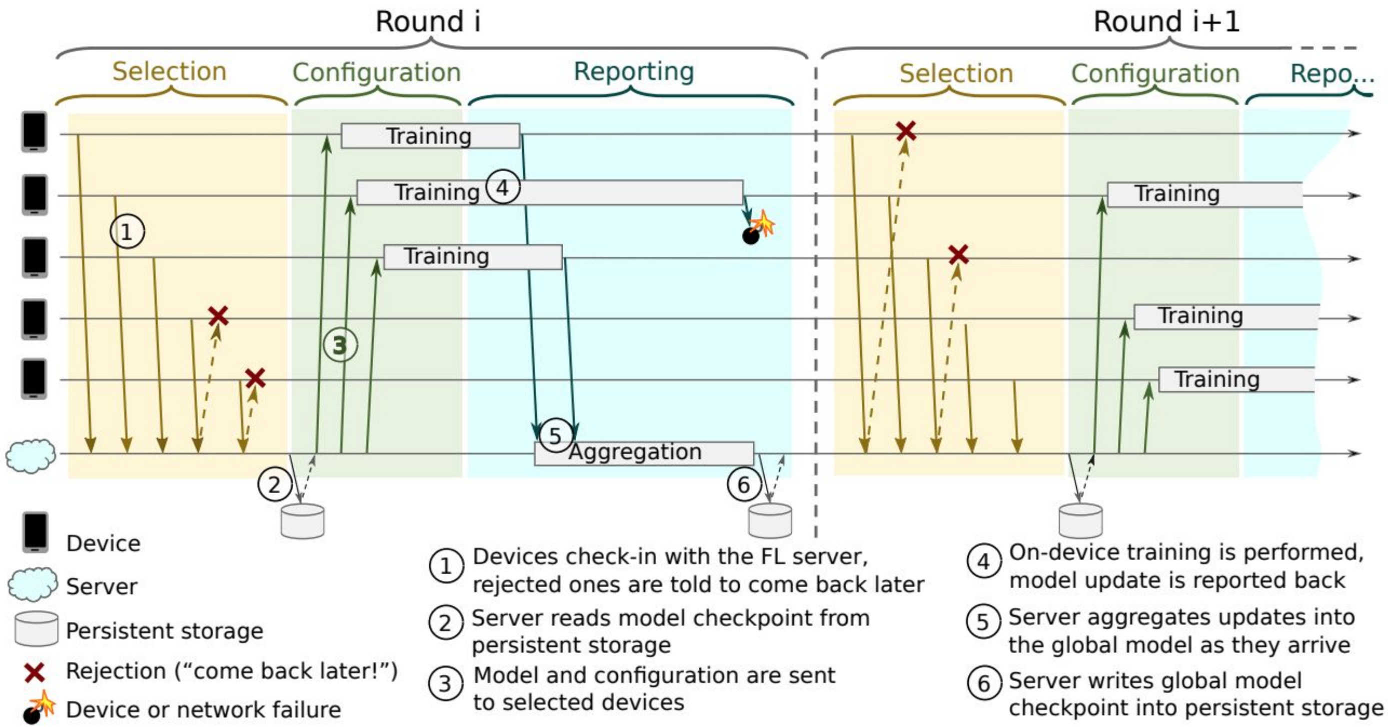


Figure 1: Federated Learning Protocol

[Bonawitz et al. 2019]
[TensorFlow Federated] on GitHub (Google AI -- Federated Learning Team)

