Abstract

Compression schemes have been extensively used in Federated Learning (FL) to reduce the communication cost of distributed learning. While most approaches rely on a bounded variance assumption of the noise produced by the compressor, this paper investigates the use of compression and aggregation schemes that produce a specific error distribution, e.g., Gaussian or Laplace, on the aggregated data. We present and analyze different aggregation schemes based on layered quantizers achieving exact error distribution. We provide different methods to leverage the proposed compression schemes to obtain compression-for-free in differential privacy applications. Our general compression methods can recover and improve standard FL schemes with Gaussian perturbations such as Langevin dynamics and randomized smoothing.

1 INTRODUCTION

Machine learning has become increasingly data-hungry, requiring vast amounts of data to train models effectively. Federated learning (FL) (McMahan et al., 2017; Karimireddy et al., 2020; Kairouz et al., 2021b) has emerged as a promising approach for collaborative machine learning in distributed settings. In FL, multiple parties with their local datasets participate in a joint model training process without exchanging their raw data. However, communication from these devices to a central server can be slow and expensive resulting in a bottleneck. Thus, compression schemes have been widely used to reduce the size of the updates before transmission. Standard compression schemes (Konečný et al., 2016; Alistarh et al., 2017; Wen et al., 2017; Lin et al., 2018; Li et al., 2022) typically assume relatively bounded variance assumptions on the noise produced by the compressor, i.e., for some $w > 0$ a (randomized) compression scheme $C : \mathbb{R}^d \rightarrow \mathbb{R}^d$ satisfies $E[\|C(x) - x\|_2^2] \leq \omega \|x\|_2^2$ for any $x \in \mathcal{X} \subseteq \mathbb{R}^d$. These schemes do not take into account the specific error distribution of the noise, which may depend on the input distribution. This can be hindering in settings where control of the noise shape can provide tighter analysis or can enable the fulfillment of additional constraints, e.g. differential privacy.

In a wider scope, different structures of the compression error have been extensively leveraged beyond FL such as in: lossy data compression for audio and music compression (Johnston, 1988; Sayood, 2017); medical imaging (Liu et al., 2017; Ammah and Owusu, 2019); error control in wireless communications (Costello et al., 1998); quantization in deep learning (Rastegari et al., 2016) with weight pruning (Han et al., 2015; Anwar et al., 2017), low-precision weights (Jacob et al., 2018; Jung et al., 2019) and data perturbation for privacy-preserving machine learning (Abadi et al., 2016).

Contributions. In light of the above context, this work aims to: (1) propose different quantized aggregation schemes based on layered quantizers that produce a specific error distribution, e.g. Gaussian or Laplace. (2) provide theoretical guarantees and practical implementation of the developed aggregate mechanisms. (3) demonstrate FL applications in which we directly benefit from an exact Gaussian noise distribution, namely differential privacy, Langevin dynamics, and randomized smoothing.

Related work. Recently, different compression techniques with a precise noise distribution have been applied to various machine learning tasks. Agustsson and Theis (2020) apply subtractively dithered quantization (Roberts, 1962; Ziv, 1985; Zamir and Feder, 1992) to ensure a uniformly distributed noise in neural compression. Relative entropy coding (Havasi et al., 2019; Flamich et al., 2020) aims at compressing the...
model weights and latent representations to a number of bits approximately given by their relative entropy from a reference distribution. Differential privacy can be achieved by ensuring a precise noise distribution, using lattice quantization (Amiri et al., 2021; Lang et al., 2023), randomized dithered quantization (Shahmiri et al., 2024), minimal random coding (Shah et al., 2022), or randomized encoding (Chaudhuri et al., 2022). These methods can be regarded as examples of channel simulation (Bennett et al., 2002; Harsha et al., 2010; Li and El Gamal, 2018), which is a point-to-point compression scheme where the output follows a precise conditional distribution given the input. Refer to (Yang et al., 2023; Theis et al., 2022) for more applications of channel simulation in neural compression.

