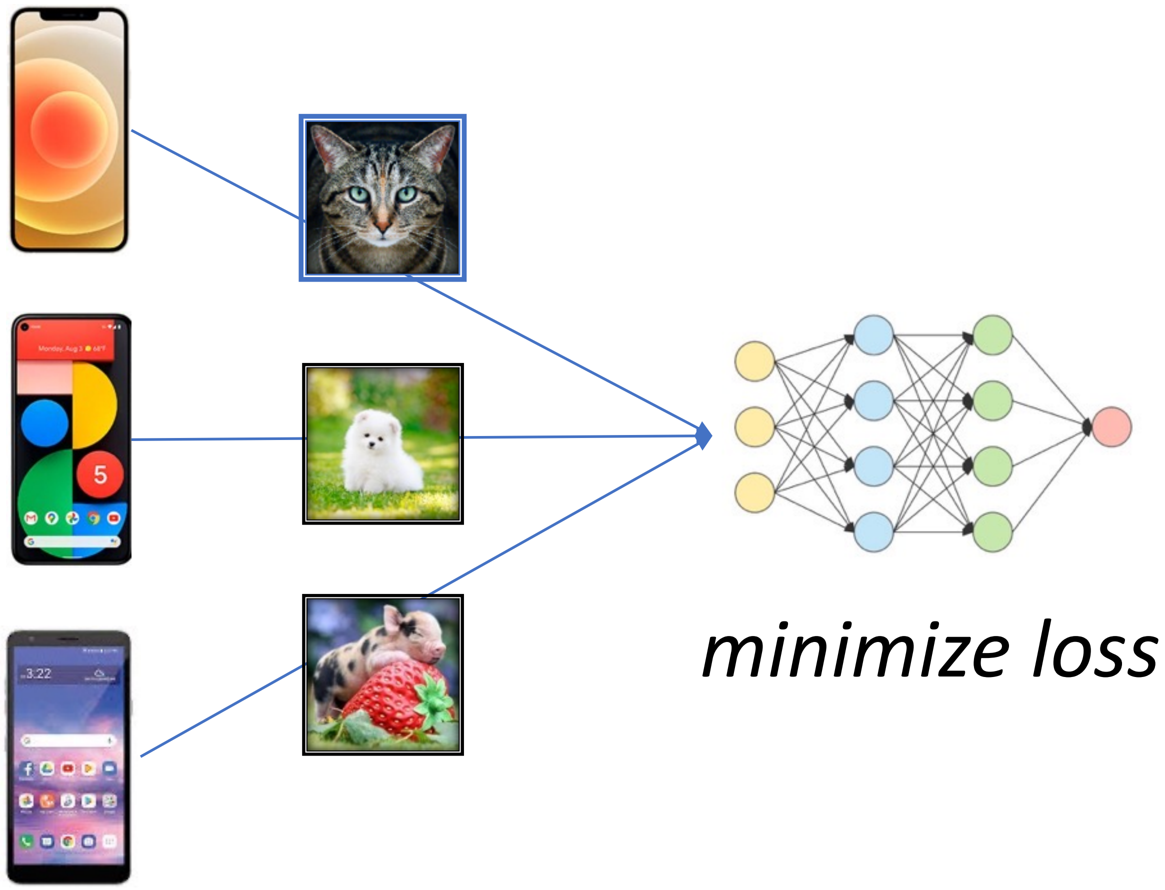
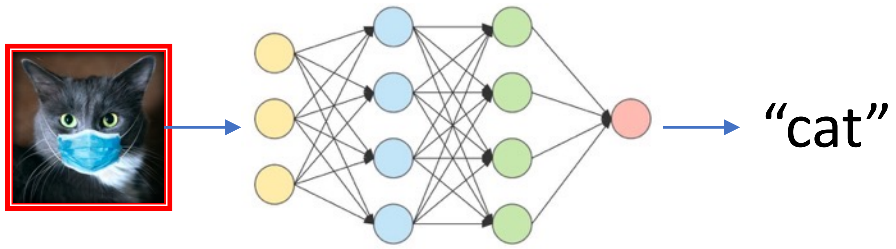
From average-case to worst-case privacy leakage in neural networks

Florian Tramèr

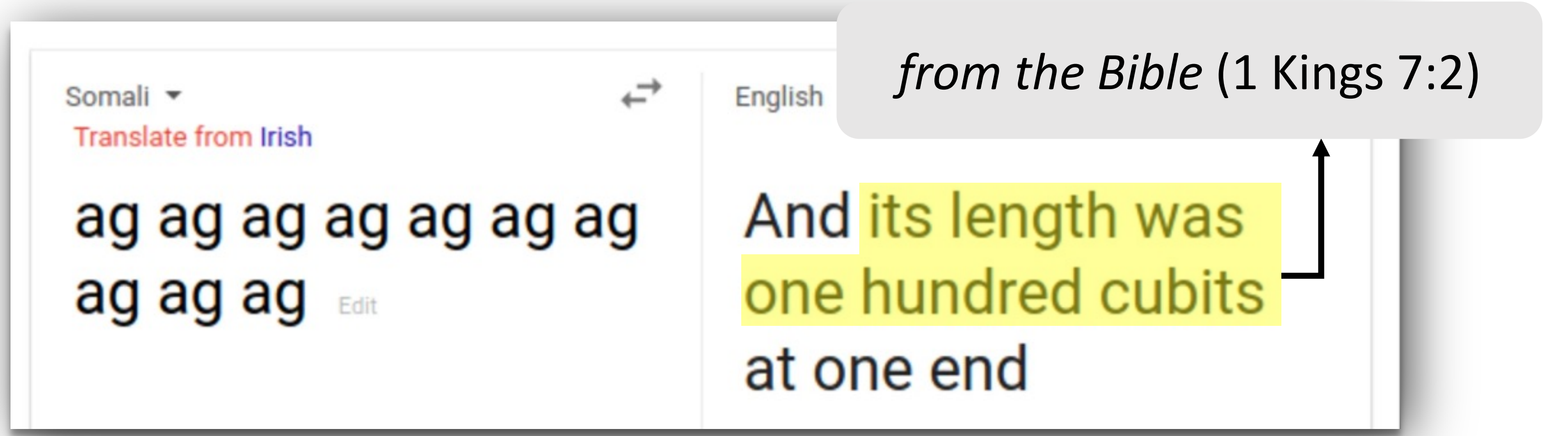
Google Brain & ETHZ

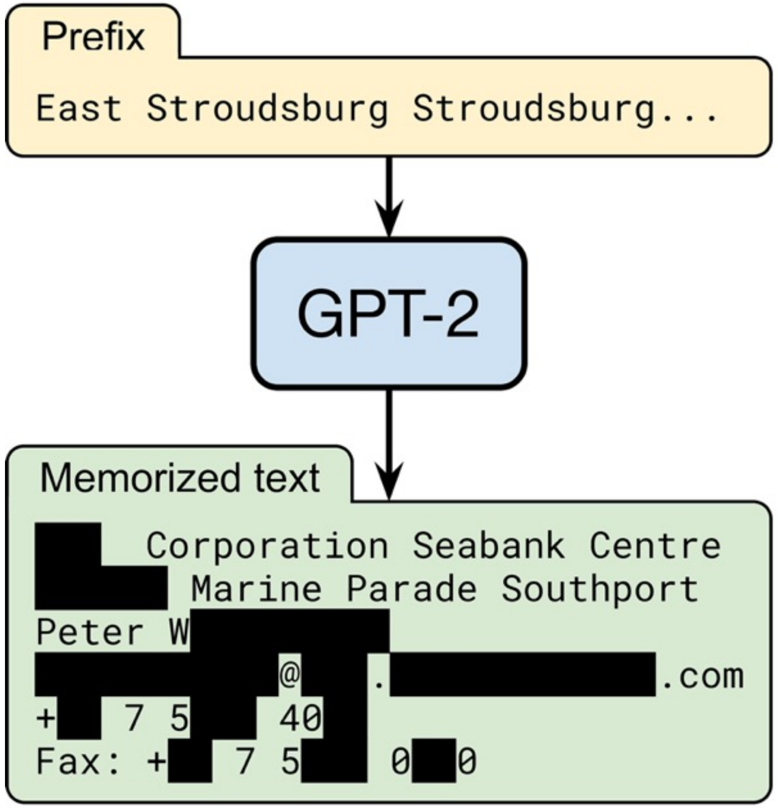
Neural networks learn from a (private) training set.



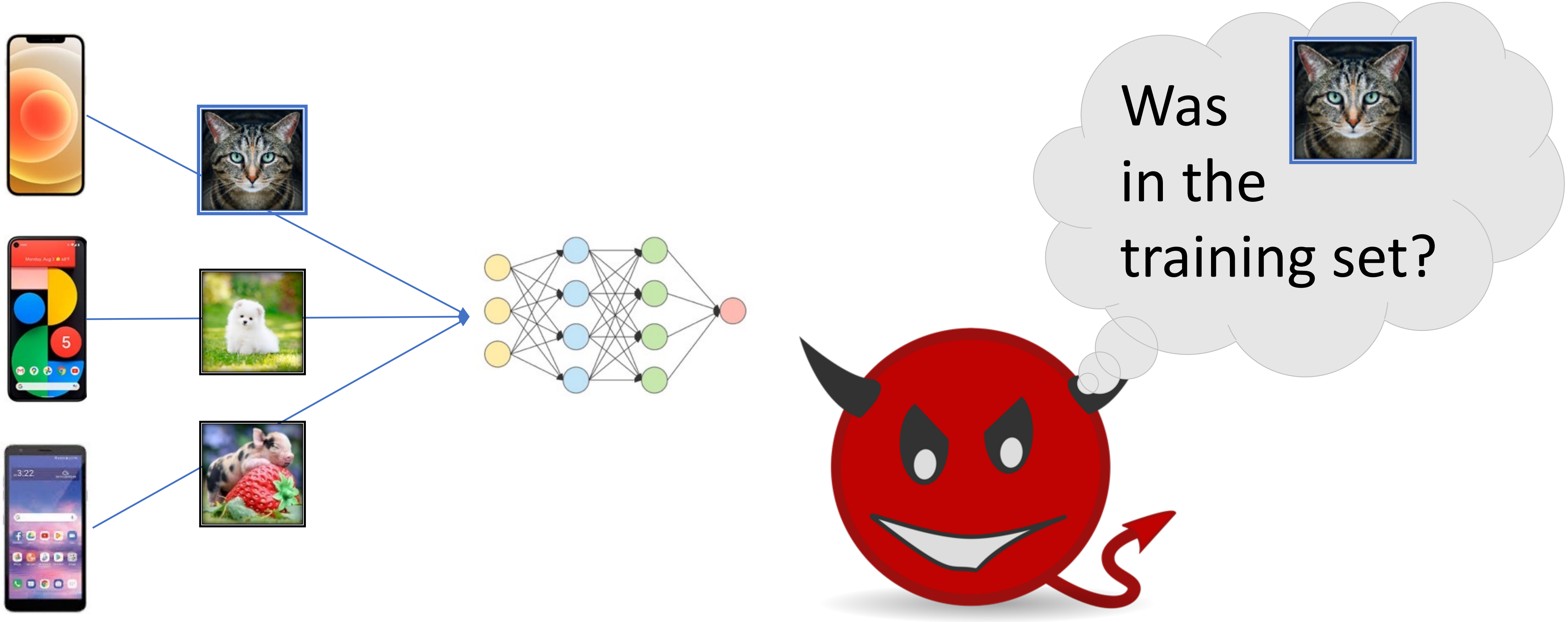


The trained model might *leak* the training set.





This talk: Membership inference attacks



Why should we care about membership inference?

1. A real attack (e.g., models trained on medical data)
2. An attack component (e.g., for data extraction)
3. A simple, formal upper-bound on data leakage

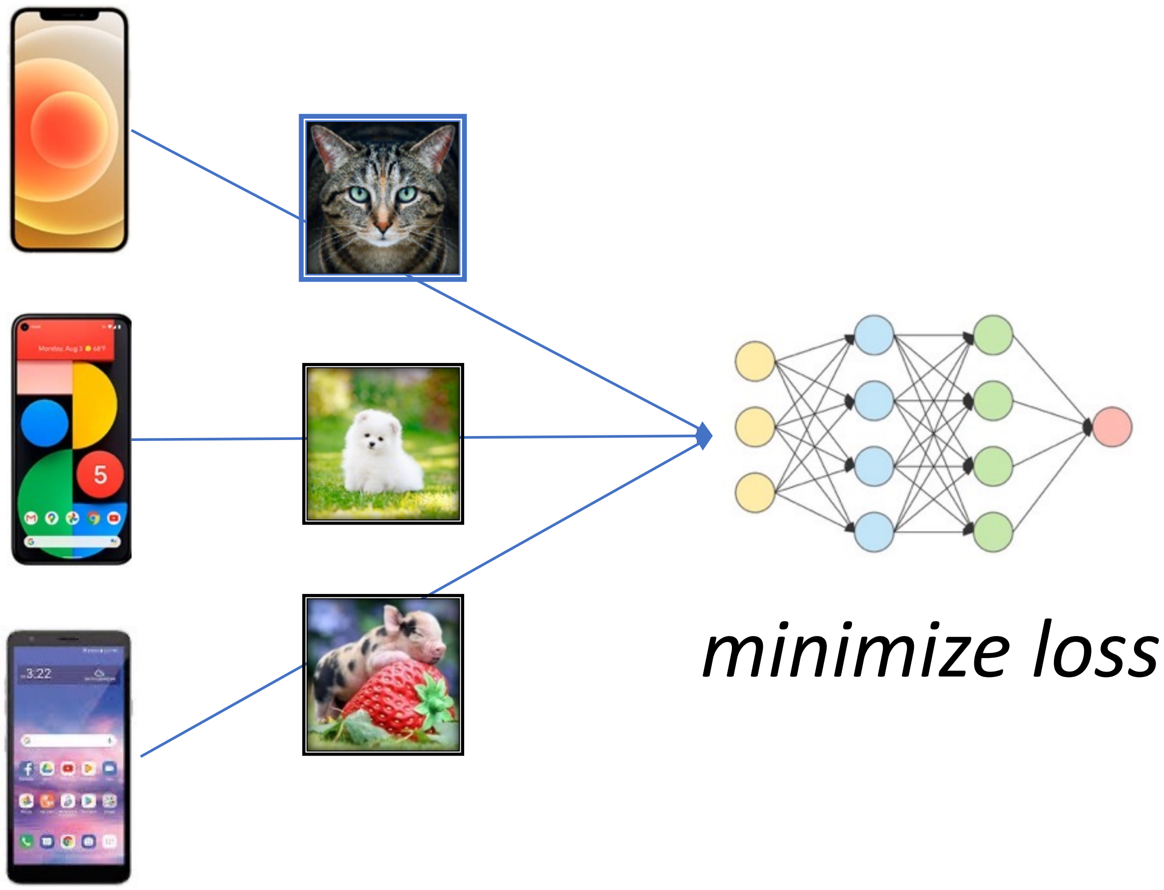
Data privacy	$\approx$	Secure encryption
Membership inference attack		Chosen plaintext attack

Outline.

