Appendix

A. Related Work

***Mixup*.** See Section 2. *Mixup* can improve both generalization and adversarial robustness (Verma et al., 2019). It has also been adapted to various learning tasks, including semi-supervised data augmentation (Berthelot et al., 2019), unsupervised image synthesis (Beckham et al., 2019), and adversarial defense at the inference stage (Pang et al., 2019). Recently, (Fu et al., 2019) combined *Mixup* with model aggregation to defend against inversion attack, and (Liu et al., 2019) proposed a novel method using *Mixup* for on-cloud privacy-preserving inference.

Differential privacy. Differential privacy for deep learning involves controlling privacy leakage by adding noise to the learning pipeline. If the noise is drawn from certain distributions, say Gaussian or Laplace, it is possible to provide guarantees of privacy (Dwork et al., 2006; Dwork & Roth, 2014). Applying differential privacy techniques to distributed deep learning is non-trivial. Shokri and Shmatikov (Shokri & Shmatikov, 2015) proposed a distributed learning scheme by directly adding noise to the shared gradients. However, the amount of privacy guaranteed drops with the number of training epochs and the size of shared parameters. DPSGD (Abadi et al., 2016) was proposed to dynamically keep track of privacy spending based on the composition theorem (Dwork, 2009). However, it still leads to an accuracy drop of about 20% on CIFAR-10 dataset. Also, to control privacy leakage, DPSGD has to start with a model pre-trained using nonprivate labeled data, and then carefully fine-tunes a few layers using private data.

Privacy using cryptographic protocols. In distributed learning setting with multiple data participants, it is possible for the participants to jointly train a model over their private inputs by employing techniques like homomorphic encryption (Gentry, 2009; Graepel et al., 2012; Li et al., 2017) or secure multi-party computation (MPC) (Yao, 1982; Beimel, 2011; Mohassel & Zhang, 2017; Dolev et al., 2019). Recent work proposed to use cryptographic methods to secure federated learning by designing a secure gradients aggregation protocol (Bonawitz et al., 2016) or encrypting gradients (Aono et al., 2017). These approaches slow down the computation by orders of magnitude, and may also require special security environment setups.

Instance Hiding. See Appendix B.

B. Instance Hiding

In the classical setting of instance hiding (Abadi et al., 1987) in cryptography, a computationally-limited Alice is trying to get more powerful computing services Bob_1 and Bob_2 to help her compute a function f on input x, without revealing x. The simplest case is that f is a linear function over a finite field (e.g., integers modulo a prime number). Then Alice can pick a random number r and "hide" the input x by asking Bob_1 for $f(x+r)$ and Bob_2 for $f(r)$, and then infer $f(x)$ from the two answers. When all arithmetic is done modulo a prime, it can be shown that neither Bob_1 nor Bob_2 individually learns anything (information-theoretically speaking) about x. This scheme can also be applied to compute polynomials instead of linear functions.

InstaHide is inspired by the special case where there is a single computational agent Bob_1. Alice has to use random values r such that she knows $f(r)$, and simply ask Bob_1 to supply $f(x+r)$. Note that such a random value r would be *use-once* (also called *nonce* in cryptography); it would not be reused when trying to evaluate a different input.

C. Attacks on *Mixup*

Here we provide more details for the attacks discussed in Section 2.

Notations. We use $\langle u, v\rangle$ to denote the inner product between vector u and v. We use $\mathbf{0}$ to denote the zero vector. For a vector x, we use $\|x\|_2$ to denote its ℓ_2 norm. For a positive integer n, we use $[n]$ to denote set $\{1, 2, \cdots, n\}$. We use $\Pr[]$ to denote the probability, use $\mathbb{E}[]$ to denote the expectation. We use $\mathcal{N}(\mu, \sigma^2)$ to denote Gaussian distribution. For any two vectors x, y, we use $\langle x, y\rangle$ to denote the inner product.

C.1. Don't mix up the same image multiple times

Let us continue with the vision task. This attack argues that given a pair of *Mixup* images $\widetilde{x}_1$ and $\widetilde{x}_2$, by simply checking $\langle \widetilde{x}_1, \widetilde{x}_2\rangle$, the attacker can determine with high probability whether $\widetilde{x}_1$ and $\widetilde{x}_2$ are derived from the same image. We show this by a simple case with $k=2$ in Theorem C.1, where k is the number of images used to generate a *Mixup* sample.

Theorem C.1. *Let $\mathcal{X} \subset \mathbb{R}^d$ with $|\mathcal{X}| = n$ and $\forall x \in \mathcal{X}$, we sample $x \sim \mathcal{N}(0, \sigma^2 I)$ where $\sigma^2 = 1/d$. Let $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ denote three disjoint sets such that $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 = \mathcal{X}$, with probability $1-\delta$, we have:*
Part 1. *For $x_1 \in \mathcal{X}_1, x_2, x_2' \in \mathcal{X}_2, x_3 \in \mathcal{X}_3$, $\langle x_3 + x_1, x_2 + x_2' \rangle \leq c_1 \cdot 4\sigma^2 \sqrt{d} \log^2(dn/\delta)$.*
Part 2. *For $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, x_3 \in \mathcal{X}_3$, $\langle x_3 + x_1, x_3 + x_2 \rangle \geq (c_2 \cdot \sqrt{\log(n/\delta)} + d)\sigma^2 - c_1 \cdot 3\sigma^2 \sqrt{d} \log^2(dn/\delta)$. where $c_1, c_2 > 0$ are two universal constants.*

Proof. **Part 1.** First, we can expand $\langle x_3 + x_1, x_2 + x_2' \rangle$,

$$\langle x_3 + x_1, x_2 + x_2' \rangle = \langle x_3, x_2 \rangle + \langle x_1, x_2 \rangle + \langle x_3, x_2' \rangle + \langle x_1, x_2' \rangle$$

For each fixed $u \in \{x_3, x_1\}$ and each fixed $v \in \{x_2, x_2'\}$, using Lemma C.8, we have

$$\Pr\left[|\langle u, v \rangle| \geq c_1 \cdot \sigma^2 \sqrt{d} \log^2(dn/\delta) \right] \leq \delta/n^2.$$

where $c_1 > 1$ is some sufficiently large constant.

