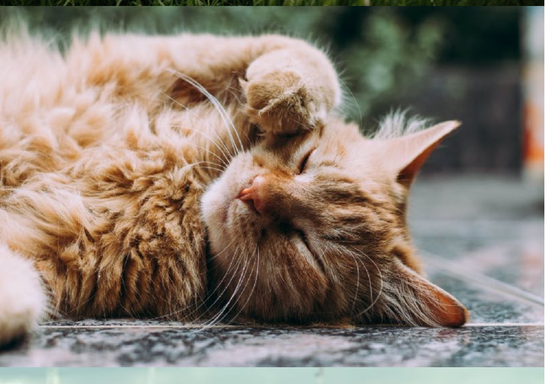
CRYPTEN

Machine Learning on Encrypted Data with **CrypTen**

CRYPTEN

What is machine learning?



FEATURES

MODEL

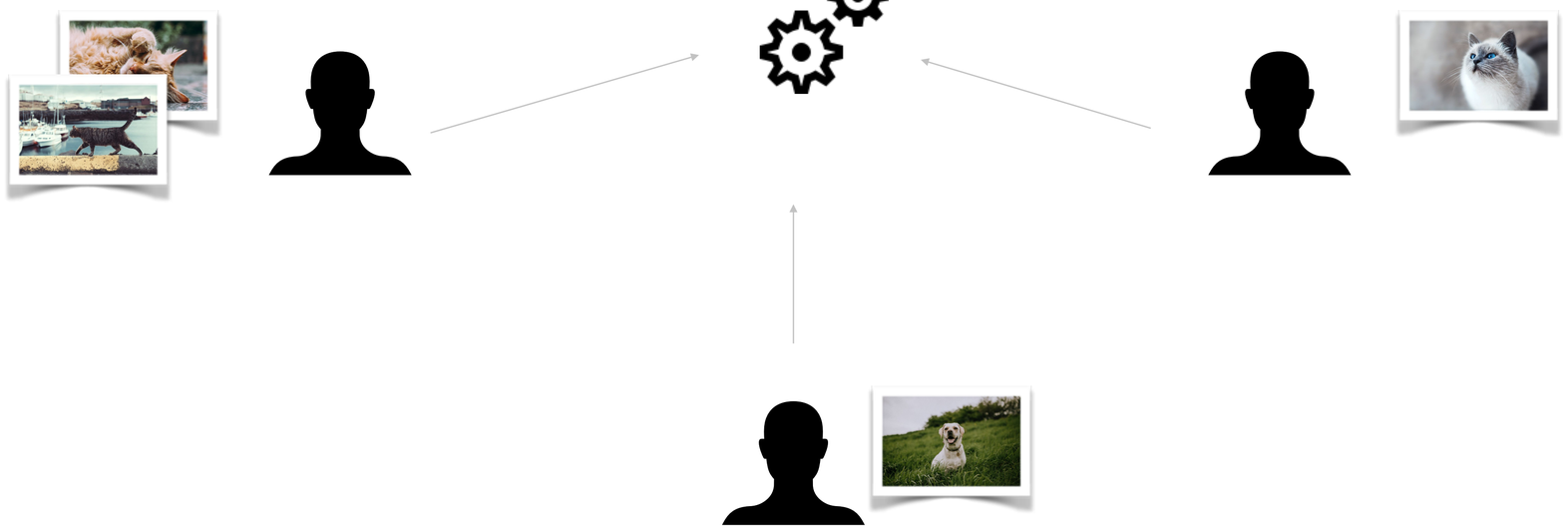
LABELS

cat *or* dog

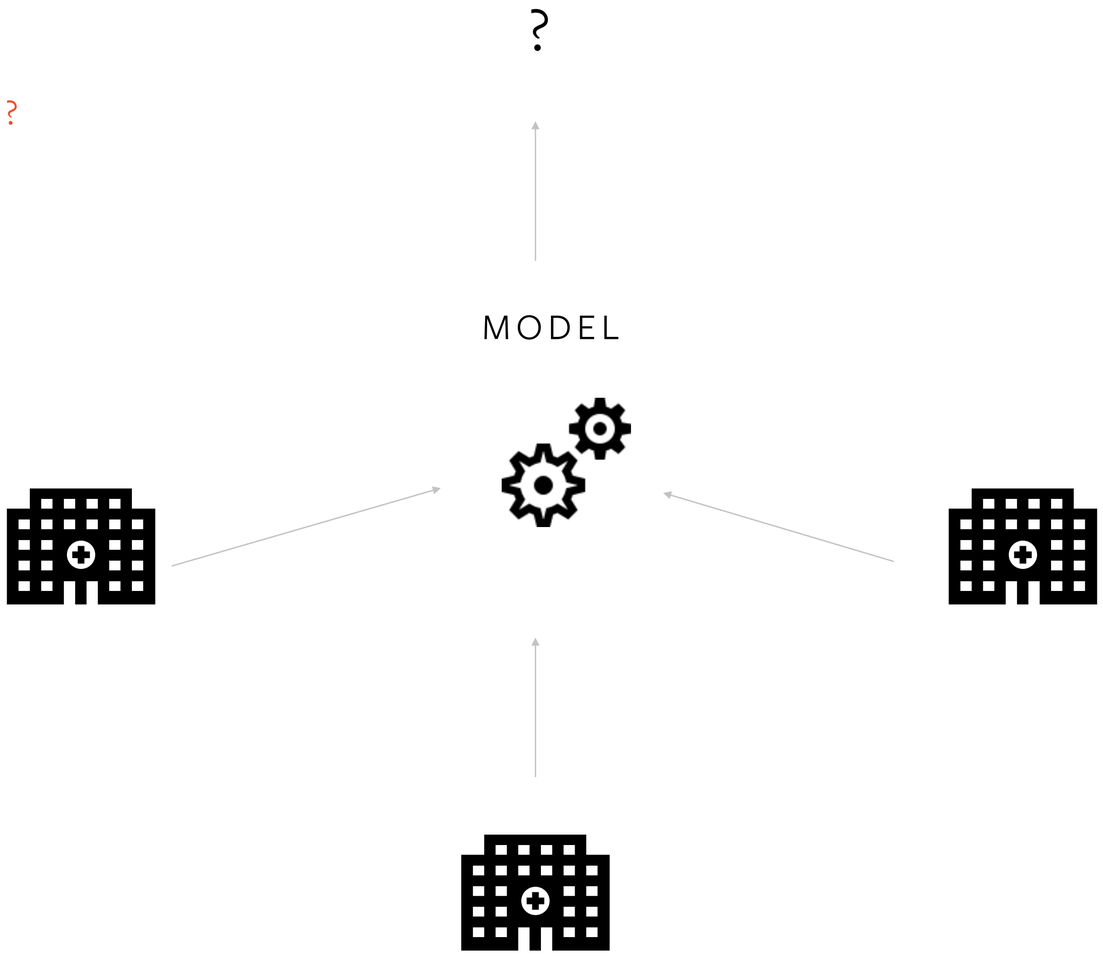
MULTIPLE PARTIES