- Most membership attacks (and their evaluations) *are flawed*
- A new principled attack that works on *outliers*
- A new stronger attack that works for *any input*
- Defenses and *how to audit them*

A simple MI attack: loss thresholding

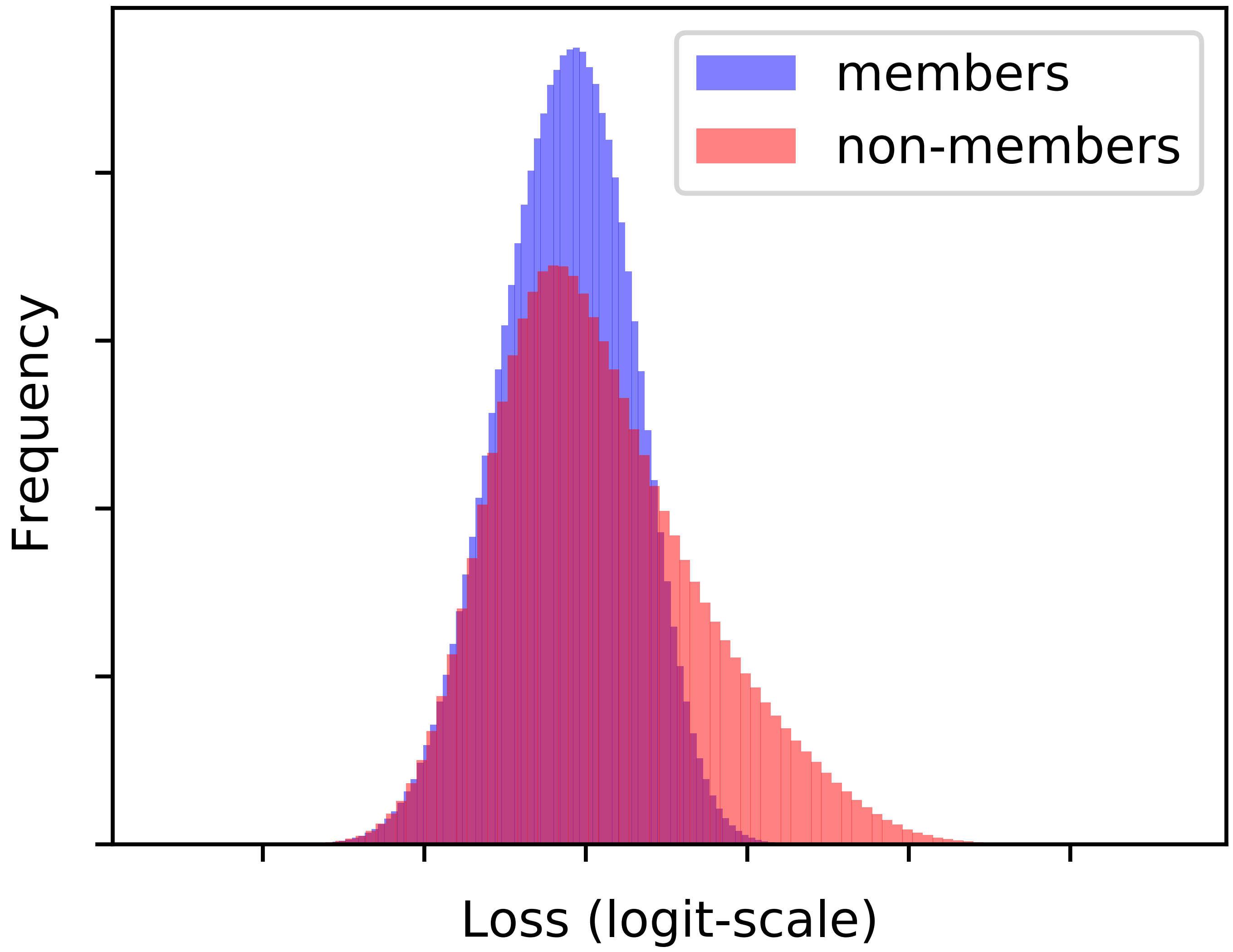
[Yeom et al.'18]



minimize loss

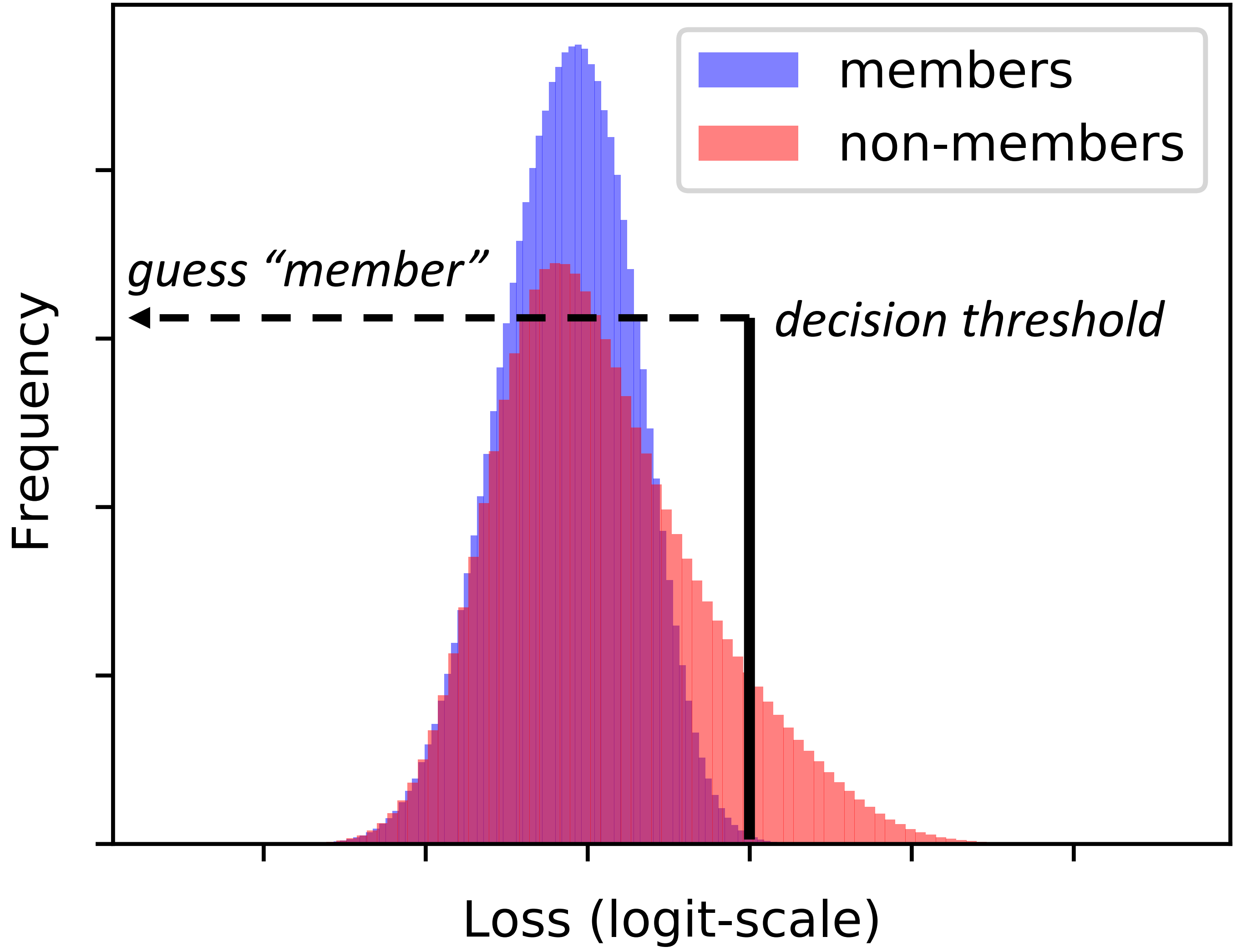
A simple MI attack: loss thresholding

[Yeom et al.'18]



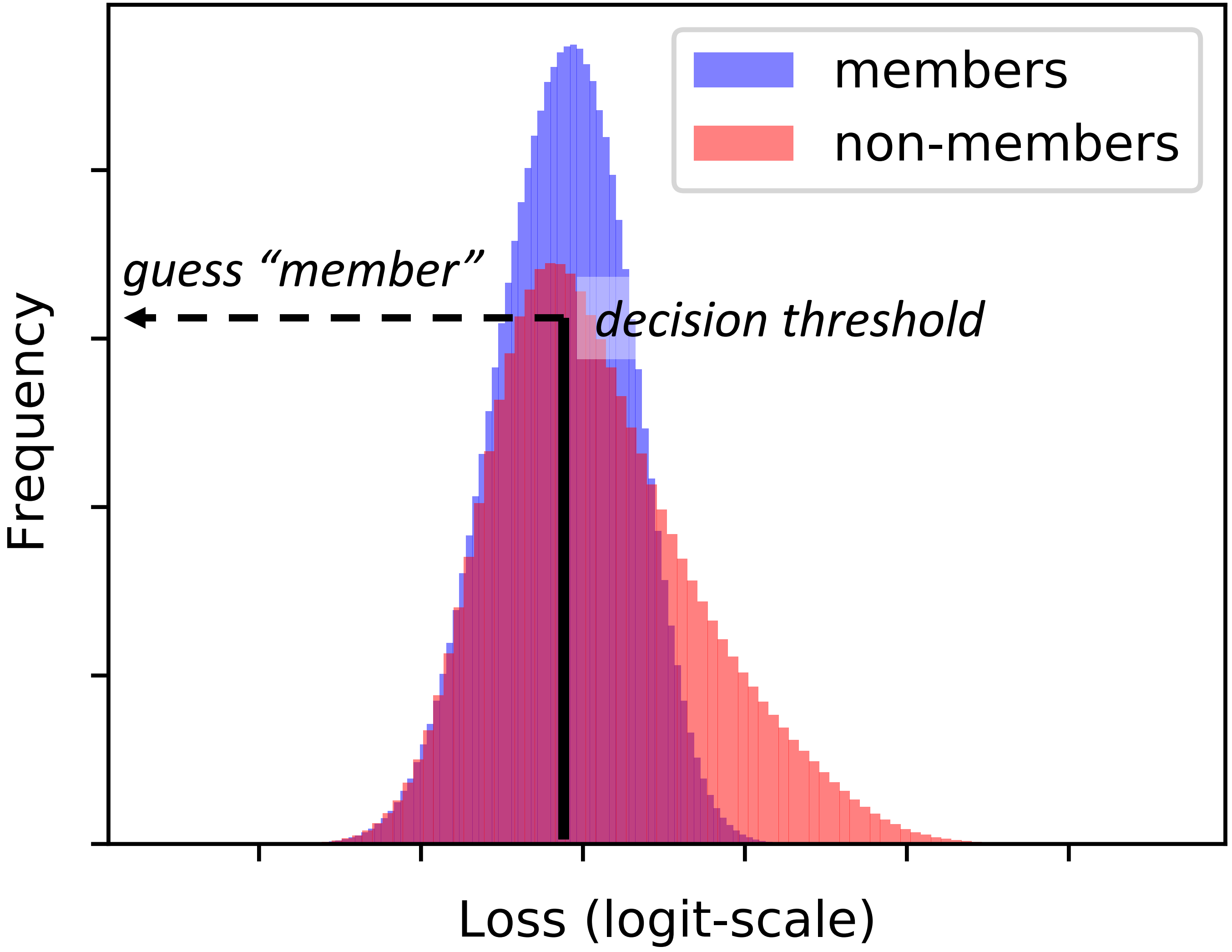
A simple MI attack: loss thresholding

[Yeom et al.'18]



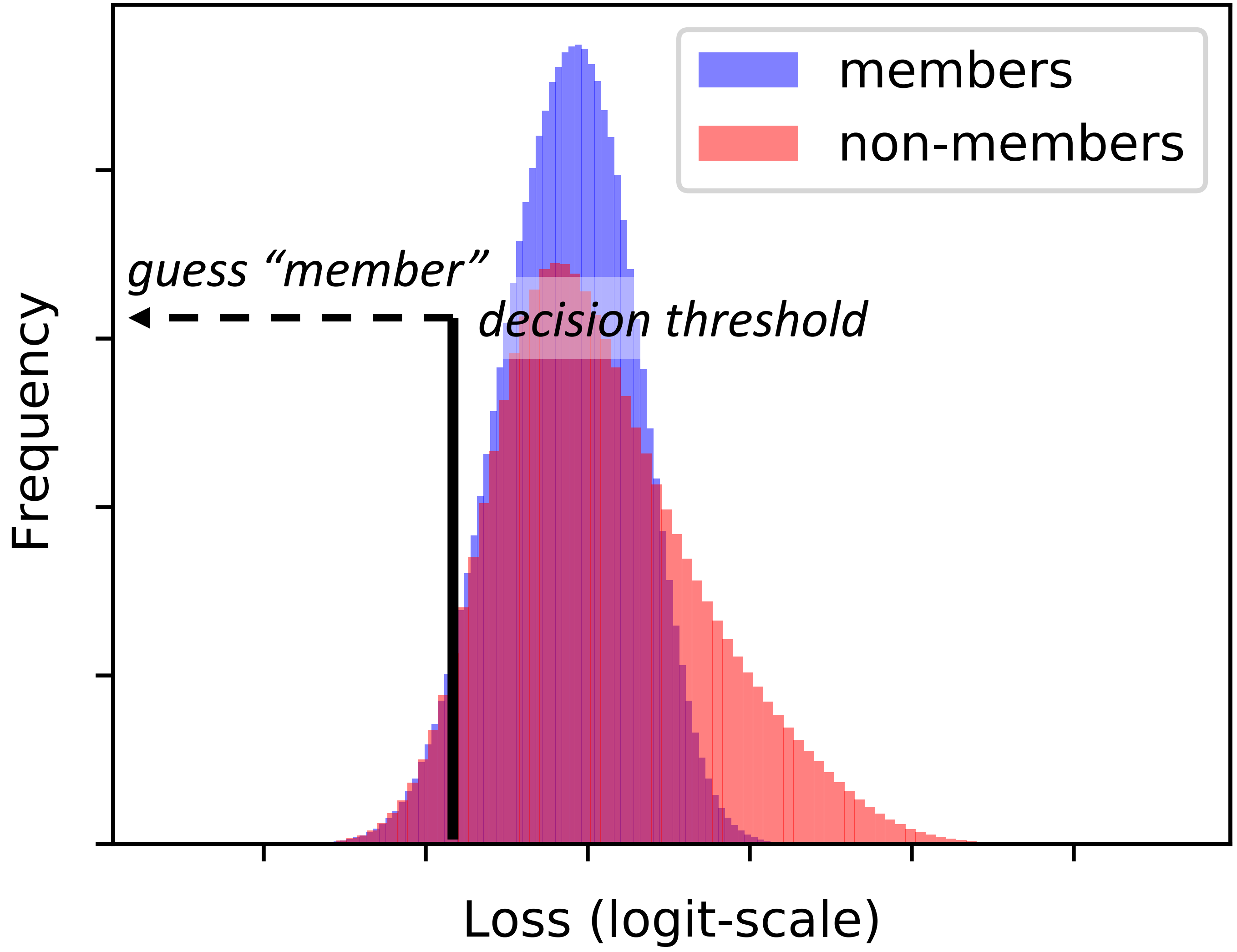
A simple MI attack: loss thresholding

[Yeom et al.'18]

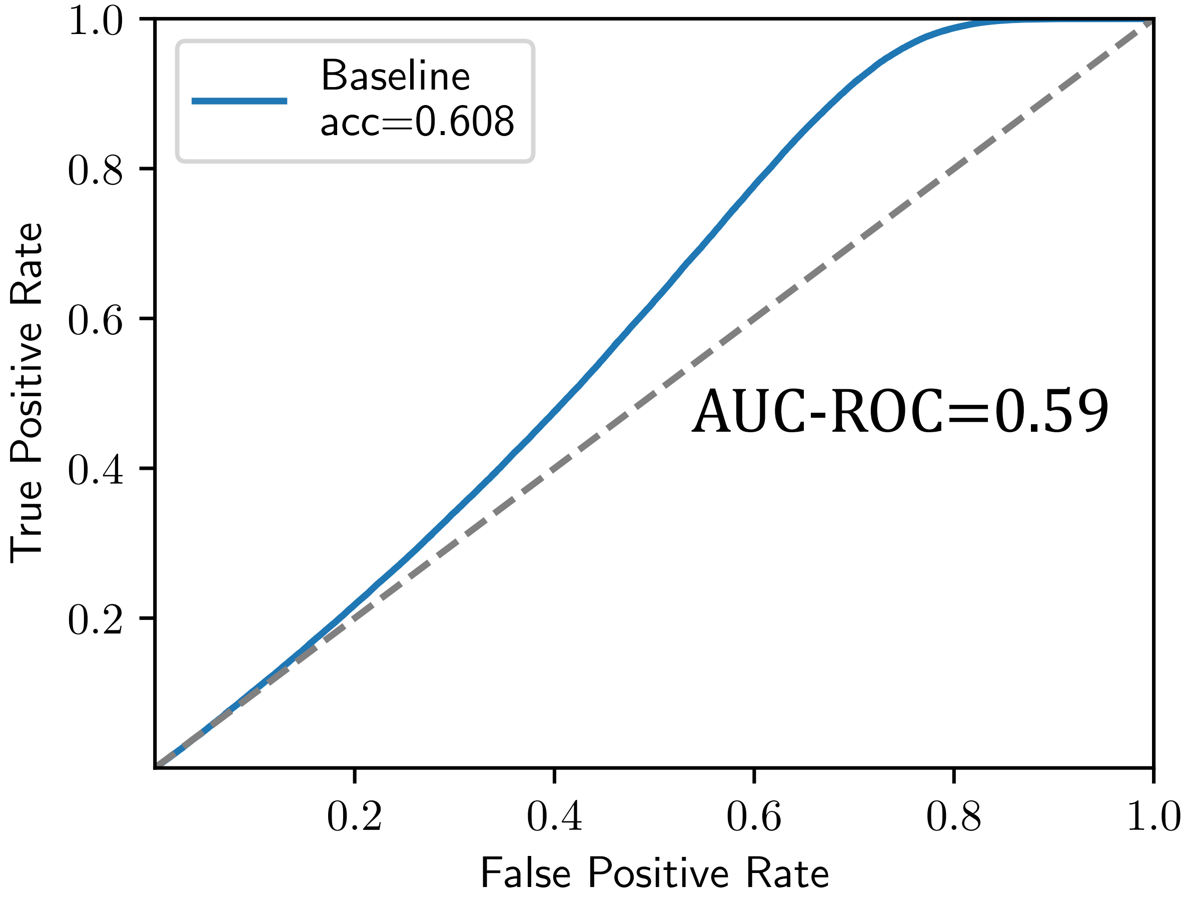


A simple MI attack: loss thresholding

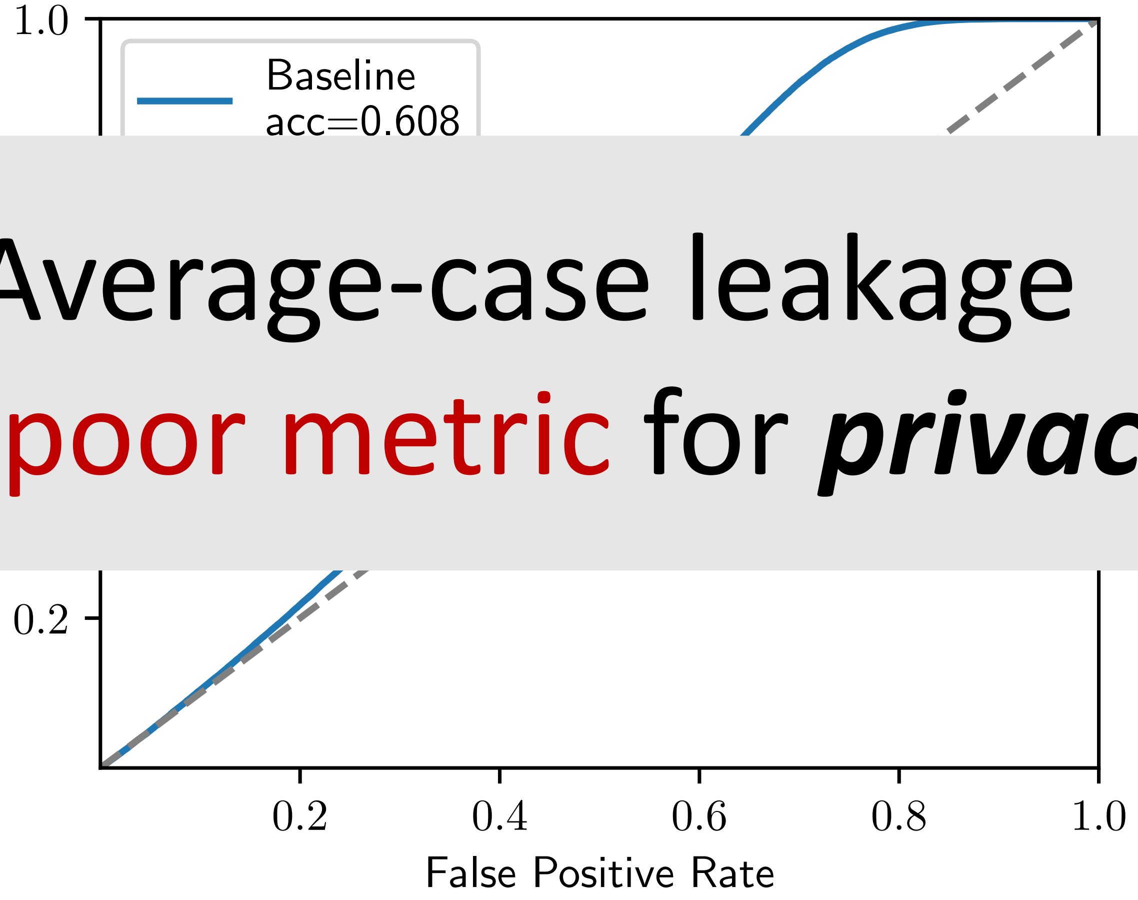
[Yeom et al.'18]