Since x_1, x_2, x_2, x_3 are independent random Gaussian vectors, taking a union bound over all pairs of u and v, we have Lemma C.8, we have

$$\Pr[|\langle x_3 + x_1, x_2 + x_2' \rangle| \geq c_1 \cdot 4\sigma^2 \sqrt{d} \log^2(dn/\delta) \Big] \leq 4\delta/n^2. \tag{1}$$

Part 2.

We can lower bound $|\langle x_3 + x_1, x_3 + x_2 \rangle|$ in the following sense,

$$\begin{aligned} |\langle x_3 + x_1, x_3 + x_2 \rangle| &= |\langle x_3, x_3 \rangle + \langle x_3, x_2 \rangle + \langle x_1, x_3 \rangle + \langle x_1, x_2 \rangle| \\ &\geq |\langle x_3, x_3 \rangle| - |\langle x_3, x_2 \rangle + \langle x_1, x_3 \rangle + \langle x_1, x_2 \rangle| \end{aligned}$$

For a fixed x_3, we can lower bound $\|x_3\|_2^2$ with Lemma C.5,

$$\Pr\left[\|x_3\|_2^2 > (c_2 \cdot \sqrt{\log(n/\delta)} + d)\sigma^2 \right] \geq 1 - \delta/n^2.$$

Since x_1, x_2, x_3 are independent random Gaussian vectors, using Lemma C.8, we have

$$\Pr[|\langle x_3, x_2 \rangle + \langle x_1, x_3 \rangle + \langle x_1, x_2 \rangle| \geq c_1 \cdot 3\sigma^2 \sqrt{d} \log^2(dn/\delta) \Big] \leq 3\delta/n^2. \tag{2}$$

where $c_1 > 1$ is some sufficiently large constant.

Thus, for a fixed x_3, we have

$$|\langle x_3 + x_1, x_3 + x_2 \rangle| \geq (c_2 \cdot \sqrt{\log(n/\delta)} + d)\sigma^2 - c_1 \cdot 3\sigma^2 \sqrt{d} \log^2(dn/\delta)$$

holds with probability $1 - \delta/n^2 - 3\delta/n^2 = 1 - 4\delta/n^2$. □

Corollary C.2. *Let $\mathcal{X} \subset \mathbb{R}^d$ with $|\mathcal{X}| = n$ and $\forall x \in \mathcal{X}$, we sample $x \sim \mathcal{N}(0, \sigma^2 I)$ where $\sigma^2 = 1/d$. Let $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ denote three disjoint sets such that $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 = \mathcal{X}$.*
For any $\beta > 1$, if $(2\beta)^{-1} \cdot \sqrt{d} \log^2(nd/\delta) \geq 4$, then with probability $1-\delta$, we have :
for $x_1 \in \mathcal{X}_1, x_2, x_2' \in \mathcal{X}_2, x_3 \in \mathcal{X}_3$,

$$|\langle x_3 + x_1, x_3 + x_2 \rangle| \geq \beta \cdot |\langle x_3 + x_1, x_2 + x_2' \rangle|.$$

Proof. With Theorem C.1, we have with probability $1-\delta$, we have :

$$\begin{aligned}\frac{|\langle x_3+x_1, x_3+x_2\rangle|}{|\langle x_3+x_1, x_2+x_2'\rangle|} \geq & \frac{(c_2\cdot\sqrt{\log(n/\delta)}+d)\sigma^2 - c_1\cdot 3\sigma^2\sqrt{d}\log^2(dn/\delta)}{c_1\cdot 4\sigma^2\sqrt{d}\log^2(nd/\delta)} \\ = & \frac{c_2\cdot\sqrt{\log(n/\delta)}+d - c_1\cdot 3\sqrt{d}\log^2(nd/\delta)}{c_1\cdot 4\sqrt{d}\log^2(nd/\delta)} \\ = & \frac{1+c_2/(c_1\cdot d)\cdot\sqrt{\log(n/\delta)} - 3/\sqrt{d}\cdot\log^2(nd/\delta)}{4/\sqrt{d}\cdot\log^2(nd/\delta)} \\ \geq & \frac{1-3/\sqrt{d}\cdot\log^2(nd/\delta)}{4/\sqrt{d}\cdot\log^2(nd/\delta)} \geq \frac{1-1/(2\beta)}{1/(2\beta)} = 2\beta-1 \geq \beta,\end{aligned}$$

where the forth step follows from choice c_1 and c_2, the fifth step follows from assumption in Lemma statement, and the last step follows from $\beta > 1$.

Thus, we complete the proof. $\square$

C.2. Attacks that run in $|\mathcal{X}|$ time

This attack says that, if a cross-dataset *Mixup* sample $\widetilde{x}$ is generated by mixing 1 sample from a privacy-sensitive original dataset and $k-1$ samples from a public dataset (say ImageNet (Deng et al., 2009)), then the attacker can crack the $k-1$ samples from the public dataset by simply checking the inner product between $\widetilde{x}$ and all images in the public dataset. We show this formally in Theorem C.3.