cat *or* dog

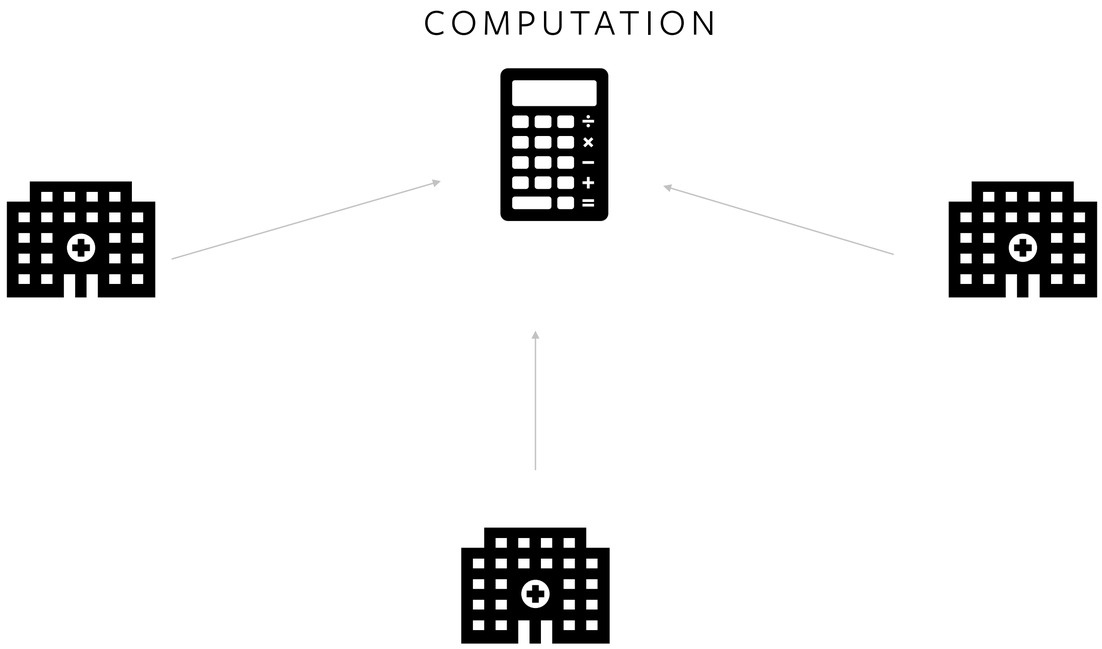
MODEL



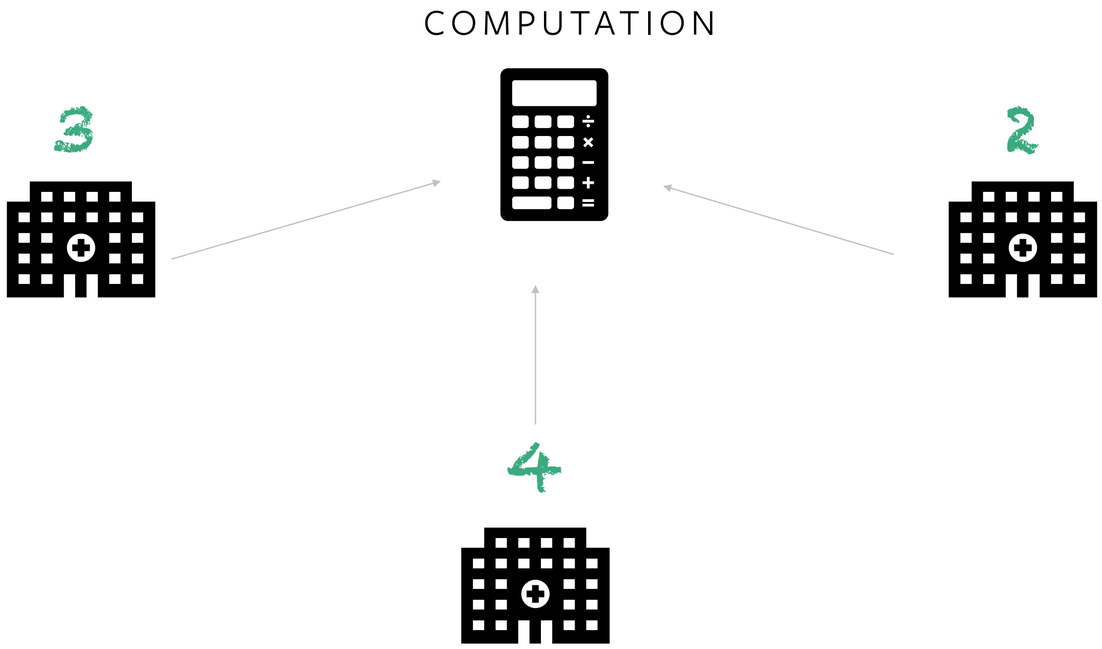
WHAT ABOUT SENSITIVE DATA?