In this paper, we not only consider a point-to-point compression setting, but also a distributed mean estimation and aggregation setting (Suresh et al., 2017), which is one of the most fundamental building blocks of FL algorithms (Kairouz et al., 2021b). There are $n$ users holding the data $x_1, \ldots, x_n$ respectively (which can be the gradient as in FedAvg, or the model weights as in FedSGD (McMahan et al., 2017)), who communicate with the server to allow the server to output an estimate $Y$ of the mean, with a precise noise distribution $Y - n^{-1} \sum_i x_i \sim Q$, which can then be used to perform model updates with a more precise behavior.

There are two approaches to this problem, namely individual mechanisms where point-to-point compression is performed between each user and the server, and the server simply averages the reconstructions of the data of each user (possibly with some postprocessing); and aggregate mechanisms where the encoding functions of all users are designed as a whole to ensure a precise noise distribution of the final estimate $Y$. For individual mechanisms, we study the communication costs of the direct and shifted layered quantizers based on (Wilson, 2000; Hegazy and Li, 2022). For aggregate mechanisms, we propose a novel method, called the aggregate Gaussian mechanism, to ensure that the overall noise distribution $Q$ is exactly Gaussian, and analyze its communication cost.

In particular, for differential privacy (DP), the above schemes allow us to consider two trust settings. First, for a completely trusted server, if the clients wish to prevent the output of the server from leaking information on their data $x_1, \ldots, x_n$, a DP restriction (Dwork et al., 2006) can be imposed on the output $Y$. This is achieved by requiring the noise distribution $Q$ to be a privacy-preserving noise distribution. For example, Gaussian noise can guarantee $(\varepsilon, \delta)$-DP (Dwork et al., 2014) and Rényi DP (Mironov, 2017). Second, when the server is less-trusted, the clients wish for the server to not know their individual datapoints but trust it to faithfully carry out some postprocessing to make the output DP against external observers. In this case, it is vital to ensure that the compression mechanism is homomorphic, and the messages sent from the users to the server can be aggregated before decoding so that it is compatible with secure aggregation (SecAgg) techniques such as (Bonawitz et al., 2017) and other homomorphic cryptosystems. The aggregate Gaussian mechanism we propose is homomorphic, making it suitable for both privacy concerns of trusted and less-trusted servers.

In a recent concurrent work, Hasircioglu and Gunduz (2023) studied the use of dithered quantization to ensure privacy in federated learning, which utilized a similar layered construction as (Wilson, 2000; Hegazy and Li, 2022) to produce a Gaussian noise. The results in this paper not only applies to Gaussian noise, but to any unimodal noise. Also, we propose homomorphic mechanisms that are compatible with SecAgg, whereas the mechanism in (Hasircioglu and Gunduz, 2023) is not homomorphic, making it less clear if it can be secured by SecAgg.

**Outline.** In section 2, we proceed by defining two families of compression mechanisms, individual and aggregate, that enable controlling the compression error. In addition, we demonstrate multiple federated learning settings, differential privacy, Langevin dynamics, and randomized smoothing, where it is advantageous to have tight control over the compression error distribution. Section 3 introduces the two individual mechanisms, the shifted layered quantizer and the direct layered quantizer. It also combines the analysis of Hegazy and Li (2022) and our bound on the conditional entropy of the shifted layered quantizer to compare the communication complexity of both schemes illustrating the settings where each is better. Subsequently, in section 4, we present and analyze a novel aggregate homomorphic compression mechanism enabling the aggregation of clients’ information before decompression. Finally, section 5 provides a more detailed analysis and experiments for leveraging our mechanisms in differential privacy. We defer the experiments on federated Langevin dynamics and the discussion on randomized smoothing to the appendix. The code for the experiments is available on GitHub.

