Privacy-Preserving ML

Project MIRAI - Meeting T2.2/T2.3

October 12, 2021 - Online

Pedro Santos – pss@isep.ipp.pt

Outline

- Overview of Security in ML Contexts
  - Goals and Target Stage
  - Attacks at Inference and Training Stage
  - Attack Surface
- Privacy Attacks in ML
  - Privacy Threat Model
  - Taxonomy of Privacy Attacks
  - Shadow Attacks
- Selected Privacy-preserving Techniques
  - Ensemble of Teacher Models
  - Differential Privacy
  - Distributed Training
  - Training over Encrypted Data
  - Trusted Execution

Overview of Security in ML Contexts

Goals and Target Stage

N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, ‘Towards the Science of Security and Privacy in Machine Learning’, arXiv:1611.03814 [cs], Nov. 2016, Accessed: Aug. 04, 2021. [Online].

Goal of attacks

- Confidentiality and Privacy
  - Such attacks attempt to extract information about the model or training data.
  - When the model itself represents intellectual property, it requires that the model and its parameters be confidential.
- Integrity and Availability
  - Here the goal is to induce model behavior as chosen by the adversary.
  - Attacks attempting to control model outputs are at the heart of integrity attacks — the integrity of the inference process is undermined.

Focus of our work

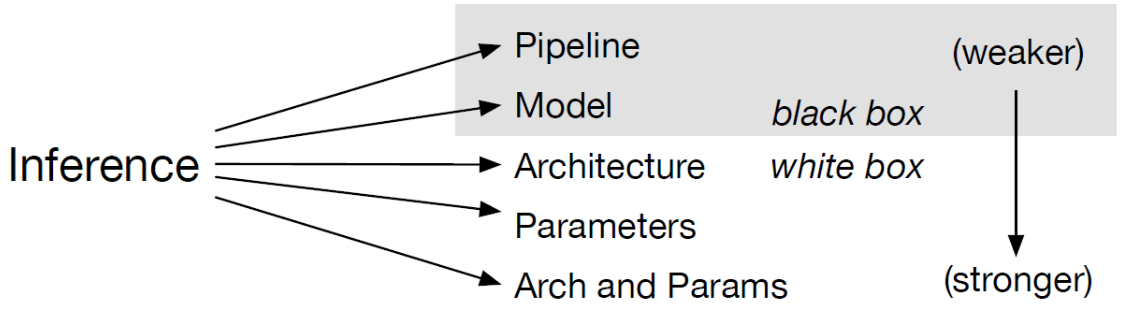
Stage Targeted by Attacker

- Attacks at Inference Time
  - Attacker causes the model to produce adversary-selected (incorrect) outputs
  - Attackers collect evidence about the model characteristics (reconnaissance).
- Attacks at Training Time
  - Attacker attempts to learn, influence or corrupt the model itself.

Attacks at Inference Time

N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, 'Towards the Science of Security and Privacy in Machine Learning', arXiv:1611.03814 [cs], Nov. 2016, Accessed: Aug. 04, 2021. [Online].

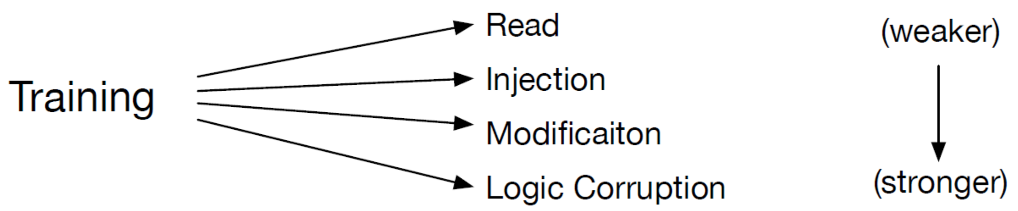
- Goals of attacks at Inference Time:
  - Attacker causes the model to produce adversary-selected (incorrect) outputs;
  - Attackers collect evidence about the model characteristics (reconnaissance).
- Attacks at inference time —exploratory attacks — do not tamper with the targeted model.
- Inference phase attacks can be classified depending on the attacker knowledge:
  - **White box attacks:** the adversary has some information about the model or its original training data, e.g., ML algorithm, model parameters, network structure, or summary, partial, or full training data.
  - **Black box attacks:** assume no knowledge about the model.



Attacks at Training Time

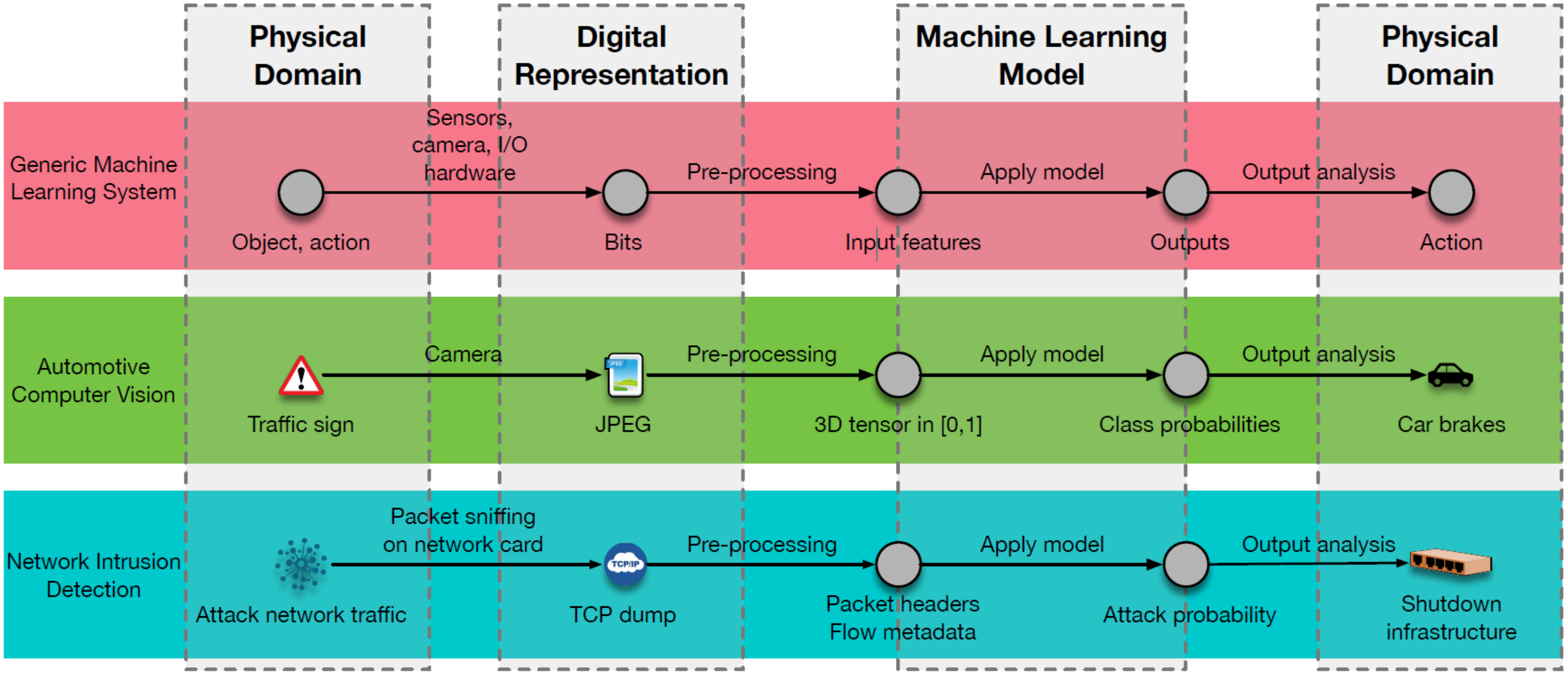
N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, 'Towards the Science of Security and Privacy in Machine Learning', arXiv:1611.03814 [cs], Nov. 2016, Accessed: Aug. 04, 2021. [Online].

- Attacks on training attempt to learn, influence or corrupt the model itself.
- Simplest attack is accessing a summary, partial or all of the training data.
  - Depending on the quality and volume of training data, the adversary can create a substitute model to mount attacks.
  - Note that these attacks are offline attempts at model reconnaissance, and thus may be used to undermine privacy.
- There are two broad attack strategies for altering the model.
  - **Alter the training data:** either by inserting adversarial inputs into the existing training data (injection) or altering the training data directly (modification). In the case of reinforcement learning, the adversary may modify the environment in which the agent is evolving.
  - **Model corruption:** the adversaries can tamper with the learning algorithm. Any adversary that can alter the learning logic (and thus controls the model itself) is very powerful and difficult to defend against.



Attack Surface

N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, 'Towards the Science of Security and Privacy in Machine Learning', arXiv:1611.03814 [cs], Nov. 2016, Accessed: Aug. 04, 2021. [Online].



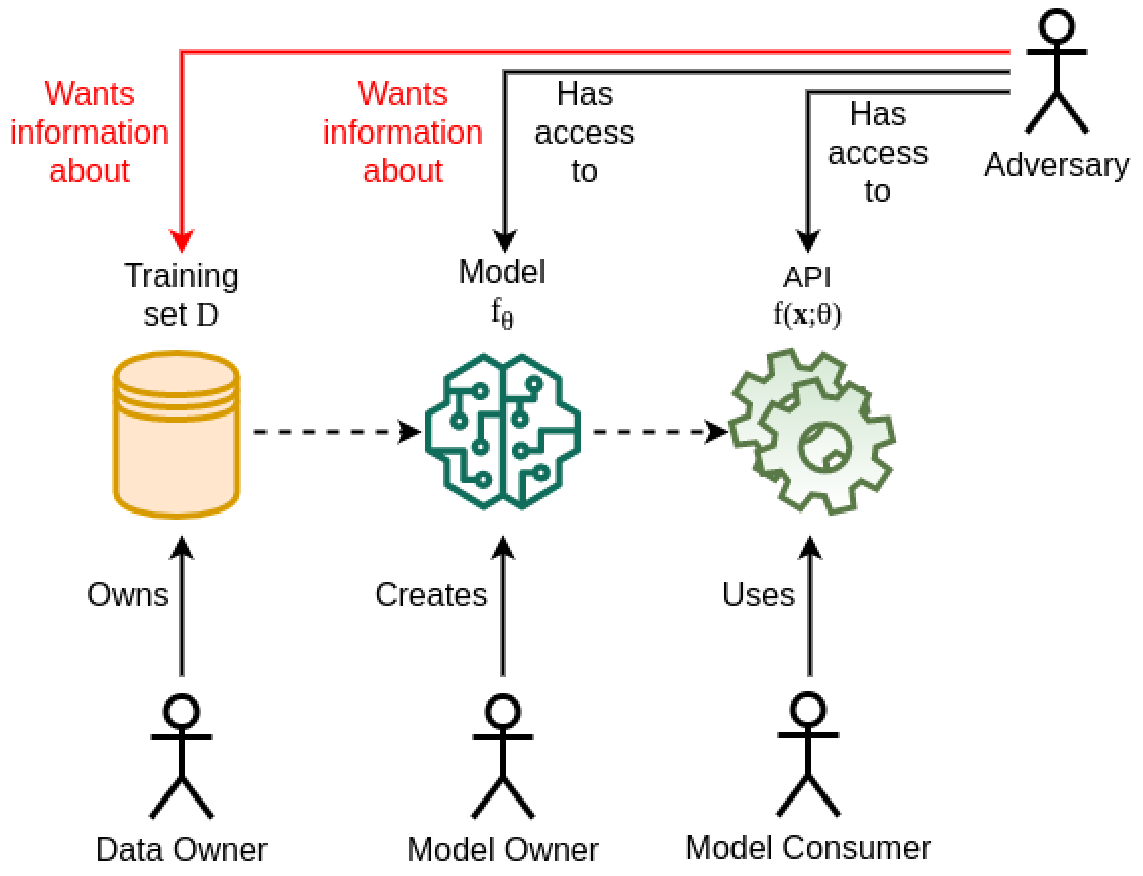
Privacy Attacks in ML

Privacy Threat Model

M. Rigaki and S. Garcia, 'A Survey of Privacy Attacks in Machine Learning', arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

