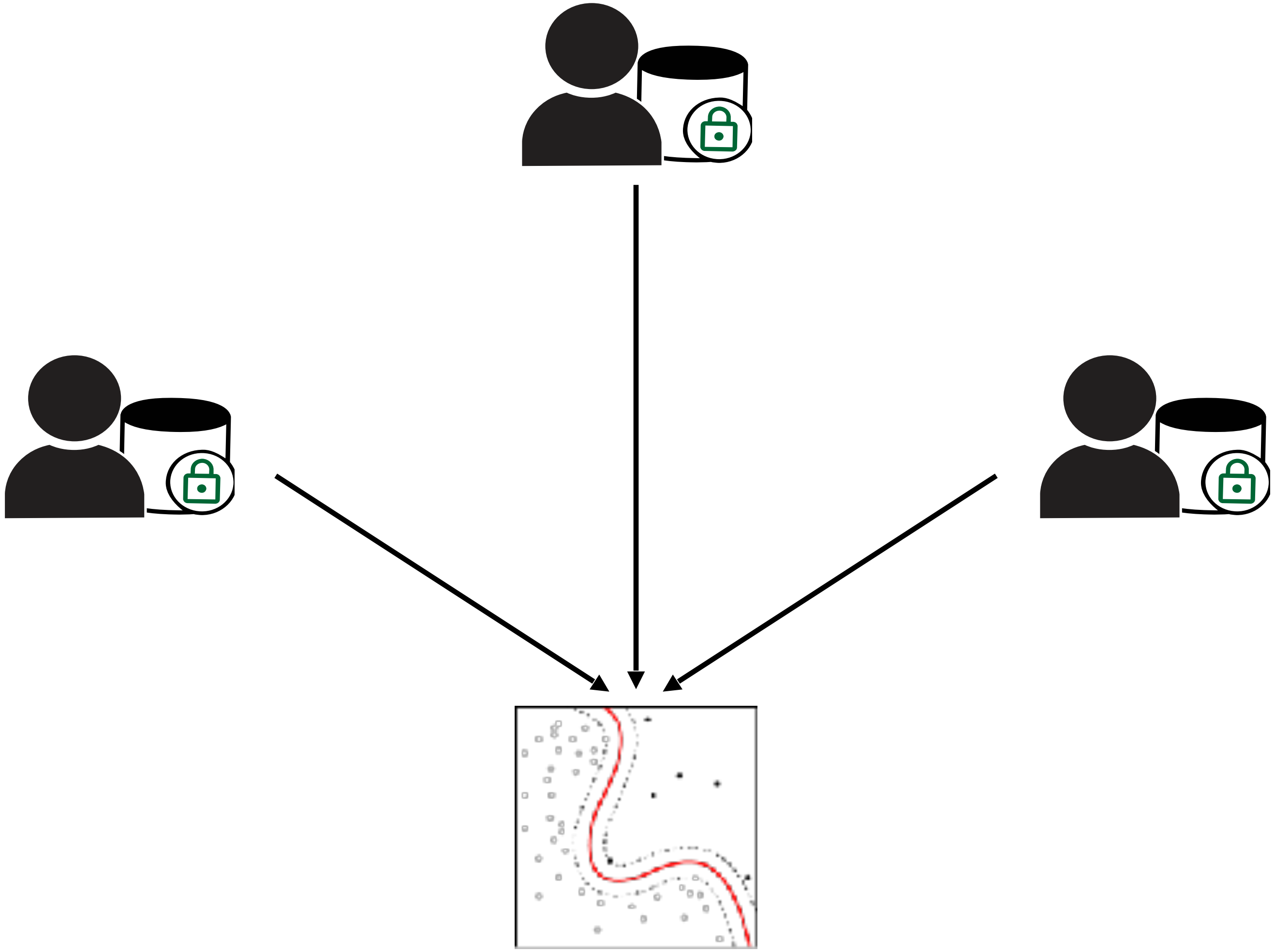
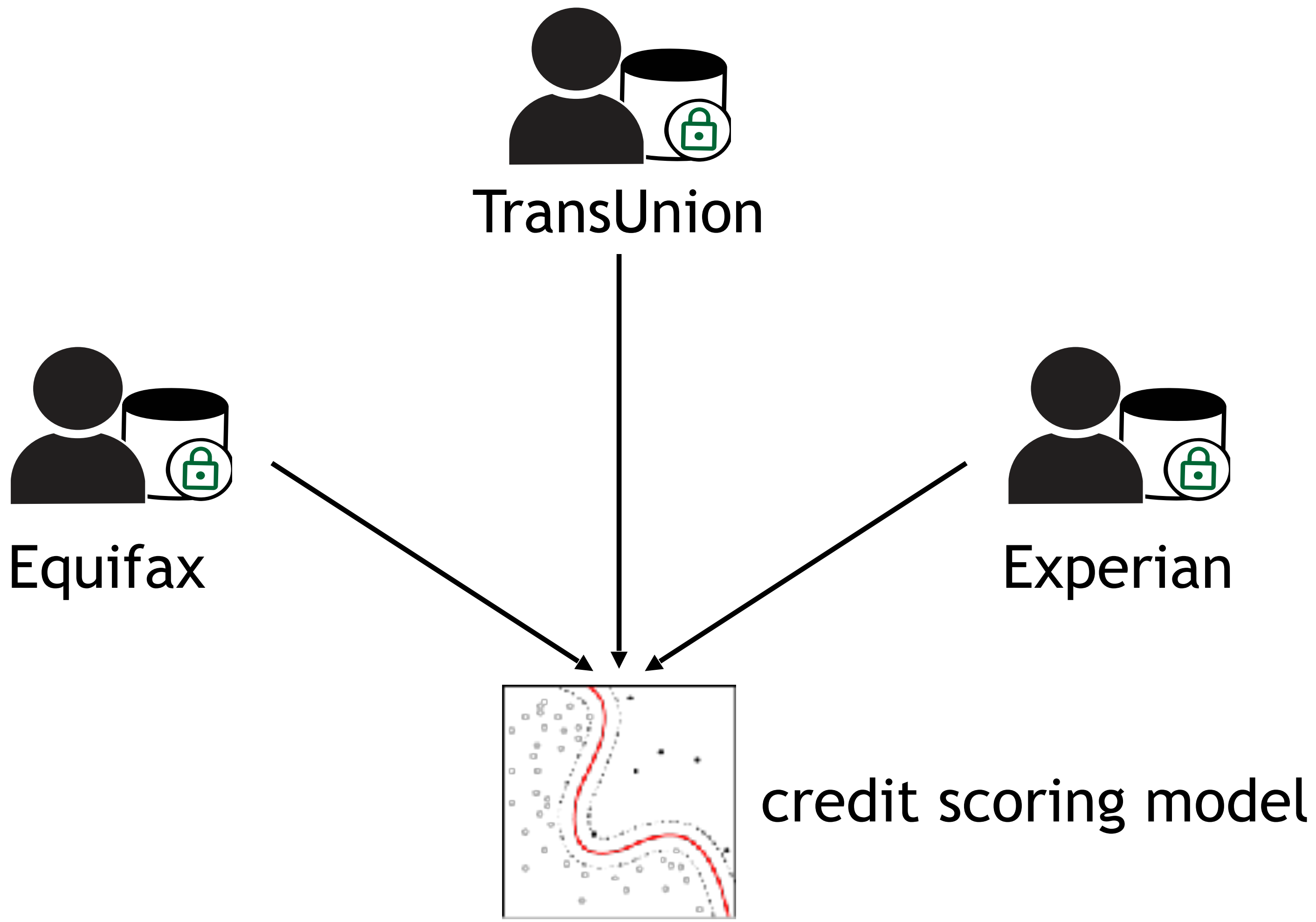
Fast & Faster Privacy-Preserving ML in Secure Hardware Enclaves

Nick Hynes, Raymond Cheng, Dawn Song | UC Berkeley & Oasis Labs
with support from the TVM team and community!