Loss thresholding leaks membership *on average.*

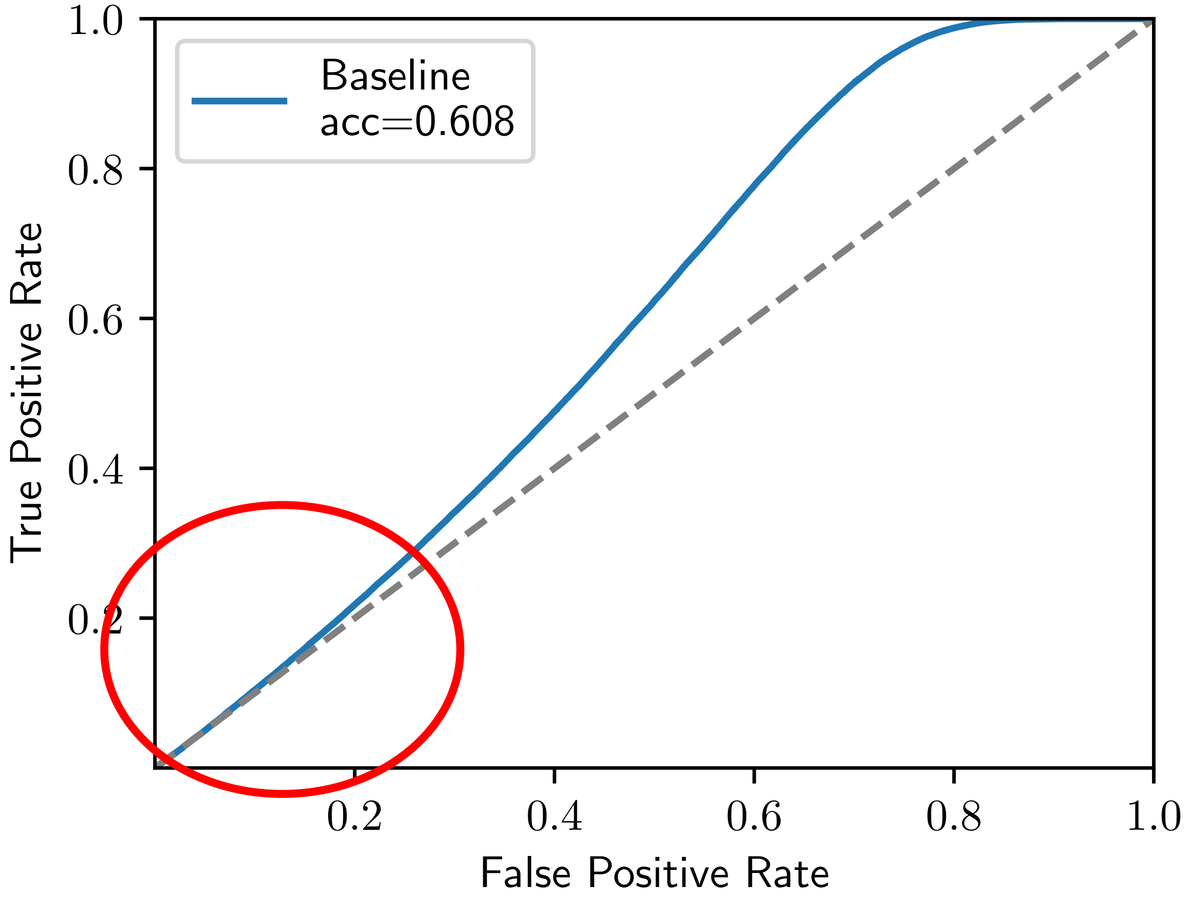


Loss thresholding leaks membership *on average.*

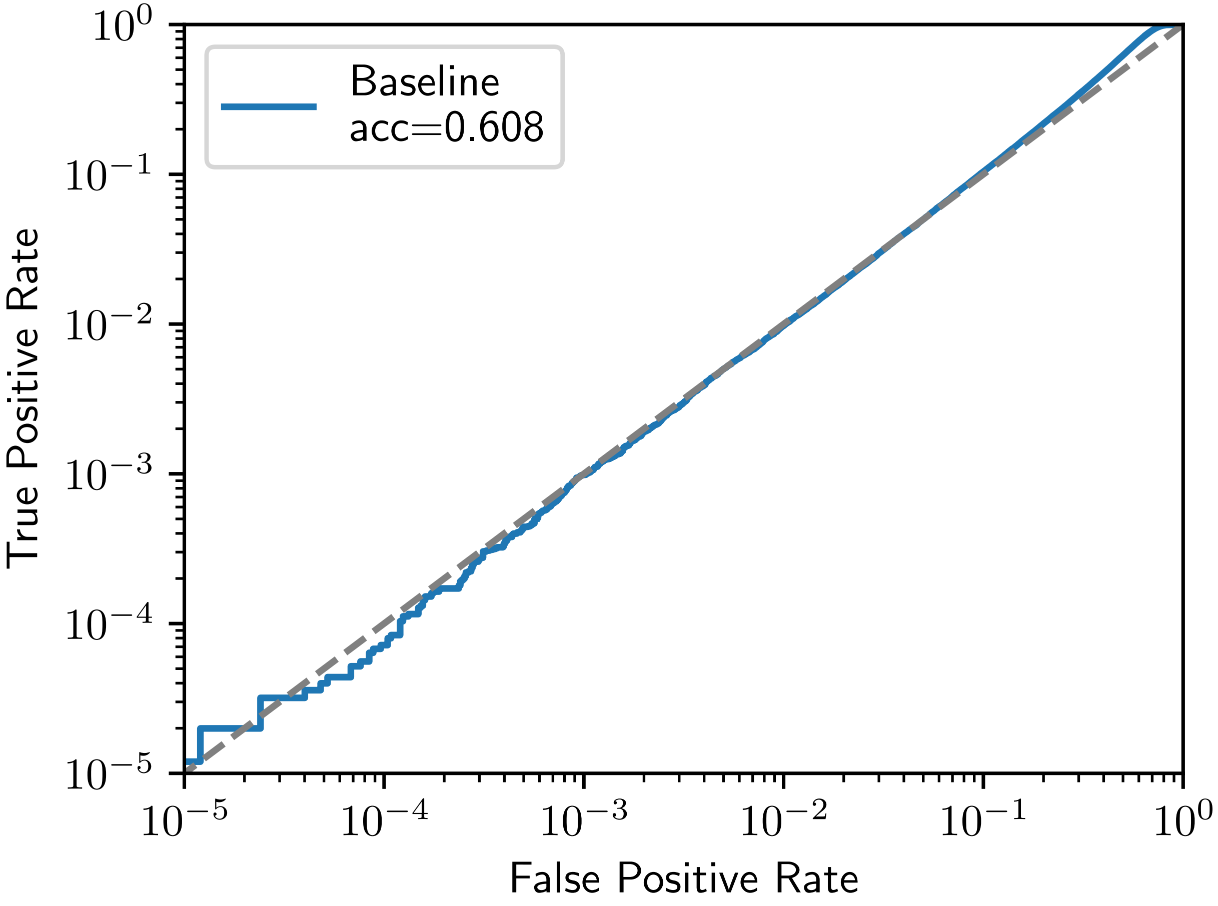


Average-case leakage
is a poor metric for ***privacy*!**

Loss thresholding doesn't *confidently* infer membership of *any* member of the training set!

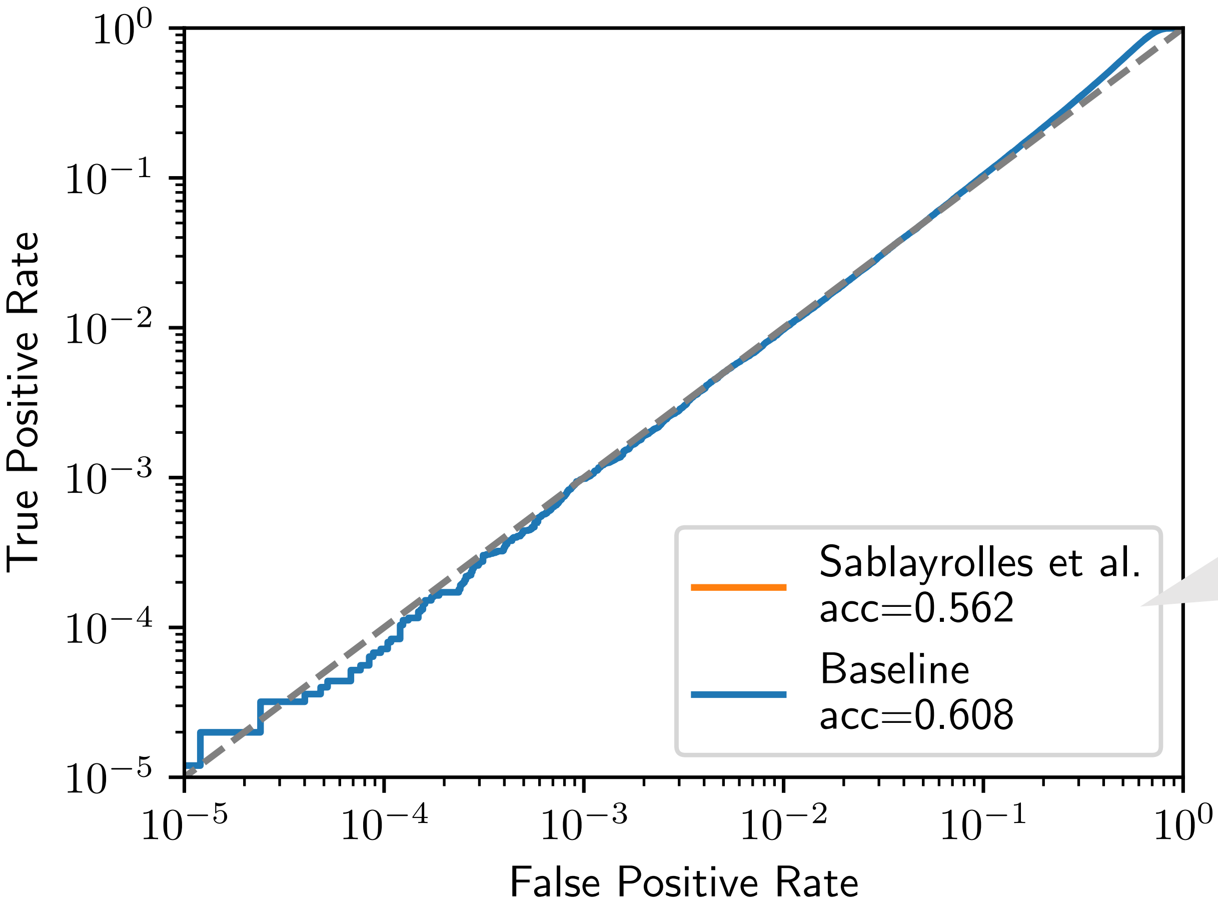


Our preferred evaluation methodology: *low* FPRs



Some attacks work! (but average-case metrics don't show it)

[Sablayrolles et al.'19]

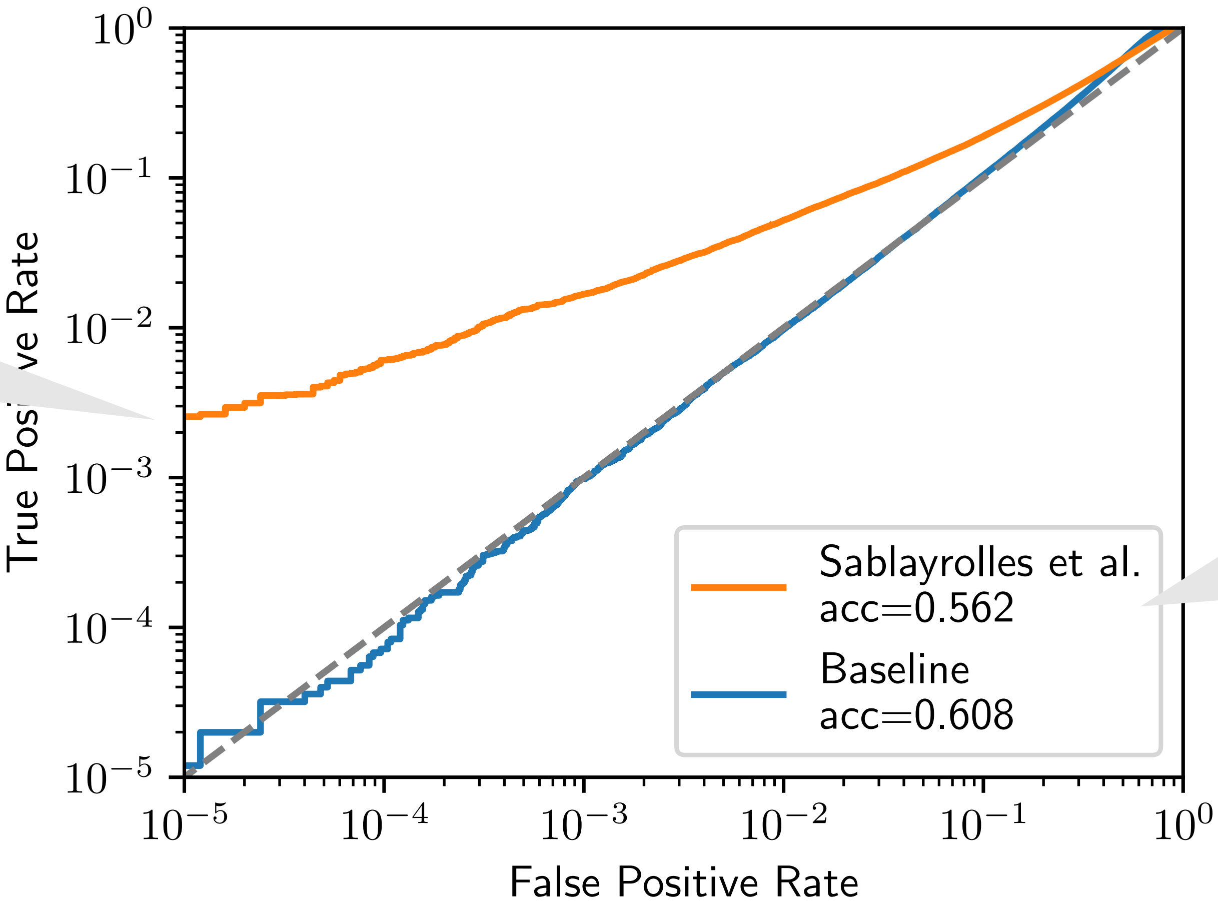


slightly **worse**
on average

Some attacks work! (but average-case metrics don’t show it)

[Sablayrolles et al.’19]

100x better in the *worst-case*



slightly **worse** *on average*

Insight: not all examples are equally "hard"

[Sablayrolles et al.'19, Long et al.'20, Feldman & Zhang'20, Watson et al.'21, Ye et al.'21]

Which is a member?