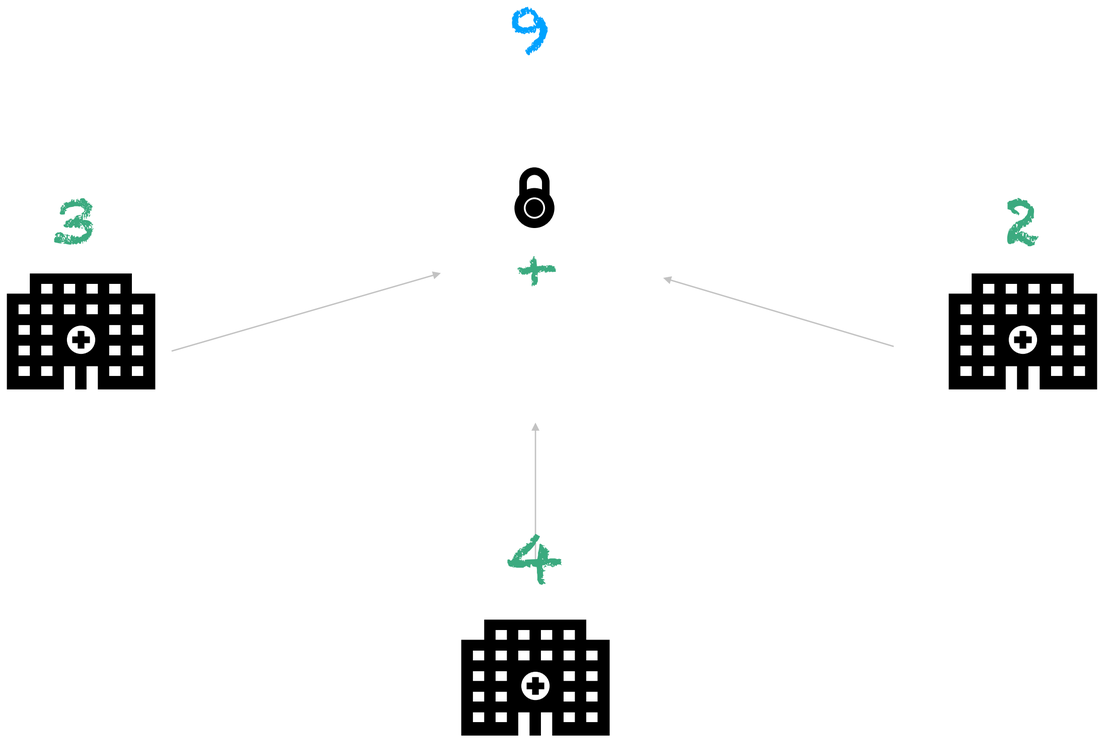
SECURE MULTI-PARTY COMPUTATION



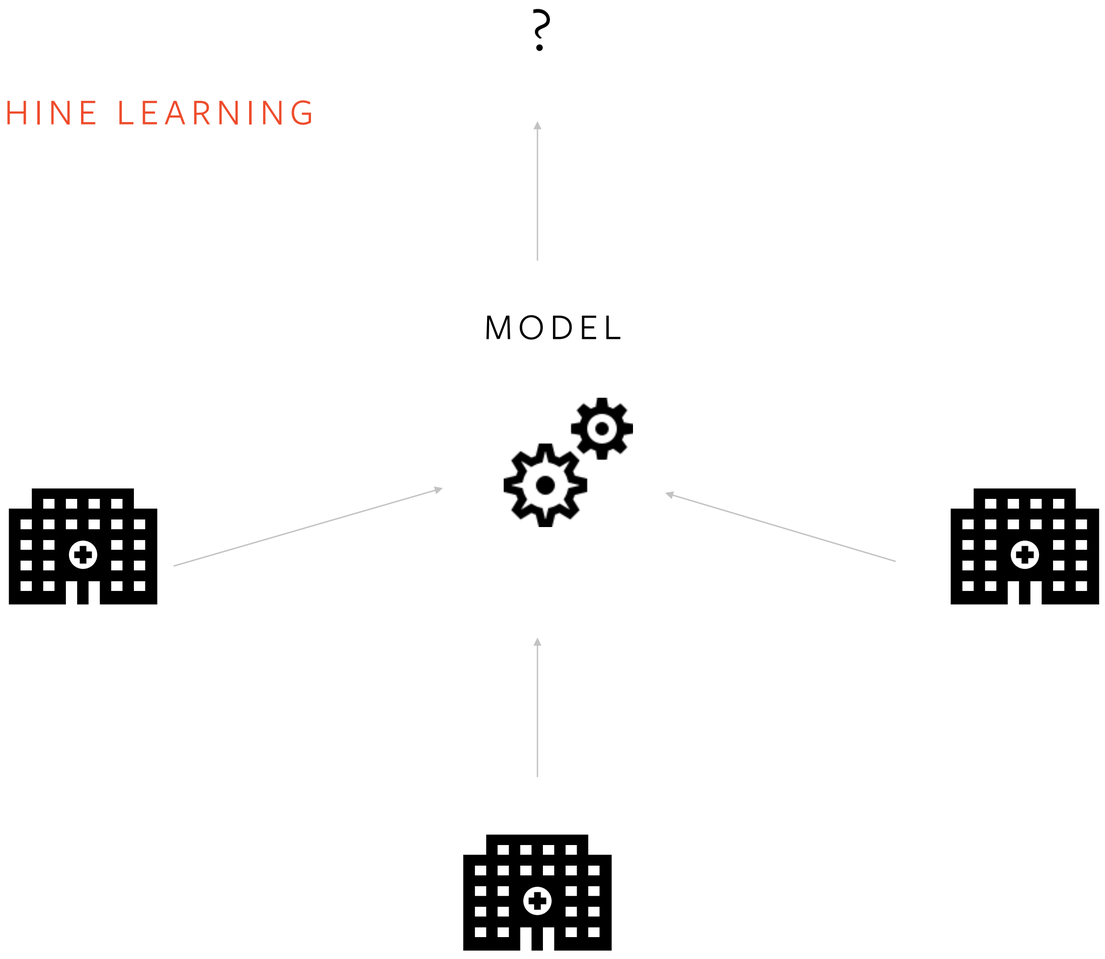
SECURE MULTI-PARTY COMPUTATION



SECURE MULTI-PARTY COMPUTATION



SECURE COMPUTATION & MACHINE LEARNING



TEAM


Brian Knott


Shobha Venkataraman


Mark Ibrahim


Shubho Sengupta


Awni Hannun


Laurens van der Maaten


Xing Zhou

INTRODUCING

CrypTen

WHAT IS CRYPTEN?