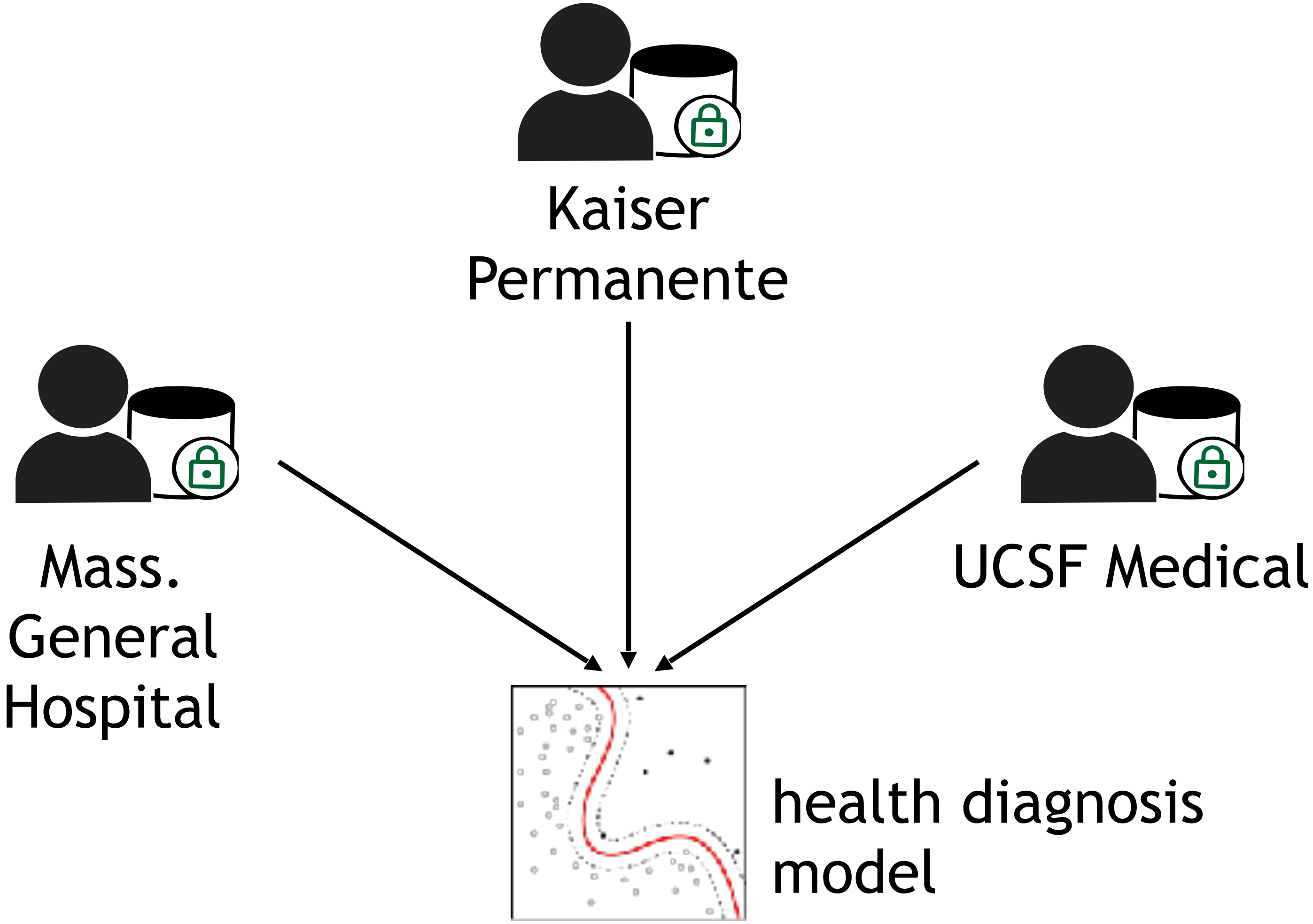
Ideal: data providers pool data to train a large, complex model



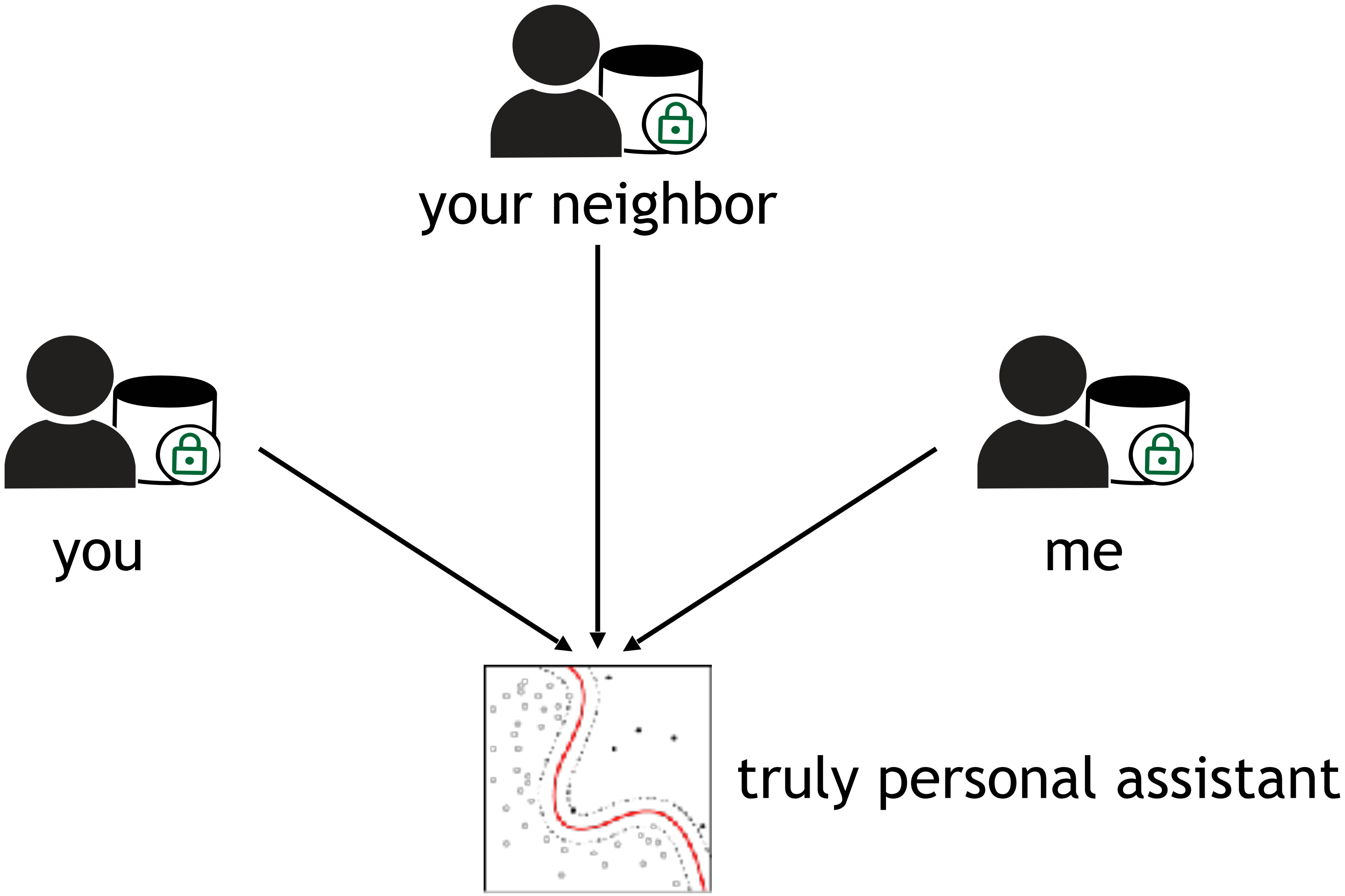
Ideal: data providers pool data to train a large, complex model