**Notation.** The term log refers to the logarithm in base 2 while ln denotes the natural logarithm. The floor and ceil functions are denoted by $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ respectively with $\lfloor x \rfloor := \lfloor x + 1/2 \rfloor$. For $n \in \mathbb{N}$, we refer to $\{1, \ldots, n\}$ with the notation $[n]$. For a discrete probability distribution $p_X$ supported on $X$ and random variable $X \sim p_X$, $H(X)$ denotes the entropy of $X$, i.e.

---

1https://github.com/mahegz/CompWithExactError
Quantization error follows a target distribution. \( H(X) = H(p_X) = -\sum_{x \in X} p_X(x) \log p_X(x) \). \( H(Y) \) denotes the conditional entropy of \( X \) given \( Y \), i.e., \( H(X|Y) = \mathbb{E}[H(p_X|Y)(Y)] \). For a continuous probability distribution \( f_X \) supported on \( X \) and random variable \( X \sim f_X \), \( h(X) \) denotes the differential entropy of \( X \), i.e. \( h(X) = -\int f_X(x) \log f_X(x)\,dx \). The Lebesgue measure is denoted by \( \lambda \) and \( U(a,b) \) with \( a < b \) refers to the uniform distribution on \((a,b)\). All proofs, additional details and experiments are available in the supplementary material.

2 BACKGROUND, MOTIVATION

Quantized aggregation. Consider \( n \) clients (\( n \geq 1 \)). In the standard FL setting, the training process relies on locally generated randomness at individual participant devices. In this paper, we allow the clients and the server to have shared randomness. Practically, shared randomness can be generated by sharing a small random seed among the clients and the server, allowing them to generate a sequence of shared random numbers. We will see later that the existence of shared randomness can greatly simplify the schemes.

Let \( S_i \in S \) be the shared randomness between client \( i \) and the server, and \( T \in T \) be the global shared randomness between all clients and the server. When building an FL algorithm, one is allowed to choose the joint distribution \( P(S_i), T \) where these variables are usually (but not necessarily) taken to be mutually independent. Client \( i \in [n] \) holds the data \( x_i \in \mathbb{R}^d \) and for privacy and communication constraints, performs encoding to produce the description \( M_i = \mathcal{D}(x_i, S_i, T) \in \mathcal{M} \), where \( \mathcal{D} : \mathbb{R}^d \times S \times T \rightarrow \mathcal{M} \) is the encoding function, and \( \mathcal{M} \) is the set of descriptions (usually taken to be \( \mathbb{Z}^d \)). Given the descriptions \( M_1, \ldots, M_n \), the server produces the reconstruction \( Y = \mathcal{D}(M_1, \ldots, M_n, S_1, \ldots, S_n, T) \) which is an estimate of the average \( n^{-1} \sum_{i=1}^n x_i \), where \( \mathcal{D} : \mathcal{M}^n \times S^n \times T \rightarrow \mathbb{R}^d \) is the overall decoding function. The goal is to control the distribution of the quantization error as described in the next definition.

Definition 1. (Aggregate AINQ mechanism) A quantization scheme with \( n \) clients holding data \( x_1, \ldots, x_n \) and a server producing \( Y \) satisfies the Additive Independent Noise Quantization (AINQ) property if the quantization error follows a target distribution \( Q \) for any \( \{x_i\}_{i=1}^n \), i.e.,

\[
Y - \left( \frac{1}{n} \sum_{i=1}^n x_i \right) \sim Q.
\]

A special case is when \( n = 1 \), which we call point-to-point AINQ mechanism. In this case, an AINQ mechanism with shared randomness \( S \in S \) can be performed in the following steps: (1) sample \( S \sim P_S \), then (2) encode \( M = \mathcal{D}(x, S) \) and (3) decode \( Y = \mathcal{D}(M, S) \). We do not require the global shared randomness \( T \) here since it plays the same role as \( S \). We require the quantization noise to follow a particular distribution \( Y - x \sim Q \). The simplest point-to-point AINQ mechanism is subtractive dithering, which produces a uniform noise distribution.