Theorem C.3. *Let $\mathcal{X}\subset\mathbb{R}^d$ with $|\mathcal{X}|=n$ and $\forall x\in\mathcal{X}$, we sample $x\sim\mathcal{N}(0,\sigma^2 I)$. Let $\widetilde{x}=\sum_{i=1}^k x_i$, where $x_i\in\mathcal{X}$, with probability $1-\delta$, we have:*
Part 1. *For all $t'\in[n]\backslash[k]$, $\langle\widetilde{x}, x_{t'}\rangle\leq c_1\cdot k\sigma^2\sqrt{d}\log^2(nd/\delta)$.*
Part 2. *For all $t\in[k]$, $\langle\widetilde{x}, x_t\rangle\geq(c_2\cdot\sqrt{\log(n/\delta)}+d)\sigma^2 - c_1\cdot(k-1)\sigma^2\sqrt{d}\log^2(nd/\delta)$.*
where $c_1, c_2>0$ are two universal constants.

We remark that the proof of Theorem C.3 is similar to the proof of Theorem C.1.

Proof. **Part 1.** We can rewrite $\langle\widetilde{x}, x_{t'}\rangle$ as follows:

$$\langle\widetilde{x}, x_{t'}\rangle = \sum_{i=1}^k\langle x_i, x_{t'}\rangle,$$

which implies

$$|\langle\widetilde{x}, x_{t'}\rangle| \leq \sum_{i=1}^k|\langle x_i, x_{t'}\rangle|.$$

For each fixed $i\in[k]$ and each fixed $t'\notin[k]$, using Lemma C.8, we have

$$\Pr\left[|\langle x_i, x_{t'}\rangle|\geq c_1\cdot\sigma^2\sqrt{d}\log^2(dn/\delta)\right]\leq\delta/n^2.$$

where $c_1>1$ is some sufficiently large constant.

Since $x_1, x_2, \cdots, x_k, x_{t'}$ are independent random Gaussian vectors, taking a union over all $i\in[k]$, we have

$$\Pr\left[|\langle\widetilde{x}, x_{t'}\rangle|\geq c_1\cdot k\sigma^2\sqrt{d}\log^2(dn/\delta)\right]\leq\delta k/n^2.$$

Taking a union bound over all $t'\in[n]\backslash[k]$, we have

$$\Pr\left[\forall t'\in[n]\backslash[k],\quad|\langle\widetilde{x}, x_{t'}\rangle|\geq c_1\cdot k\sigma^2\sqrt{d}\log^2(dn/\delta)\right]\leq\delta(n-k)k/n^2.$$

Part 2.

We can lower bound $|\langle \widetilde{x}, x_t \rangle|$ as follows:

$$|\langle \widetilde{x}, x_t \rangle| = \Big| \langle x_t, x_t \rangle + \sum_{i \in [k] \backslash \{t\}} \langle x_i, x_t \rangle \Big| \geq |\langle x_t, x_t \rangle| - \Big| \sum_{i \in [k] \backslash \{t\}} \langle x_i, x_t \rangle \Big|$$
$$\geq |\langle x_t, x_t \rangle| - \sum_{i \in [k] \backslash \{t\}} |\langle x_i, x_t \rangle|.$$

First, we can bound $\|x_t\|_2^2$ with Lemma C.5,

$$\Pr\Big[\|x_t\|_2^2 > (c_2 \cdot \sqrt{\log(n/\delta)} + d)\sigma^2 \Big] \geq 1 - \delta/n^2.$$

For each $i \in [k] \backslash \{t\}$, using Lemma C.8, we have

$$\Pr\Big[|\langle x_i, x_t \rangle| \geq c_1 \cdot \sigma^2 \sqrt{d} \log^2(dn/\delta) \Big] \leq \delta/n^2.$$

where $c_1 > 1$ is some sufficiently large constant.

Taking a union bound over all $i \in [k] \backslash \{t\}$, we have

$$\Pr\Big[\sum_{i \in [k] \backslash \{t\}} |\langle x_i, x_t \rangle| \geq c_1 \cdot (k-1)\sigma^2 \sqrt{d} \log^2(dn/\delta) \Big] \leq \delta(k-1)/n^2.$$

Thus, for a fixed $t \in [k]$, we have

$$|\langle \widetilde{x}, x_t \rangle| \geq (c_2 \cdot \sqrt{\log(n/\delta)} + d)\sigma^2 - c_1 \cdot (k-1)\sigma^2 \sqrt{d} \log^2(dn/\delta)$$

holds with probability $1 - \delta/n^2 - \delta(k-1)/n^2 = 1 - \delta k/n^2$.

Taking a union bound over all $t \in [k]$, we complete the proof. $\square$

Corollary C.4. *Let $\mathcal{X} \subset \mathbb{R}^d$ with $|\mathcal{X}| = n$ and $\forall x \in \mathcal{X}$, we sample $x \sim \mathcal{N}(0, \sigma^2 I)$. Let $\widetilde{x} = \sum_{i=1}^k x_i$, where $x_i \in \mathcal{X}$. For any $\beta > 1$, if $k \leq (2\beta)^{-1} \cdot \sqrt{d} \log^2(nd/\delta)$, then with probability $1 - \delta$, we have : for all $t' \in [n] \backslash [k]$ and all $t \in [k]$*

$$|\langle \widetilde{x}, x_t \rangle| \geq \beta \cdot |\langle \widetilde{x}, x_{t'} \rangle|.$$