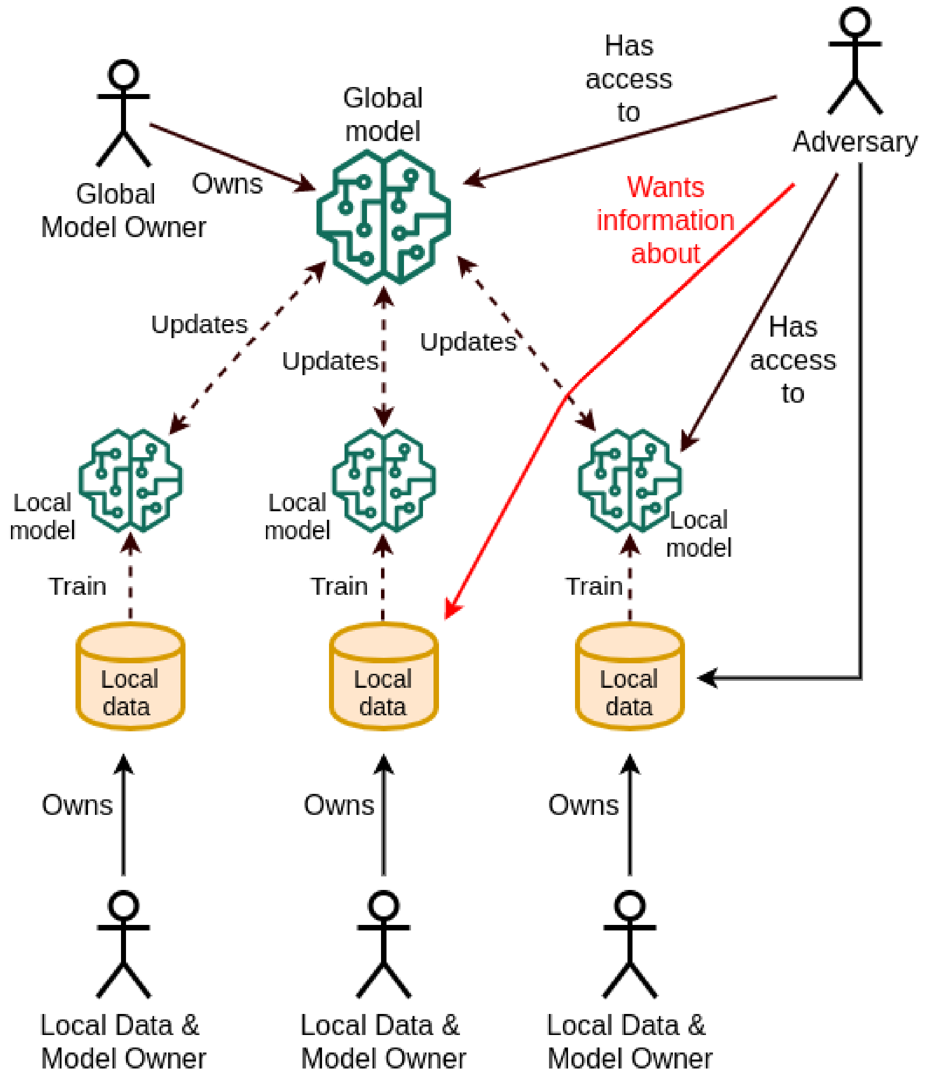
Actors

1. The data owners, whose data may be sensitive.
2. The model owners, which may or may not own the data and may or may not want to share information about their models.
3. The model consumers, that use the services that the model owner exposes, usually via some sort of programming or user interface.
4. The adversaries, that may also have access to the model's interfaces as a normal consumer does, e.g. feeding new input data for inference. If the model owner allows, they may have access to the model itself.



Distributed Learning Scenario

M. Rigaki and S. Garcia, ‘A Survey of Privacy Attacks in Machine Learning’, arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

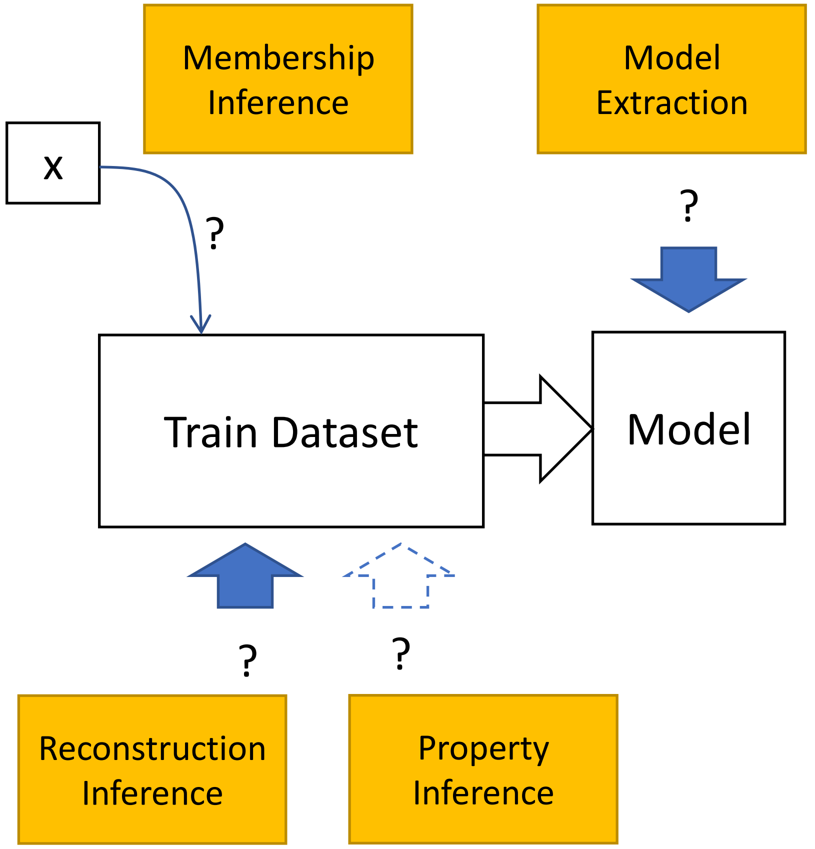


Taxonomy of Privacy Attacks

M. Rigaki and S. Garcia, ‘A Survey of Privacy Attacks in Machine Learning’, arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

Types of attack

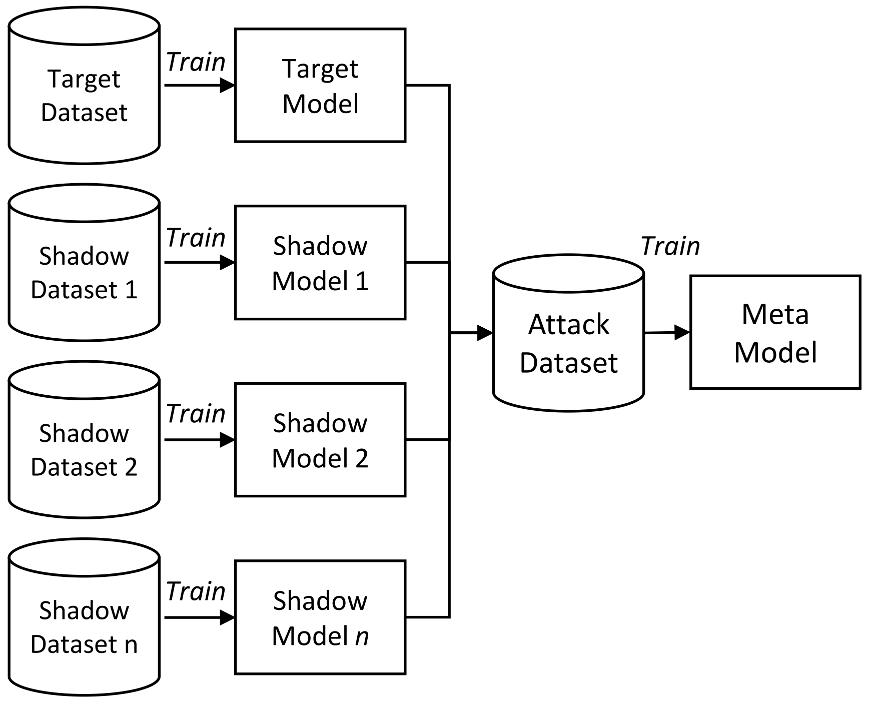
- **Membership Inference Attacks:** Membership inference tries to determine whether an input sample *x* was used as part of the training set *D*.
- **Reconstruction Attacks:** Reconstruction attacks try to recreate one or more training samples and/or their respective training labels.
- **Property Inference Attacks:** The ability to extract dataset properties which were not explicitly encoded as features or were not correlated to the learning task, is called property inference.
- **Model Extraction Attacks:** Model extraction is a class of black-box attacks where the adversary tries to extract information and potentially fully reconstruct a model by creating a substitute model $f^{\hat{}}$ that behaves very similarly to the model under attack f.



Shadow Models (1/2)

R. Shokri, M. Stronati, C. Song, and V. Shmatikov, 'Membership Inference Attacks Against Machine Learning Models', in 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, May 2017, pp. 3–18.

- Some supervised learning attacks use **shadow models** and meta-models to infer membership from target datasets / target models.
- The main intuition behind such attacks is that **models behave differently when they see data that does not belong to the training dataset**.
- The objective of the attacker is to construct an attack model that can recognize such differences in the target model's behavior, and use them to distinguish members from non-members of the target model's training dataset (based solely on the target model's output).