CrypTen is a research platform joining

machine learning & secure-computation

DESIGN PRINCIPLES

MACHINE-LEARNING FIRST

Hello **CrypTensor!**

EAGER EXECUTION

No compilers! Easy debugging and learning.

REALISTIC

Parties and communication are real.

PyTorch

1. Make a **Tensor.**

```
x = torch.tensor([1.0, 2.0, 3.0])
```

2. Add Tensors

```
x + y
```

3. Multiply Tensors

```
z = x * y
print(z)
```

CrypTen

1. Make a **CrypTensor.**

```
x_enc = crypten.cryptensor([1.0, 2.0, 3.0]) # encrypts tensor
```

2. Add **CrypTensors**

```
x_enc + y_enc
```

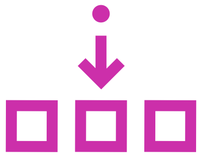
3. Multiply **CrypTensors**

```
z = x_enc * y_enc
print(z.get_plain_text())                    # decrypt
```

DESIGN PRINCIPLES

MACHINE-LEARNING FIRST

Hello CrypTensor!



EAGER EXECUTION

No compilers! Easy debugging and learning.

REALISTIC

Parties and communication are real.

DESIGN PRINCIPLES

MACHINE-LEARNING FIRST

Hello CrypTensor!

EAGER EXECUTION

No compilers! Easy debugging and learning.

REALISTIC

Parties and communication are real.

HELLO CRYPTENSOR

1. Make a **CrypTensor.**

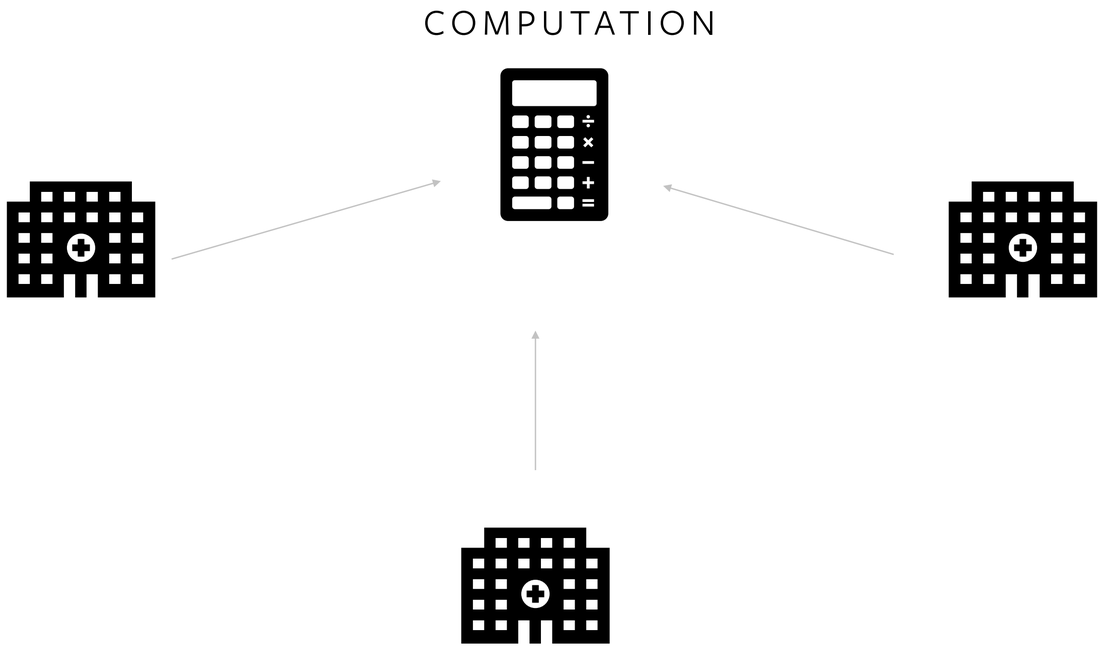
```
import crypten

crypten.init()                          # sets up communication

x_enc = crypten.cryptensor([1.0, 2, 3]) # encrypts tensor

x_dec = x_enc.get_plain_text()          # decrypts tensor
```

SECURE MULTI-PARTY COMPUTATION

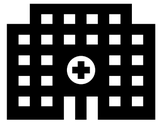


INTUITION: SECURE MULTI-PARTY COMPUTATION

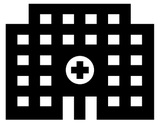
SECURE ADDITION:

7 + 10

7

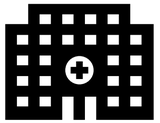


PARTY A



10

PARTY C



PARTY B

INTUITION: SECURE MULTI-PARTY COMPUTATION

PARTY **A** SHARES AN ENCRYPTED VALUE

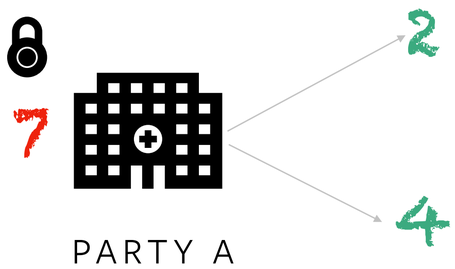
PARTY A

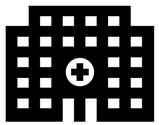
PARTY C

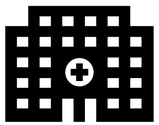
PARTY B

INTUITION: SECURE MULTI-PARTY COMPUTATION

PARTY **A** SHARES AN ENCRYPTED VALUE



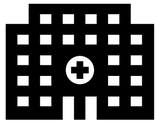




INTUITION: SECURE MULTI-PARTY COMPUTATION

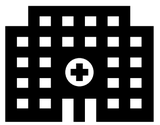
7

PARTY A



2

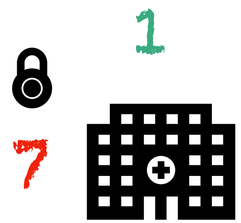
PARTY C



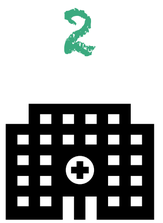
4

PARTY B

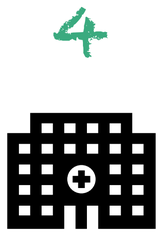
INTUITION: SECURE MULTI-PARTY COMPUTATION