Example 1. (Subtractive Dithering) For a given step size \( w > 0 \) and input \( X \), subtractive dithering works by sampling \( S \sim U((-1/2, 1/2)) \), encoding the message \( M = [X/w + S] \), and decoding \( Y = (M - S)w \). Then \( Y - X \sim U(-w/2, w/2) \) and is independent of \( X \).

An aggregate AINQ mechanism for \( n \) users can be constructed via a point-to-point AINQ mechanism. For this approach to be applicable, the overall quantization noise must have a divisible distribution that can be expressed as a sum of \( n \) i.i.d. random variables.

Definition 2. (Individual AINQ mechanism) An individual AINQ mechanism is an aggregate AINQ mechanism built via a point-to-point AINQ mechanism where the required quantization noise \( Y - n^{-1} \sum_i x_i \) follows a divisible distribution, \( T \) is empty, the shared randomness is \( S_1, \ldots, S_n \) which are i.i.d. copies of \( S \), user \( i \) produces \( M_i = \mathcal{D}(x_i, S_i) \) and the server outputs \( Y = n^{-1} \sum_i \mathcal{D}(M_i, S_i) \).

Application 1: FL and Differential Privacy. The inherent sensitivity of individual data raises significant concerns about privacy breaches in the distributed setting. To address these challenges, the integration of differential privacy into federated learning has emerged as a compelling approach (Abadi et al., 2016; Truex et al., 2019; Wei et al., 2020; Noble et al., 2022).

Definition 3. (Differential-Privacy (DP)) Any algorithm \( A \) is \((\varepsilon, \delta)\)-differentially private, if for all adjacent datasets \( D_1 \) and \( D_2 \) and all subsets \( E \subset \text{Im}(A) \)

\[
P(A(D_1) \in E) \leq e^\varepsilon P(A(D_2) \in E) + \delta,
\]

where \( P \) is over the randomness used by algorithm \( A \) and \( \text{Im} \) is the image set of \( A \).

The Gaussian mechanism injects controlled noise into computations, allowing for a balance between privacy protection and data utility. For a function \( f \) that operates on a dataset \( \mathcal{D} \), it is defined as

\[
G(D) = f(D) + \mathcal{N}(0, \sigma^2 I)
\]

and it is guaranteed to be \((\varepsilon, \delta)\)-DP if the noise satisfies \( \sigma^2 \geq 2 \Delta_2^2 \ln(1.25/\delta)/\varepsilon^2 \) (Dwork et al., 2014) where \( \Delta_2 = \sup_{D_1, D_2} ||f(D_1) - f(D_2)||_2 \) for \( D_1 \) and \( D_2 \) differing on one element. While common privacy-preserving approaches rely on adding a Gaussian or Laplace noise on top of compression schemes, one can leverage AINQ
mechanisms to directly obtain privacy guarantees with a reduced communication cost. For example, setting the compression error to be a properly scaled Gaussian recovers the Gaussian mechanism of Eq. (3).

**Application 2: FL and Langevin dynamics.** When solving the Bayesian inference problem in the FL setting (Vono et al., 2022), compression operators \( \mathcal{C} : \mathbb{R}^d \rightarrow \mathbb{R}^d \) are commonly unbiased and have a bounded variance. For derivative function \( H \) of a potential, the stochastic Langevin dynamics starts from \( \theta_0 \in \mathbb{R}^d \) and is updated as

\[
\theta_{k+1} = \theta_k - \gamma H(\theta_k) + \sqrt{2\gamma} Z_{k+1},
\]

with \( Z_k \sim \mathcal{N}(0, I_d) \) and \( \gamma > 0 \). Using compression with exact error distribution, one can precisely control the distribution of the error \( \mathcal{C}(X) - X \) and recover the QLSD scheme of Vono et al. (2022) with a reduced communication cost. This may be done via the quantizer \( \mathcal{C} \), such that \( \mathcal{C}(X) - X \sim \mathcal{N}(0, 2I_d/\gamma) \) along with the update rule \( \theta_{k+1} = \theta_k - \gamma \mathcal{C}(H(\theta_k)) \). (See Appendix C.2 for more details)