Main idea:

1. The attacker trains shadow models (one per output class of the target model) using shadow datasets $D_{shadow} = \{\boldsymbol{x}_{shadow,i}, \boldsymbol{y}_{shadow,i}\}_{i=1}^{n}$, that are of the same format and distribution as the target dataset.
2. After training the shadow models, the adversary constructs an attack dataset $D_{attack} = \{f_i(x_{shadow,i}), y_{shadow,i}\}_{i=1}^{n}$, where f_i is the respective shadow model.
3. The attack dataset is used to train the meta-model, which essentially performs inference based on the outputs of the shadow models.
4. The trained meta-model is used for testing using the outputs of the target model.

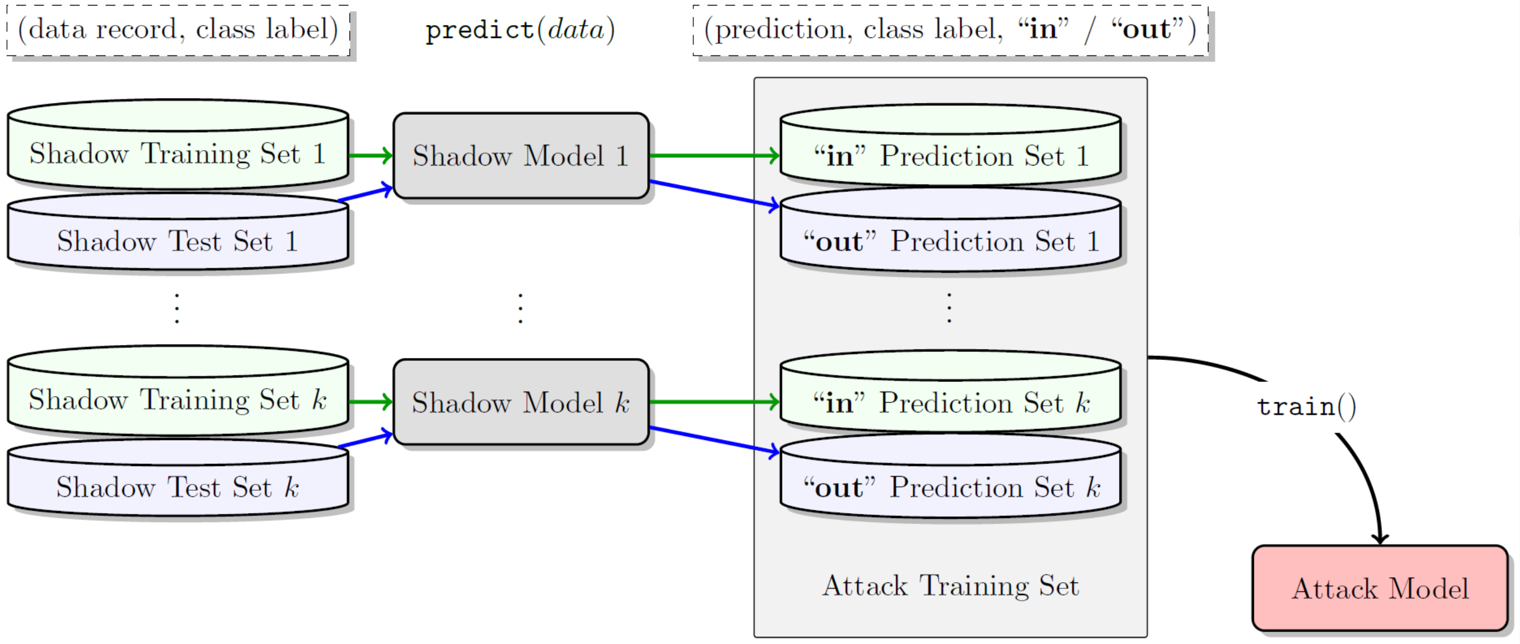
Our results show that learning how to infer membership in shadow models' training datasets (for which we know the ground truth and can easily compute the cost function during supervised training) produces an attack model that successfully infers membership in the target model's training dataset, too.

Shadow Models (2/2)

R. Shokri, M. Stronati, C. Song, and V. Shmatikov, 'Membership Inference Attacks Against Machine Learning Models', in 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, May 2017, pp. 3–18.

In more detail:

1. For all records in the training dataset of a shadow model, the model is queried and obtain the output. These output vectors are labeled *"in"* and added to the attack model's training dataset.
2. The shadow model is also queried with a test dataset disjoint from its training dataset. The outputs on this set are labeled *"out"* and also added to the attack model's training dataset.
3. Having constructed a dataset that reflects the black-box behavior of the shadow models on their training and test datasets, we train a collection of c_{target} attack models, one per each output class of the target model.

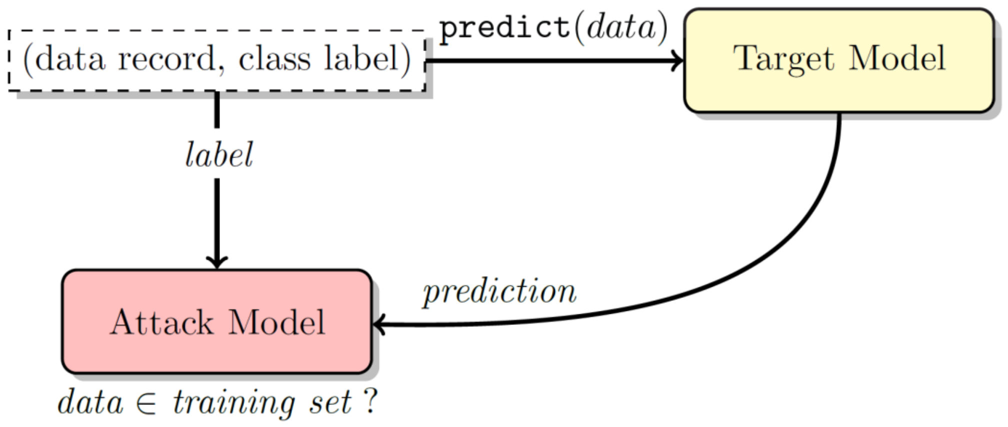


Shadow Models (2/2)

R. Shokri, M. Stronati, C. Song, and V. Shmatikov, 'Membership Inference Attacks Against Machine Learning Models', in 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, May 2017, pp. 3–18.

In more detail:

1. For all records in the training dataset of a shadow model, the model is queried and obtain the output. These output vectors are labeled *"in"* and added to the attack model's training dataset.
2. The shadow model is also queried with a test dataset disjoint from its training dataset. The outputs on this set are labeled *"out"* and also added to the attack model's training dataset.
3. Having constructed a dataset that reflects the black-box behavior of the shadow models on their training and test datasets, we train a collection of c_{target} attack models, one per each output class of the target model.



Membership Inference Attacks

M. Rigaki and S. Garcia, 'A Survey of Privacy Attacks in Machine Learning', arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

Membership inference tries to determine whether an input sample x was used as part of the training set D.

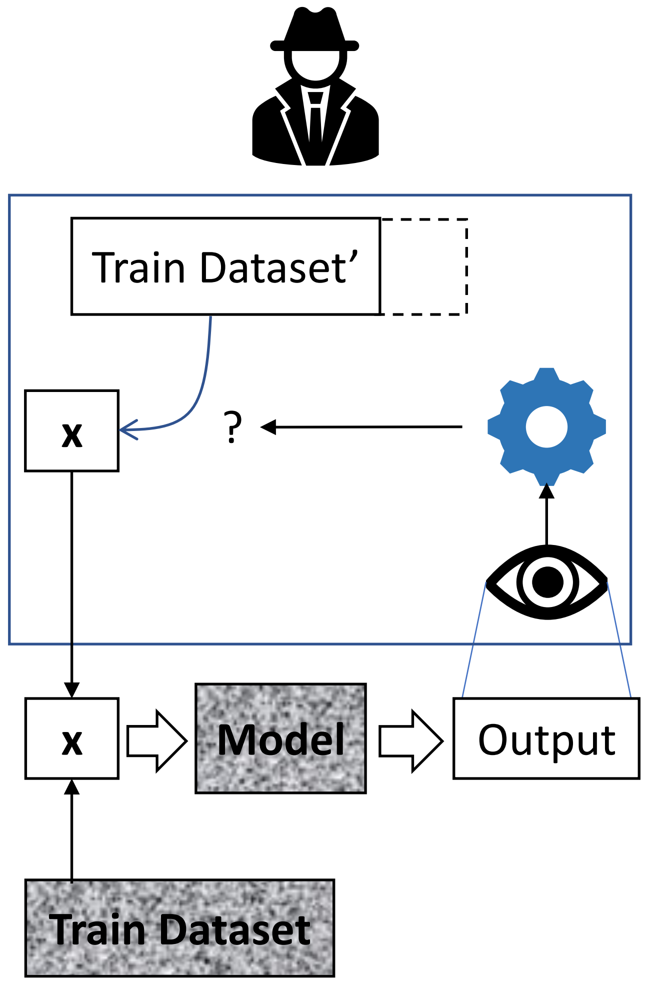
- Only assumes knowledge of the model's output prediction vector (typically a black-box attack)
- Targets supervised machine learning models (and generative models such as GANs and VAEs)
- White-box: if attacker has access to the model parameters and gradients, then this allows for more accurate membership inference determination.

Implementation:

- The meta-model is trained to recognize patterns in the prediction vector output of the target model.

Defense:

- **Differential Privacy.** if two databases differ only by one record and are used by the same algorithm (or mechanism), the output of that algorithm should be similar. Differential privacy offers a trade-off between privacy protection and utility or model accuracy.
- **Regularization techniques.** Regulation techniques aim to reduce overfitting and increase model generalization performance.
- **Prediction vector tampering**. As many models assume access to the prediction vector during inference, one can restrict the output to the top *k* classes or predictions of a model.



Reconstruction Attacks

M. Rigaki and S. Garcia, 'A Survey of Privacy Attacks in Machine Learning', arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

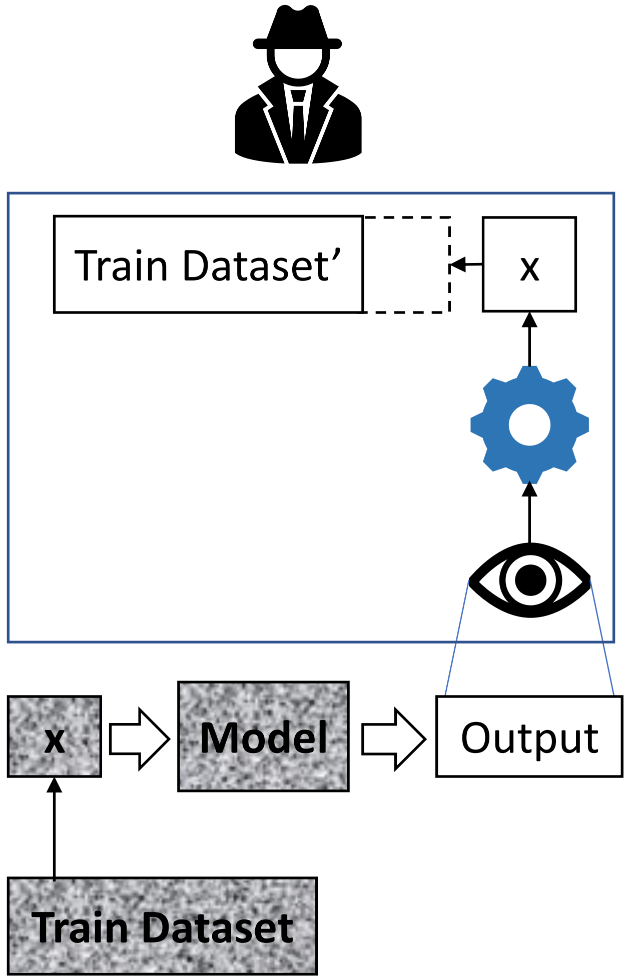
Reconstruction attacks try to recreate one or more training samples and/or their respective training labels.