Proof. With Theorem C.3, we have with probability $1 - \delta$, we have : for all $t \in [k]$ and $t' \notin [k]$,

$$\frac{|\langle \widetilde{x}, x_t \rangle|}{|\langle \widetilde{x}, x_{t'} \rangle|} \geq \frac{(c_2 \cdot \sqrt{\log(n/\delta)} + d)\sigma^2 - c_1 \cdot (k-1)\sigma^2 \sqrt{d} \log^2(nd/\delta)}{c_1 \cdot k\sigma^2 \sqrt{d} \log^2(nd/\delta)}$$
$$= \frac{c_2 \cdot \sqrt{\log(n/\delta)} + d - c_1 \cdot (k-1)\sqrt{d} \log^2(nd/\delta)}{c_1 \cdot k\sqrt{d} \log^2(nd/\delta)}$$
$$= \frac{1 + c_2/(c_1 \cdot d) \cdot \sqrt{\log(n/\delta)} - (k-1)/\sqrt{d} \cdot \log^2(nd/\delta)}{k/\sqrt{d} \cdot \log^2(nd/\delta)}$$
$$\geq \frac{1 - k/\sqrt{d} \cdot \log^2(nd/\delta)}{k/\sqrt{d} \cdot \log^2(nd/\delta)} \geq \frac{1 - 1/(2\beta)}{1/(2\beta)} = 2\beta - 1 \geq \beta,$$

where the forth step follows from the choice of c_1 and c_2, the fifth step follows from the assumption in the Lemma statement, and the last step follows from $\beta > 1$.

Thus, we complete the proof. $\square$

C.3. Chi-square concentration and Bernstein inequality

We state two well-known probability tools in this section. One is the concentration inequality for Chi-square and the other is Bernstein inequality.

First, we state a concentration inequality for Chi-square:

Lemma C.5 (Lemma 1 on page 1325 of Laurent and Massart (Laurent & Massart, 2000))**.** *Let $X \sim \mathcal{X}_k^2$ be a chi-squared distributed random variable with k degrees of freedom. Each one has zero mean and σ^2 variance. Then*

$$\Pr[X - k\sigma^2 \geq (2\sqrt{kt} + 2t)\sigma^2] \leq \exp(-t), \quad \text{and} \quad \Pr[k\sigma^2 - X \geq 2\sqrt{kt}\sigma^2] \leq \exp(-t).$$

We state the Bernstein inequality as follows:

Lemma C.6 (Bernstein inequality (Bernstein, 1924))**.** *Let $X_1, \cdots, X_n$ be independent zero-mean random variables. Suppose that $|X_i| \leq M$ almost surely, for all $i \in [n]$. Then, for all $t > 0$,*

$$\Pr\left[\sum_{i=1}^n X_i > t\right] \leq \exp\left(-\frac{t^2/2}{\sum_{j=1}^n \mathbb{E}[X_j^2] + Mt/3}\right).$$

C.4. Inner product between a random Gaussian vector a fixed vector

The goal of this section is to prove Lemma C.7. It provides a high probability bound for the absolute value of inner product between one random Gaussian vector with a fixed vector.

Lemma C.7 (Inner product between a random Gaussian vector and a fixed vector)**.** *Let $u_1, \cdots, u_d$ denote i.i.d. random Gaussian variables where $u_i \sim \mathcal{N}(0, \sigma_1^2)$.*

Then, for any fixed vector $e \in \mathbb{R}^d$, for any failure probability $\delta \in (0, 1/10)$, we have

$$\Pr_u\left[|\langle u, e\rangle| \geq 2\sigma_1\|e\|_2\sqrt{\log(d/\delta)} + \sigma_1\|e\|_\infty \log^{1.5}(d/\delta)\right] \leq \delta.$$

Proof. First, we can compute $\mathbb{E}[u_i]$ and $\mathbb{E}[u_i^2]$

$$\mathbb{E}[u_i] = 0, \quad \text{and} \quad \mathbb{E}[u_i^2] = \sigma_1^2.$$

Next, we can upper bound $|u_i|$ and $|u_i e_i|$.

$$\Pr_u[|u_i - \mathbb{E}[u_i]| \geq t_1] \leq \exp(-t_1^2/(2\sigma_1^2)).$$

Take $t_1 = \sqrt{2\log(d/\delta)}\sigma_1$, then for each fixed $i \in [d]$, we have, $|u_i| \leq \sqrt{2\log(d/\delta)}\sigma_1$ holds with probability $1 - \delta/d$.

Taking a union bound over d coordinates, with probability $1 - \delta$, we have : for all $i \in [d]$, $|u_i| \leq \sqrt{2\log(d/\delta)}\sigma_1$.

Let E_1 denote the event that, $\max_{i\in[d]} |u_i e_i|$ is upper bounded by $\sqrt{2\log(d/\delta)}\sigma_1\|e\|_\infty$. $\Pr[E_1] \geq 1 - \delta$.

Using Bernstein inequality (Lemma C.6), we have

$$\begin{aligned}\Pr_u[|\langle u, e\rangle| \geq t] \leq &\ \exp\left(-\frac{t^2/2}{\|e\|_2^2\,\mathbb{E}[u_i^2] + \max_{i\in[d]}|u_i e_i| \cdot t/3}\right)\\ \leq &\ \exp\left(-\frac{t^2/2}{\|e\|_2^2\sigma_1^2 + \sqrt{2\log(d/\delta)}\sigma_1\|e\|_\infty \cdot t/3}\right) \leq \ \delta,\end{aligned}$$

where the second step follows from $\Pr[E_1] \geq 1 - \delta$ and $\mathbb{E}[u_i^2] = \sigma_1^2$, and the last step follows from choice of t:

$$t = 2\sigma_1\|e\|_2\sqrt{\log(d/\delta)} + \sigma_1\|e\|_\infty \log^{1.5}(d/\delta)$$

Taking a union bound with event E_1, we have $\Pr[|\langle u, e\rangle| \geq t] \leq 2\delta$. Finally, rescaling δ finishes the proof. □

C.5. Inner product between two random Gaussian vectors

The goal of this section is to prove Lemma C.8. It provides a high probability bound for the absolute value of inner product between two random (independent) Gaussian vectors.