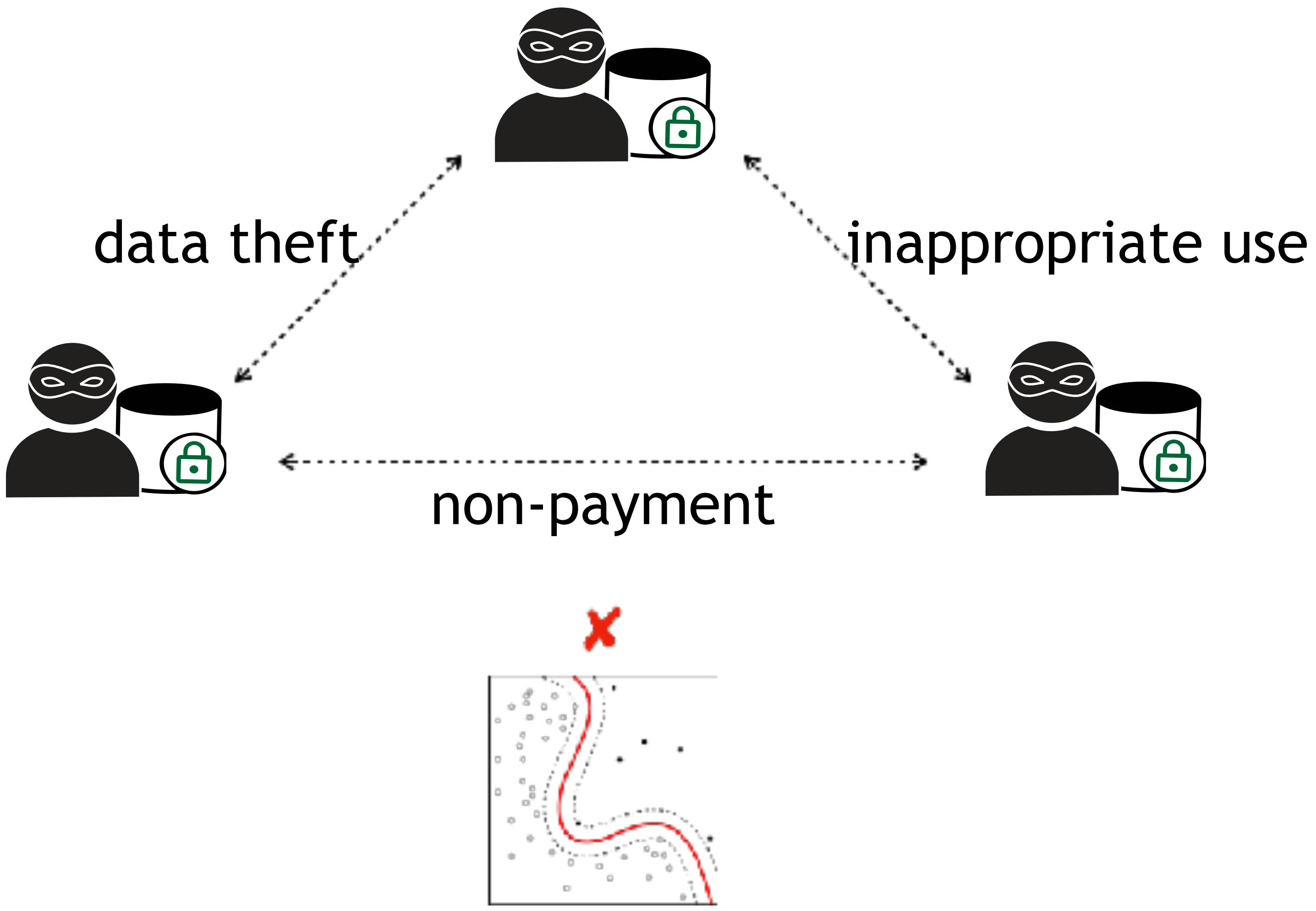
Ideal: data providers pool data to train a large, complex model



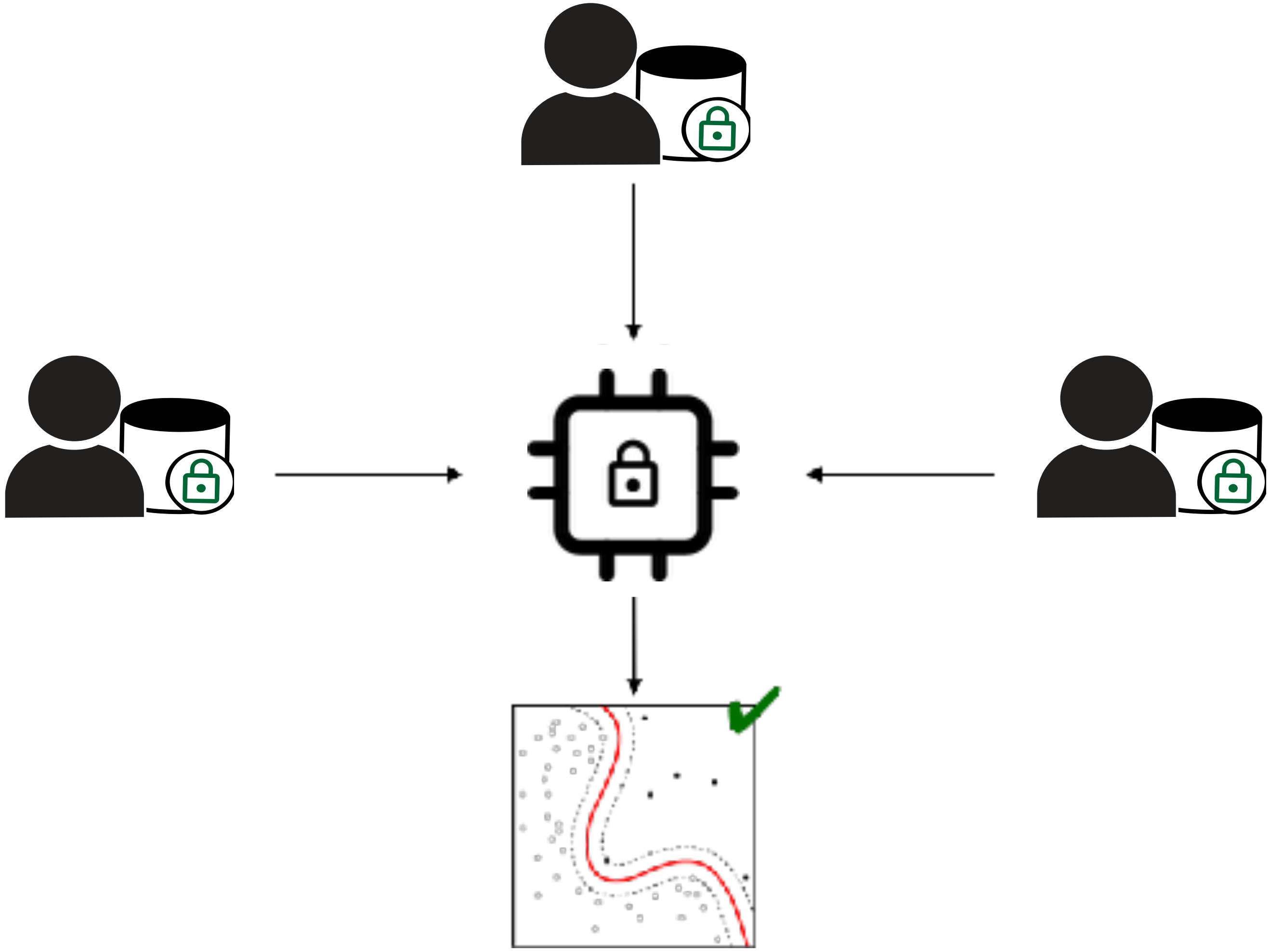
Ideal: data providers pool data to train a large, complex model



Reality: data providers are mutually distrusting!