- Also referred to as attribute inference or model inversion: given output labels and partial knowledge of some features, the attacker tries to recover sensitive features or the full data sample.

Two types of such attacks:
- Create an actual reconstruction of the data;
- Create class representatives or probable values of sensitive features that do not necessarily belong to the training dataset (e.g., faces of a person).

Implementation:
- Assumes that the adversary has access to the model f, the priors of the sensitive and non-sensitive features, and the output of the model for a specific input x.
- The attack is based on estimating the values of sensitive features, given the values of non-sensitive features and the output label.
- This method uses a maximum a posteriori (MAP) estimate of the attribute that maximizes the probability of observing the known parameters.

Defense:
- Reconstruction attacks often require access to the loss gradients during training.
- Most of the defences against reconstruction attacks propose techniques that affect the information retrieved from these gradients.



Property Inference Attacks

M. Rigaki and S. Garcia, 'A Survey of Privacy Attacks in Machine Learning', arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

Property inference is the ability to extract dataset properties which were not explicitly encoded as features or were not correlated to the learning task.

Property inference aims to extract information that:

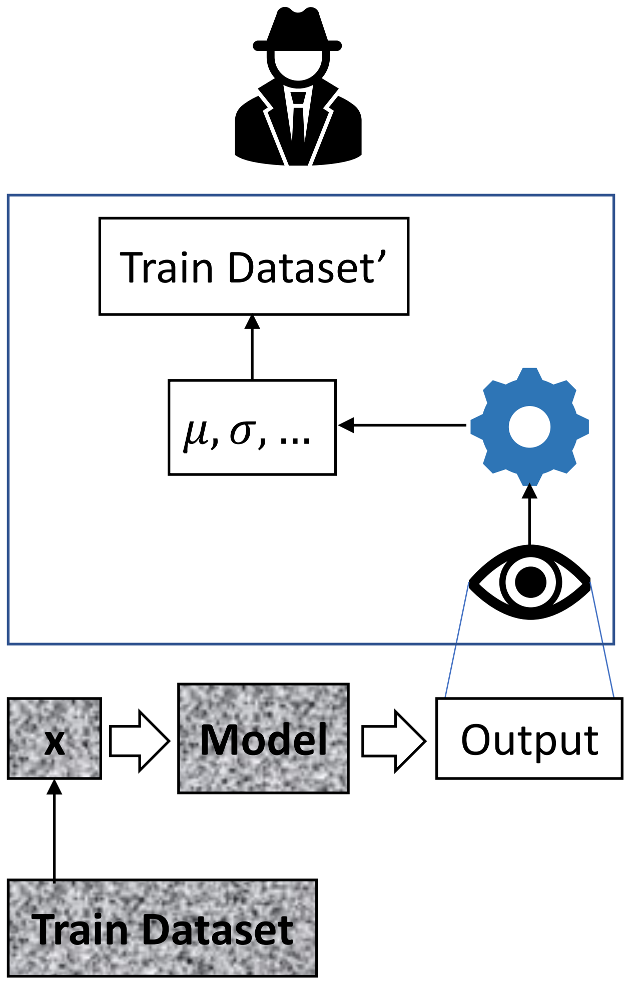
- Was learned from the model unintentionally – even well generalized models may learn properties that are relevant to the whole input data distribution and sometimes this is unavoidable or even necessary for the learning process.
- May be used to gain insights about the training dataset.

Examples:

- Extraction of information about the ratio of women and men in a patient dataset, when this information was not an encoded attribute or a label of the dataset.
- Neural network that performs gender classification and can be used to infer if people in the training dataset wear glasses or not.

Property inference target either:

- Dataset-wide properties
- Emergence of properties within a batch of data (on the collaborative training of a model).



Model Extraction Attacks

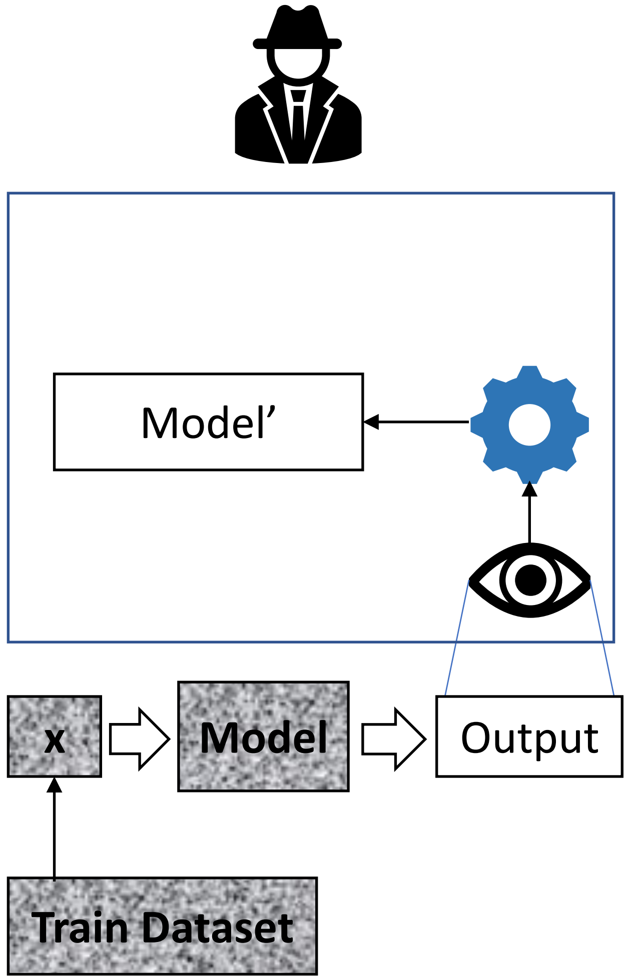
M. Rigaki and S. Garcia, ‘A Survey of Privacy Attacks in Machine Learning’, arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646

Model extraction is a class of black-box attacks where the adversary tries to extract information and potentially fully reconstruct a model by creating a substitute model f' that behaves very similarly to the model under attack f.

<ins>Two types of model extraction attacks:</ins>

- Task accuracy extraction
  - Create models that match the accuracy of target model f in a test set that is drawn from the input data distribution and related to the learning task.
  - The adversary is interested in creating a substitute that learns the same task as the target model equally well.
- Fidelity extraction
  - Create a substitute model f' that matches f at a set of input points that are not necessarily related to the learning task.
  - The adversary aims to create a substitute that replicates the decision boundary of f as faithfully as possible.

- In both cases, it is assumed that the adversary wants to use as few queries as possible.
- Knowledge of the target model architecture is not strictly necessary, if the adversary’s substitute model of same or higher complexity than model under attack.
- Some works focus on recovering information from the target model, e.g., hyper-parameters in the objective function; in neural networks: activation types, optimisation algorithm, number of layers, etc.



An Example of Model Inversion Attacks

M. Fredrikson, S. Jha, and T. Ristenpart, ‘Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures’, in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver Colorado USA, Oct. 2015, pp. 1322–1333. doi: 10.1145/2810103.2813677.

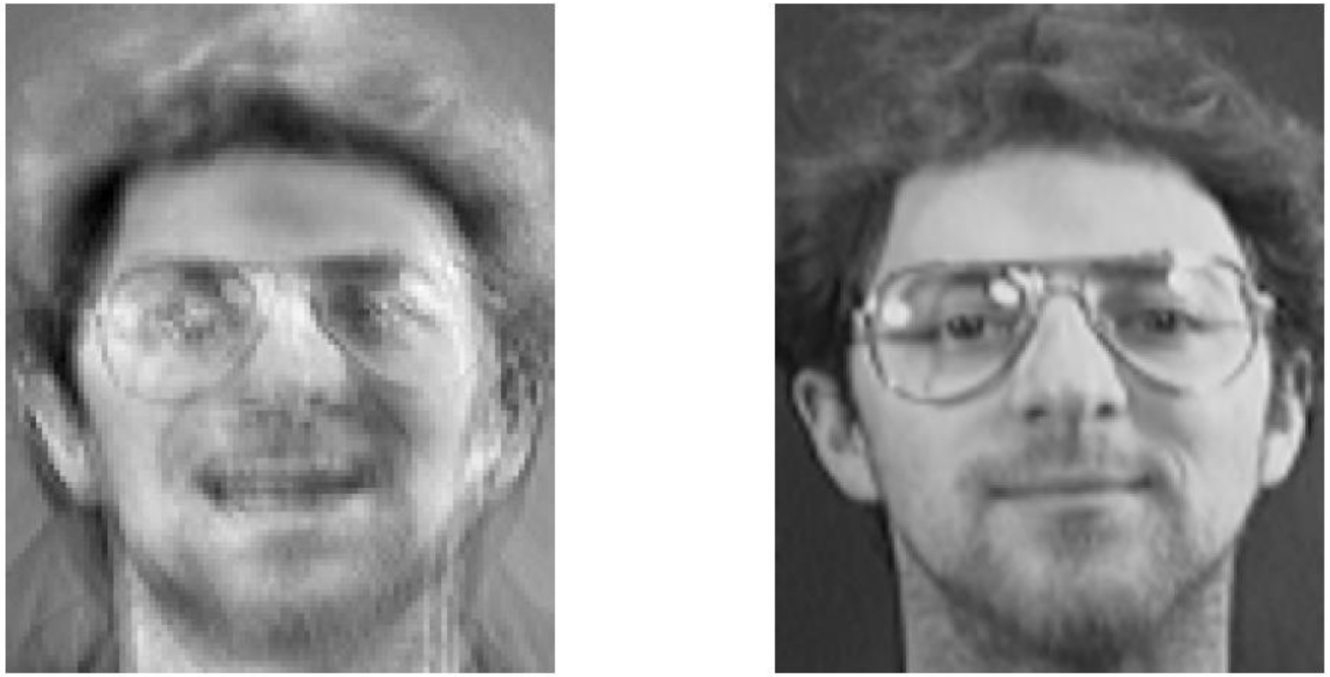
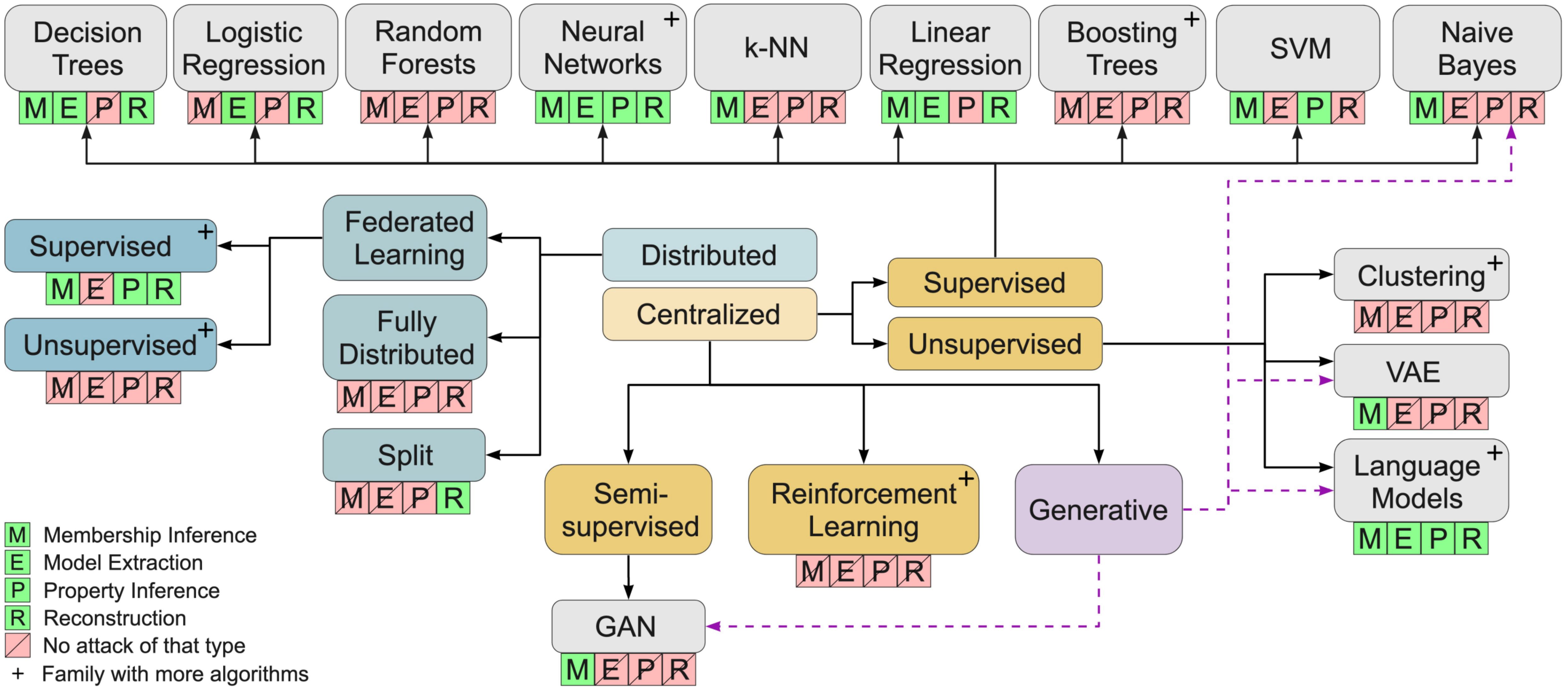