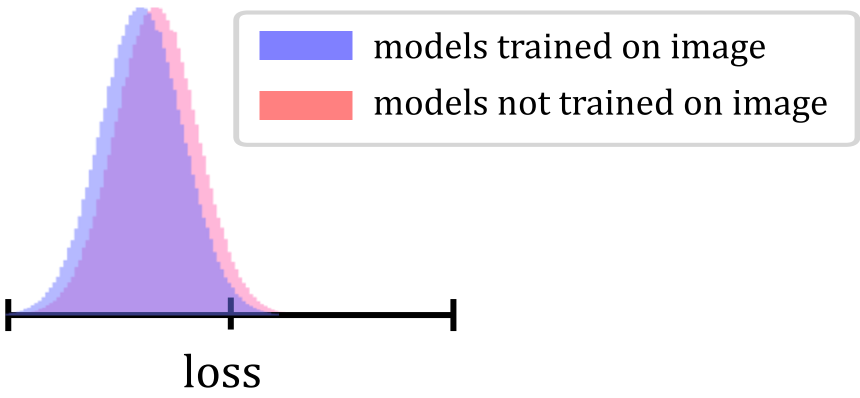
Confidence: 90% cat

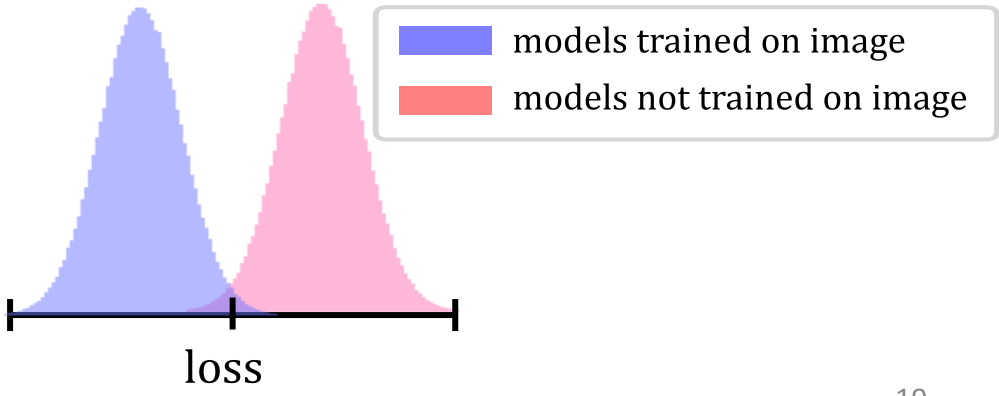
Confidence: 85% truck

Insight: not all examples are equally "hard"

[Sablayrolles et al.'19, Long et al.'20, Feldman & Zhang'20, Watson et al.'21, Ye et al.'21]

Which is a member?

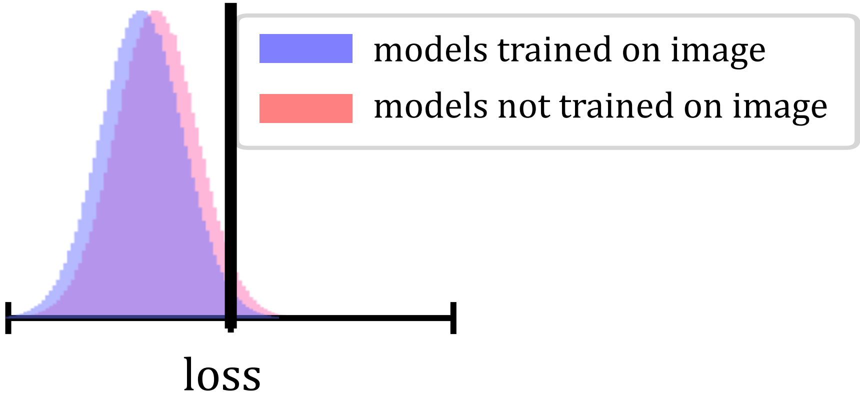


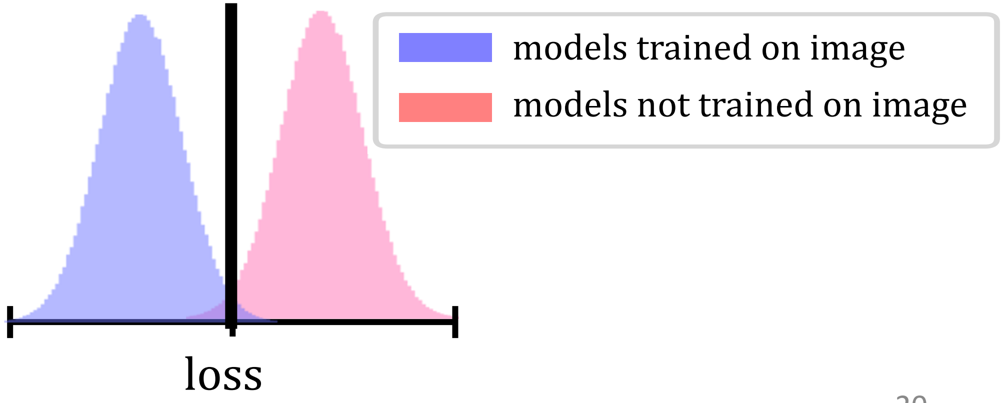


Insight: not all examples are equally "hard"

[Sablayrolles et al.'19, Long et al.'20, Feldman & Zhang'20, Watson et al.'21, Ye et al.'21]

Which is a member?





LIRA: Likelihood Ratio Attack

Carlini et al., "Membership Inference Attacks From First Principles", IEEE S&P '22

1. Query model loss $l = f(x)$

guess "member"
if $\Lambda > \tau$

5. Output *likelihood ratio*: $\Lambda = \dfrac{\Pr[l \mid x \text{ is a member}]}{\Pr[l \mid x \text{ is not a member}]}$

LIRA: Likelihood Ratio Attack

Carlini et al., "Membership Inference Attacks From First Principles", IEEE S&P '22

1. Query model loss $l = f(x)$

5. Output *likelihood ratio*: $\Lambda = \dfrac{\Pr[l \mid x \text{ is a member}]}{\Pr[l \mid x \text{ is not a member}]}$

LIRA: Likelihood Ratio Attack

Carlini et al., “Membership Inference Attacks From First Principles”, IEEE S&P ‘22

sampled from the same distribution as the training set of f

1. Query model loss $l = f(x)$

2. Train N “shadow models” $g^i_{\text{out}} \leftarrow \text{Train}(\mathcal{D})$, $g^i_{\text{in}} \leftarrow \text{Train}(\mathcal{D} \cup x)$

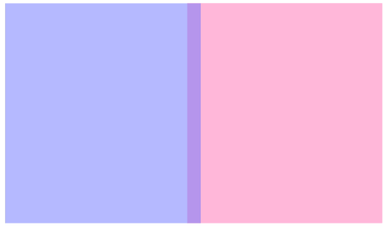
5. Output *likelihood ratio*: $\Lambda = \dfrac{\Pr[l \mid x \text{ is a member}]}{\Pr[l \mid x \text{ is not a member}]}$

LIRA: Likelihood Ratio Attack

Carlini et al., "Membership Inference Attacks From First Principles", IEEE S&P '22

1. Query model loss $l = f(x)$

2. Train N "shadow models" $g^i_{\text{out}} \leftarrow \text{Train}(\mathcal{D})$, $g^i_{\text{in}} \leftarrow \text{Train}(\mathcal{D} \cup x)$
3. Compute losses $L_{\text{out}} = \{\, g^i_{\text{out}}(\mathrm{x}) \,\}_i$, $L_{\text{in}} = \{\, g^i_{\text{in}}(\mathrm{x}) \,\}_i$



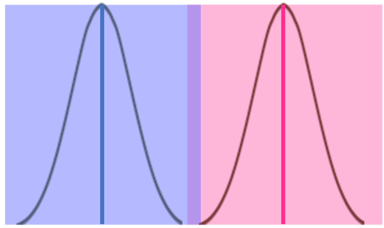
5. Output *likelihood ratio*: $\Lambda = \dfrac{\Pr[l \mid x \text{ is a member}]}{\Pr[l \mid x \text{ is not a member}]}$

LIRA: Likelihood Ratio Attack

Carlini et al., "Membership Inference Attacks From First Principles", IEEE S&P '22

1. Query model loss $l = f(x)$

2. Train N "shadow models" $g^i_{\text{out}} \leftarrow \text{Train}(\mathcal{D})$, $g^i_{\text{in}} \leftarrow \text{Train}(\mathcal{D} \cup x)$
3. Compute losses $L_{\text{out}} = \{ g^i_{\text{out}}(\text{x}) \}_i$, $L_{\text{in}} = \{ g^i_{\text{in}}(\text{x}) \}_i$
4. **Fit Gaussians to $\boldsymbol{L_{\text{out}}}$ and $\boldsymbol{L_{\text{in}}}$**

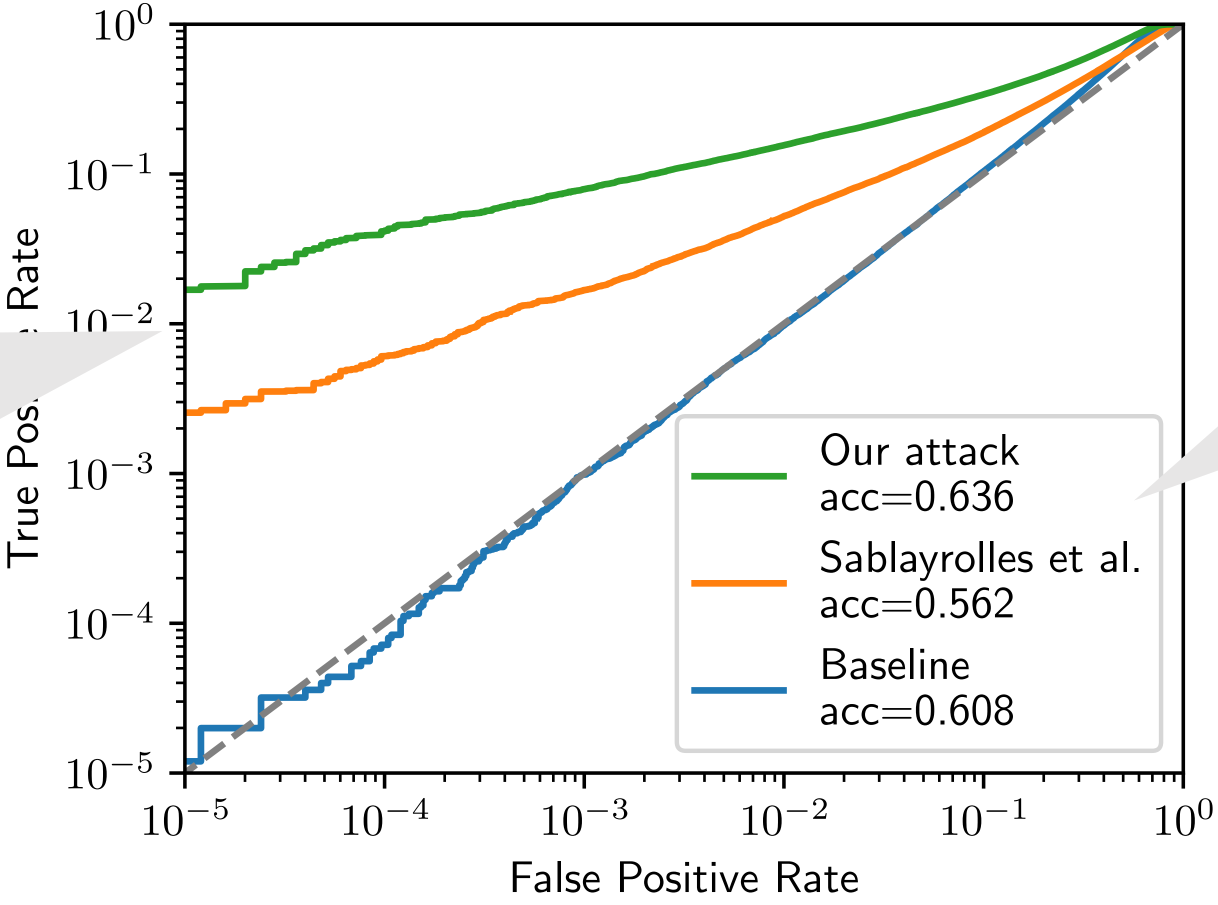


5. Output *likelihood ratio*: $\Lambda = \dfrac{\Pr[l \mid \mathcal{N}(\mu_{\text{in}}, \sigma_{\text{in}})]}{\Pr[l \mid \mathcal{N}(\mu_{\text{out}}, \sigma_{\text{out}})]}$

Results (CIFAR-10)

>10x better in the *worst case*

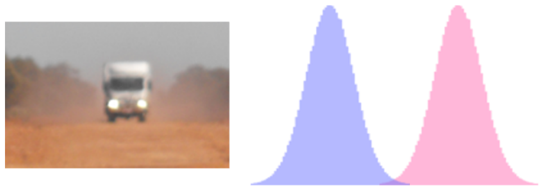
(thanks to Gaussian fitting + numeric stability + multiple queries + ...)

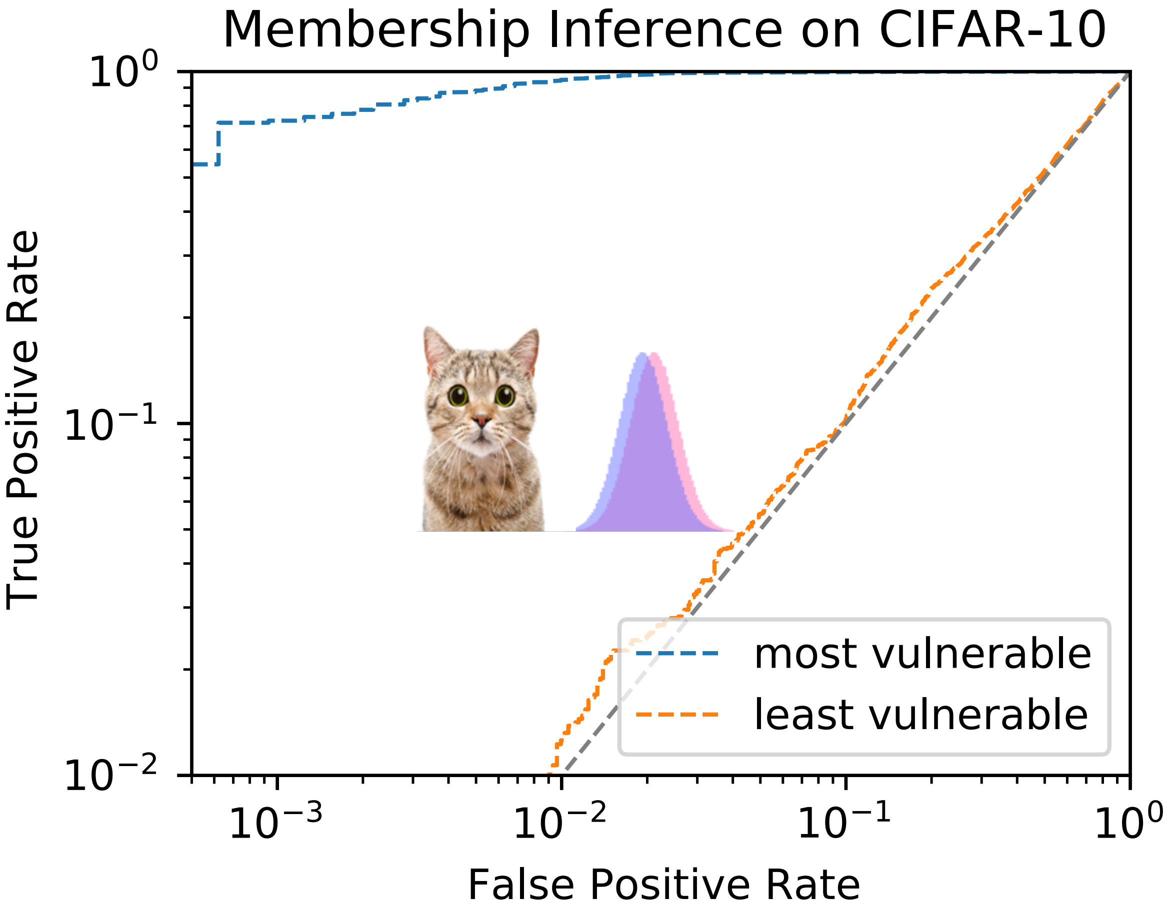


slightly better *on average*

Membership inference only works on *outliers.*

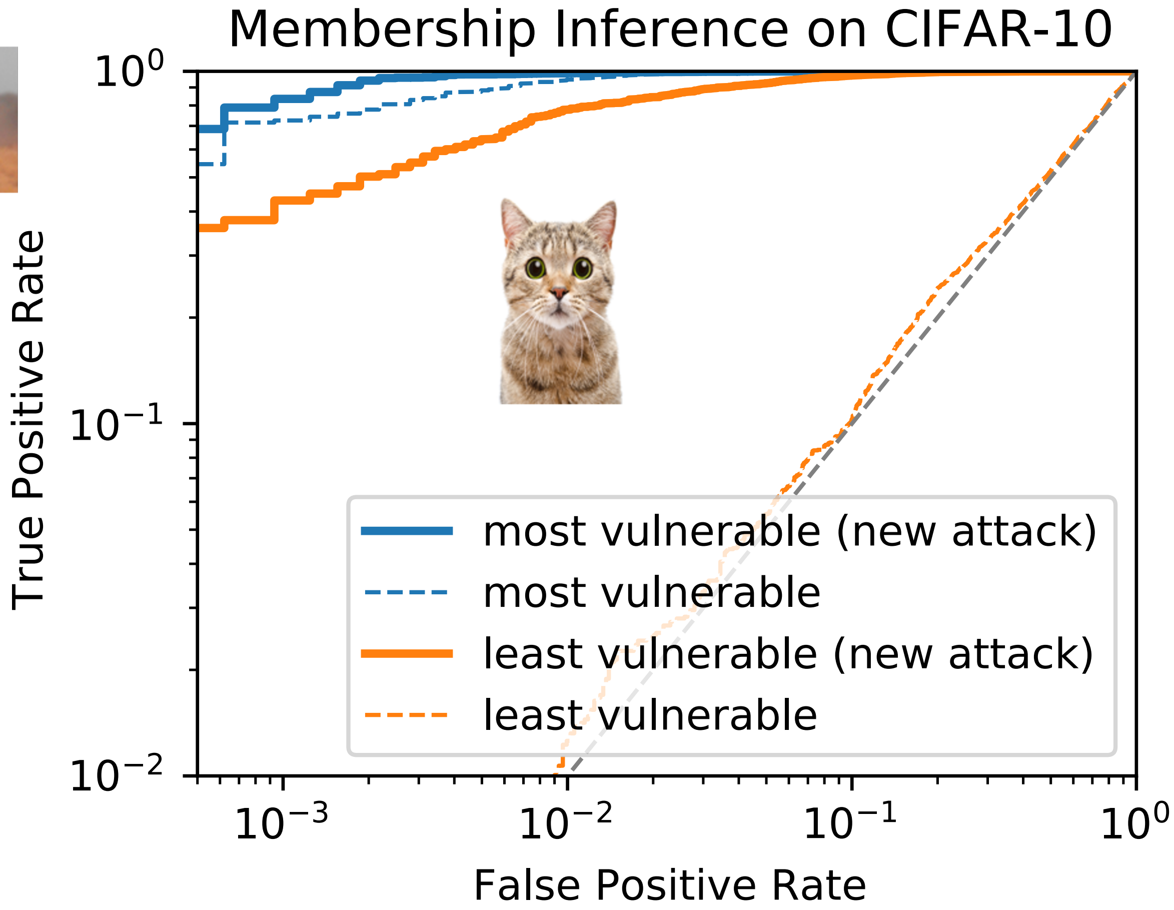
("worst-case" examples)