Lemma C.8 (Inner product between two random Gaussian vectors)**.** *Let $u_1, \cdots, u_d$ denote i.i.d. random Gaussian variables where $u_i \sim \mathcal{N}(0, \sigma_1^2)$ and $e_1, \cdots, e_d$ denote i.i.d. random Gaussian variables where $e_i \sim \mathcal{N}(0, \sigma_2^2)$.*

Then, for any failure probability $\delta \in (0, 1/10)$, we have

$$\Pr_{u,e}\left[|\langle u, e\rangle| \geq 10^4 \sigma_1 \sigma_2 \sqrt{d} \log^2(d/\delta)\right] \leq \delta.$$

Proof. First, using Lemma C.5, we compute the upper bound for $\|e\|_2^2$

$$\Pr_e[\|e\|_2^2 - d\sigma_2^2 \geq (2\sqrt{dt} + 2t)\sigma_2^2] \leq \exp(-t).$$

Take $t = \log(1/\delta)$, then with probability $1 - \delta$,

$$\|e\|_2^2 \leq (d + 3\sqrt{d \log(1/\delta)} + 2\log(1/\delta))\sigma_2^2 \leq 4d \log(1/\delta)\sigma_2^2.$$

Thus

$$\Pr_e[\|e\|_2 \leq 4\sqrt{d \log(1/\delta)}\sigma_2] \geq 1 - \delta.$$

Second, we compute the upper bound for $\|e\|_\infty$ (the proof is similar to Lemma C.7)

$$\Pr_e[|\|e\|_\infty \leq \sqrt{\log(d/\delta)}\sigma_2] \geq 1 - \delta.$$

We define t and t' as follows

$$t = 4 \cdot (\sigma_1 \|e\|_2 \sqrt{\log(d/\delta)} + \sigma_1 \|e\|_\infty \log^{1.5}(d/\delta)), \quad \text{and} \quad t' = 8 \cdot (\sigma_1\sigma_2\sqrt{d}\log(d/\delta) + \sigma_1\sigma_2 \log^2(d/\delta))$$

From the above calculations, we can show

$$\Pr_e[t' \geq t] \geq 1 - 2\delta.$$

By Lemma C.7, for fixed e, we have

$$\Pr_u[|\langle u, e\rangle| \geq t] \leq \delta$$

Overall, we have

$$\Pr_{e,u}[|\langle u, e\rangle| \geq t'] \leq 3\delta$$

Therefore, rescaling δ completes the proof. □

D. Computational hardness results of d-dimensional k-SUM

The basic components in InstaHide schemes are inspired by computationally hard problems derived from the classic SUBSET-SUM problem: given a set of integers, decide if there is a non-empty subset of integers whose integers sum to 0. It is a version of knapsack, one of the Karp's 21 NP-complete problems (Karp, 1972). The k-SUM (Erickson, 1995) is the parametrized version of the SUBSET-SUM. Given a set of integers, one wants to ask if there is a subset of k integers sum to 0. The k-SUM problem can be solved in $O(n^{\lceil k/2 \rceil})$ time. For any integer $k \geq 3$ and constant $\epsilon > 0$, whether

k-SUM can be solved in $O(n^{\lfloor k/2 \rfloor - \epsilon})$ time has been a long-standing open problem. Patrascu (Patrascu, 2010), Abboud and Lewi (Abboud & Lewi, 2013) conjectured that such algorithm doesn't exist.

It is natural to extend definition from one-dimensional scalar/number case to the high-dimensional vector case. For d-dimensional k-sum over finite field, Bhattacharyya, Indyk, Woodruff and Xie (Bhattacharyya et al., 2011) have shown that any algorithm that solves this problem has to take $\min\{2^{\Omega(d)}, n^{\Omega(k)}\}$ time unless Exponential Time Hypothesis is false. Here, Exponential Time Hypothesis is believed to be true, it states that there is no $2^{o(n)}$ time to solve 3SAT with n variables.

In this work, we observe that privacy of InstaHide can be interpreted as d-dimensional k-SUM problem and thus could be intractable and safe. For real field, d-dimensional is equivalent to 1-dimensional due to (Abboud et al., 2014). Therefore, Patrascu (Patrascu, 2010), Abboud and Lewi (Abboud & Lewi, 2013)'s conjecture also suitable for d-dimensional k-SUM problem, and several hardness results in $d = 1$ also can be applied to general $d > 1$ directly.

D.1. k-SUM

We hereby provide a detailed explanation for the d-dimensional k-SUM problem. Let us start with the special case of $d = 1$ and all values are integers.

Definition D.1 (SUBSET-SUM)**.** *Given a set of integers, if there is a subset of integers whose integers sum to* 0.

SUBSET-SUM is a well-known NP-complete problem.

The k-SUM is the parameterized version of the SUBSET-SUM,

Definition D.2 (k-SUM)**.** *Given a set of integers, if there is a subset of k integers whose integers sum to* 0.

The k-SUM problem can be solved in $O(n^{\lceil k/2 \rceil})$ time. For $k = 3$, Baran, Demaine and Patrascu (Baran et al., 2008) introduced algorithm that takes $O(n^2/\log^2 n)$ time. It has been a longstanding open problem to solve k-SUM for some k in time $O(n^{\lceil k/2 \rceil - \epsilon})$. Therefore, complexity communities made the following conjecture,

Conjecture D.3 (The k-SUM Conjecture, (Abboud & Lewi, 2013))**.** *There does not exist a $k \geq 2$, an $\epsilon > 0$, and a randomized algorithm that succeeds (with high probability) in solving* k-SUM *in time $O(n^{\lceil k/2 \rceil - \epsilon})$.*

Although the $n^{\lceil (k/2) \rceil}$ hardness for k-SUM is not based on anything else at the moment, an $n^{\Omega(k)}$ lower bound under ETH is already known due to Abboud and Lewi (Abboud & Lewi, 2013). Recently, Abboud (Abboud, 2019) also shows a weaker $n^{\Omega(k/\log k)}$ lower bound under the Set Cover Conjecture, but it has the advantage that it holds for any fixed $k > 2$.