Figure 1: An image recovered using a new model inversion attack (left) and a training set image of the victim (right). The attacker is given only the person’s name and access to a facial recognition system that returns a class confidence score.

Overview of Privacy Attacks

M. Rigaki and S. Garcia, 'A Survey of Privacy Attacks in Machine Learning', arXiv:2007.07646 [cs], Apr. 2021, Accessed: Aug. 06, 2021. [Online]. Available: http://arxiv.org/abs/2007.07646



Selected Privacy-Preserving Techniques

Selected Privacy-Preserving Techniques

1. Ensemble of Teacher Models
2. Differential Privacy
3. Distributed Training
4. Training over Encrypted Data
5. Trusted Execution

Ensemble of Teacher Models

1. Ensemble of Teacher Models

N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, 'Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data', arXiv:1610.05755 [cs, stat], Mar. 2017, Accessed: Sep. 19, 2021.

Motivation and Proposal

- Models may inadvertently or implicitly store some of its training data; careful analysis of the model may reveal sensitive data.
- Private Aggregation of Teacher Ensembles (PATE): approach to providing strong privacy guarantees for training data

How does it work?

- PATE combines, in a black-box fashion, multiple models trained with disjoint datasets (e.g., records of different subsets of users).
- As the models rely directly on sensitive data, their are not published, but instead used as "teachers" for a "student" model.
- The student learns to predict an output chosen by noisy voting among all of the teachers, and cannot directly access an individual teacher or the underlying data or parameters.

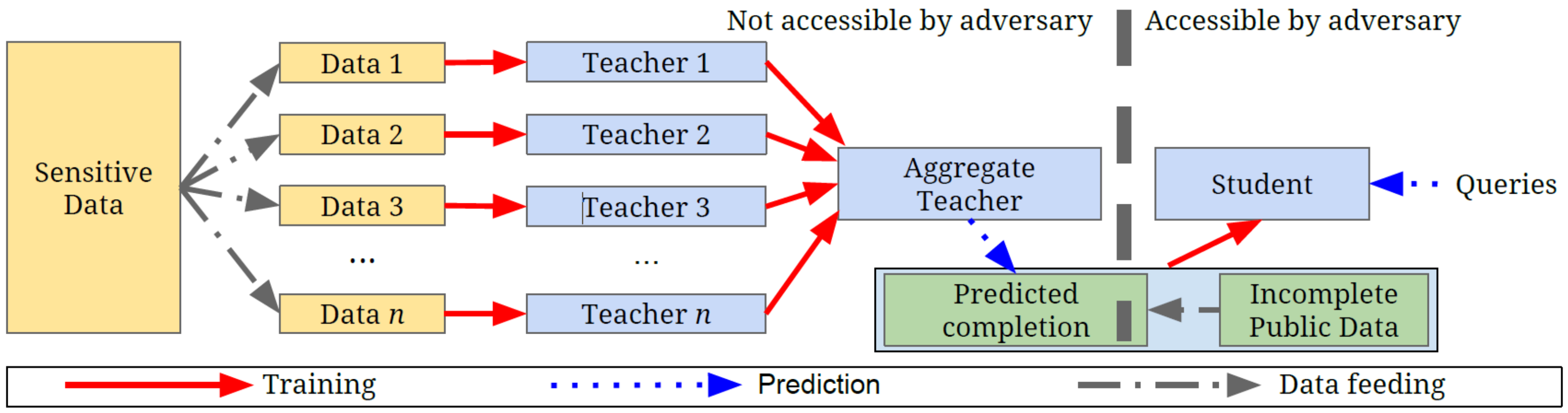


Figure 1: Overview of the approach: (1) an ensemble of teachers is trained on disjoint subsets of the sensitive data, (2) a student model is trained on public data labeled using the ensemble.

1. Ensemble of Teacher Models

Properties and Applicability

- The student’s privacy properties can be understood both intuitively (since no single teacher and thus no single dataset dictates the student’s training) and formally, in terms of differential privacy.
- These properties hold even if an adversary can not only query the student but also inspect its internal workings.
- Compared with previous work, the approach imposes only weak assumptions on how teachers are trained: it applies to any model, including non-convex models like DNNs.

Differential Privacy

2. Differential Privacy

M. Abadi et al., 'Deep Learning with Differential Privacy', Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308–318, Oct. 2016, doi: 10.1145/2976749.2978318.

Differential privacy: constitutes a strong standard for privacy guarantees for algorithms on aggregate databases.

- It is defined in terms of the application-specific concept of adjacent databases.
- Consider each training dataset is a set of image-label pairs: two of these sets are adjacent if they differ in a single entry, i.e., if one image-label pair is present in one set and absent in the other.
- Definition. A randomized mechanism *M* : *D* → *R* with domain D and range R satisfies (ε, δ)-differential privacy if for any two adjacent inputs d, d0 ∈ D and for any subset of outputs S ⊆ R it holds that

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

Adding Noise

- A common paradigm for approximating a deterministic real-valued function *f* : *D* → *R* with a differentially private mechanism is **via additive noise** calibrated to *f*'s sensitivity S_f, which is defined as the maximum of the absolute distance |*f(d)* - *f(d')*| where *d* and *d'* are adjacent inputs.
- Consider the Gaussian noise mechanism defined by $M(d) \triangleq f(d) + N(0, S_f^2\sigma^2)$
- A single application of the Gaussian mechanism to function *f* of sensitivity S_f satisfies (ε, δ)-differential privacy if

$$\delta \geq \frac{4}{5}\exp(-(\sigma\varepsilon)^2/2) \quad \text{and} \quad \varepsilon > 1$$

2. Differential Privacy

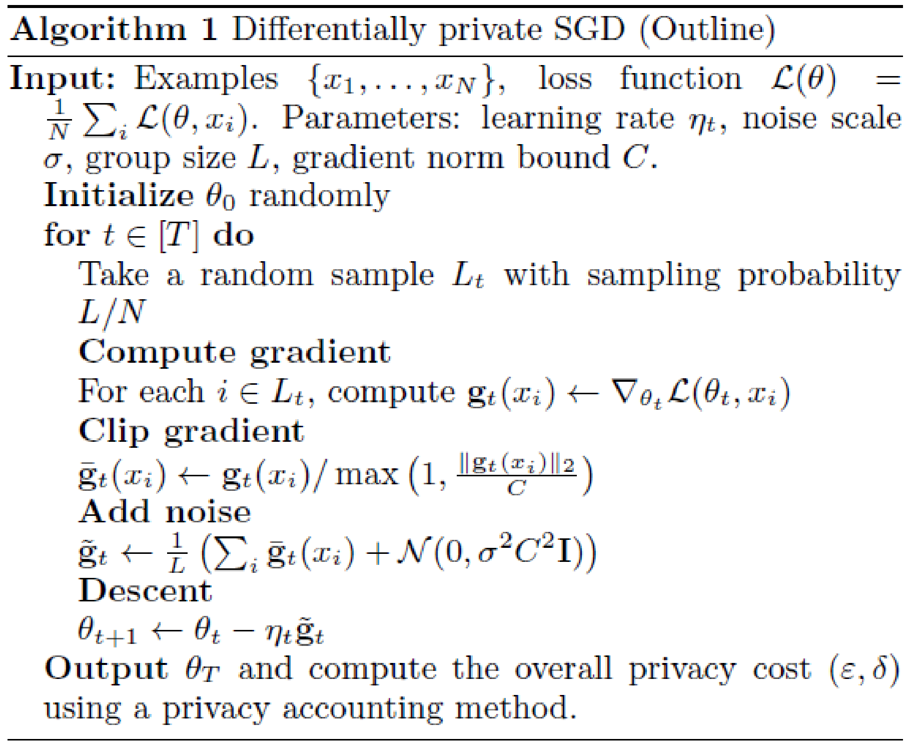
M. Abadi et al., ‘Deep Learning with Differential Privacy’, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308–318, Oct. 2016, doi: 10.1145/2976749.2978318.