TF Encrypted

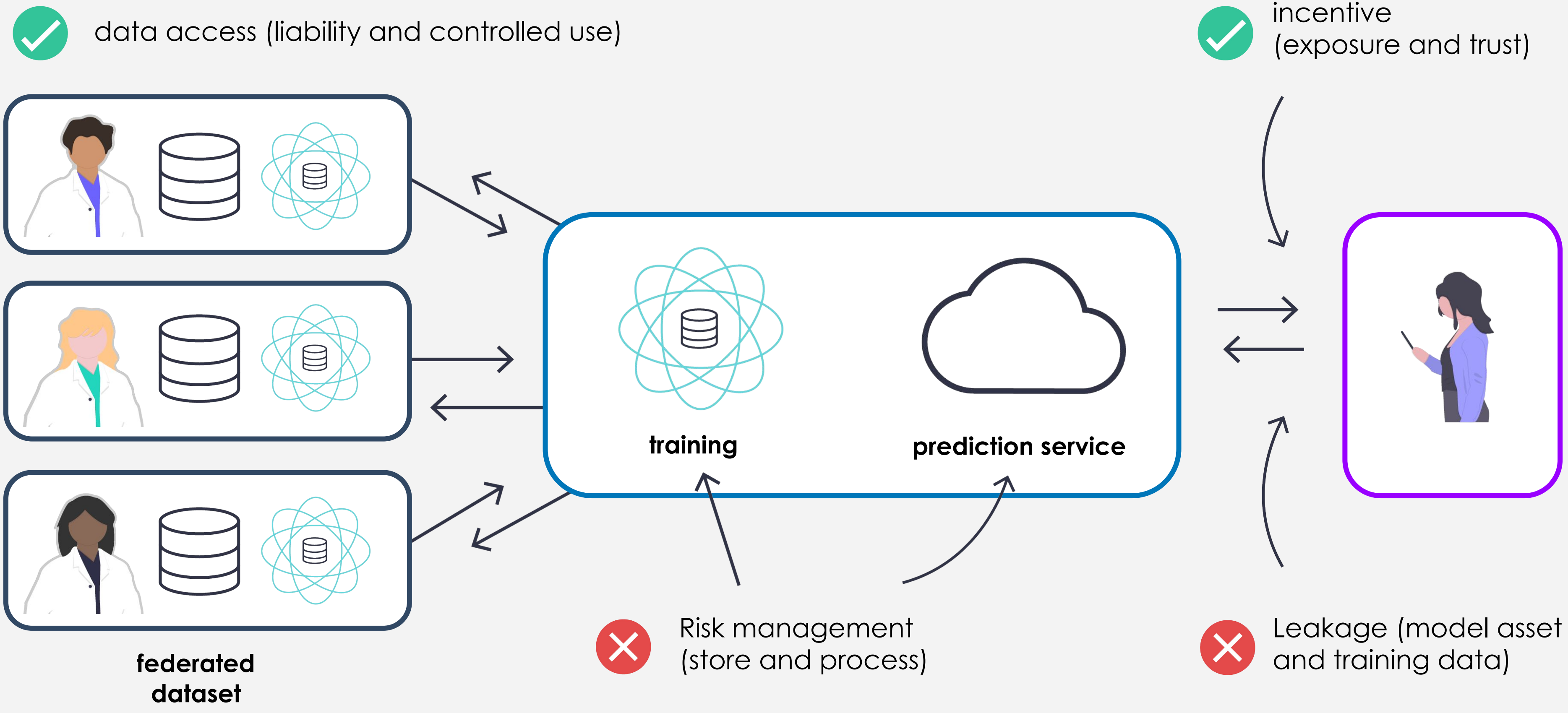
TF Encrypted

A Framework for Deep Learning on Encrypted Data

The benefits:

- Usability
- Integration
- Performance
- Extensibility

Federated Learning



Privacy Leakage With Federated Learning

Problem:

- Can reconstruct data based on gradients updates.

Solution:

- Distribute trust among data owners using **secure aggregation**

Secure Computation

<u>Objective</u>

Evaluate a publicly-known function $f({}^{*}x, w) = y$ such that ${}^{*}x$, w, and y are all kept secret[1]

<u>Method</u>

Encryptor **Decryptor**

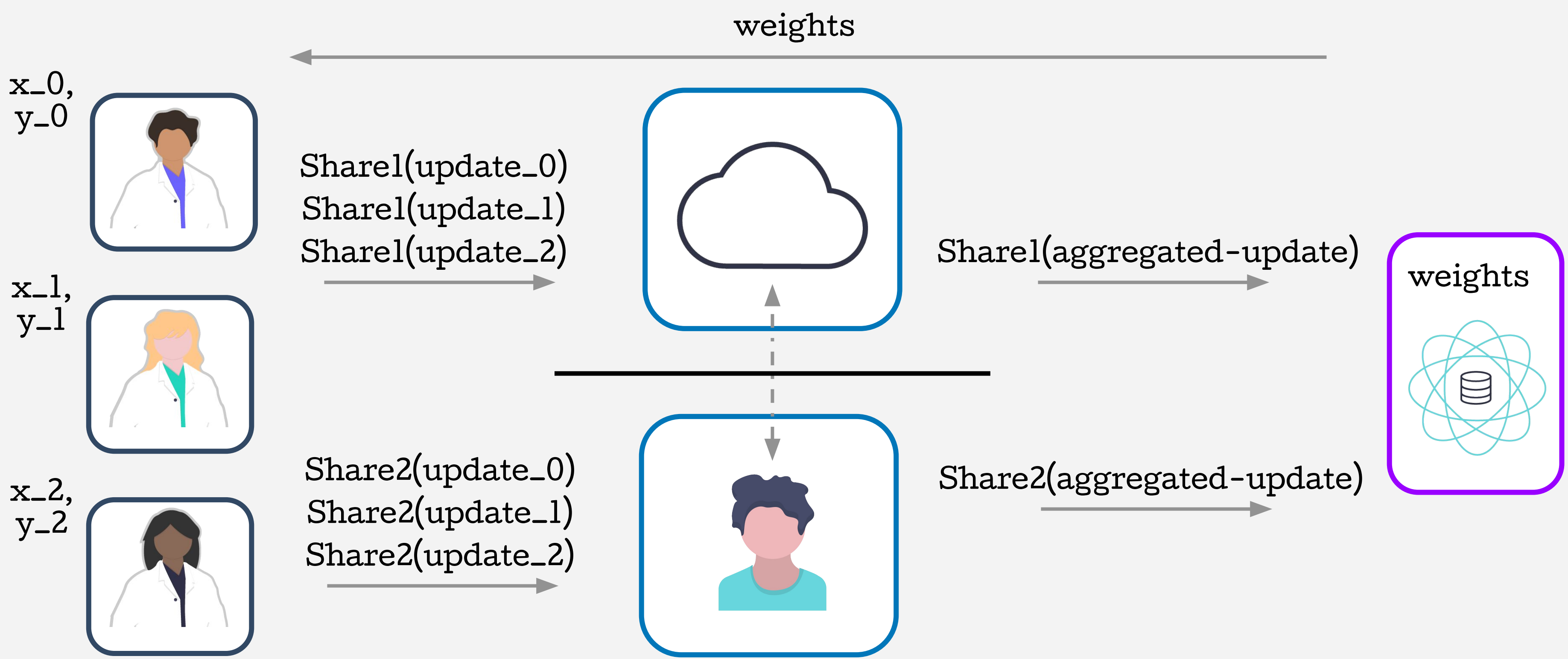
Find a function $\boldsymbol{\varphi}$ such that $\boldsymbol{\varphi}$ is homomorphic through f and $\boldsymbol{\varphi}^{-1}$ is crypto-hard to determine without exact knowledge of some or all inputs.