**Application 3: FL and Randomized Smoothing.** Fast rates for non-smooth optimization problems of the form \( \min_{\theta \in \mathbb{R}^d} f(\theta) = \sum_{i=1}^n f_i(\theta) \) can be attained using the smoothing approach of Duchi et al. (2012); Scaman et al. (2018). These accelerated algorithms such as Distributed Randomized Smoothing (DRS) rely on a smoothed version \( f_\sigma \) of \( f \) defined by

\[
f_\sigma(\theta) = \mathbb{E}_\xi[f(\theta + \sigma \xi)],
\]

where \( \xi \sim \mathcal{N}(0, I_d) \) and \( \sigma > 0 \). Each client approximate the smoothed gradient with a subgradient \( g_i \) evaluated at \( m \) perturbed points \( g_i(\theta + \sigma \xi_j) \) for \( j \in [m] \). Interestingly, the sampling steps may be replaced with compressors that produce exact error distribution. In the spirit of Philippenko and Dieuleveut (2021), one can first compress the model parameter \( \theta \) with a Gaussian error distribution as \( \mathcal{C}(\theta) = \theta + \sigma \xi \) and then evaluate the subgradients at compressed point as \( g_i(\mathcal{C}(\theta)) \) to recover the classical DRS algorithm.

### 3 INDIVIDUAL MECHANISMS

In order to obtain a target noise distribution exactly, we describe two point-to-point AINQ mechanisms based on Hegazy and Li (2022) and on the layered multishift coupling in Wilson (2000), which can be used to construct n-client individual AINQ mechanisms using Def. 2. For geometric interpretations, we refer to them respectively as the direct layered quantizer and the shifted layered quantizer. They both rely on subtractive dithering with step size \( w \) to generate an error which follows a uniform distribution \( U(-w/2, w/2) \), as in Example 1, but with a random step size \( w \).

![Figure 1: Illustration of the sampling area for different distributions: (shifted) Gaussian and (shifted) Laplace.](image)

#### 3.1 Individual Gaussian Mechanisms via Direct and Shifted Layered Quantizers

Consider a random variable \( Z \) following a unimodal distribution \( f_Z \). Instead of having a deterministic value of \( w \), it is randomly sampled from an appropriate distribution ensuring that the marginal error follows \( f_Z \). Define the superlevel set \( \mathcal{L}_z(f_Z) := \{ u \in \mathbb{R} : f_Z(u) \geq z \} \) for any \( z \in \mathbb{R} \) and let \( \tilde{Z} := \max f_Z \). For any \( x \in (0, \tilde{Z}) \), define \( b_{\tilde{Z}}(x) = \inf \mathcal{L}_x(f_Z) \) and \( b_{\tilde{Z}}^+(x) = \sup \mathcal{L}_x(f_Z) \).

As illustrated in Figure 1, for a unimodal distribution, sampling under the graph can be thought of as sampling from an infinite mixture of uniform distributions over continuous intervals.

**Direct Layered Quantizer.** In order to sample from a real continuous random variable, it is sufficient to uniformly sample in the area under its graph and then project onto the \( x \)-axis. The construction below has been mentioned in the special case of Gaussian noise in Agustsson and Theis (2020), and studied in the general unimodal case in Hegazy and Li (2022).