PARTY A



PARTY C



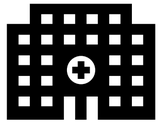
PARTY B

INTUITION: SECURE MULTI-PARTY COMPUTATION

7

1

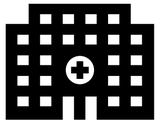
7



PARTY A

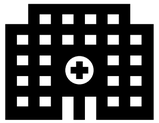
+

2



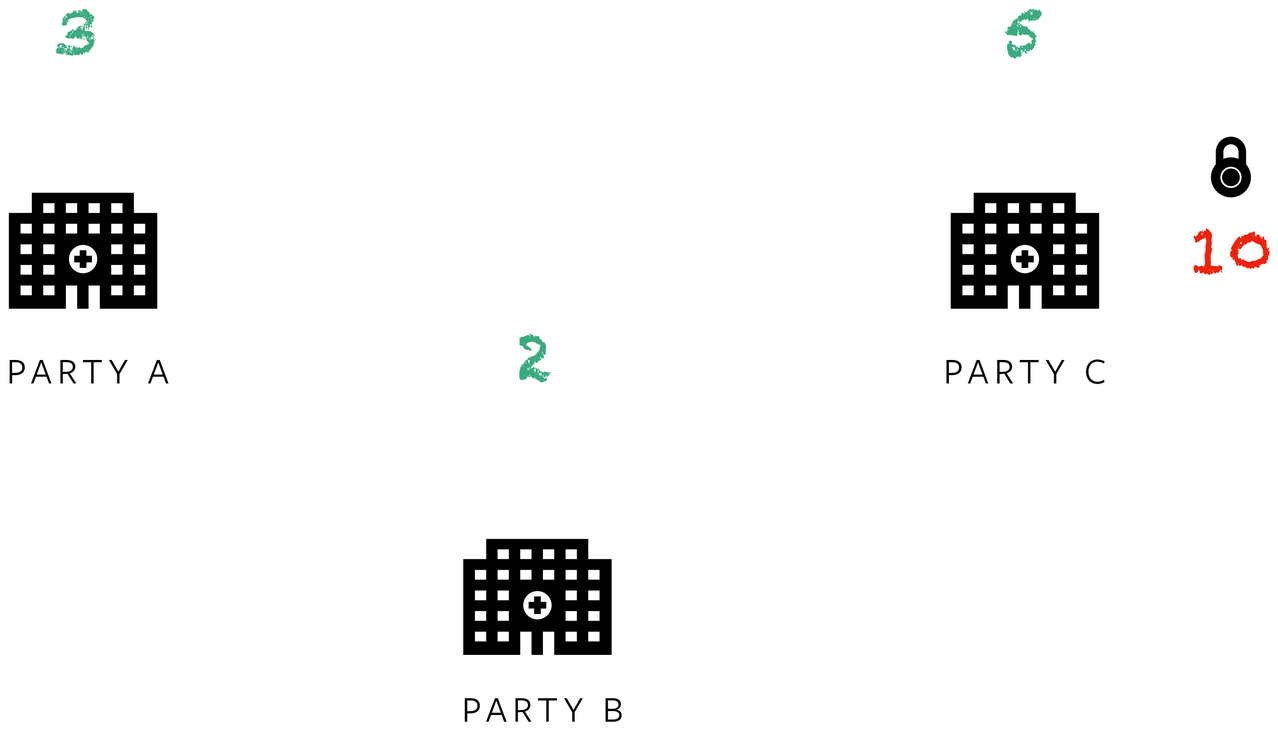
PARTY C

4

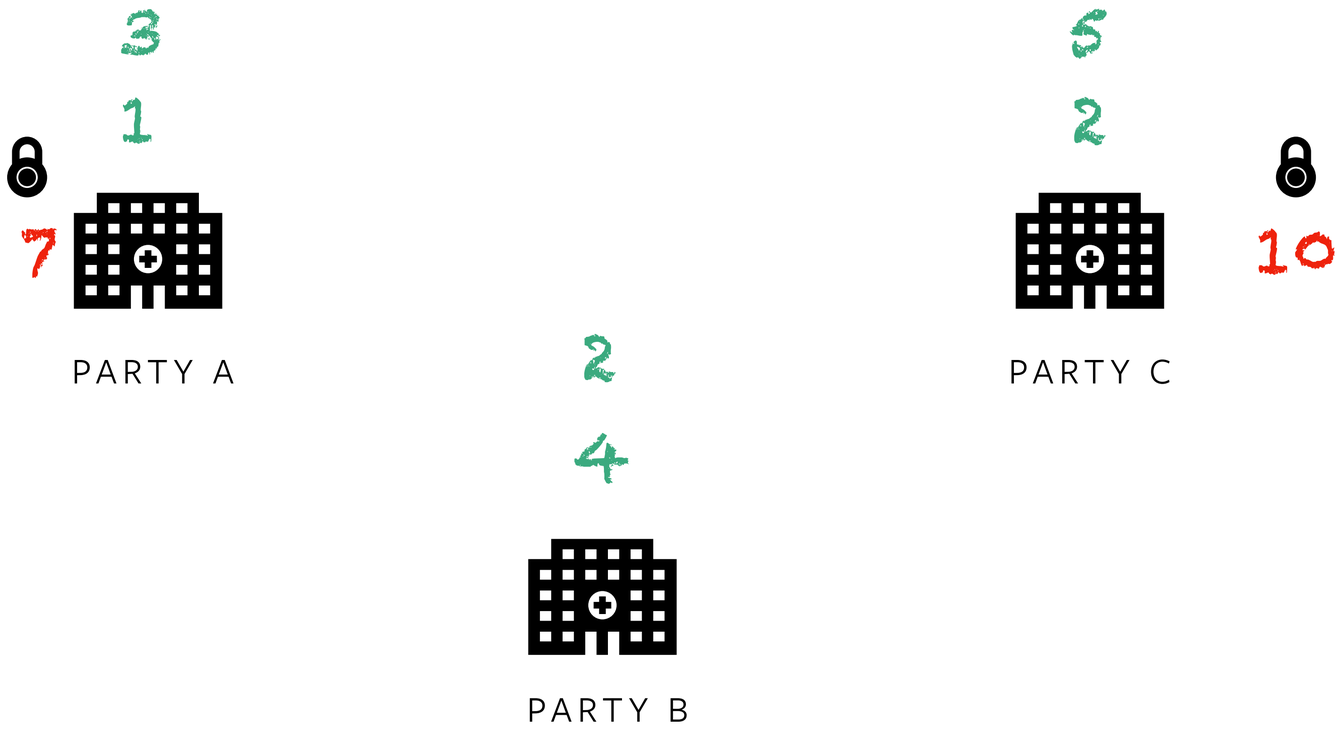


PARTY B

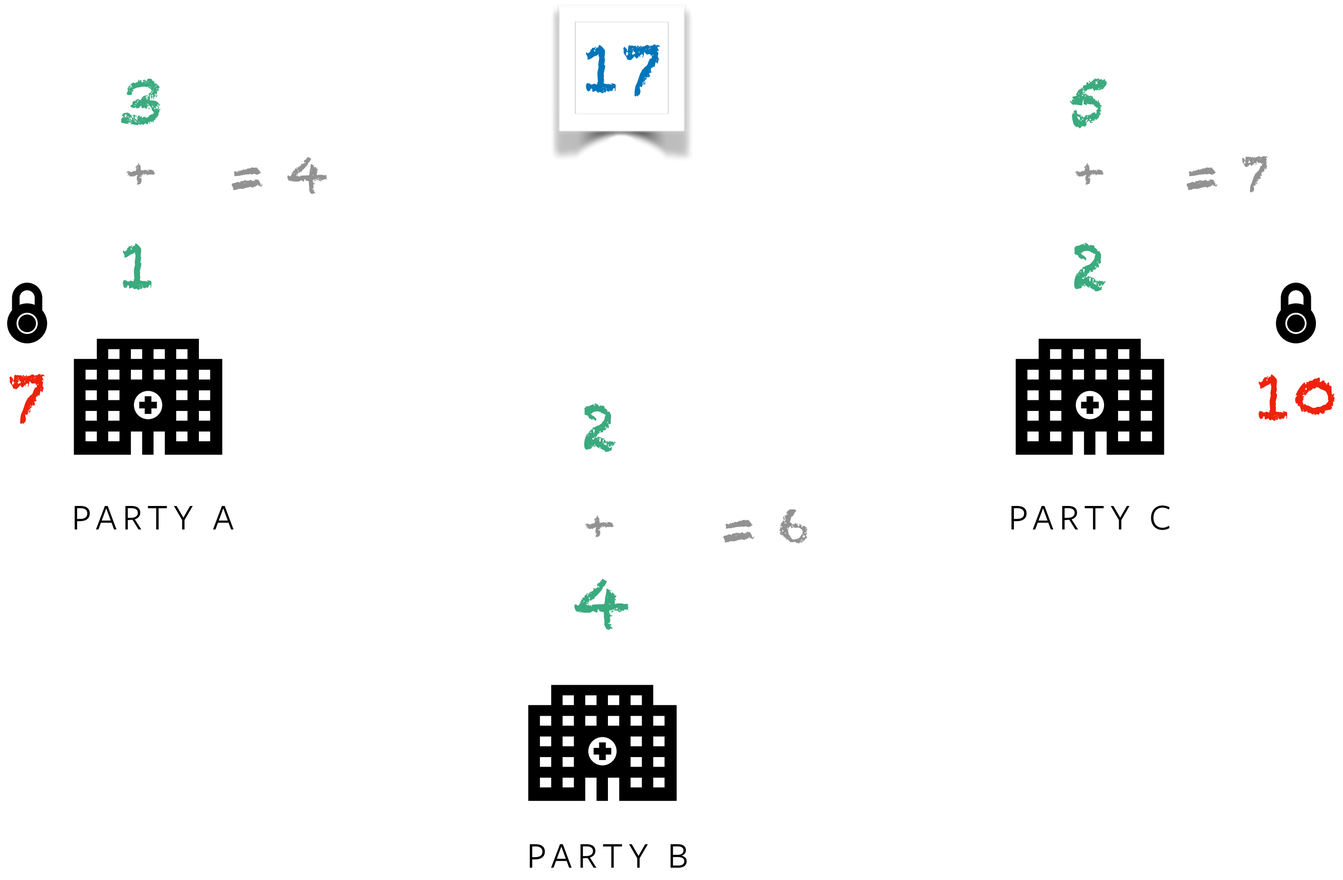
INTUITION: SECURE MULTI-PARTY COMPUTATION



INTUITION: SECURE MULTI-PARTY COMPUTATION



INTUITION: SECURE MULTI-PARTY COMPUTATION



ENCRYPTED TRAINING

1. Make a **CrypTen Model.**
2. Encrypt Data
3. Train!

```
import crypten


crypten.init()                          # sets up communication


class LogisticRegression(crypten.nn.Module):

    def __init__(self):
       super().__init__()
       self.linear = crypten.nn.Linear(28 * 28, 10)

    def forward(self, x):
        return self.linear(x)



model = LogisticRegression().encrypt()         # encrypts tensor
```

Training Across Parties

1. Join **Encrypted Data**
2. Encrypt Model
3. Train!

```
import crypten


crypten.init()                          # sets up communication


alice_images_enc = crypten.load("/tmp/data/alice_images.pth", src=ALICE)
bob_labels_enc = crypten.load("/tmp/data/bob_labels.pth", src=BOB)

model = LogisticRegression().encrypt()
train_model(model, alice_images_enc, bob_labels_enc)
```

CRYPTEN

Photo Credits

Photo by Ludemeula Fernandes on Unsplash
Photo by Willian Justen de Vasconcellos on Unsplash
Photo by The Lucky Neko on Unsplash
Photo by Mitchell Orr on Unsplash