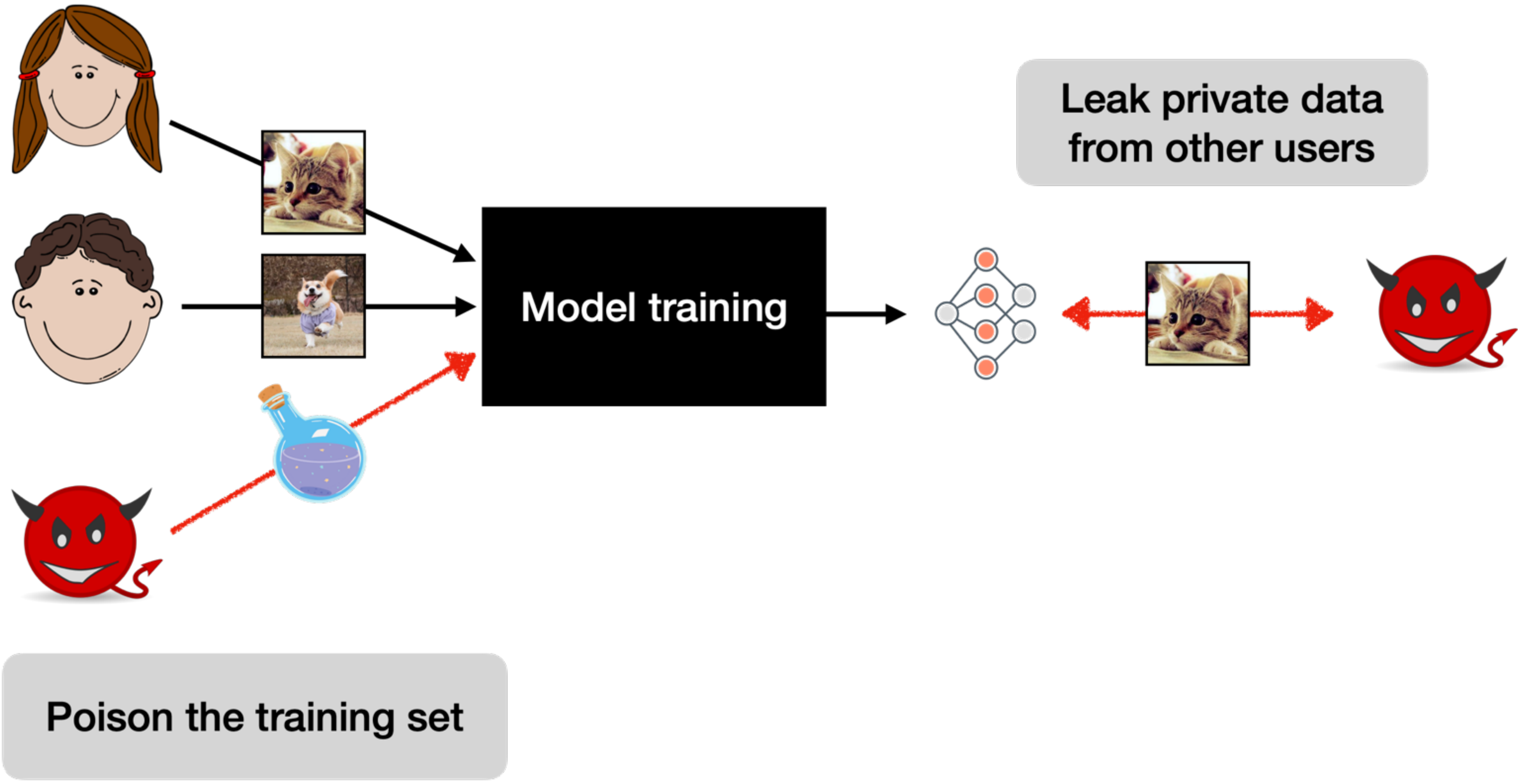
Next: a new attack that also works on *inliers!*

("average-case" examples)

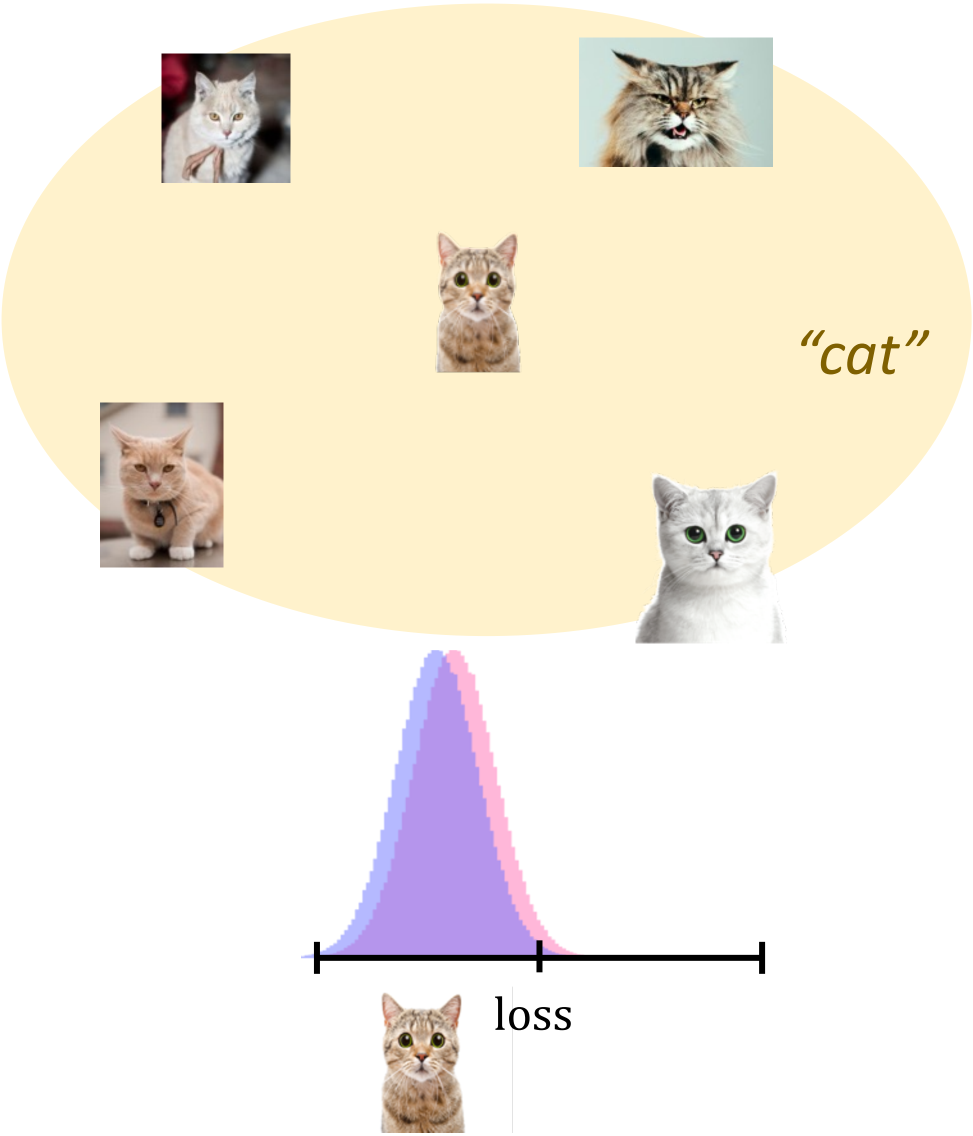


A new threat model: *privacy poisoning*

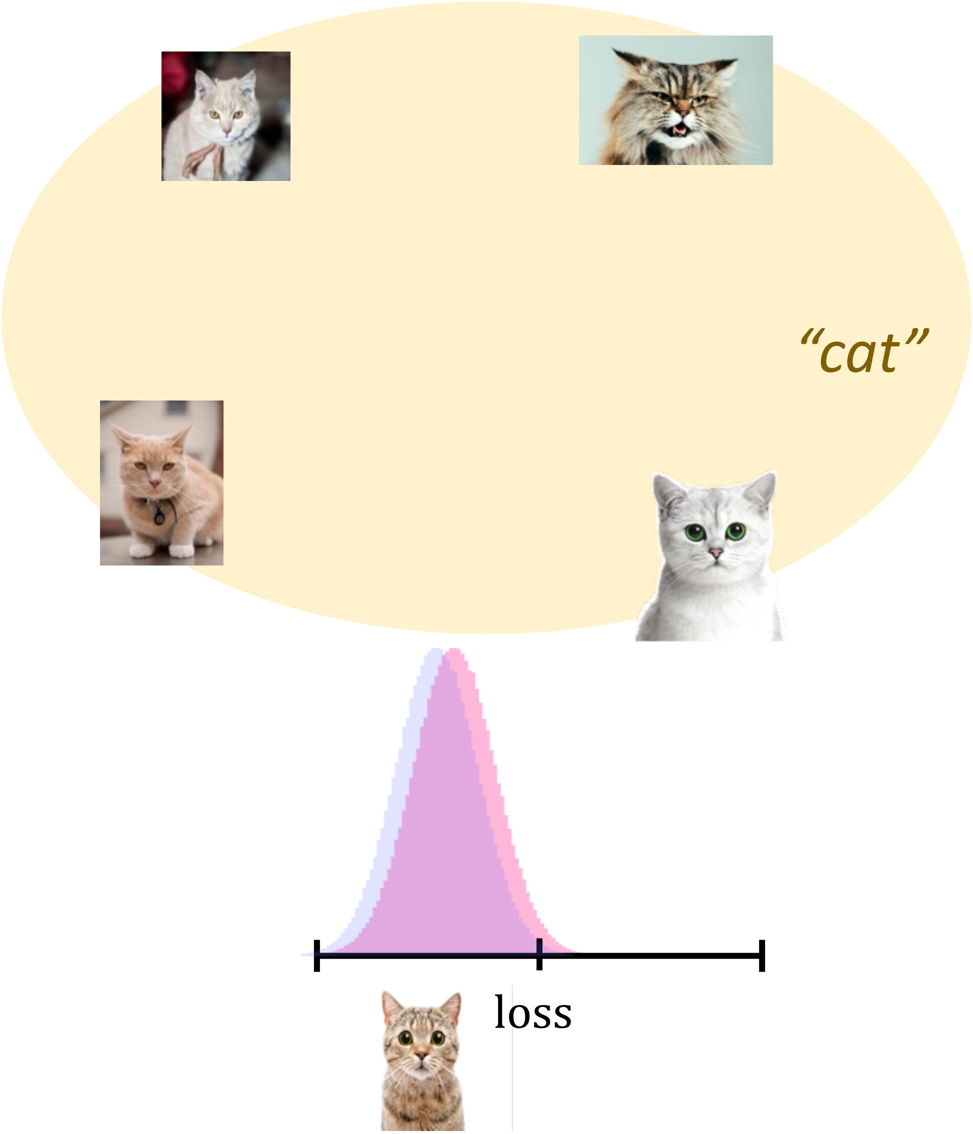
T et al. "Truth Serum: Poisoning Machine Learning Models to Reveal Their Secrets"



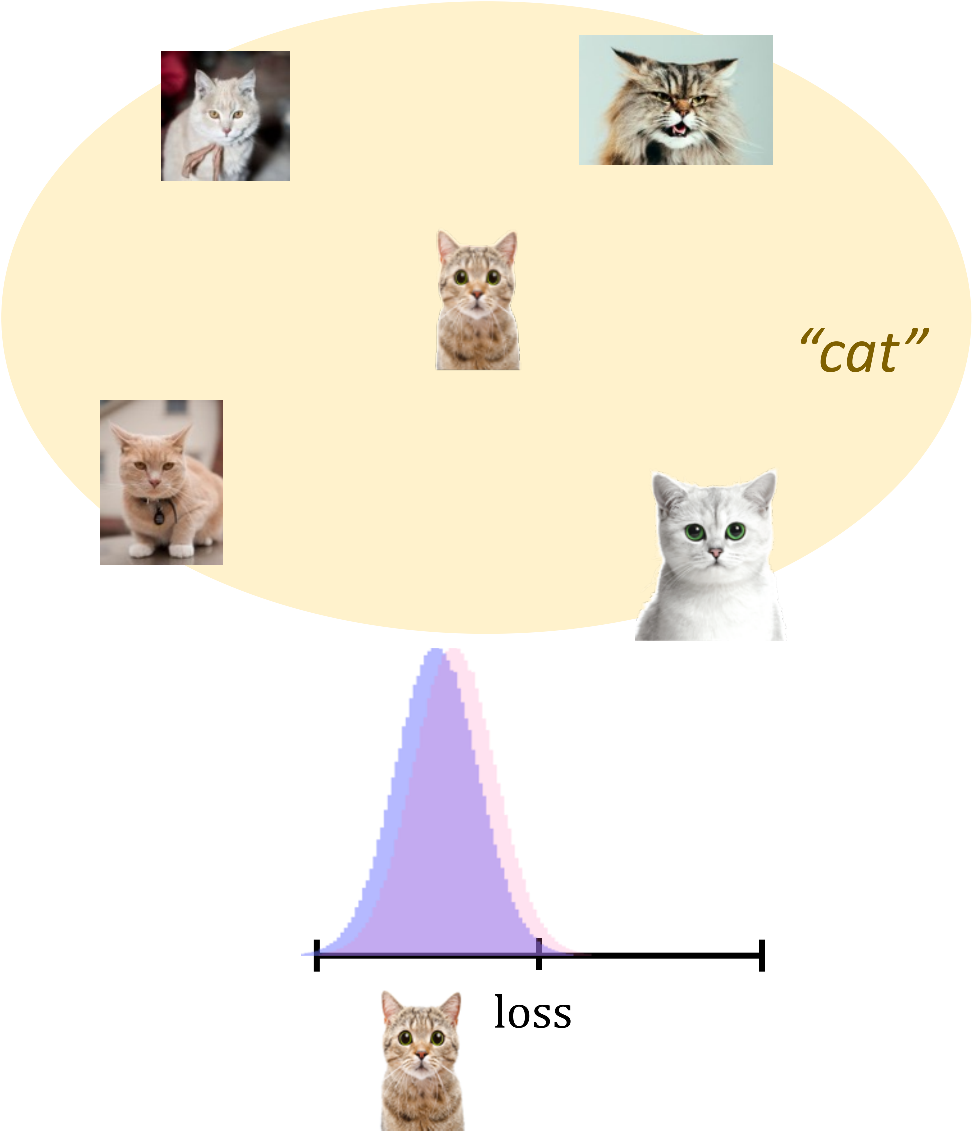
Poisoning can *transform* inliers into outliers.