$$f(\boldsymbol{\varphi}(x), \boldsymbol{\varphi}(w)) = \boldsymbol{\varphi}(f(x, w))$$

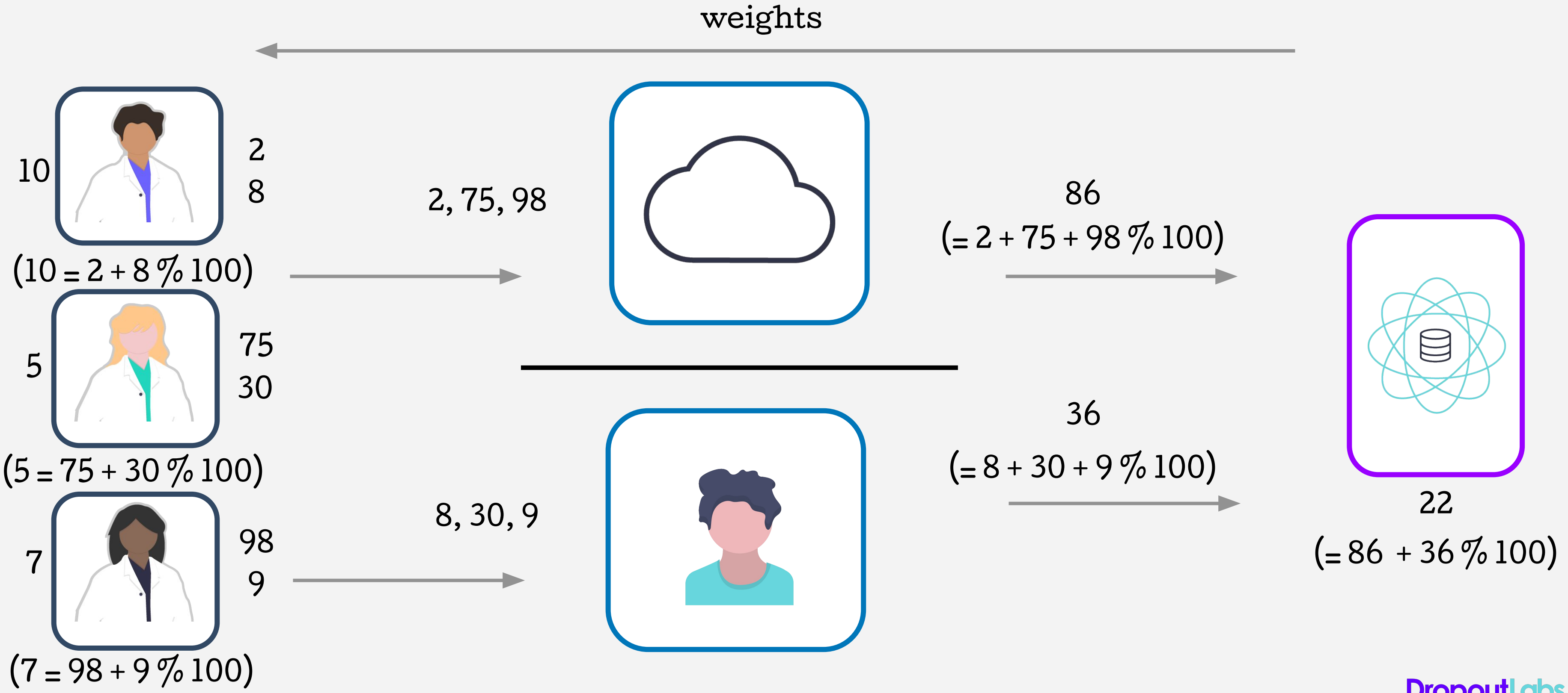
$$(\boldsymbol{\varphi}^{-1} \circ f)(\boldsymbol{\varphi}(x), \boldsymbol{\varphi}(w)) = f(x, w)$$