**Definition 4.** (Direct layered quantizer) Given any \( Z \) with a unimodal p.d.f. \( f_Z \), the direct layered quantizer, producing an error \( Z \), is defined using the encoder \( \mathcal{E} \), the decoder \( \mathcal{D} \), and \( P_S \) such that \( S = (U, D_Z) \) with \( U \sim U(0, 1) \) independent of \( D_Z \sim f_D \) where the p.d.f. \( f_D \) is defined through the superlevel sets of \( f_Z \). For all \( x \in \mathbb{R} \), \( f_D(x) = \lambda(\mathcal{L}_x(f_Z)) = (b_{\tilde{Z}}^+(x) - b_{\tilde{Z}}^-(x)) \mathbf{1}_{(0, \tilde{Z})}(x) \) and

\[
M = \mathcal{E}(X, S) = \frac{X f_D(D_Z) + U}{\lambda(\mathcal{L}_x(f_Z))}.
\]

**Shifted Layered Quantizer.** Another approach for leveraging subtractive dithering is based on multishift coupling, as described in Wilson (2000). It is based on the idea of using a sequence of shifted uniform distributions to generate a sequence of target distributions. The key idea is that the sampling from an area \( A \) and then projecting onto the \( x \)-axis is equivalent to the sampling from any horizontal reflection of \( A \) and then projecting onto the \( x \)-axis. Thus, one can can flip one side of a unimodal distribution and still take samples...
from the part under the flipped side (see Figure 1 with the shifted distributions).

**Definition 5.** *(Shifted layered quantizer)* Given any $Z$ with a unimodal p.d.f. $f_Z$, the layered randomized quantizer, producing an error $Z$, is defined using the encoder $\mathcal{E}$, the decoder $\mathcal{D}$ and randomness $\mathcal{P}_S$ such that $S = (U, W_Z)$ with $U \sim \mathcal{U}(0, 1)$ independent of $W_Z \sim f_W$ where the p.d.f. $f_W$ is defined through the superlevel sets of $f_Z$. For all $x \in \mathbb{R}$, 

$$f_W(x) = (b^+_{Z}(x) - b^-_{Z}(Z - x)) 1_{(a, b)}(x),$$

$$\mathcal{E}(X, S) = [X/f_W(W_Z) + U],$$

$$\mathcal{D}(M, S) = (M-U) f_W(W_Z) + \frac{b^+_{Z}(W_Z) + b^-_{Z}(Z-W_Z)}{2}.$$

By construction, the shifted layered quantizer satisfies the AINQ property (see details in Appendix B.1). Note that in definitions 4 and 5 we implicitly assumed that the distribution is bounded i.e. $Z$ exists. This may be directly relaxed for the construction of the direct layered quantizer by setting the indicator to be always true.

### 3.2 Communication Complexity

To transmit the integer codeword $M$, we should encode it into bits. There are two general approaches. First, if a fixed-length code is used where $M$ is always encoded to the same number of bits, then $\lceil \log |\text{Supp } M| \rceil$ bits are required. Second, if a variable-length code is used, for example, if we encode $M$ using the Huffman code (Huffman, 1952) on the conditional distribution $p_{M|S}$, then we require an expected encoding length bounded between $\mathcal{H}(M|S)$ and $\mathcal{H}(M|S)+1$. We will see that the direct layered quantizer gives a better variable-length performance, whereas the shifted layered quantizer gives a better fixed-length performance.

For variable-length codes, it has been shown in (Hegazy and Li, 2022, Theorem 4) that every AINQ scheme with error distribution $f_Z$ and uniform input $X \sim \mathcal{U}(0, t)$ must satisfy

$$\mathcal{H}(M|S) \geq \log(t) + h(D_Z), \tag{4}$$

where $D_Z$ is defined in Definition 4, and $-h(D_Z)$ is called the *layered entropy*. It has also been shown in Hegazy and Li (2022) that the direct layered quantizer is almost optimal, in the sense that as long as $Z$ has a unimodal distribution and $\text{Supp}(X) \subseteq [0, t]$, it achieves

$$\mathcal{H}(M|S) \leq \log(t) + \frac{8 \log(e)}{t} \sqrt{\mathbb{V}[Z]} + h(D_Z). \tag{5}$$

The gap between (4) and (5) tends to 0 as $t \to \infty$.