D.2. k Vector Sum over Finite Field

Now we move onto the d-dimensional k vector sum problem.

Definition D.4 (d-dimensional k-VEC-SUM over finite field)**.** *Given a set of n vectors in $\mathbb{F}_q^d$, if there is a subset of k vectors such that the summation of those k vectors is an all 0 vector.*

A more general definition that fits the InstaHide setting is called k-VEC-T-SUM (where T denotes the "target"),

Definition D.5 (d-dimensional k-VEC-T-SUM over finite field)**.** *Given a set of n vectors in $\mathbb{F}_q^d$, if there is a subset of k vectors such that the summation of those k vectors is equal to some vector z in $\mathbb{F}_q^d$.*

d-dimensional k-VEC-T-SUM and d-dimensional k-VEC-SUM are considered to have the same hardness. Bhattacharyya, Indyk, Woodruff and Xie (Bhattacharyya et al., 2011) proved hardness result for the problem defined in Definition D.4 and Definition D.5. Before stating the hardness result, we need to define several basic concepts in complexity. We introduce the definition of 3SAT and Exponential Time Hypothesis(ETH). For the details and background of 3SAT problem, we refer the readers to (Arora & Barak, 2009).

Definition D.6 (3SAT problem)**.** *Given n variables and m clauses conjunctive normal form* CNF *formula with size of each clause at most 3, the goal is to decide whether there exits an assignment for the n boolean variables to make the* CNF *formula be satisfied.*

We state the definition of Exponential Time Hypotheis, which can thought of as a stronger assumption than P$\neq$NP.

Hypothesis D.7 (Exponential Time Hypothesis (ETH) (Impagliazzo et al., 1998)). *There is a $\delta > 0$ such that* 3SAT *problem defined in Definition D.6 cannot be solved in $O(2^{\delta n})$ running time.*

Now, we are ready to state the hardness result.

Theorem D.8 ((Bhattacharyya et al., 2011)). *Assuming Exponential Time Hypothesis (*ETH*), any algorithm solves k-*VEC-SUM *or k-*T-VEC-SUM *requires $\min\{2^{\Omega(d)}, n^{\Omega(k)}\}$ time.*

D.3. k Vector Sum over Bounded Integers

For the bounded integer case, we can also define the k-VEC-SUM problem,

Definition D.9 (d-dimensional k-VEC-SUM over bounded integers). *For integers k, n, M, $d > 0$, the k-*VEC-SUM *problem is to determine, given vectors $x_1, \cdots, x_n, x \in [0, kM]^d$, if there is a size-$k$ subset $S \subseteq [n]$ such that $\sum_{i \in S} x_i = z$.*

Abboud, Lewi, and Williams proved that the 1-dimensional k-VEC-SUM and d-dimensional k-VEC-SUM are equivalent in bounded integer setting,

Lemma D.10 (Lemma 3.1 in (Abboud et al., 2014)). *Let $k, p, d, s, M \in \mathbb{N}$ satisfy $k < p$, $p^d \geq kM+1$, and $s = (k+1)^{d-1}$. There is a collection of mappings $f_1, \cdot, f_s : [0, M] \times [0, kM] \to [-kp, kp]^d$, each computable in time $O(\mathrm{poly} \log M + k^d)$, such that for all numbers $x_1, \cdots, x_k \in [0, M]$ and targets $t \in [0, kM]$,*

$$\sum_{j=1}^{k} x_j = t \iff \exists i \in [s] \sum_{j=1}^{k} f_i(x_j, t).$$

Due to the above result, as long as we know a hardness result for classical k-SUM, then it automatically implies a hardness result for d-dimensional k-VEC-SUM.

E. Phase Retrieval

E.1. Compressive Sensing

Compressive sensing is a powerful mathematical framework the goal of which is to reconstruct an approximately k-sparse vector $x \in \mathbb{R}^n$ from linear measurements $y = \Phi x$, where $\Phi \in \mathbb{R}^{m \times n}$ is called "sensing" or "sketching" matrix. The mathematical framework was initiated by (Candes et al., 2006; Donoho, 2006).

We provide the definition of the ℓ_2/ℓ_2 compressive sensing problem.

Definition E.1 (Problem 1.1 in (Nakos & Song, 2019)). *Given parameters ϵ, k, n, and a vector $x \in \mathbb{R}^n$. The goal is to design some matrix $\Phi \in \mathbb{R}^{m \times n}$ and a recovery algorithm $\mathcal{A}$ such that we can output a vector x' based on measurements $y = \Phi x$,*

$$\|x' - x\|_2 \leq (1+\epsilon) \min_{k-\mathrm{sparse}\ z \in \mathbb{R}^n} \|z - x\|_2.$$

We primarily want to minimize m (which is the number of measurements), the running time of $\mathcal{A}$(which is the decoding time) and column sparsity of Φ.

The ℓ_2/ℓ_2 is the most popular framework, and the state-of-the-art result is due to (Gilbert et al., 2010; Nakos & Song, 2019).

E.2. Phase Retrieval

In compressive sensing, we can observe $y = \Phi x$. However, in phase retrieval, we can only observe $y = |\Phi x|$. Here, we provide the definition of ℓ_2/ℓ_2 phase retrieval problem.