1: cryptographically-hard to learn for relevant parties

Participants



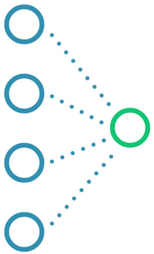
Participants



DropoutLabs

Secure Federated Learning Tutorial

https://github.com/dropoutlabs/tf-world-tutorial/tree/master/federated-learning



Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

Reptile: Pseudocode

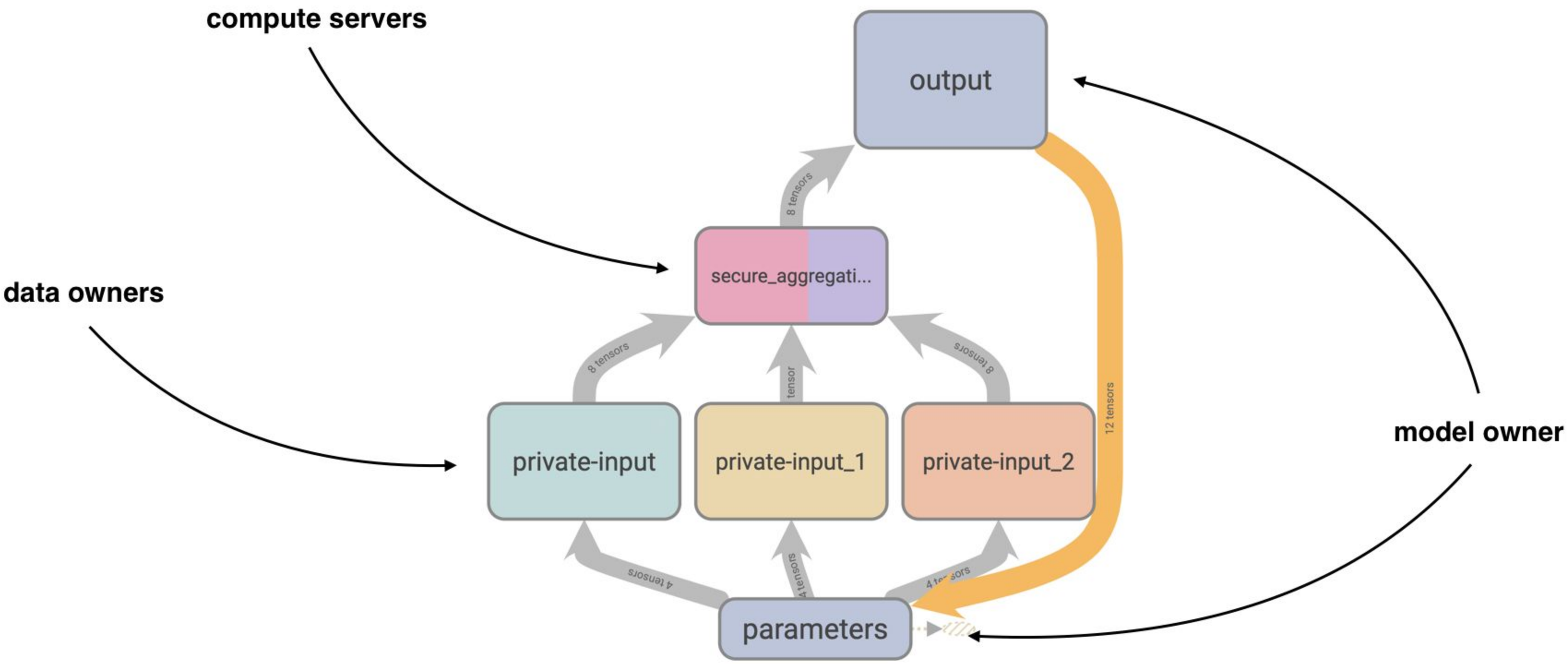
Reptile model function:

Sample batch from DataOwner's dataset
Run *k* steps of SGD (or whichever optimizer has been specified
Return the new weights

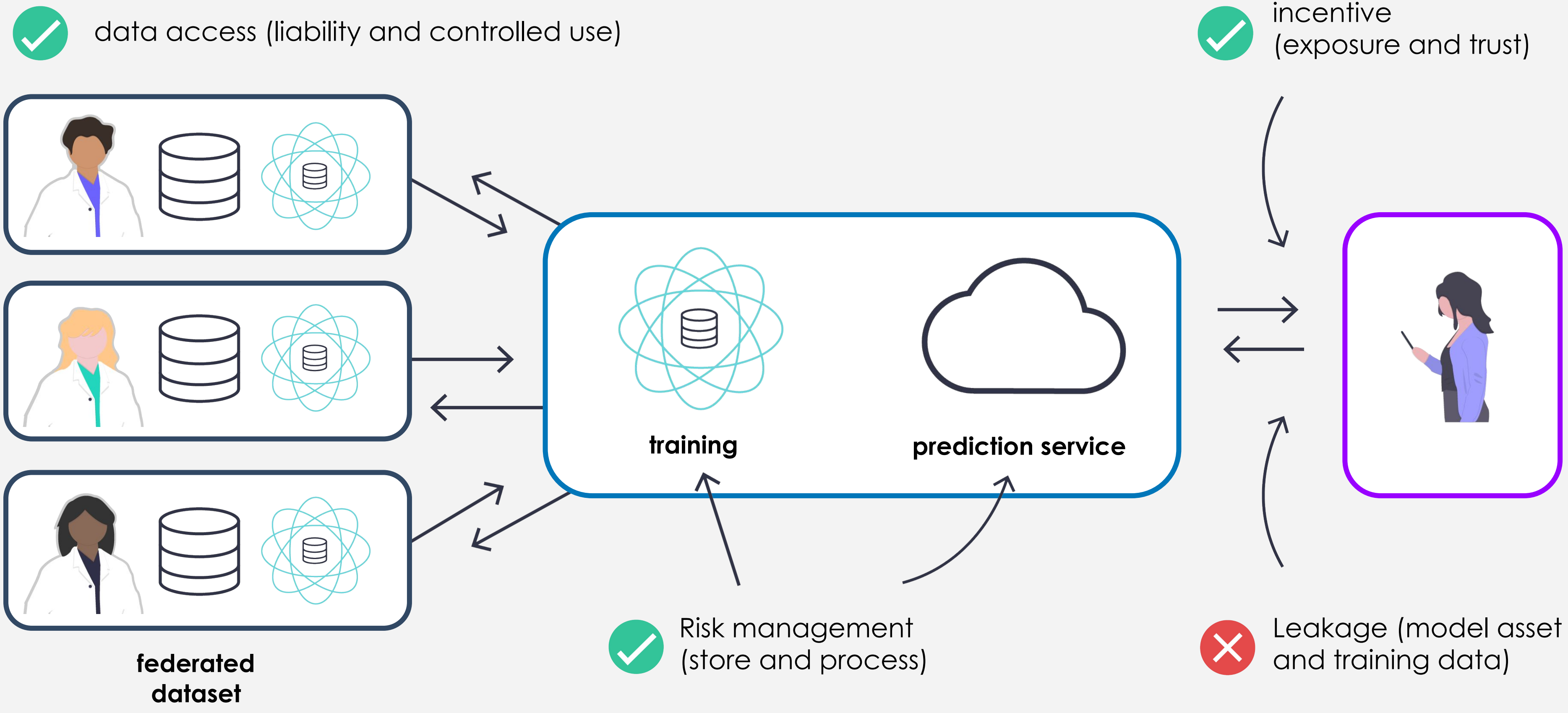
Reptile aggregation function:

Securely average the model weights
Subtract newly averaged weights from old weights for approx. gradient
Update master model with this approximate gradient using SGD.