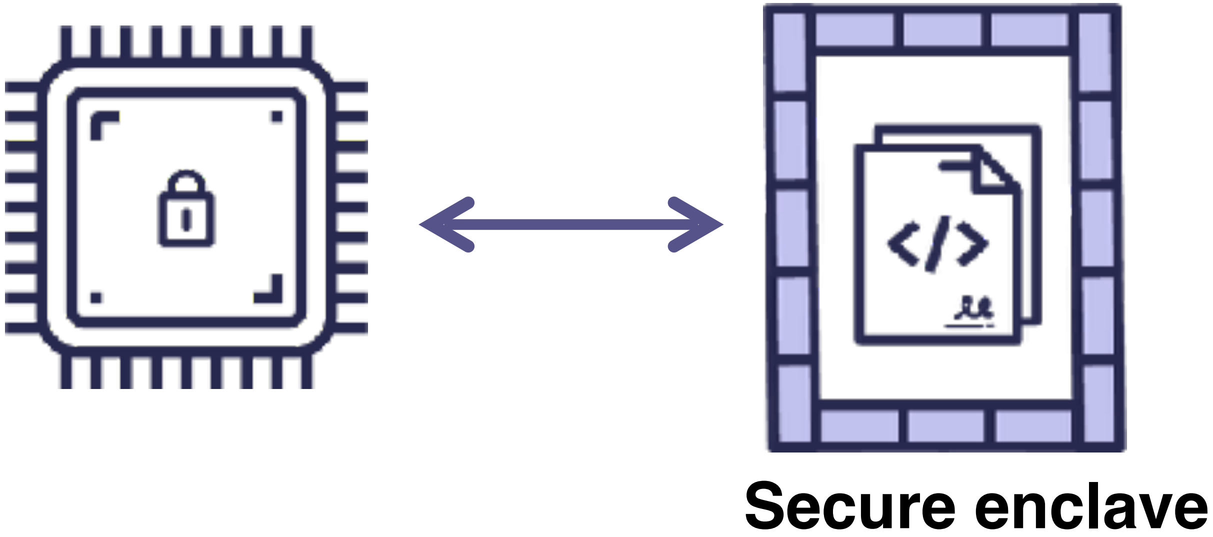
Solution: providers cooperate via a virtual trusted third party



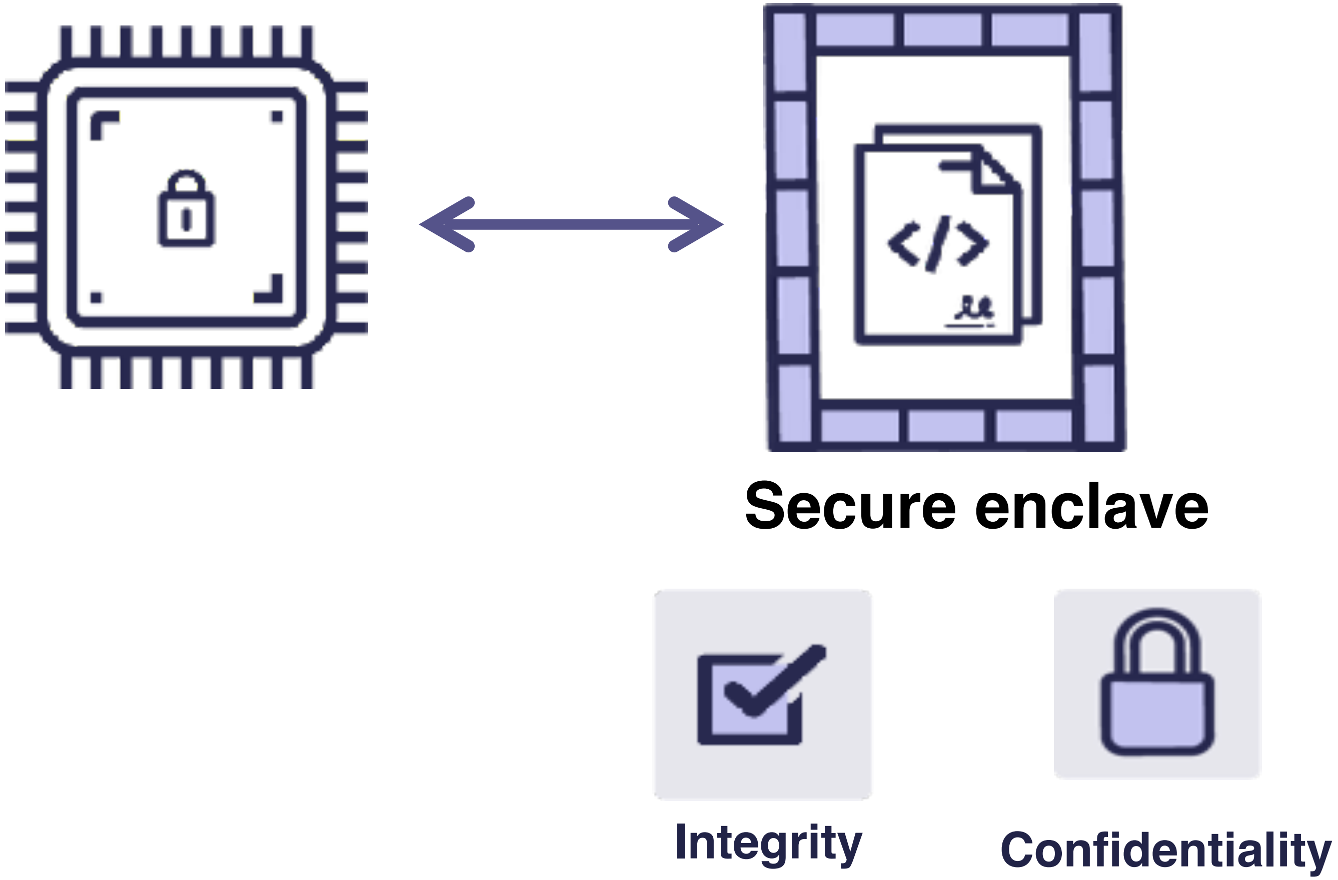
Secure Computation Techniques

	Performance	Support for practical ML models	Security mechanisms
Trusted Execution Env. (TEE)	●●●	●●●	Secure hardware
Secure multi-party computation	●○○	●○○	Cryptography, distributed trust
Zero-knowledge proof	●○○	●○○	Cryptography, local computation
Fully homomorphic encryption	○○○	●○○	Cryptography

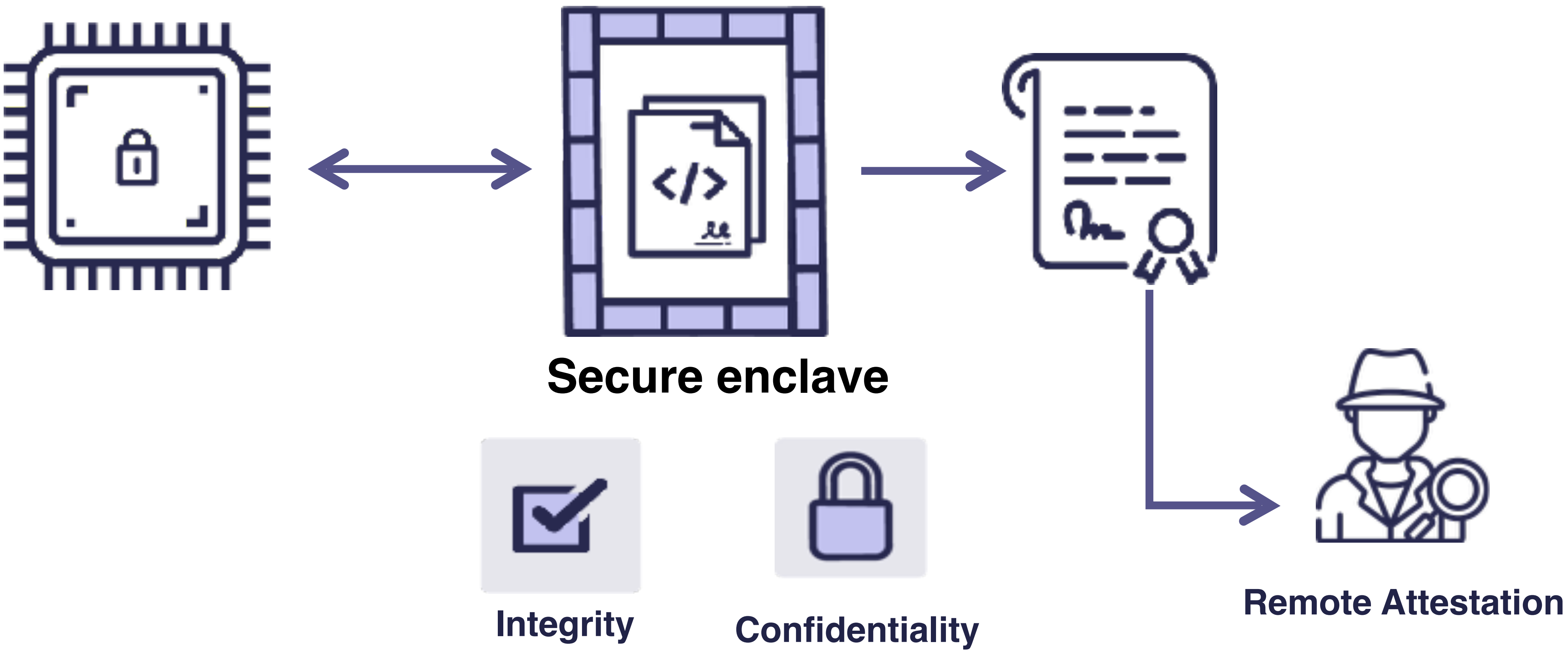
Secure Enclaves



Secure Enclaves



Secure Enclaves



TEE Implementations

- Intel SGX: in your laptop, Azure, Alibaba Cloud, and IBM Cloud

TEE Implementations

- Intel SGX: in your laptop, Azure, Alibaba Cloud, and IBM Cloud

- Keystone: the first open-source end-to-end secure enclave
  - runs on RISCV chips and FPGAs
  - keystone-enclave/keystone