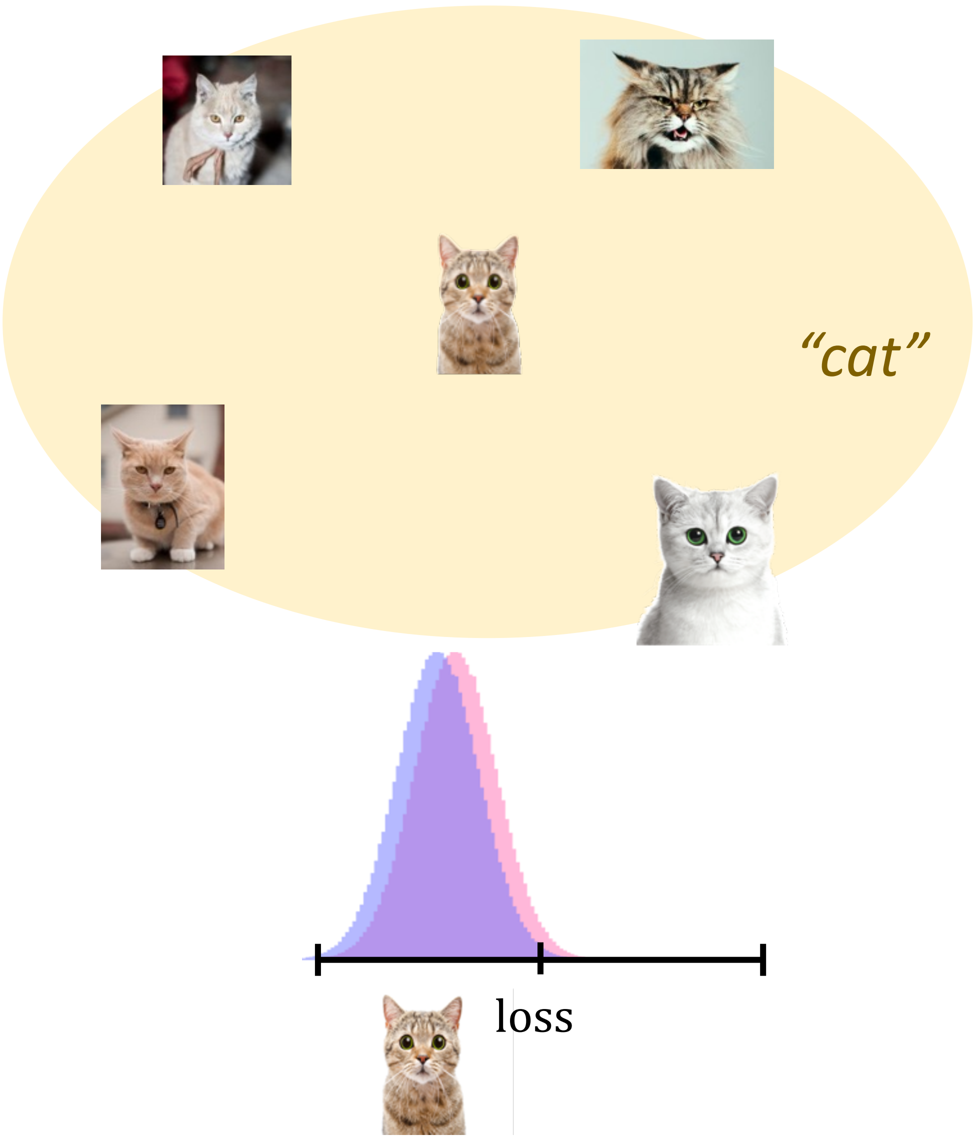
Poisoning can *transform* inliers into outliers.



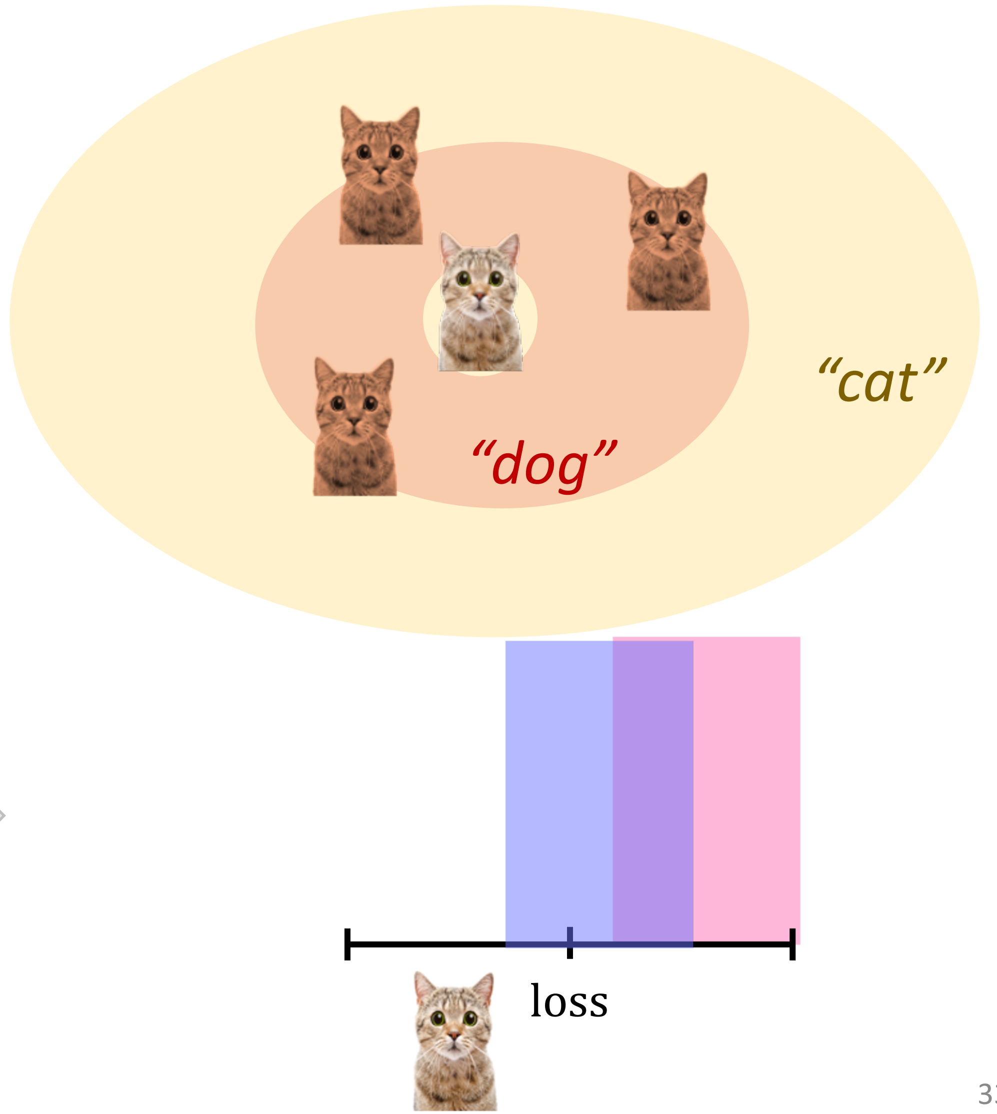
Poisoning can *transform inliers into outliers.*



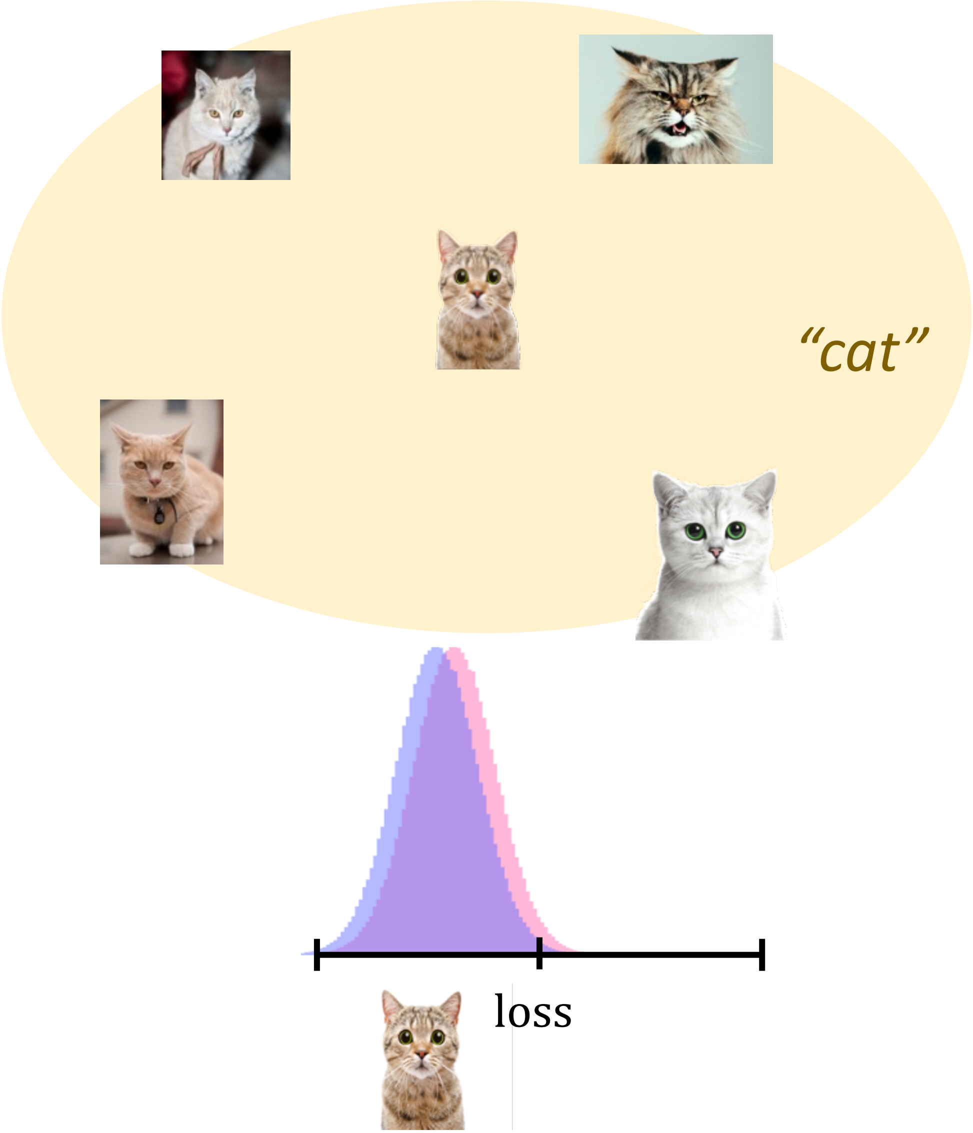
Poisoning can *transform inliers into outliers.*

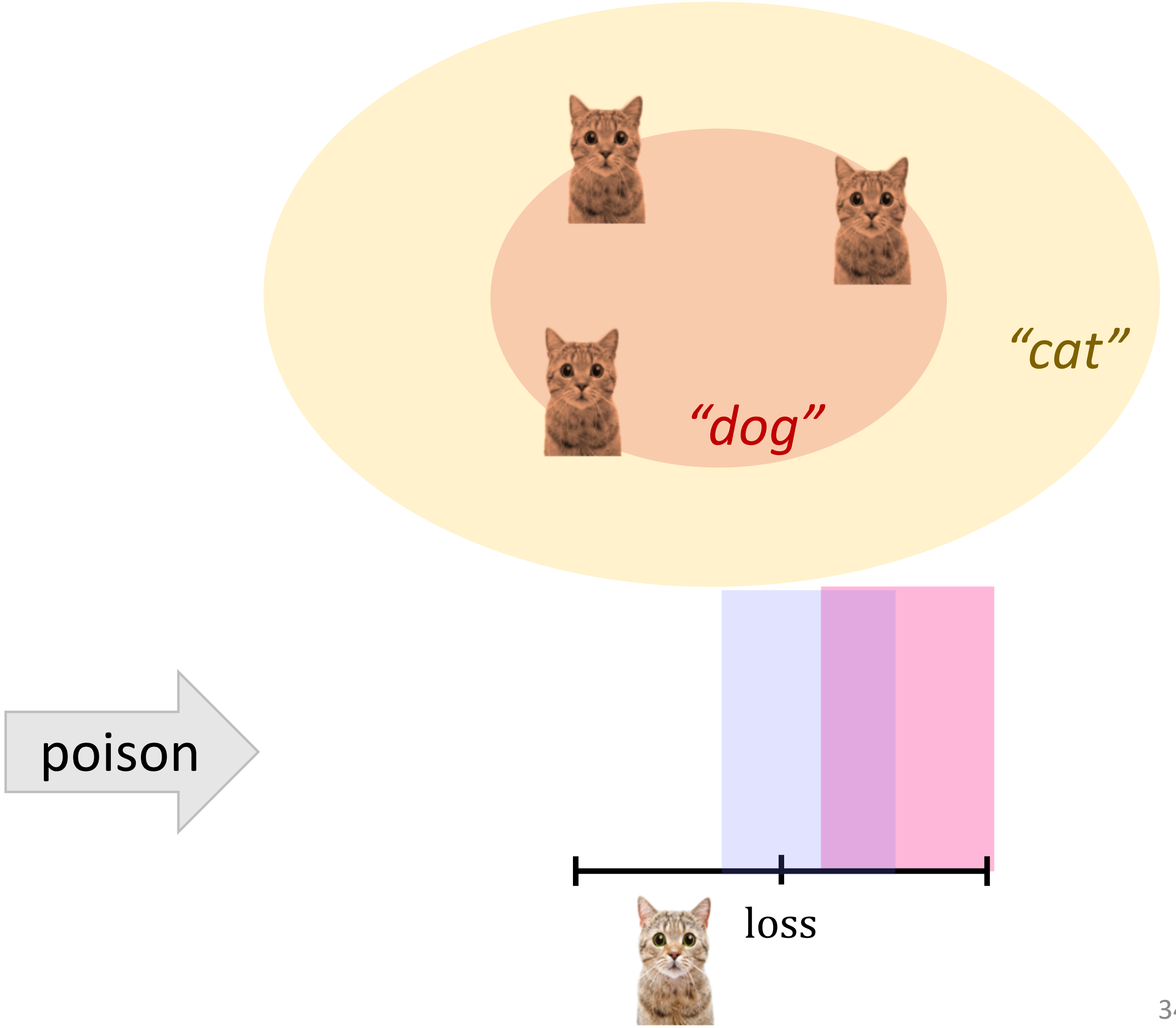


poison

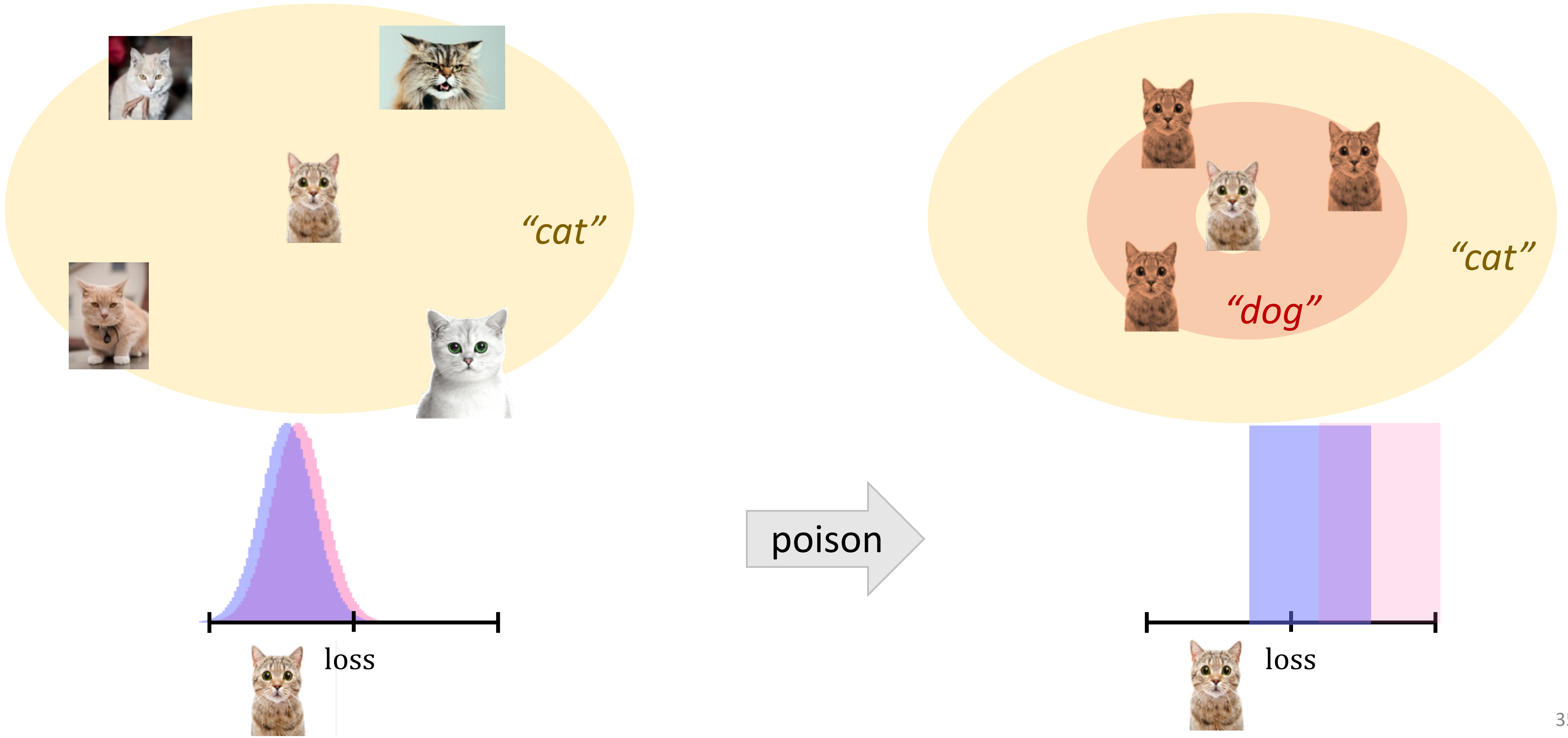


Poisoning can *transform* inliers into outliers.