Overall Computation



Secure Federated Learning



Encrypted Predictions

Third Use Case

Opportunity:

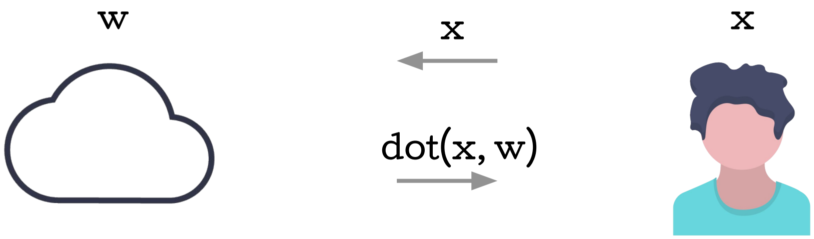
- Provide skin cancer detection to new users.

Problems:

- Users won’t share their data for privacy reasons.
- Your model is an asset. Ergo, you don’t want to send it to their device.

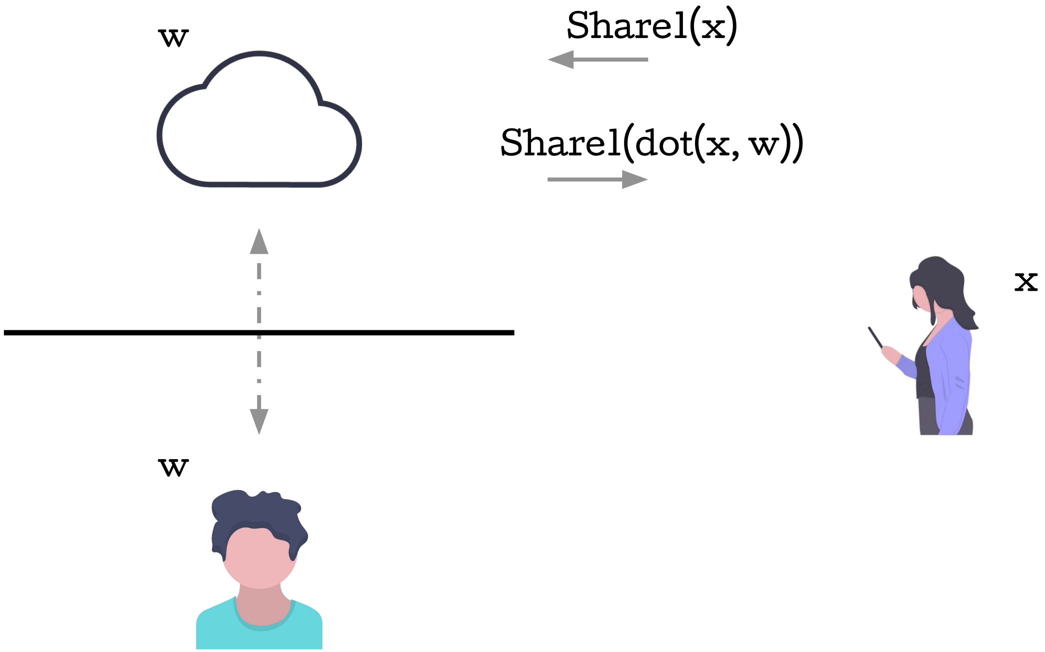
Secure Computation

Prediction with Linear Model



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

... using Secret Sharing



… using Secret Sharing

w

Share1(x)

Share1(dot(x, w))

homomorphic secret sharing

public multiplication

private addition

$$\mathrm{dot}\left(\begin{bmatrix}\mathrm{Share}(x0) & \mathrm{Share}(x1) & \mathrm{Share}(x2)\end{bmatrix}, \begin{bmatrix} w0 \\ w1 \\ w2 \end{bmatrix}\right) = \mathrm{Share}(x0)*w0 + \mathrm{Share}(x1)*w1 + \mathrm{Share}(x2)*w2$$

$$= \mathrm{Share}(x0*w0) + \mathrm{Share}(x1*w1) + \mathrm{Share}(x2*w2)$$

$$= \mathrm{Share}(x0*w0 + x1*w1 + x2*w2)$$

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

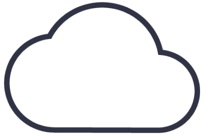
$x1$ $\quad$ $y1$ $\quad$ $z1 = x1 + y1$

$x2$ $\quad$ $y2$ $\quad$ $z2 = x2 + y2$

$x = x1 + x2$ $\quad$ $y = y1 + y2$

$$\begin{aligned} x + y \\ &= (x1 + x2) + (y1 + y2) \\ &= (x1 + y1) + (x2 + y2) \\ &= z1 + z2 \end{aligned}$$