TEE Implementations

- Intel SGX: in your laptop, Azure, Alibaba Cloud, and IBM Cloud
- Keystone: the first open-source end-to-end secure enclave
    - runs on RISCV chips and FPGAs
    - keystone-enclave/keystone
- Ginseng: a drop-in enclave framework for FPGA ML accelerators

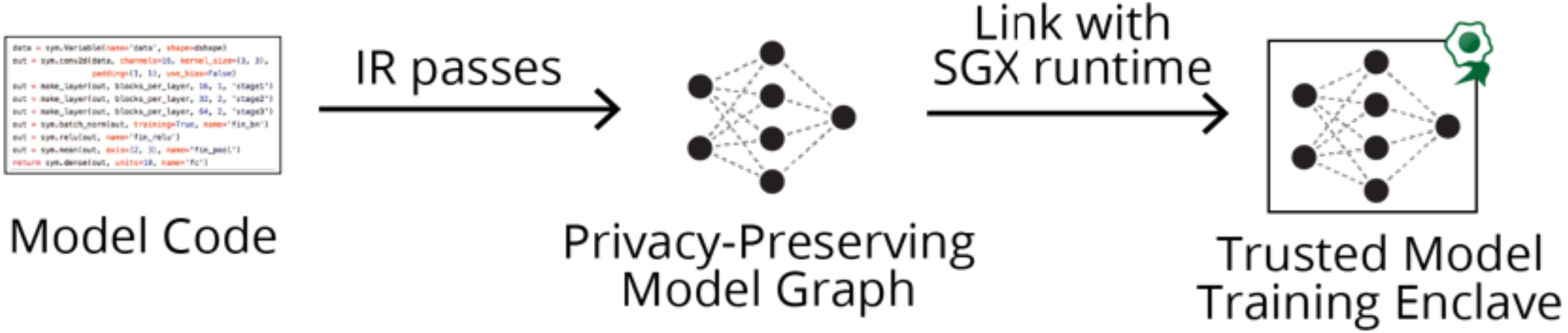
1. Privacy-Preserving ML & Secure Enclaves

2. Myelin: Efficient Private ML in CPU Enclaves

3. Ginseng: Accelerated Private ML in FPGA Enclaves

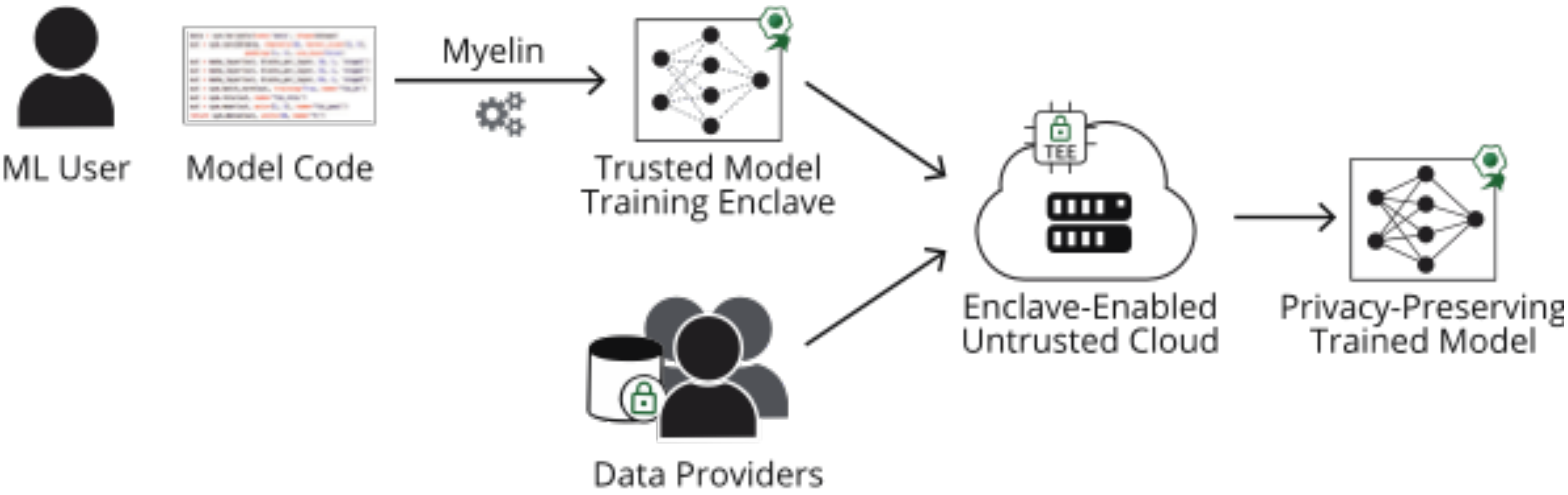
4. Sterling: A Privacy-Preserving Data Marketplace

Myelin: Efficient Private ML in CPU Enclaves



dmlc/tvm/apps/sgx
dmlc/tvm/rust

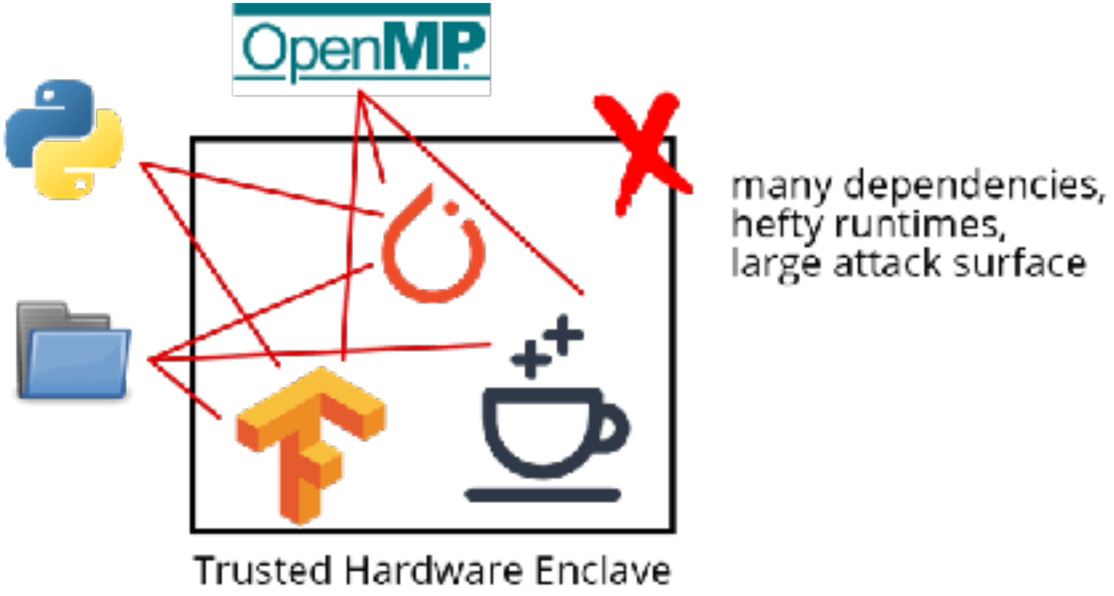
Myelin: Efficient Private ML in CPU Enclaves