Definition E.2 ((Li & Nakos, 2018)). *Given parameters ϵ, k, n, and a vector $x \in \mathbb{R}^n$. The goal is to design some matrix $\Phi \in \mathbb{R}^{m \times n}$ and a recovery algorithm $\mathcal{A}$ such that we can output a vector x' based on measurements $y = |\Phi x|$,*

$$\|x' - x\|_2 \leq (1+\epsilon) \min_{k-\mathrm{sparse}\ z \in \mathbb{R}^n} \|z - x\|_2.$$

We primarily want to minimize m (which is the number of measurements), the running time of $\mathcal{A}$ (which is the decoding time) and column sparsity of Φ.

The state-of-the-art result is due to (Li & Nakos, 2018).

The problem formulation of compressive sensing and phase retrieval have many variations, for more details we refer the readers to several surveys (Price, 2013; Nakos, 2019; Song, 2019).

E.3. Comments on Phase Retrieval

We would like to point out that although *InstaHide* can be formulated as a phase retrieval problem, classical techniques will fail as an attack.

To show the phase retrieval formulation of *InstaHide*, we first argue applying the random pixel-wise sign is equivalent to taking the absolute value, namely (A): $= \sigma \circ f_{\text{mix}}(x)$ is equivalent to (B): $= |f_{\text{mix}}(x)|$, where $f_{\text{mix}}(\cdot)$ denote the mixing function. This is because, to reduce from B to A, we can just take absolute value in each coordinate; and to reduce from A to B, we can just select a random mask (i.e., σ) and apply it.

Classical phase retrieval (Fienup, 1978; 1982; Moravec et al., 2007) aims to recover x given $y = |\Phi x|$, where $\Phi \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. In the *InstaHide* setting, one can think of Φ as the concatcation of the private and public dataset, where $n = n_{\text{private}} + n_{\text{public}}$, and x is a k-sparse vector which selects k out of n data points to mixup. Therefore, $y = |\Phi x|$ is the 'encrypted' sample using *InstaHide*. It has been shown that solving x can be formulated as a linear program (Moravec et al., 2007). It is well-known that linear program can be solved in polynomial in number of constraints/variables (Jiang et al., 2020) and thus $m = O(k^2 \log(n/k^2))$ gives a polynomial-time optimization procedure. However, this result does not naturally serve as an attack on *InstaHide*. First, Φ in phase retrieval needs to satisfy certain properties which may not be true for Φ in *InstaHide*. More importantly, running LP becomes impossible for *InstaHide* case, since part of Φ remains unknown.

F. Experiments

F.1. Network architecture and hyperparameters

Table 3 provides Implementation details of the deep models. All experiments are conducted on 24 NVIDIA RTX 2080 Ti GPUs.

	MNIST (LeCun et al., 2010)	**CIFAR-10** (Krizhevsky, 2009)	**CIFAR-100** (Krizhevsky, 2009)
Input normalization parameters (normalized = (input-mean)/std)	mean: (0.50, 0.50, 0.50) std: (0.50, 0.50, 0.50)	mean: (0.49, 0.48, 0.45) std: (0.20, 0.20, 0.20)	mean: (0.49, 0.48, 0.45) std: (0.20, 0.20, 0.20)
Number of Epochs	30	200	200
Network architecture	ResNet-18 (He et al., 2016)	ResNet-18 (He et al., 2016)	NasNet (Zoph et al., 2018)
Optimizer	SGD (momentum = 0.9) (Qian, 1999)		
Initial learning rate (vanilla training)	0.05	0.1	0.1
Initial learning rate (*InstaHide*)	0.05	0.1	0.1
Learning rate decay	None	by a factor of 0.1 at 100 and 150 epochs	by a factor of 0.2 at 60, 120 and 160 epochs
Regularization	ℓ_2-regularization (10^{-4})		
Batch size	128		

Table 3: Implementation details of network architectures and training schemes.

F.2. Details of attacks

We hereby provide more details for the attacks in Section 5.3.

Algorithm 3 Gradients matching attack

```
1:  Require :
2:  The function $F(x;W)$ can be thought of as a neural network, for each $l \in [L]$, we define $W_l \in \mathbb{R}^{m_l \times m_{l-1}}$ to be the
    weight matrix in $l$-th layer, and $m_0 = d_i$ and $m_l = d_o$. $W = \{W_1, W_2, \cdots, W_L\}$ denotes the weights over all layers.
3:  Let $(x_0, y_0)$ denote a private (image, label) pair.
4:  Let $\mathcal{L} : \mathbb{R}^{d_o \times d_o} \to \mathbb{R}$ denote loss function
5:  Let $g(x,y) = \nabla \mathcal{L}(F(x;W), y)$ denote the gradients of loss function, and $\widehat{g} = g(x,y)|_{x=x_0, y=y_0}$ is the gradients
    computed on $x_0$ with label $y_0$
6:  procedure InputRecoveryfromGradients()
7:      $x^{(1)} \leftarrow \mathcal{N}(0,1)$, $y^{(1)} \leftarrow \mathcal{N}(0,1)$          ▷ Random initialization of the input and the label
8:      for $t = 1 \to T$ do
9:          Let $D_g(x,y) = \|g(x,y) - \widehat{g}\|_2^2$
10:         $x^{(t+1)} \leftarrow x^{(t)} - \eta \cdot \nabla_x D_g(x,y)|_{x=x^{(t)}}$
11:         $y^{(t+1)} \leftarrow y^{(t)} - \eta \cdot \nabla_y D_g(x,y)|_{y=y^{(t)}}$
12:     end for
13:     return $x^{(T+1)}$, $y^{(T+1)}$
14: end procedure
```