Privacy Accountant

- Differential privacy for repeated applications of additive noise mechanisms follows from composition theorems.
- The task of keeping track of the accumulated privacy loss in the course of execution of a composite mechanism, and enforcing the applicable privacy policy, can be performed by the privacy accountant.

Differentially Private Stochastic Gradient Descent Algorithm

- Method for training a model with parameters by minimizing empirical loss function $\mathcal{L}(\theta)$.
- At each step of the SGD:
    1. compute the gradient $\nabla_\theta \mathcal{L}(\theta, x_i)$ for a random subset of examples;
    2. clip the ℓ_2 norm of each gradient,
    3. compute the average,
    4. add noise in order to protect privacy,
    5. take a step in the opposite direction of this average noisy gradient.
    6. compute privacy loss of the mechanism *(based on the information maintained by the privacy accountant)*.



```
Algorithm 1 Differentially private SGD (Outline)
Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate $\eta_t$, noise scale
  $\sigma$, group size $L$, gradient norm bound $C$.
  Initialize $\theta_0$ randomly
  for $t \in [T]$ do
    Take a random sample $L_t$ with sampling probability $L/N$
    Compute gradient
    For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$
    Clip gradient
    $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$
    Add noise
    $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
    Descent
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$
  Output $\theta_T$ and compute the overall privacy cost $(\varepsilon, \delta)$
  using a privacy accounting method.
```

Distributed Learning

3. Distributed Learning

R. Shokri and V. Shmatikov, 'Privacy-Preserving Deep Learning', in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver Colorado USA, Oct. 2015, pp. 1310–1321. doi: 10.1145/2810103.2813687.

- Recall the fundamentals of distributed or federated learning
  1. Nodes train with local data and produce model updates
  2. Model updates are leveraged to train a global model
  3. Nodes download globally-improved models or parameters

Distributed/federated learning offers an inherent level of privacy, as local data does not leave the respective node

- The work of [Shokri15] leverages and applies this architecture to enable multiple parties to jointly learn an accurate neural network model for a given objective, without sharing their input datasets.
- The training of neural networks relies to a great extent on Gradient Descent; the authors propose Selective Stochastic Gradient Descent (SSGD).
- The main intuition behind selective parameter update is that during SGD, some parameters contribute much more to the neural network's objective function and thus undergo much bigger updates during a given iteration of training.

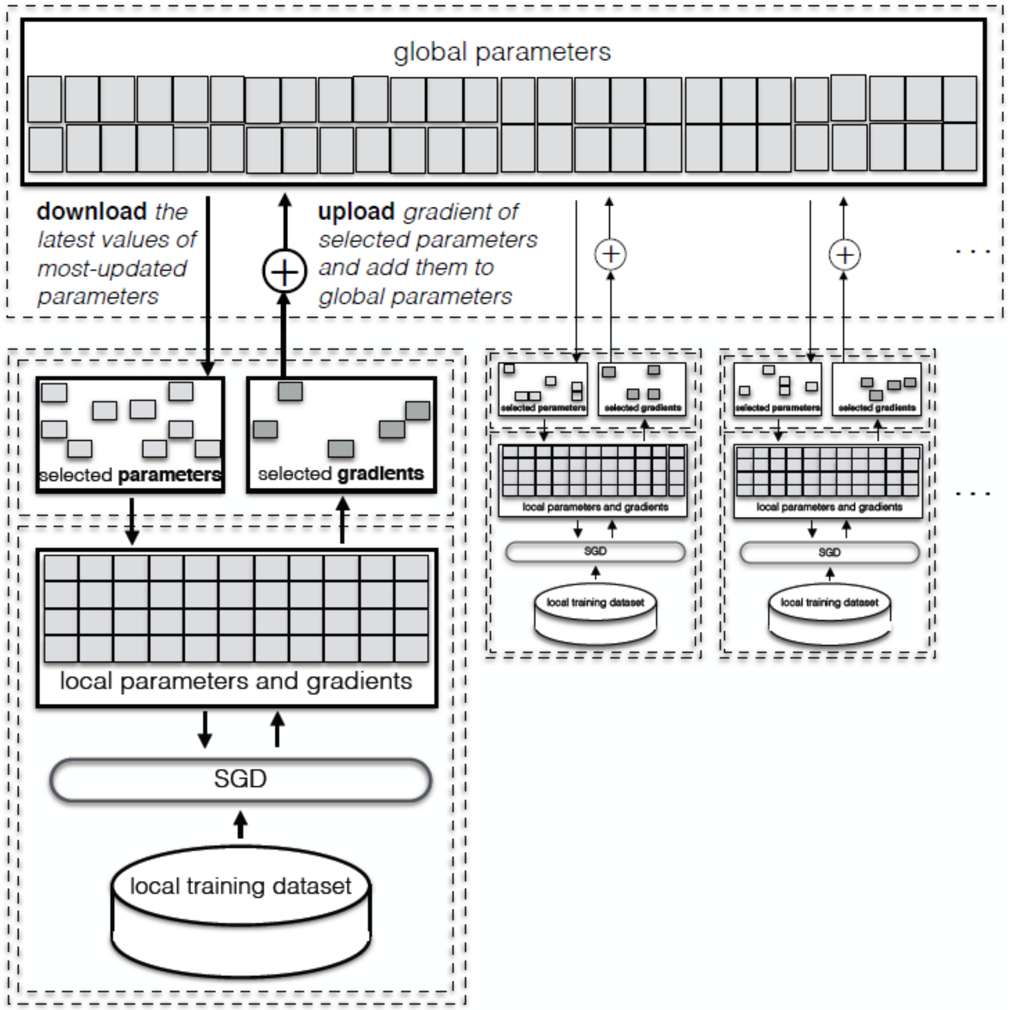

global parameters
download *the latest values of most-updated parameters*
upload *gradient of selected parameters and add them to global parameters*
selected **parameters**
selected **gradients**
local parameters and gradients
SGD
local training dataset
. . .

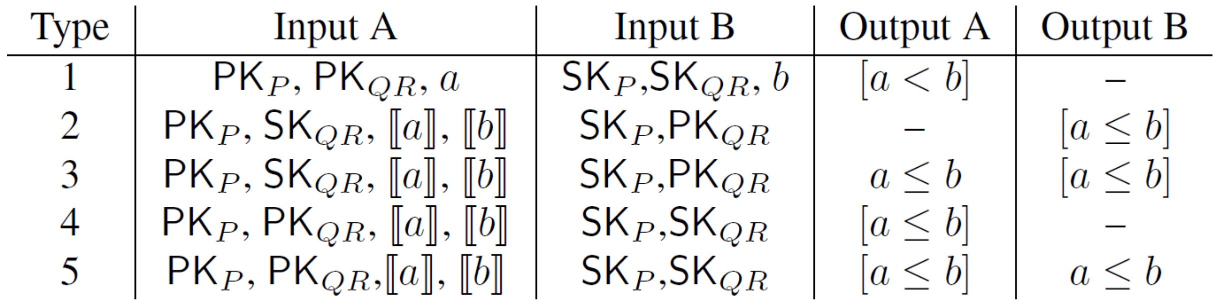
Figure 2: High-level architecture of our deep learning system. An abstract model of the parameter server, which maintains global values for the parameters, is depicted at the top.

ML over Encrypted Data

4. Classification over Encrypted Data

R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, ‘Machine Learning Classification over Encrypted Data’, presented at the Network and Distributed System Security Symposium, San Diego, CA, 2015. doi: 10.14722/ndss.2015.23241.

- Computations can be performed over encrypted data – Homomorphic
- Authors [Bost2015] design and present three classifiers that satisfy this privacy constraint:
  - Hyperplane decision | Naïve Bayes | Decision trees
- The authors define a library of building blocks with which the classifiers are implemented:
  1. Comparison with unencrypted inputs
  2. Comparison with encrypted inputs
  3. Reversed comparison over encrypted data
  4. Negative integers comparison and sign determination
  5. argmax over encrypted data
  6. Computing dot products
  7. Dealing with floating point numbers



Type	Input A	Input B	Output A	Output B
1	PK_P, PK_{QR}, a	SK_P,SK_{QR}, b	$[a < b]$	–
2	PK_P, SK_{QR}, $[\![a]\!]$, $[\![b]\!]$	SK_P,PK_{QR}	–	$[a \leq b]$
3	PK_P, SK_{QR}, $[\![a]\!]$, $[\![b]\!]$	SK_P,PK_{QR}	$a \leq b$	$[a \leq b]$
4	PK_P, PK_{QR}, $[\![a]\!]$, $[\![b]\!]$	SK_P,SK_{QR}	$[a \leq b]$	–
5	PK_P, PK_{QR},$[\![a]\!]$, $[\![b]\!]$	SK_P,SK_{QR}	$[a \leq b]$	$a \leq b$

- The authors then describe how to implement the 3 classifiers above with the building blocks

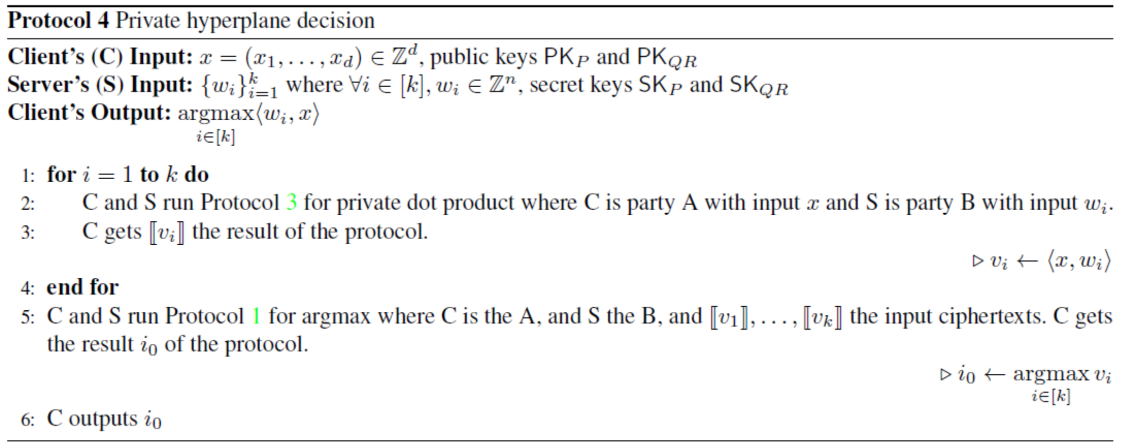
4. Classification over Encrypted Data

R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, ‘Machine Learning Classification over Encrypted Data’, presented at the Network and Distributed System Security Symposium, San Diego, CA, 2015. doi: 10.14722/ndss.2015.23241.

- **Private hyperplane decision**
  - This classifier computes Argmax

$$k^* = \underset{i \in [k]}{\operatorname{argmax}} \langle w_i, x \rangle$$

Argmax: an operation that finds the argument that gives the maximum value from a target function.