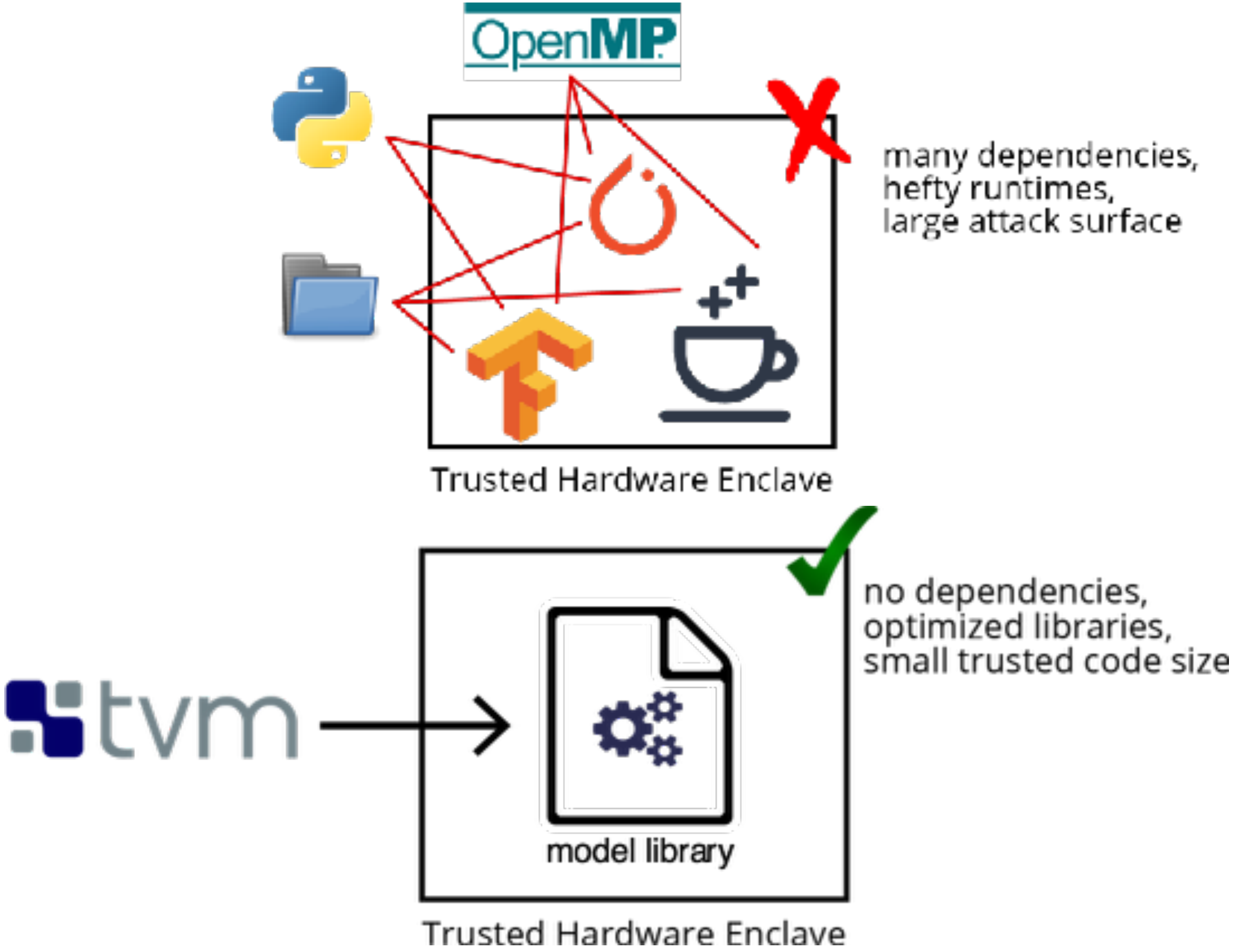
[3] Efficient Per-Example Gradient Computations. Goodfellow. 2015

Step 1: Get the ML in the Enclave

Step 1: Get the ML in the Enclave



Step 1: Get the ML in the Enclave



Step 2: Add *Differential* Privacy

Step 2: Add *Differential* Privacy

- DP offers a strong, formal definition of privacy

Step 2: Add *Differential* Privacy

- DP offers a strong, formal definition of privacy
- privacy risk to any individual is the same whether or not they contributed data

Step 2: Add *Differential* Privacy

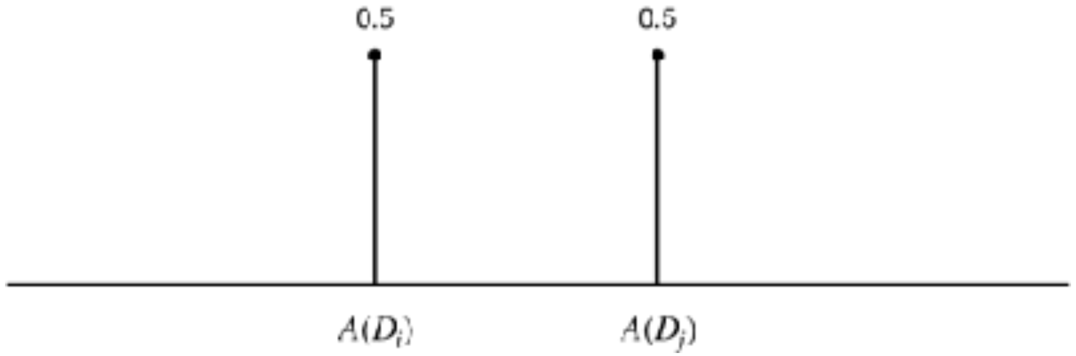
- DP offers a strong, formal definition of privacy
- privacy risk to any individual is the same whether or not they contributed data
- adds noise so that that model trained on neighboring datasets are indistinguishable

Step 2: Add *Differential* Privacy

- DP offers a strong, formal definition of privacy
- privacy risk to any individual is the same whether or not they contributed data
- adds noise so that that model trained on neighboring datasets are indistinguishable
- slow in standard frameworks

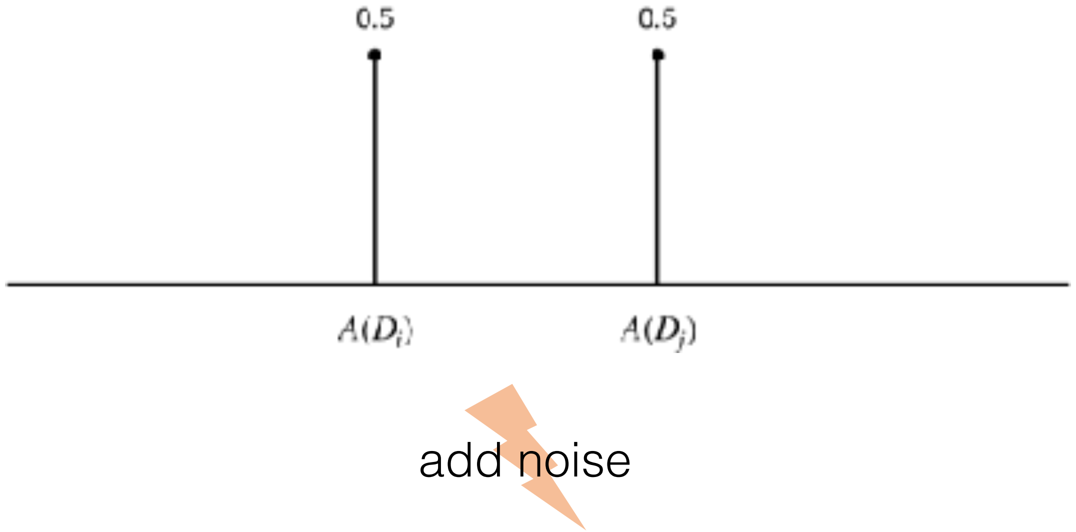
Step 2: Add *Differential* Privacy

- DP offers a strong, formal definition of privacy
- privacy risk to any individual is the same whether or not they contributed data
- adds noise so that that model trained on neighboring datasets are indistinguishable
- slow in standard frameworks