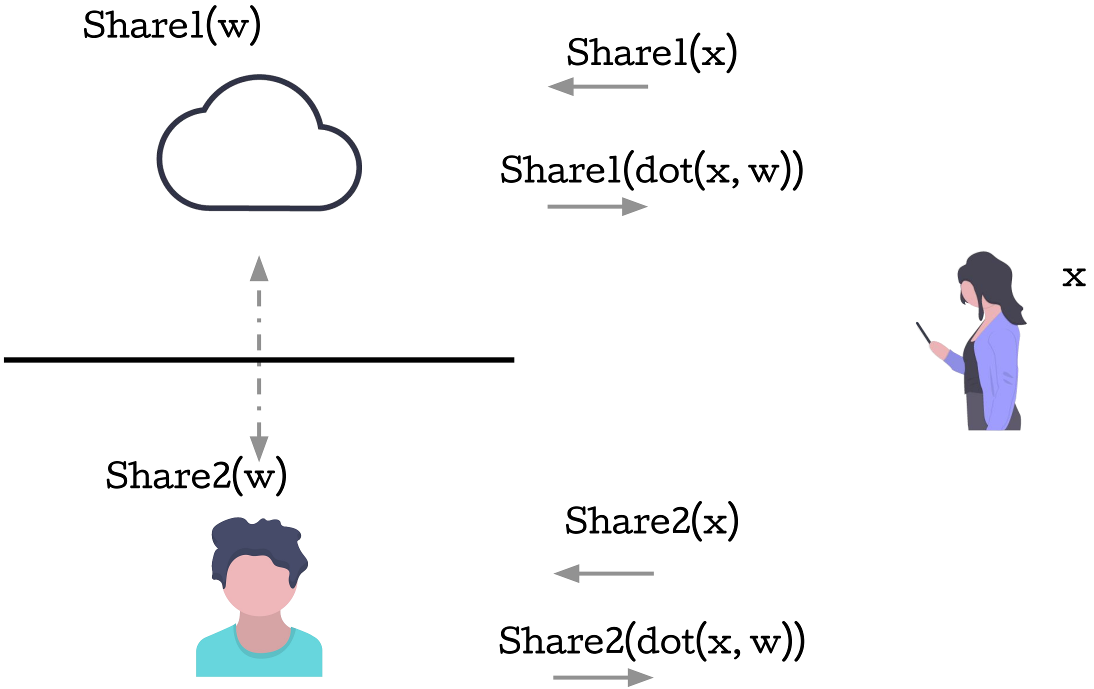
Public Multiplication with Secret Sharing

	$x1$	w	$z1 = x1 * w$
	$x2$	w	$z2 = x2 * w$

$x = x1 + x2$

$$\begin{aligned} & x * w \\ &= (x1 + x2) * w \\ &= (x1 * w) + (x2 * w) \\ &= z1 + z2 \end{aligned}$$

… using Secret Sharing, with Private Model



... using Secret Sharing, with Private Model

Share1($\mathbf{w}$)

Share1(x)

Share1(dot(x, w))

private multiplication

		Share(w0)	
dot(Share1(x0) Share1(x1) Share1(x2) ,		Share(w1)	) = Share1(x0)*Share(w0) +
		Share(w2)	= Share1(x0*w0) +
			= Share1(x0*w0 + x1*w1 + x2*w2)

DropoutLabs

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

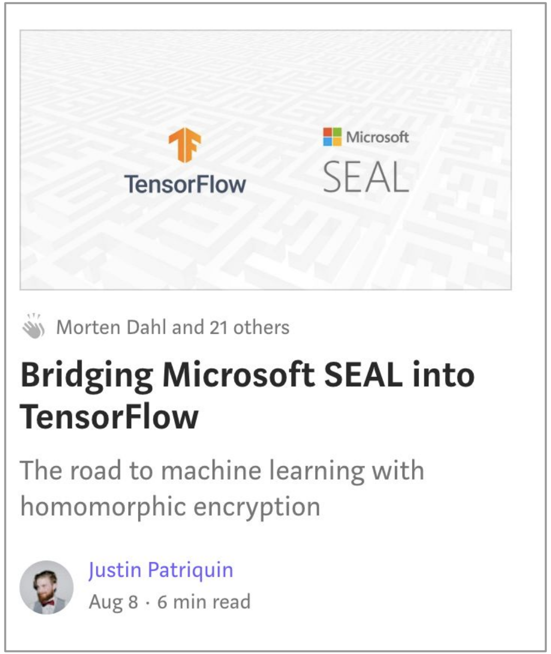
Encrypted Predictions Tutorial

https://github.com/dropoutlabs/tf-world-tutorial/tree/master/federated-learning

Characteristics

	Secure Enclaves	Garbled Circuits	Homomorphic Encryption	Secret Shares
Compute	Low	Medium	Very High	Low
Network	Low	Medium	Low	High
Operations	Any	Boolean	Arithmetic	Arithmetic
Runs on	Hardware	Software	Software	Software
Parties	2+	2+	2+	3+

TF Seal: Homomorphic Encryption coming soon to TF Encrypted!