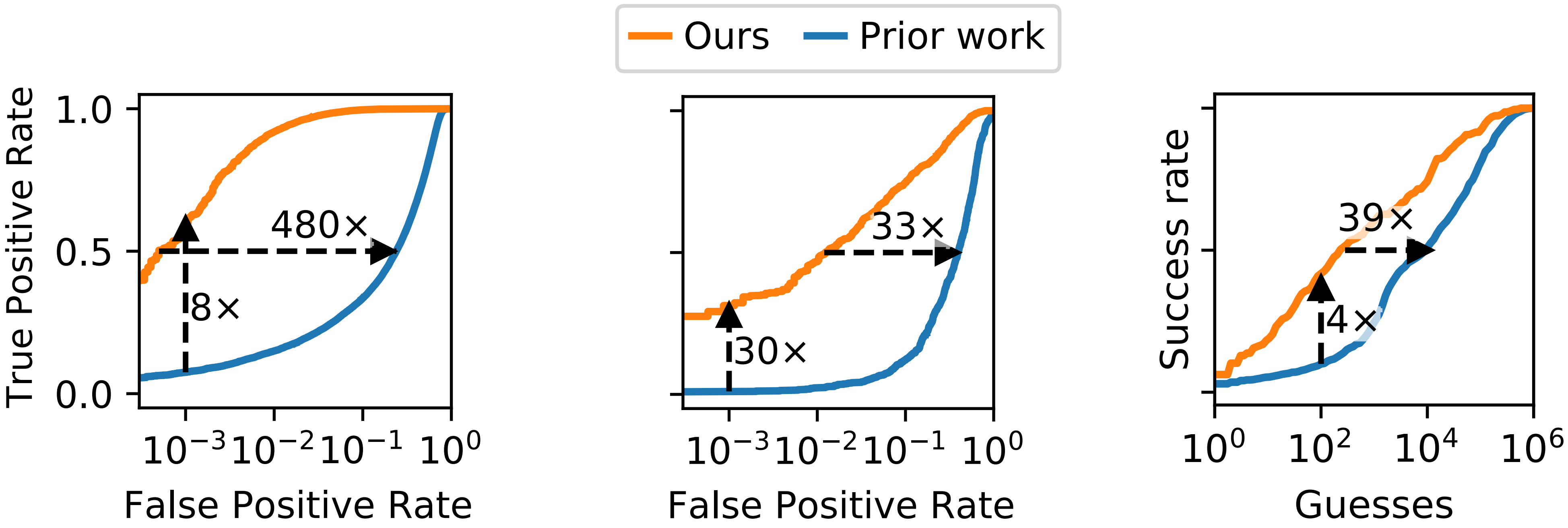




Poisoning can *transform inliers into outliers.*



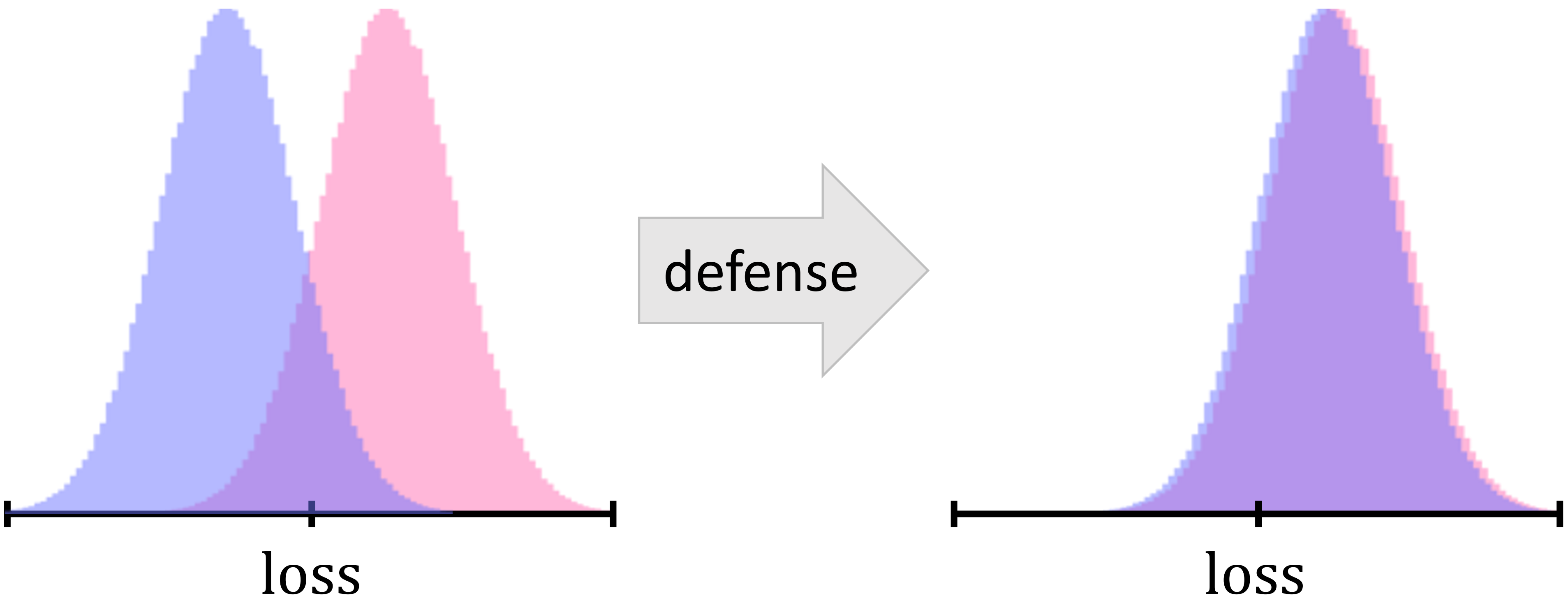
Poisoned models leak more than membership.



(a) Membership Inference (b) Attribute Inference (c) Data Extraction

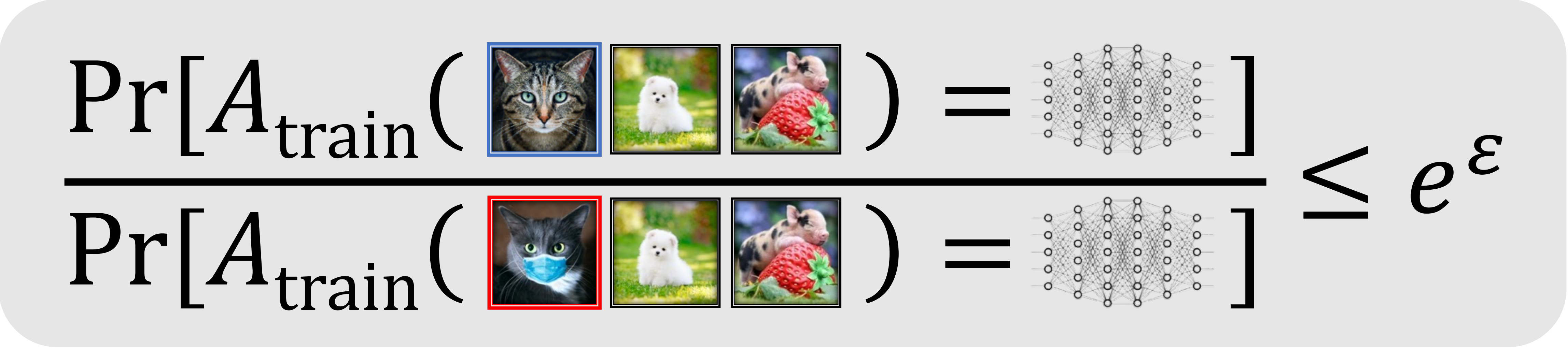
with targeted poisoning of **<0.1%** of the training set

How to defend against membership leakage?



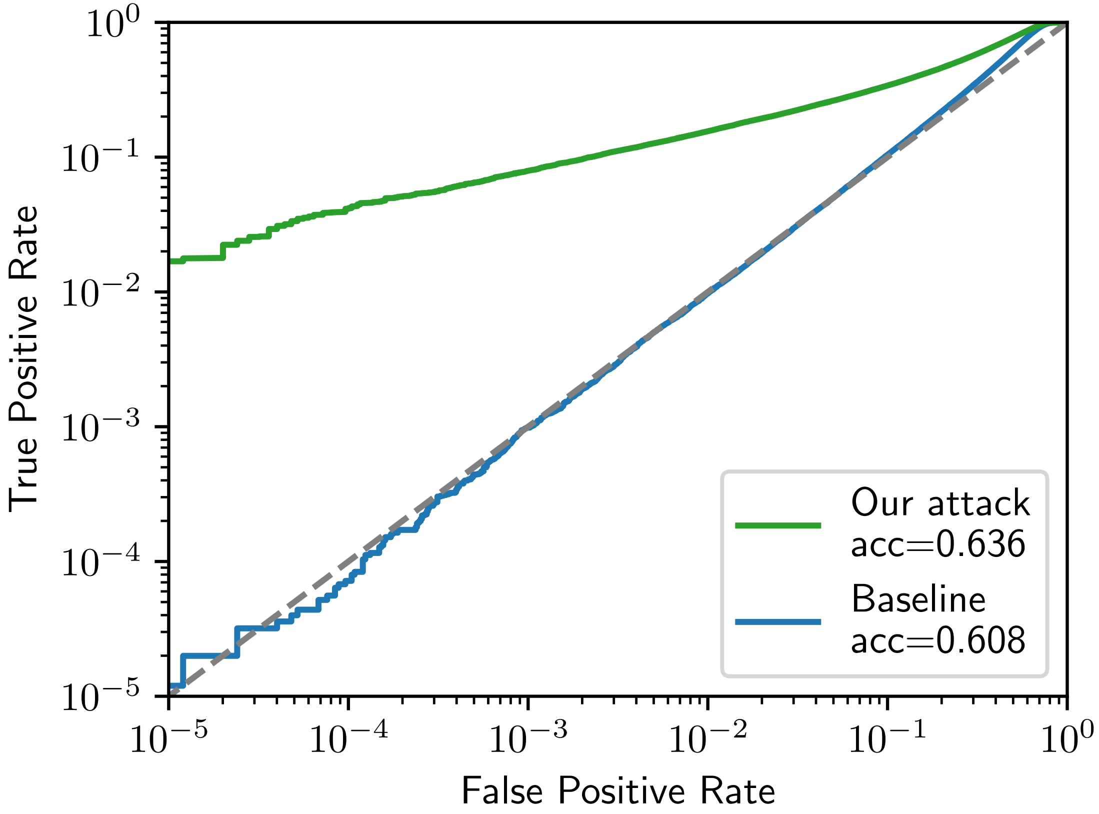
Differential privacy prevents all our attacks.

*DP guarantee holds for **any** pair of datasets that differ in **any** single element*



DP bounds the success of ***any*** MI attack.

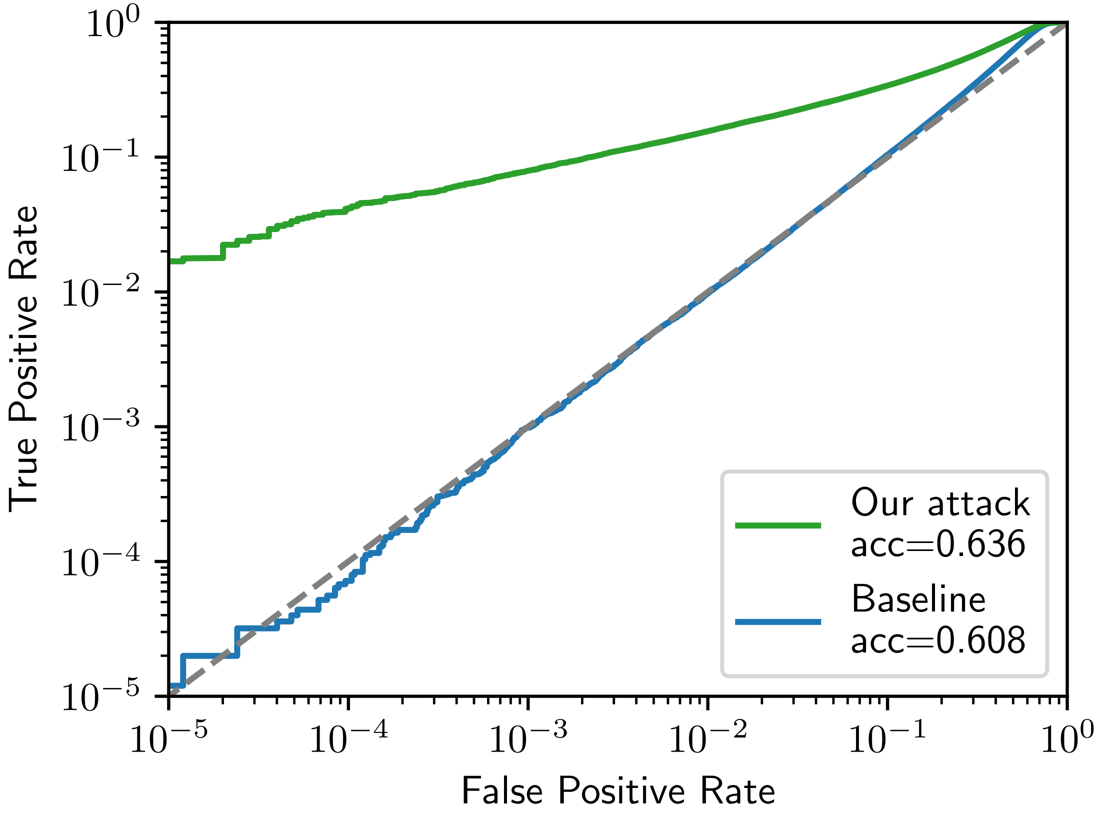
[Kairouz et al. '15]



$$\frac{TPR}{FPR} \leq e^{\varepsilon}$$

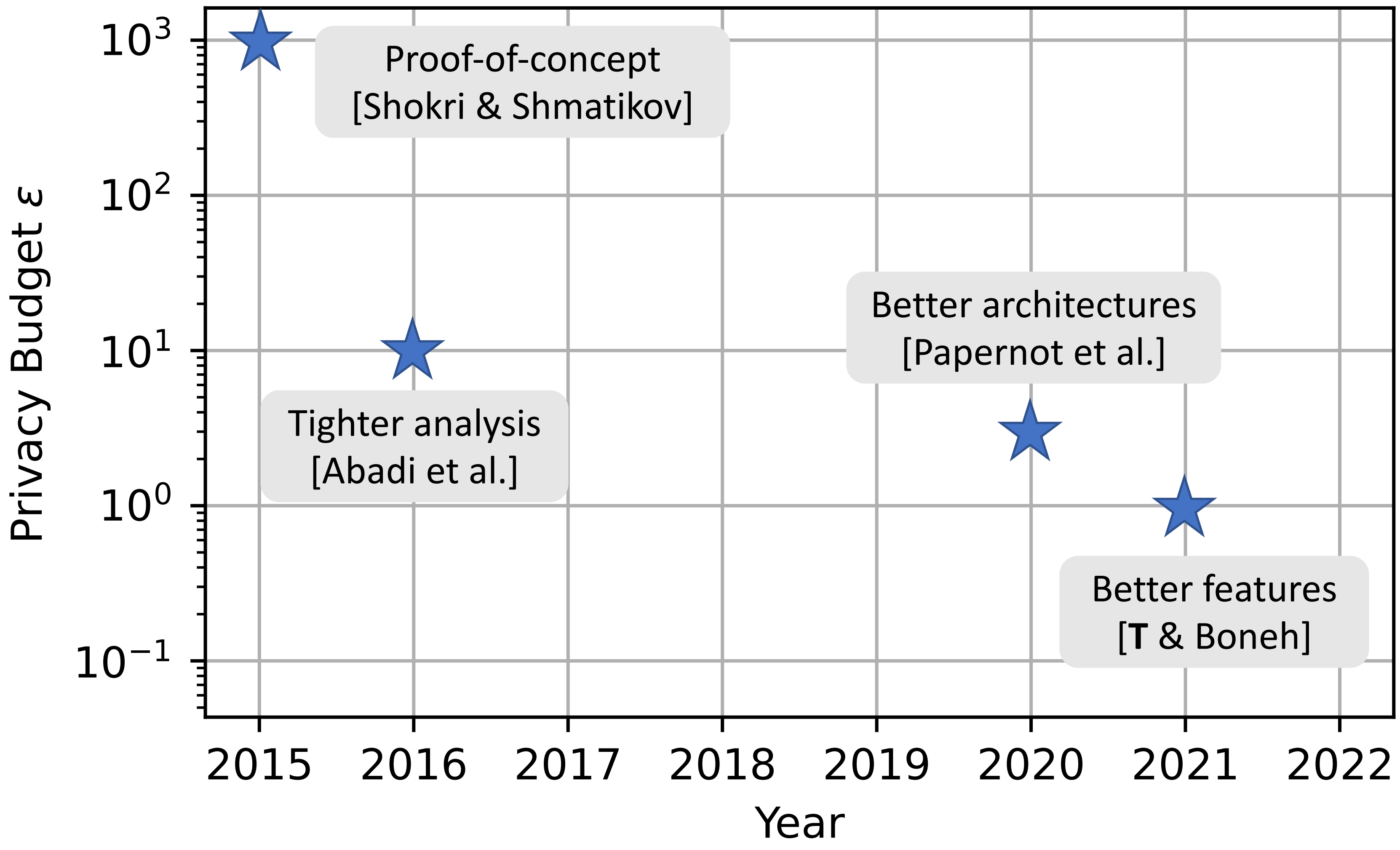
Corollary: MI attacks can be used to *audit* privacy.