Protocol 4 Private hyperplane decision

Client's (C) Input: $x = (x_1, \ldots, x_d) \in \mathbb{Z}^d$, public keys PK_P and PK_{QR}
Server's (S) Input: $\{w_i\}_{i=1}^k$ where $\forall i \in [k], w_i \in \mathbb{Z}^n$, secret keys SK_P and SK_{QR}
Client's Output: $\underset{i \in [k]}{\operatorname{argmax}} \langle w_i, x \rangle$

```
1: for $i = 1$ to $k$ do
2:     C and S run Protocol 3 for private dot product where C is party A with input $x$ and S is party B with input $w_i$.
3:     C gets $[\![v_i]\!]$ the result of the protocol.
                                                              $\triangleright\ v_i \leftarrow \langle x, w_i \rangle$
4: end for
5: C and S run Protocol 1 for argmax where C is the A, and S the B, and $[\![v_1]\!], \ldots, [\![v_k]\!]$ the input ciphertexts. C gets
   the result $i_0$ of the protocol.
                                                              $\triangleright\ i_0 \leftarrow \underset{i \in [k]}{\operatorname{argmax}}\, v_i$
6: C outputs $i_0$
```

Protocol for implementing private hyperplane decision

  - Example of a building block: argmax
  - Computation count:
    - k - 1 encrypted comparisons of *L* bits integers;
    - 7(k - 1) homomorphic operations (refreshes, multiplications, subtractions);
    - k - 1 roundtrips to the comparison protocol

4. Classification over Encrypted Data

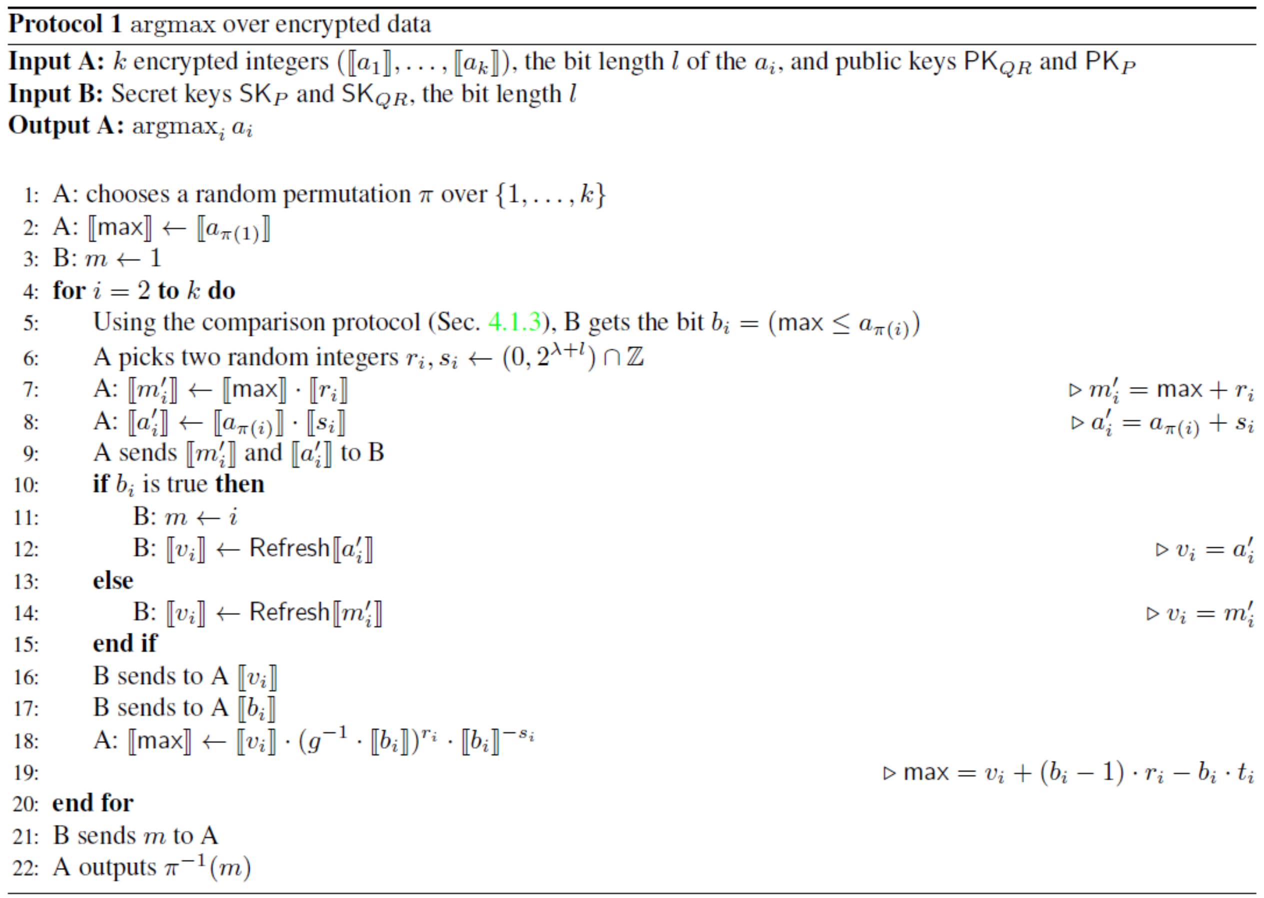
R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, 'Machine Learning Classification over Encrypted Data', presented at the Network and Distributed System Security Symposium, San Diego, CA, 2015. doi: 10.14722/ndss.2015.23241.

- **Private hyperplane decision**
  - This classifier computes Argmax

$$k^* = \underset{i \in [k]}{\operatorname{argmax}} \langle w_i, x \rangle$$

Argmax: an operation that finds the argument that gives the maximum value from a target function.

  - Example of a building block: argmax
  - Computation count:
    - k - 1 encrypted comparisons of *L* bits integers;
    - 7(k - 1) homomorphic operations (refreshes, multiplications, subtractions);
    - k - 1 roundtrips to the comparison protocol



Protocol 1 argmax over encrypted data

Input A: k encrypted integers $(\llbracket a_1 \rrbracket, \dots, \llbracket a_k \rrbracket)$, the bit length l of the a_i, and public keys PK_{QR} and PK_P
Input B: Secret keys SK_P and SK_{QR}, the bit length l
Output A: $\operatorname{argmax}_i a_i$

```
1:  A: chooses a random permutation $\pi$ over $\{1, \dots, k\}$
2:  A: $\llbracket \mathsf{max} \rrbracket \leftarrow \llbracket a_{\pi(1)} \rrbracket$
3:  B: $m \leftarrow 1$
4:  for $i = 2$ to $k$ do
5:      Using the comparison protocol (Sec. 4.1.3), B gets the bit $b_i = (\mathsf{max} \le a_{\pi(i)})$
6:      A picks two random integers $r_i, s_i \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$
7:      A: $\llbracket m'_i \rrbracket \leftarrow \llbracket \mathsf{max} \rrbracket \cdot \llbracket r_i \rrbracket$        $\triangleright m'_i = \mathsf{max} + r_i$
8:      A: $\llbracket a'_i \rrbracket \leftarrow \llbracket a_{\pi(i)} \rrbracket \cdot \llbracket s_i \rrbracket$        $\triangleright a'_i = a_{\pi(i)} + s_i$
9:      A sends $\llbracket m'_i \rrbracket$ and $\llbracket a'_i \rrbracket$ to B
10:     if $b_i$ is true then
11:         B: $m \leftarrow i$
12:         B: $\llbracket v_i \rrbracket \leftarrow \mathsf{Refresh}\llbracket a'_i \rrbracket$        $\triangleright v_i = a'_i$
13:     else
14:         B: $\llbracket v_i \rrbracket \leftarrow \mathsf{Refresh}\llbracket m'_i \rrbracket$        $\triangleright v_i = m'_i$
15:     end if
16:     B sends to A $\llbracket v_i \rrbracket$
17:     B sends to A $\llbracket b_i \rrbracket$
18:     A: $\llbracket \mathsf{max} \rrbracket \leftarrow \llbracket v_i \rrbracket \cdot (g^{-1} \cdot \llbracket b_i \rrbracket)^{r_i} \cdot \llbracket b_i \rrbracket^{-s_i}$
19:                                                  $\triangleright \mathsf{max} = v_i + (b_i - 1) \cdot r_i - b_i \cdot t_i$
20: end for
21: B sends $m$ to A
22: A outputs $\pi^{-1}(m)$
```

Protocol for implementing argmax, a building block for classifiers, over encrypted data

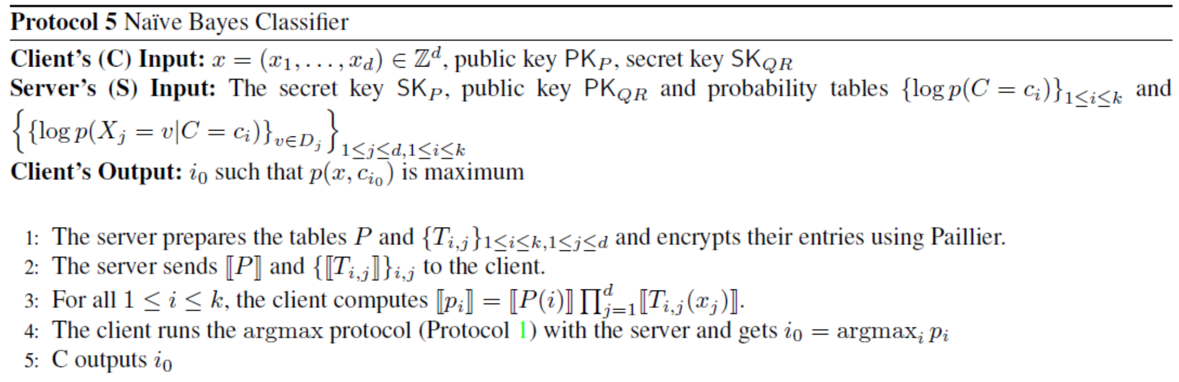
4. Classification over Encrypted Data

R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, '**Machine Learning Classification over Encrypted Data**', presented at the Network and Distributed System Security Symposium, San Diego, CA, 2015. doi: 10.14722/ndss.2015.23241.