GitHub:
https://github.com/tf-encrypted/tf-seal

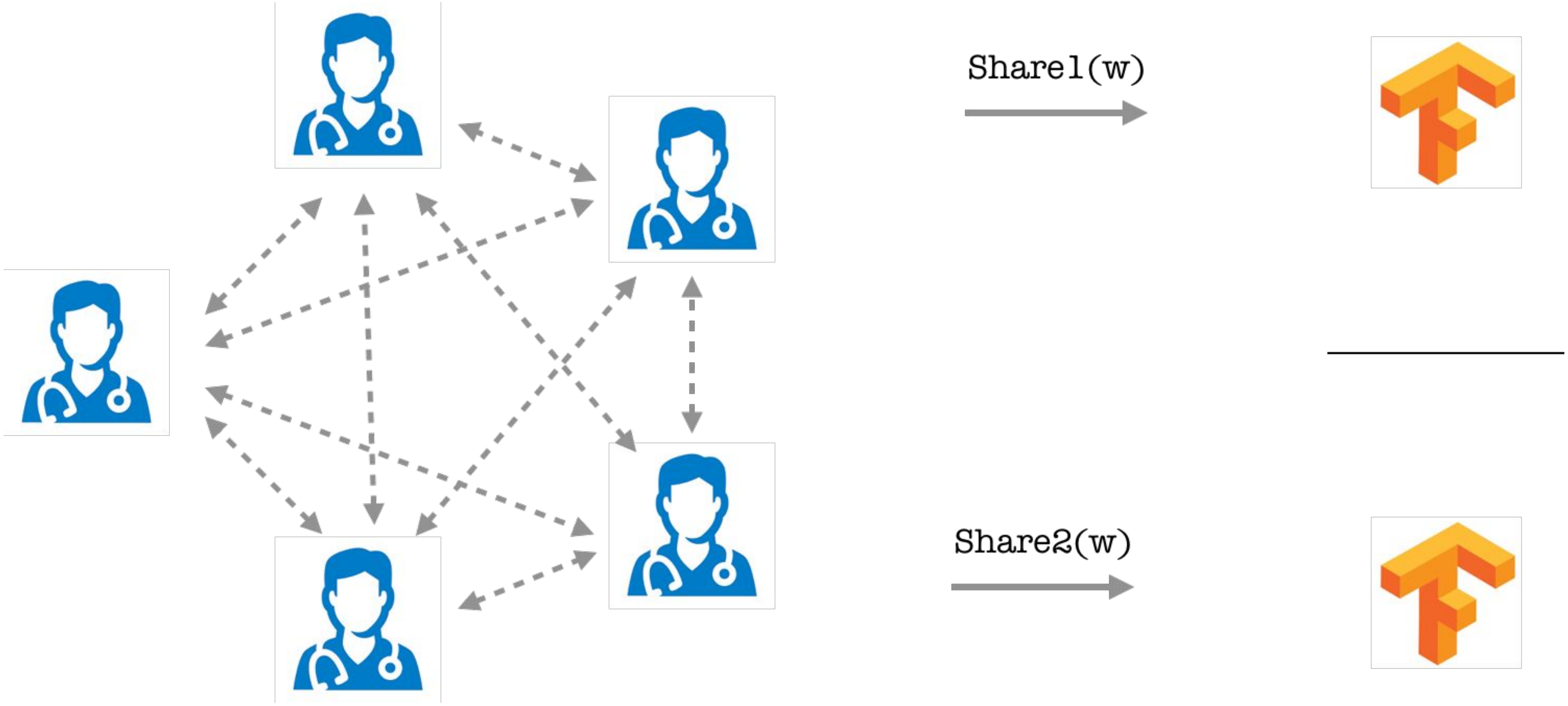
Medium:
https://medium.com/dropoutlabs/bridging-microsoft-seal-into-tensorflow-b04cc2761ad4

TF Trusted: Secure Enclaves coming soon to TF Encrypted!

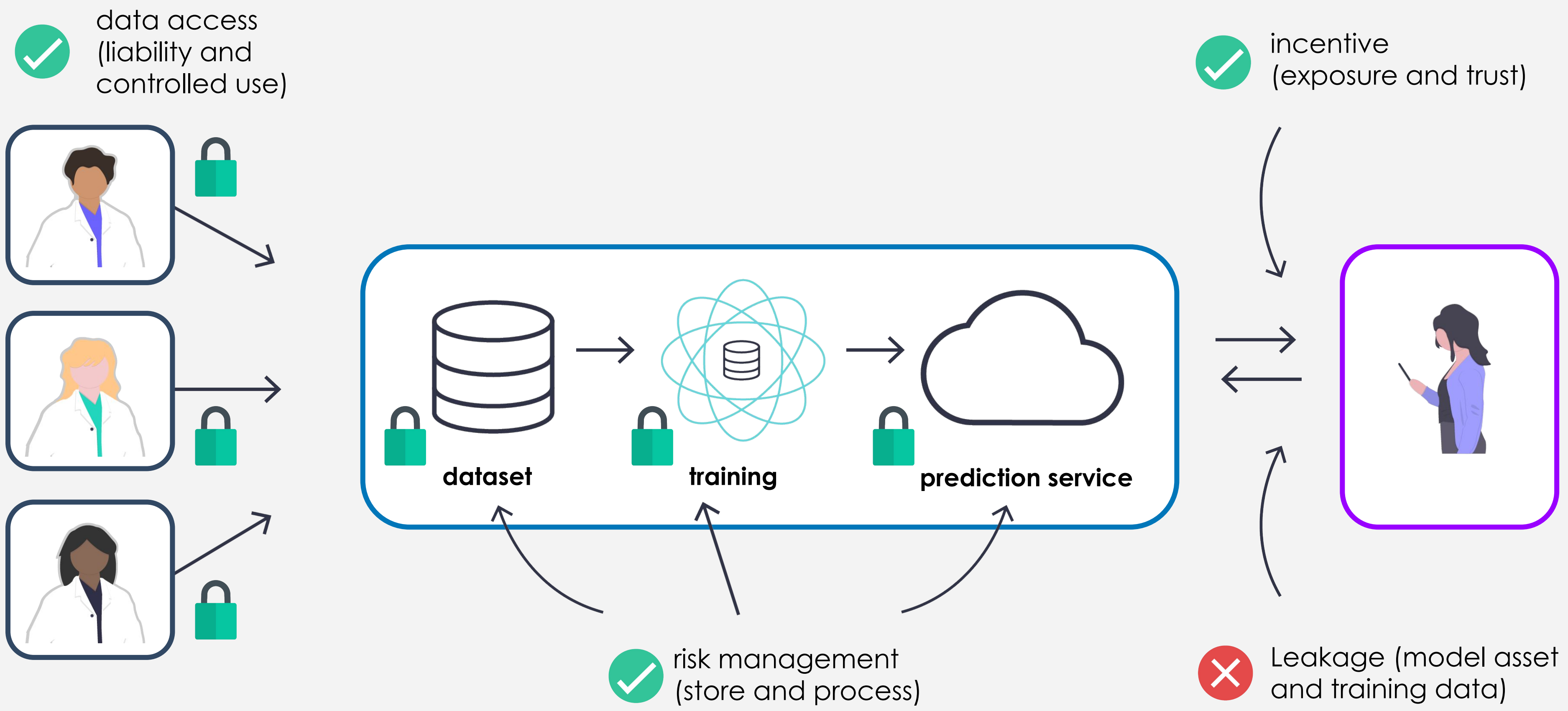
Github:
https://github.com/dropoutlabs/tf-trusted