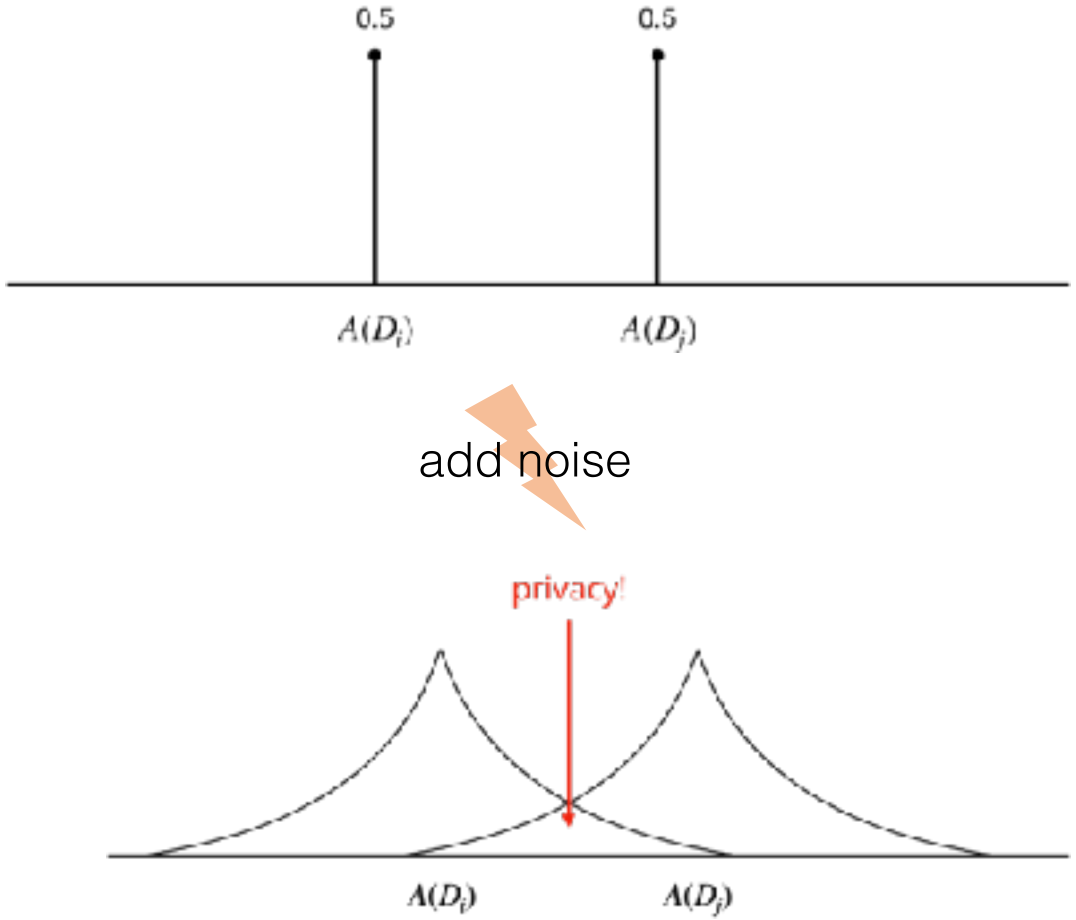
Step 2: Add *Differential* Privacy

- DP offers a strong, formal definition of privacy
- privacy risk to any individual is the same whether or not they contributed data
- adds noise so that that model trained on neighboring datasets are indistinguishable
- slow in standard frameworks



Step 2: Add *Differential* Privacy

- DP offers a strong, formal definition of privacy
- privacy risk to any individual is the same whether or not they contributed data
- adds noise so that that model trained on neighboring datasets are indistinguishable
- slow in standard frameworks



Step 3: Make it Fast

Differentially Private SGD

1. compute forward pass for mini-batch of *m* examples
2. compute *per-example gradients*
3. rescale each example's gradient to have unit norm
4. average them up
5. add noise
6. take gradient step

Step 3: Make it Fast

Differentially Private SGD

1. compute forward pass for mini-batch of *m* examples
2. compute *per-example gradients*
3. rescale each example's gradient to have unit norm
4. average them up
5. add noise
6. take gradient step

add a pass to fuse these

Step 3: Make it Fast

Differentially Private SGD

1. compute forward pass for batch of *m* examples
2. compute *per-example gradients*
3. rescale each example's gradient to have unit norm
4. average + noise+ gradient step

autograd takes O(m) [4]
O(1) with custom IR ops

[4] Efficient Per-Example Gradient Computations. Goodfellow. 2015

Step 4: Benchmark

Performance on CIFAR-10

	1 Myelin Enclave	non-private CPU	related work
VGG-9 (training)	21.3 img/s	27.2 img/s	Chiron (4 enclaves) [5] 24.7 img/s
ResNet-32 (training)	12.4 img/s	13.6 img/s	-
MobileNet (inference)	32.4 img/s	-	Slalom (enclave+GPU) [6] 35.7 img/s

[5] Chiron: Privacy-preserving machine learning as a service. Hunt, Song, Shokri, Shmatikov, and Witchel. 2018
[6] Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. Tramer and

State of the Art Performance for ML in Single CPU Enclave

- but a CPU is a CPU: ½ day to train a ResNet is emotionally unsatisfying
- no GPU TEEs (yet), but we *can* do FPGAs!

1. Privacy-Preserving ML & Secure Enclaves

2. Myelin: Efficient Private ML in CPU Enclaves

3. Ginseng: Accelerated Private ML in FPGA Enclaves

4. Sterling: A Privacy-Preserving Data Marketplace

Ginseng, the Learning TEE

- Main idea: FPGA can be programmed with ML accelerator (VTA) *and* the components required to make a TEE
  - memory encryption
  - key generation
  - remote attestation

- TEEs are general-purpose; ML is very particular
  We get big efficiency wins from specializing TEE to ML workloads

Ginseng = VTA + Tensor Encryption + Secure OS

Ginseng = VTA + Tensor Encryption + Secure OS

- Tensor Encryption Core (TEC) safeguards the tensors in memory
  - protects entire models’ tensors for virtually no overhead

Ginseng = VTA + Tensor Encryption + Secure OS

- Tensor Encryption Core (TEC) safeguards the tensors in memory
  - protects entire models' tensors for virtually no overhead

- Ginseng Secure OS protects the end-to-end workflow
  - built atop formally verified components
  - minimal trusted computing base
  - side-channel resistant

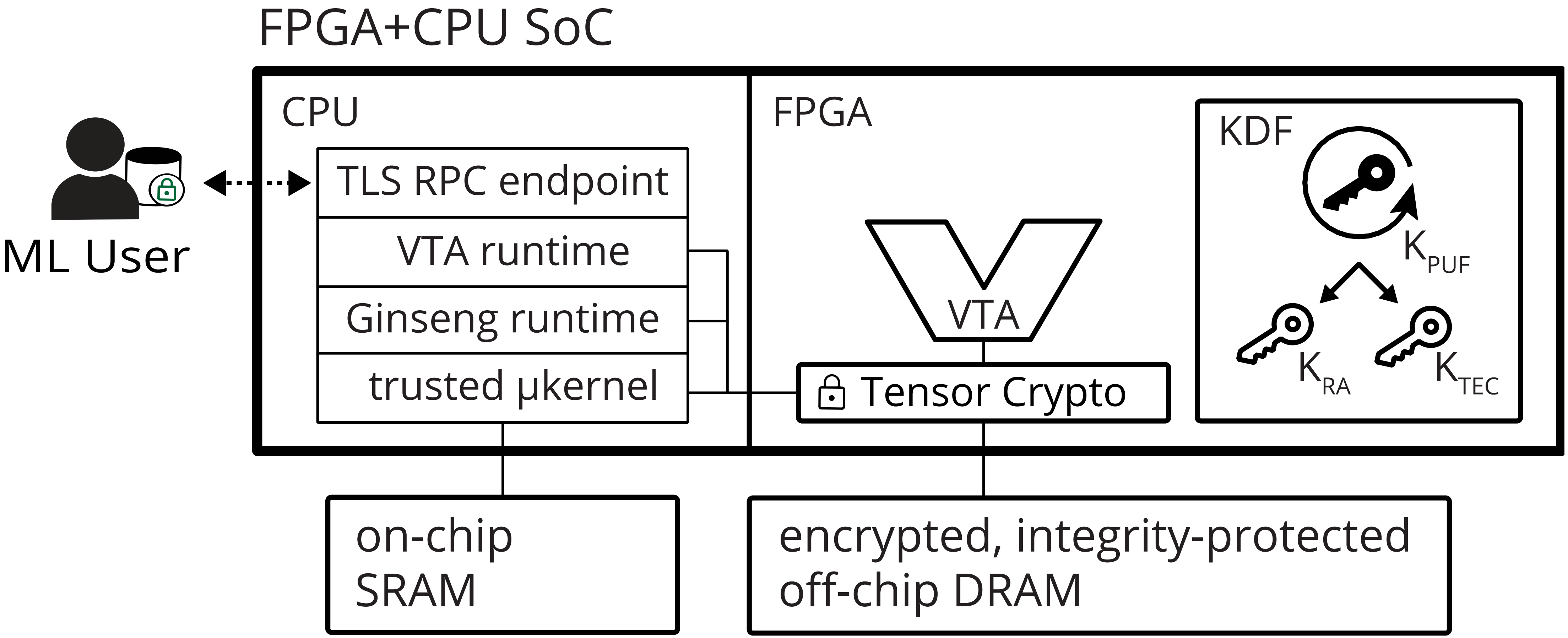
Ginseng = VTA + Tensor Encryption + Secure OS