[Jagielsky et al. ‘20, Nasr et al. ’21]

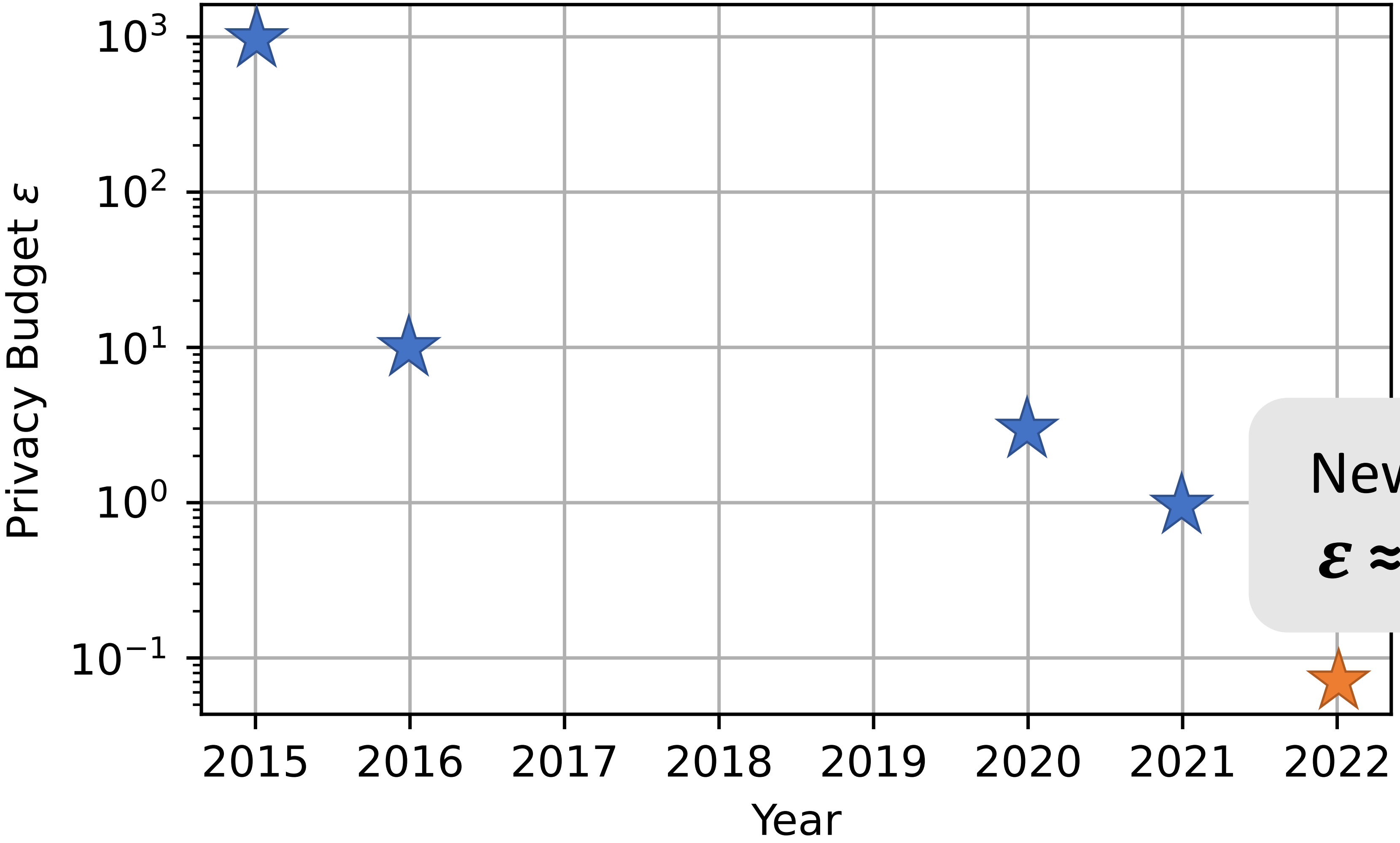


$$e^{\varepsilon} \geq \frac{TPR}{FPR}$$

Example: DP with 98% accuracy on MNIST

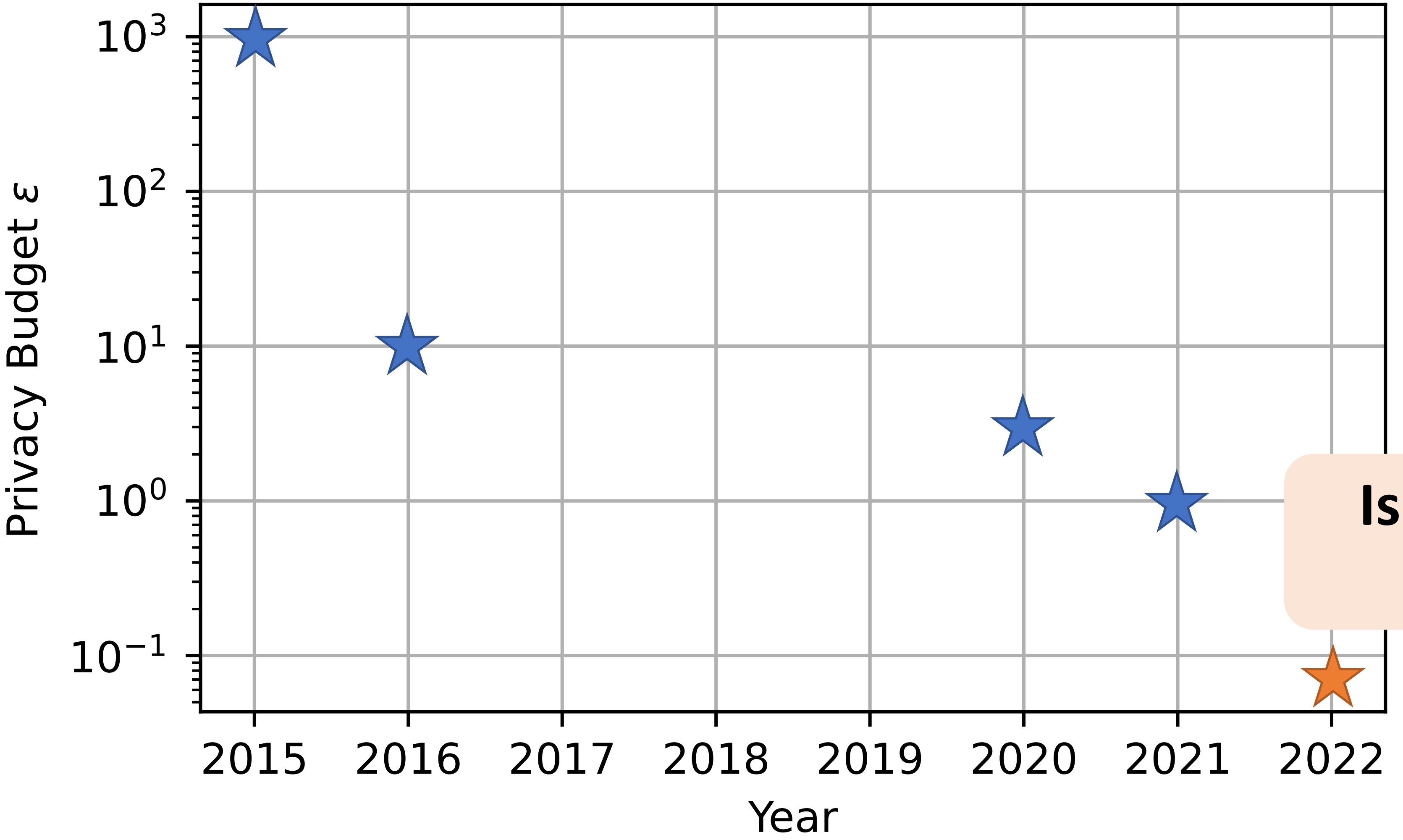


Example: DP with 98% accuracy on MNIST



New algorithm
$\boldsymbol{\varepsilon} \approx$ 0.08 (!!!)

Example: DP with 98% accuracy on MNIST



Is this claim *correct*?

How to verify a privacy claim?

➢ Check the **proof**

$$
\begin{aligned}
&c(o_{1:k}; \mathcal{M}_{1:k}, o_{1:(k-1)}, d, d') \\
&\quad = \log \frac{\Pr[\mathcal{M}_{1:k}(d; o_{1:(k-1)}) = o_{1:k}]}{\Pr[\mathcal{M}_{1:k}(d'; o_{1:(k-1)}) = o_{1:k}]} \\
&\quad = \log \prod_{i=1}^{k} \frac{\Pr[\mathcal{M}_i(d) = o_i \mid \mathcal{M}_{1:(i-1)}(d) = o_{1:(i-1)}]}{\Pr[\mathcal{M}_i(d') = o_i \mid \mathcal{M}_{1:(i-1)}(d') = o_{1:(i-1)}]} \\
&\quad = \sum_{i=1}^{k} \log \frac{\Pr[\mathcal{M}_i(d) = o_i \mid \mathcal{M}_{1:(i-1)}(d) = o_{1:(i-1)}]}{\Pr[\mathcal{M}_i(d') = o_i \mid \mathcal{M}_{1:(i-1)}(d') = o_{1:(i-1)}]} \\
&\quad = \sum_{i=1}^{k} c(o_i; \mathcal{M}_i, o_{1:(i-1)}, d, d').
\end{aligned}
$$

Thus

$$
\begin{aligned}
&\mathbb{E}_{o'_{1:k} \sim \mathcal{M}_{1:k}(d)} \left[\exp(\lambda c(o'_{1:k}; \mathcal{M}_{1:k}, d, d')) \;\middle|\; \forall i < k \colon o'_i = o_i \right] \\
&\quad = \mathbb{E}_{o'_{1:k} \sim \mathcal{M}_{1:k}(d)} \left[\exp\left(\lambda \sum_{i=1}^{k} c(o'_i; \mathcal{M}_i, o_{1:(i-1)}, d, d') \right) \right] \\
&\quad = \mathbb{E}_{o'_{1:k} \sim \mathcal{M}_{1:k}(d)} \left[\prod_{i=1}^{k} \exp\left(\lambda c(o'_i; \mathcal{M}_i, o_{1:(i-1)}, d, d') \right) \right] \\
&\qquad\qquad \text{(by independence of noise)}
\end{aligned}
$$

How to verify a privacy claim?

- Check the proof
- Check the **code**

```
def process_microbatch(i, sample_state):
  """Process one microbatch (record) with privacy helper."""
  microbatch_loss = tf.reduce_mean(
      input_tensor=tf.gather(microbatches_losses, [i]))
  with gradient_tape.stop_recording():
    grads = gradient_tape.gradient(microbatch_loss, var_list)
  sample_state = self._dp_sum_query.accumulate_record(
      sample_params, sample_state, grads)
  return sample_state

for idx in range(self._num_microbatches):
  sample_state = process_microbatch(idx, sample_state)

grad_sums, self._global_state, _ = (
    self._dp_sum_query.get_noised_result(sample_state,
                                         self._global_state))
```

How to verify a privacy claim?

- Check the proof

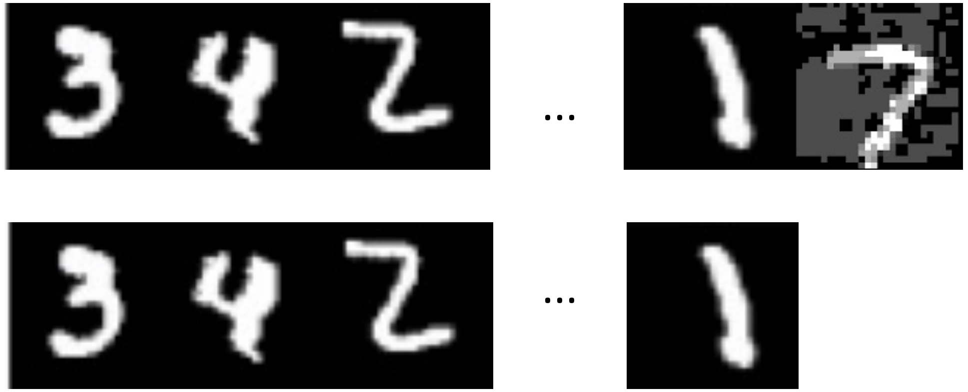
- Check the code

- Launch a **MI attack**!

DP bounds should hold for **any** data point.

$D_0 =$

$D_1 =$

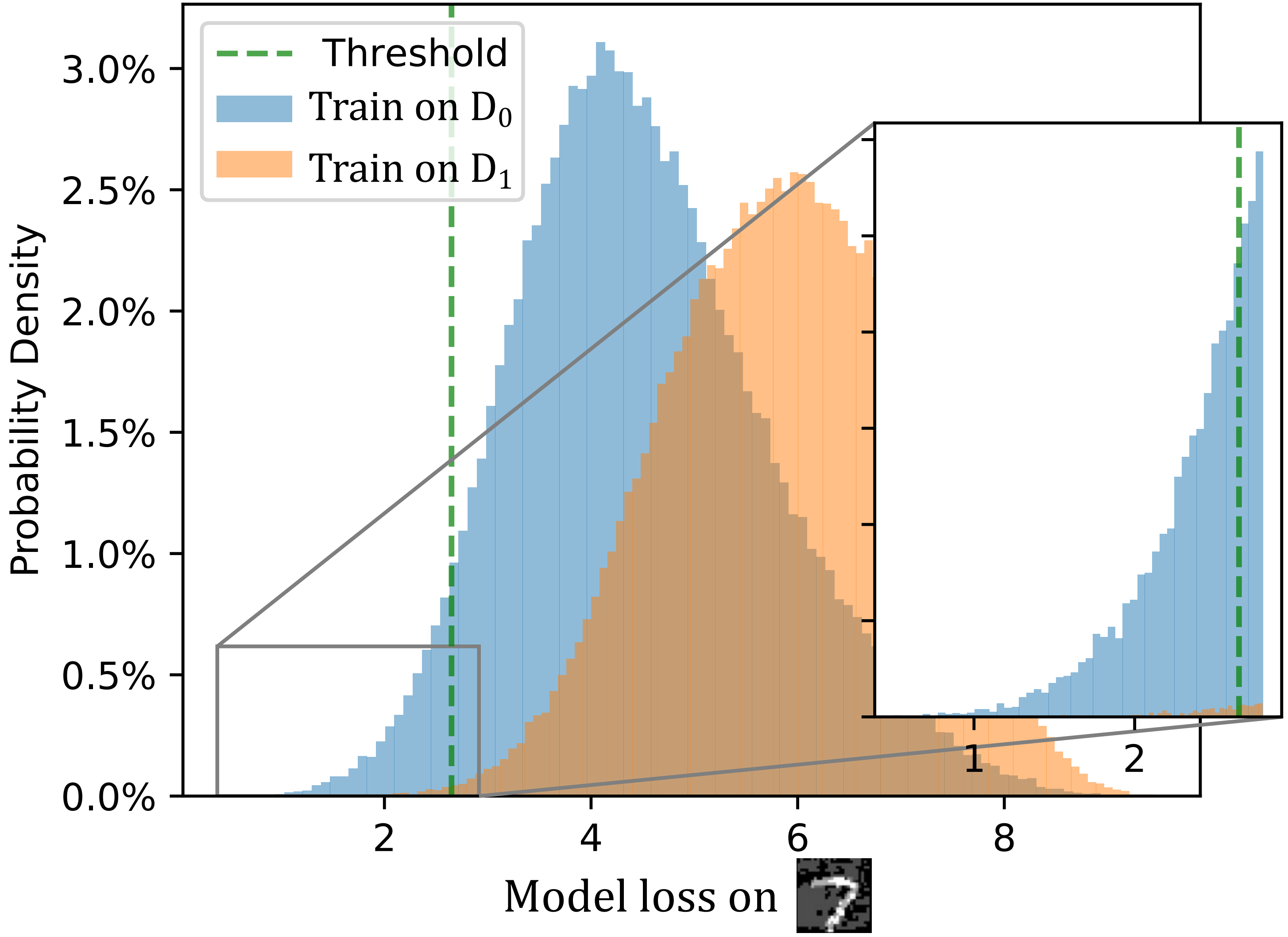


worst-case out-of-distribution data point

Attack goal: guess if DP model was trained on D_0 or D_1

Run the attack 100’000 times…

T et al. “Debugging Differential Privacy: A Case Study for Privacy Auditing”



$\boldsymbol{\varepsilon > 2.7}$

(claim was $\varepsilon = 0.08$)

Conclusion

- Average-case leakage is a poor metric for privacy!
- We must reevaluate what we "know" about MI attacks & defenses
- Poisoning can turn average-case inputs into worst-case inputs
- Worst-case MI attacks are a useful tool for catching DP bugs

https://floriantramer.com florian.tramer@inf.ethz.ch