Medium:
https://medium.com/dropoutlabs/secure-logistic-regression-mpc-vs-enclave-benchmark-65cf5f03a81f

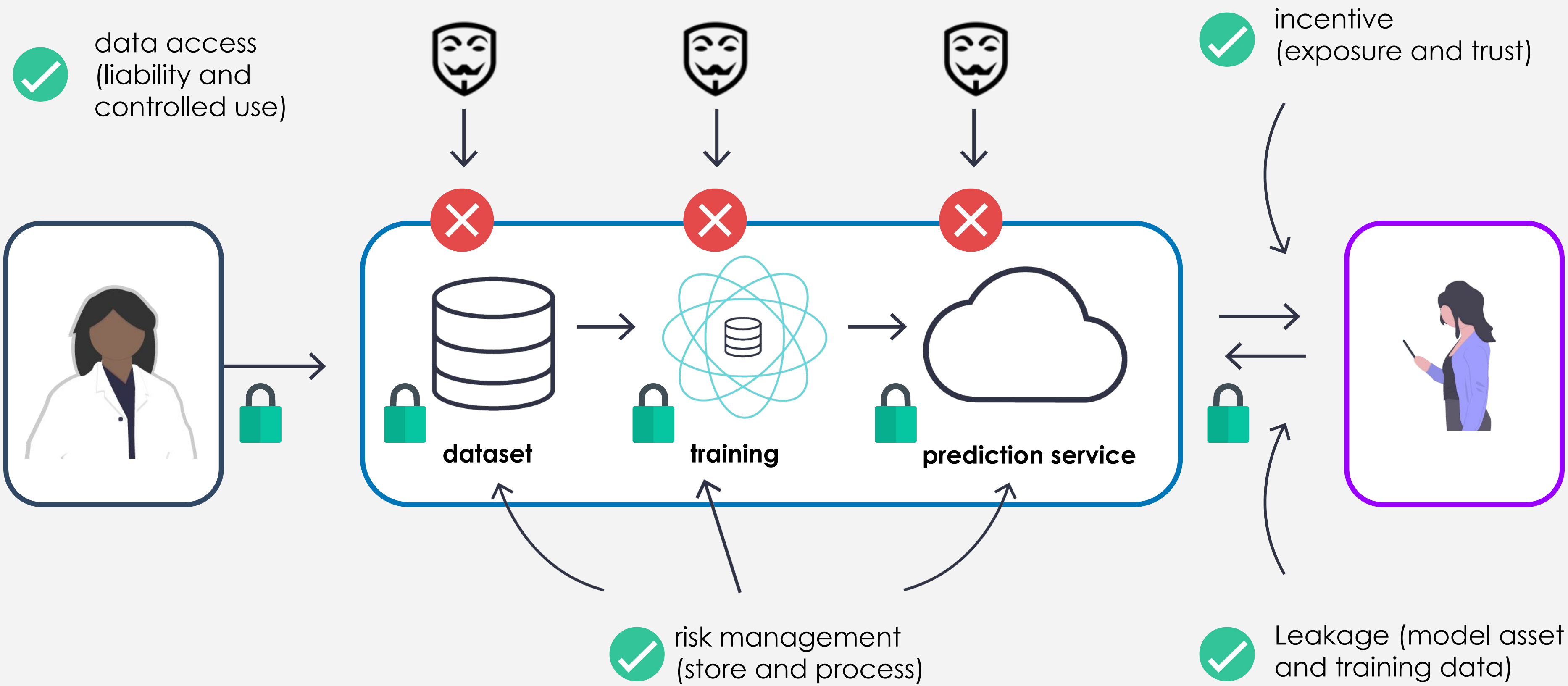
Multi-party training



Secure Computation



Secure Computation + Differential Privacy



Open Questions

Can we make the currently prohibitive performance cost marginal? Or is secure computation fated to only enable use cases that aren't currently possible (e.g. due to regulatory concerns)

Or can we combine MPC with other approaches like federated learning to give it a privacy guarantee? (e.g. secure aggregation from Google Federated Learning team)

Similar to what people are trying to do now with differential privacy, are there alternative definitions of privacy/security that would lead to better performance?

PrivacyAI

www.privacy.ai

PrivacyAI

Contact Us Request Demo

A Platform for Secure, Privacy-Preserving Machine Learning

Build **better models** by working with **more valuable data**

Request Demo

Self-serve access to protected data

Integrated with your existing workflow

Powered by modern privacy techniques

Governed by security policies and permissions