- **Secure Naïve Bayes classifier**
  - Also based on argmax

$$k^* = \operatorname*{argmax}_{i\in[k]} \log p(C = c_i | X = x)$$

$$= \operatorname*{argmax}_{i\in[k]} \left\{ \log p(C = c_i) + \sum_{j=1}^{d} \log p(X_j = x_j | C = c_i) \right\}$$



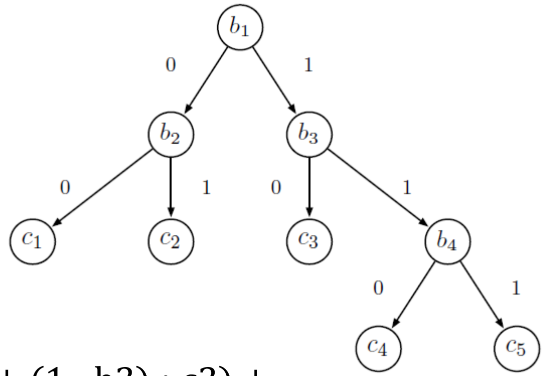
Protocol 5 Naïve Bayes Classifier

Client's (C) Input: $x = (x_1, \dots, x_d) \in \mathbb{Z}^d$, public key PK_P, secret key SK_{QR}
Server's (S) Input: The secret key SK_P, public key PK_{QR} and probability tables $\{\log p(C = c_i)\}_{1\le i\le k}$ and $\left\{\{\log p(X_j = v | C = c_i)\}_{v\in D_j}\right\}_{1\le j\le d, 1\le i\le k}$
Client's Output: i_0 such that $p(x, c_{i_0})$ is maximum

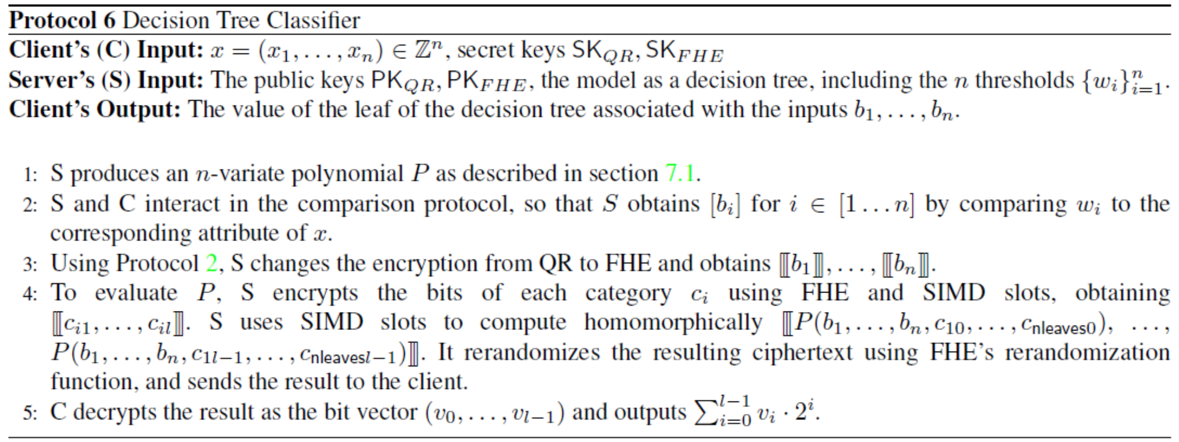
```
1: The server prepares the tables $P$ and $\{T_{i,j}\}_{1\le i\le k, 1\le j\le d}$ and encrypts their entries using Paillier.
2: The server sends $[P]$ and $\{[T_{i,j}]\}_{i,j}$ to the client.
3: For all $1 \le i \le k$, the client computes $[p_i] = [P(i)] \prod_{j=1}^{d} [T_{i,j}(x_j)]$.
4: The client runs the argmax protocol (Protocol 1) with the server and gets $i_0 = \operatorname{argmax}_i p_i$
5: C outputs $i_0$
```

Protocol for implementing private naïve Bayes

- **Private decision trees**
  - Decision trees can be described by polynomials



P(b1, b2, b3, b4, c1, … , c5) =
b1 (b3 · (b4 · c5 + (1 - b4) · c4) + (1 - b3) · c3) +
(1 - b1) (b2 · c2 + (1 - b2) · c1)



Protocol 6 Decision Tree Classifier

Client's (C) Input: $x = (x_1, \dots, x_n) \in \mathbb{Z}^n$, secret keys $\mathsf{SK}_{QR}, \mathsf{SK}_{FHE}$
Server's (S) Input: The public keys $\mathsf{PK}_{QR}, \mathsf{PK}_{FHE}$, the model as a decision tree, including the n thresholds $\{w_i\}_{i=1}^{n}$.
Client's Output: The value of the leaf of the decision tree associated with the inputs $b_1, \dots, b_n$.

```
1: S produces an $n$-variate polynomial $P$ as described in section 7.1.
2: S and C interact in the comparison protocol, so that $S$ obtains $[b_i]$ for $i \in [1 \dots n]$ by comparing $w_i$ to the corresponding attribute of $x$.
3: Using Protocol 2, S changes the encryption from QR to FHE and obtains $[\![b_1]\!], \dots, [\![b_n]\!]$.
4: To evaluate $P$, S encrypts the bits of each category $c_i$ using FHE and SIMD slots, obtaining $[\![c_{i1}, \dots, c_{il}]\!]$. S uses SIMD slots to compute homomorphically $[\![P(b_1, \dots, b_n, c_{10}, \dots, c_{\text{nleaves}0}), \dots, P(b_1, \dots, b_n, c_{1l-1}, \dots, c_{\text{nleaves}l-1})]\!]$. It rerandomizes the resulting ciphertext using FHE's rerandomization function, and sends the result to the client.
5: C decrypts the result as the bit vector $(v_0, \dots, v_{l-1})$ and outputs $\sum_{i=0}^{l-1} v_i \cdot 2^i$.
```

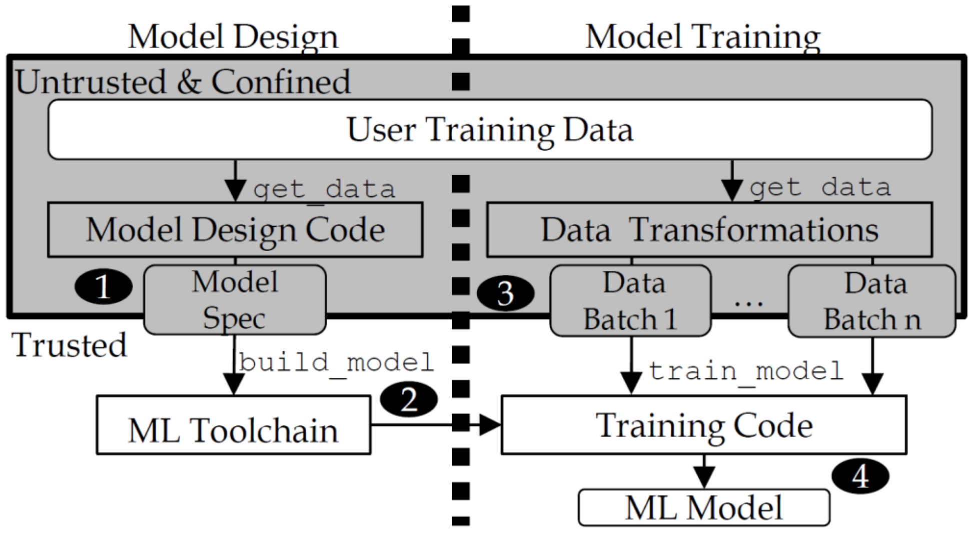
Protocol for implementing private decision trees

Trusted Execution

5. Trusted Execution

T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, 'Chiron: Privacy-preserving Machine Learning as a Service', arXiv:1803.05961 [cs], Mar. 2018, Accessed: Sep. 19, 2021. [Online]. Available: http://arxiv.org/abs/1803.05961

- Machine learning as a service (MLaaS)
  1. The service provider is free to choose the type of the model to train, how to configure and train it, and what transformations, if any, to apply to the inputs into the model.
     These choices can adaptively depend on the user's data and ML task.
  2. The user obtains API access to the trained model but no other information about it.
- Intel Software Guard Extensions (SGX) provide enclaves. Code in an enclave can safely operate on secret data without fear of unintentional disclosure to the platform.
- Chiron is a privacy-preserving architecture for MLaaS:
  - Chiron conceals the training data from the service operator:
    1. Service provider provides enclaves to user.
    2. User establishes secure communication channels with enclaves to transfer data directly.
  - Chiron reveals neither the training algorithm nor the model structure to the user, providing only black-box access to the trained model.



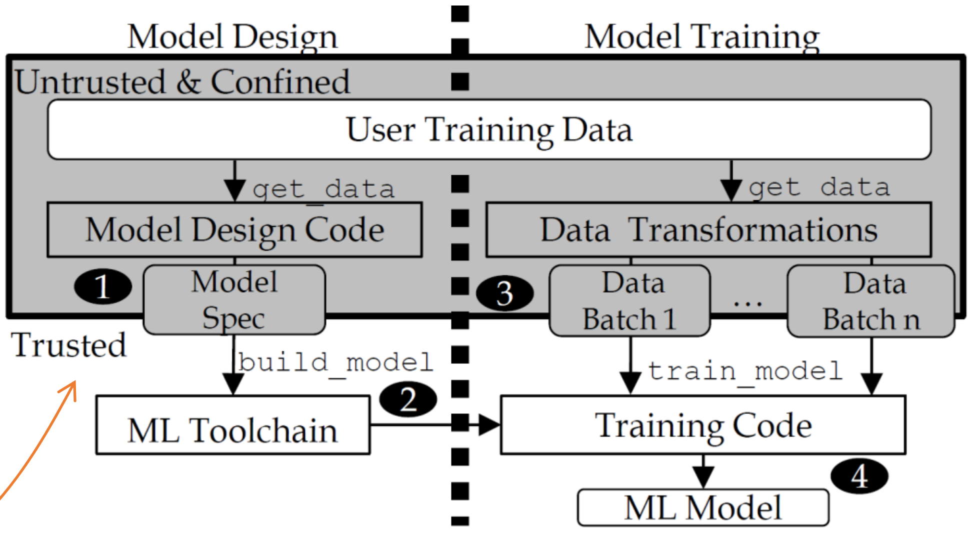
Training enclave architecture.
1) Untrusted service provider code examines data, generates model spec and passes it to the ML toolchain.
2) ML toolchain uses the spec to generate model-training code.
3) Service provider code transforms data and breaks it into batches for training.
4) Model-training code is invoked for each batch, updating the model.

5. Trusted Execution

T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, 'Chiron: Privacy-preserving Machine Learning as a Service', arXiv:1803.05961 [cs], Mar. 2018, Accessed: Sep. 19, 2021. [Online]. Available: http://arxiv.org/abs/1803.05961

Trusted vs. Untrusted code - Sandbox

- To enforce data confidentiality while allowing the provider to select, configure, and train a model any way they want, Chiron employs a Ryoan sandbox (itself based on the hardware-protected enclave).
- An enclave alone is insufficient because it only protects trusted code executing on an untrusted platform.
- Code can only be trusted if it is public and thus can be checked by users.
- In Chiron, however, the ML service provider's code is untrusted, thus users must be assured that this code is not stealing their data even though they cannot inspect it.



Sandbox (Ryoan)

- Ryoan [41] enables service providers to keep proprietary code secret while simultaneously ensuring users that the confined code cannot leak their data.
- Instead of asking users to trust the provider's code, Ryoan asks them to trust the sandbox that confines this code.
- Users can audit Ryoan to gain confidence in its correctness.

Thank You