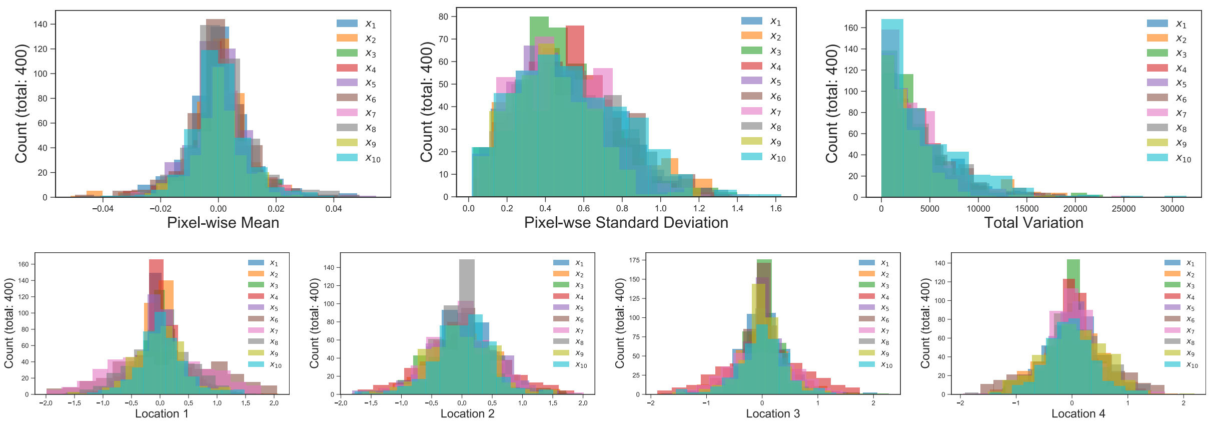

Figure 8: Empirical distributions of 3 statistics (first row) and 4 random locations (second row) for 400 *InstaHide* encryptions of 10 different images (cross-dataset, $k = 4$). Distributions of encryptions of different images are indistinguishable.

Gradients matching attack. Algorithm 3 describes the gradients matching attack (Zhu et al., 2019). This attack aims to recover the original image from model gradients computed on it. In the InstaHide setting, the goal becomes to recover $\widetilde{x}$, the image after InstaHside. As we have shown in Section 4, the upper bound on the privacy loss in gradients matching attack is the loss when attacker is given $\widetilde{x}$.

F.3. Results of the Kolmogorov–Smirnov Test

In order to further understand whether there is significant difference among distributions of *InstaHide* encryptions of different x', we run the Kolmogorov-Smirnov (KS) test (Kolmogorov, 1933; Smirnov, 1948).

Specifically, we randomly pick 10 different private x_i's, $i \in [10]$, and generate 400 encryptions for each x_i (4,000 in total). We sample $\widetilde{x}_{ij}, j \in [400]$, the encryption of a given x_i, and run KS-test under two different settings:

- $\widetilde{x}_{ij}$ v.s. all encryptions (All): KS-test(statistics of $\widetilde{x}_{ij}$, statistics of all $\widetilde{x}_{uj}$'s for $u \in [10]$)
- $\widetilde{x}_{ij}$ v.s. encryptions of other x_u's, $u \neq i$ (Other): KS-test(statistics of $\widetilde{x}_{ij}$, statistics of all $\widetilde{x}_{uj}$'s for $u \in [10], u \neq i$)

For each x_i, we run the Kolmogorov–Smirnov test for 50 independent $\widetilde{x}_{ij}$'s, and report the averaged p-value as below (a higher p-value indicates a higher probability that $\widetilde{x}_{ij}$ comes from the tested distribution). We test 7 different statistics: pixel-wise mean, pixel-wise standard deviation, total variation, and the pixel values of 4 random locations. KS-test results suggest that, there is no significant differences among distribution of encryptions of different images.

	Pixel-wise Mean		**Pixel-wise Std**		**Total Variation**		**Location 1**		**Location 2**		**Location 3**		**Location 4**	
	All	Other	All	Other	All	Other	All	Other	All	Other	All	Other	All	Other
x_1	0.49	0.49	0.54	0.54	0.54	0.54	0.61	0.61	0.58	0.58	0.61	0.61	0.59	0.59
x_2	0.53	0.52	0.49	0.49	0.49	0.49	0.65	0.65	0.56	0.56	0.61	0.61	0.74	0.74
x_3	0.54	0.53	0.55	0.55	0.55	0.55	0.61	0.61	0.68	0.68	0.69	0.69	0.58	0.58
x_4	0.49	0.48	0.49	0.49	0.49	0.49	0.48	0.48	0.40	0.41	0.70	0.70	0.70	0.70
x_5	0.51	0.51	0.50	0.50	0.50	0.50	0.60	0.60	0.72	0.72	0.66	0.66	0.50	0.50
x_6	0.44	0.43	0.52	0.51	0.52	0.51	0.66	0.66	0.69	0.69	0.52	0.52	0.60	0.59
x_7	0.45	0.45	0.48	0.48	0.48	0.47	0.62	0.62	0.52	0.52	0.64	0.65	0.67	0.66
x_8	0.46	0.46	0.50	0.49	0.50	0.49	0.75	0.76	0.69	0.69	0.63	0.63	0.73	0.73
x_9	0.53	0.53	0.53	0.53	0.53	0.53	0.60	0.60	0.67	0.67	0.54	0.54	0.59	0.59
x_{10}	0.44	0.44	0.37	0.37	0.38	0.37	0.66	0.67	0.65	0.65	0.67	0.67	0.65	0.65

Table 4: Averaged p-values for running KS-test on 50 encryptions for each $x_i, i \in [10]$. For each row, 'All' and 'Other' tests give similar p-values, suggesting there is no significant differences among distribution of encryptions of different images.