While the shifted layered quantizer is not asymptotically optimal, the optimality gap (the gap between the $\mathcal{H}(M|S)$ attained and its optimal value) is still relatively small as shown below.

**Proposition 1.** *(Optimality Gap)* For a target unimodal noise distribution $f_Z$ that is symmetric, the shifted layered quantizer achieves

$$\mathcal{H}(M|S) \leq \log(t) + \frac{8 \log(e)}{t} \sqrt{|\mathbb{V}[Z]|} + h(W_Z),$$

and the optimality gap of using the shifted layered quantizer is upper bounded by $(8 \log(e)/t) \sqrt{|\mathbb{V}[Z]|} + 2$.

Figure 2 below shows the conditional entropy $\mathcal{H}(M|S)$ needed to simulate a Gaussian noise and a Laplace noise with standard deviation $\sigma \in \{1; 3\}$ according to the support size $t$ where $X \sim \mathcal{U}(0, t)$. The gap is smaller than 1 bit for all the values computed.

While the direct layered quantizer provides better performance with respect to conditional entropy, the shifted layered quantizer gives a better worst-case guarantee and is only slightly sub-optimal with respect to conditional entropy. The advantage of the shifted layered quantizer is its fixed-length performance. A fixed-length code has the advantage that we do not have to build a Huffman code on the conditional distribution $p_{M|S}$ for each $S$, which may be infeasible. Since the shifted layered quantizer has a quantization step size bounded away from 0 as shown in Figure 1, it provides a fixed upper bound on the number of quantization bits.

**Figure 2:** Conditional entropy $\mathcal{H}(M|S)$ of the layered quantizers with Gaussian/Laplace error distribution.

**Proposition 2.** *(Minimal step size for shifted layered)*

*For a required $f_Z$, denote by $\eta_Z$ the minimal step size of the shifted layer quantizer, i.e. $\eta_Z := \min f_W$. Assume that the input $X$ lies in a fixed interval of length $t$.*

- **If** $Z \sim \text{Laplace}(0, \sigma/\sqrt{2})$ then $\eta_Z = \sigma \sqrt{2}/2$ and $|\text{Supp } M| \leq 2 + t/(\sigma \sqrt{2} \ln 2)$.

- **If** $Z \sim \mathcal{N}(0, \sigma^2)$ then $\eta_Z = 2\sigma \sqrt{\ln 4}$ and $|\text{Supp } M| \leq 2 + t/(2\sigma \sqrt{\ln 4})$. 
4 AGGREGATE MECHANISMS

4.1 Homomorphic Mechanisms

For an individual AINQ mechanism, the descriptions \( M_1, \ldots, M_n \) sent from the users must all be available to the server. The descriptions cannot be first aggregated into a single number before decoding. We study aggregate AINQ mechanisms where such “intermediate aggregation” of descriptions is possible.

Definition 6. (Homomorphic) An aggregate AINQ mechanism is homomorphic if there exist functions \( \mathcal{D} : \mathcal{M} \times \mathcal{S} \times T \rightarrow \mathbb{R}^d \) (called the homomorphic decoding function) such that the overall decoding function is

\[
\mathcal{D}(m_1, s_1, \ldots, m_n, s_n, t) = \frac{1}{n} \mathcal{D} \left( \sum_{i=1}^{n} m_i, \sum_{i=1}^{n} s_i, t \right)
\]

and for all \( t, \mathcal{D}(:,:,t) : \mathcal{M} \times \mathcal{S} \rightarrow \mathbb{R}^d \) is a homomorphism, i.e., \( \mathcal{D}(m, s, t) + \mathcal{D}(m', s', t) = \mathcal{D}(m + m', s + s', t) \) for all \( m, m', s, s' \).