- Tensor Encryption Core (TEC) safeguards the tensors in memory
  - protects entire models' tensors for virtually no overhead

- Ginseng Secure OS protects the end-to-end workflow
  - built atop formally verified components
  - minimal trusted computing base
  - side-channel resistant

- End result: an end-to-end secure, speedy ML pipeline

Ginseng = VTA + Tensor Encryption + Secure OS



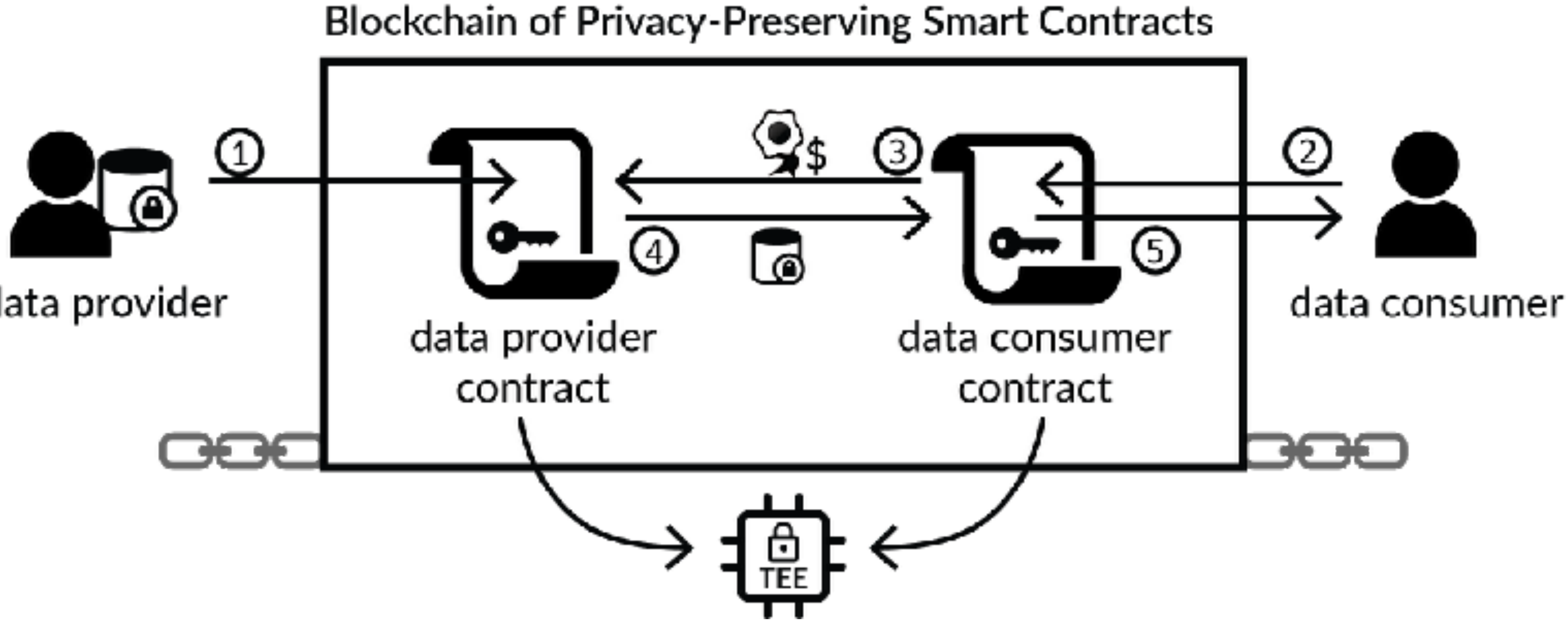
1. Privacy-Preserving ML & Secure Enclaves

2. Myelin: Efficient Private ML in CPU Enclaves

3. Ginseng: Accelerated Private ML in FPGA Enclaves

3. Sterling: A Privacy-Preserving Data Marketplace

Sterling: A Privacy-Preserving Data Marketplace built on the Oasis blockchain and TVM



[1] A Demonstration of Sterling: A Privacy-Preserving Data Marketplace. VLDB 2018.
[2] Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contract Execution. 2018.

Sterling workflow

Sterling workflow

1. data provider encrypts data and uploads to Oasis blockchain
   access to data is controlled by a confidential smart contract

Sterling workflow

1. data provider encrypts data and uploads to Oasis blockchain
   access to data is controlled by a confidential smart contract
2. data consumer uploads a model training smart contract
   which satisfies constraints of provider contract

Sterling workflow

1. data provider encrypts data and uploads to Oasis blockchain
   access to data is controlled by a confidential smart contract
2. data consumer uploads a model training smart contract
   which satisfies constraints of provider contract
3. consumer contract requests data from provider contract
   sends over payment and credentials

Sterling workflow

1. data provider encrypts data and uploads to Oasis blockchain
   access to data is controlled by a confidential smart contract
2. data consumer uploads a model training smart contract
   which satisfies constraints of provider contract
3. consumer contract requests data from provider contract
   sends over payment and credentials
4. provider contract checks that consumer contract satisfies constraints
   and sends back data

Sterling workflow

1. data provider encrypts data and uploads to Oasis blockchain
   access to data is controlled by a confidential smart contract
2. data consumer uploads a model training smart contract
   which satisfies constraints of provider contract
3. consumer contract requests data from provider contract
   sends over payment and credentials
4. provider contract checks that consumer contract satisfies constraints
   and sends back data
5. consumer contract trains a privacy-preserving model and returns it to
   the data consumer

Sterling & TVM to the Moon

Sterling & TVM to the Moon

- Sterling facilitates a distributed, trustless, *uncoordinated* data marketplace

Sterling & TVM to the Moon

- Sterling facilitates a distributed, trustless, *uncoordinated* data marketplace
- builds on the efficiency of TVM with the portability and security of Web Assembly
    - also uses the new TVM Rust runtime!

Sterling & TVM to the Moon

- Sterling facilitates a distributed, trustless, *uncoordinated* data marketplace
- builds on the efficiency of TVM with the portability and security of Web Assembly
  - also uses the new TVM Rust runtime!
- TVM modules run in secure enclaves provided by the Oasis blockchain

Roadmap

Roadmap

- Training on VTA and CPU! Super excited for Relay autograd
    - Much better than the FExpandCompute ~~kludge~~ pass we're using now

Roadmap

- Training on VTA and CPU! Super excited for Relay autograd
  - Much better than the FExpandCompute ~~kludge~~ pass we're using now

- Deploy Ginseng to AWS F1 once VTA Chisel port is ready

Roadmap

- Training on VTA and CPU! Super excited for Relay autograd
    - Much better than the FExpandCompute ~~kludge~~ pass we're using now

- Deploy Ginseng to AWS F1 once VTA Chisel port is ready

Roadmap

- Training on VTA and CPU! Super excited for Relay autograd
  - Much better than the FExpandCompute ~~kludge~~ pass we're using now
- Deploy Ginseng to AWS F1 once VTA Chisel port is ready
- automatically checking TVM models for differential privacy
  (on the blockchain, of course)

Thanks!