Here we require \( \mathcal{M} \) and \( \mathcal{S} \) to be abelian groups (e.g., \( \mathcal{M} = \mathbb{Z}^d, \mathcal{S} = \mathbb{R}^d \)) so we can perform addition over \( \mathcal{M} \) and \( \mathcal{S} \). For a homomorphic scheme, the server only requires the common randomness \( (S_i)_i, T \) and the sum of the descriptions \( \sum_{i=1}^{n} M_i \), not the individual descriptions \( M_1, \ldots, M_n \).

In federated learning, homomorphic mechanisms have several advantages. First, the descriptions \( M_1, \ldots, M_n \) can be passed from the clients to the server through a network for sum computation (Ramamoorthy and Langberg, 2013; Rai and Dey, 2012; Qu et al., 2021; Rizk and Sayed, 2021), resulting in a reduction of communication cost. A simple method is to have each node in the network add all incoming signals and pass the sum to the next node. The internal nodes in the network do not need to access \( (S_i)_i, T \). Second, homomorphic mechanisms are compatible with SecAgg (Bonawitz et al., 2017) and other additive homomorphic cryptosystems. We can perform SecAgg on \( M_1, \ldots, M_n \) so that the server can only know \( \sum_{i=1}^{n} M_i \), but not the individual \( M_i \)'s so as to preserve privacy.

The layered quantizers of the previous section do not produce homomorphic mechanisms. The reason is that the quantization step sizes of the \( n \) users are randomly drawn and different. Thus, the quantized data \( M_i \)'s are at different scales and cannot be added together. A simple and naive way to obtain an homomorphic mechanism is to rely on subtractive dithering (Example 1) with a fixed step size \( w \) for each user.

4.2 Irwin-Hall Mechanism

When applying subtractive dithering for each of the \( n \) users, the resultant noise of the individual AINQ mechanism follows a scaled Irwin-Hall distribution. More precisely, consider \( S = (S_1, \ldots, S_n) \) to be an i.i.d. sequence of \( \mathcal{U}(-1/2, 1/2) \) random variables and \( T = 0 \) to be degenerate. The encoding function is \( M_i = \delta(x_i, S_i) = [x_i/w + S_i] \) where \( w := 2\sigma\sqrt{3n} \), and the decoding function is \( Y = w(\sum_i M_i - \sum_i S_i) \). The noise is a scaled Irwin-Hall distribution \( IH(n, 0, \sigma^2) \), where \( IH(n, \mu, \sigma^2) \) denotes the distribution of \( n^{-1}\sum_{i=1}^{n} Z_i + \mu \) with \( Z_1, \ldots, Z_n \) \( \overset{iid}{\sim} \mathcal{U}(-\sigma\sqrt{3n}, \sigma\sqrt{3n}) \). Note that \( IH(n, 0, \sigma^2) \) has mean 0 and variance \( \sigma^2 \), and approximates the Gaussian distribution \( \mathcal{N}(0, \sigma^2) \) when \( n \) is large. We call this the Irwin-Hall mechanism.

While this mechanism is simple and homomorphic (the decoding function only depends on \( \sum_i M_i \) and \( \sum_i S_i \)), an obvious downside is that the resultant noise distribution is the Irwin-Hall distribution, not the Gaussian distribution. The Irwin-Hall distribution itself is not a privacy-preserving noise for \((\varepsilon, \delta)\)-differential privacy nor Rényi privacy. Moreover, specific FL applications such as stochastic Langevin dynamics (Welling and Teh, 2011; Vono et al., 2022) or randomized smoothing (Duchi et al., 2012; Scaman et al., 2018) specifically require a Gaussian distribution. This motivates the development of advanced aggregate AINQ mechanisms.

4.3 Aggregate Mechanism

Let \( P = IH(n, 0, \sigma^2) \) be the Irwin-Hall distribution. We have seen in Section 4.2 that the Irwin-Hall mechanism produces a noise with distribution \( P \). The idea is now to compose the desired noise distribution \( Q \) (e.g., Gaussian) into a mixture of shifted and scaled versions of \( P \) in the form \( aP + b \), use the global common randomness \( T \) to select the shifting and scaling factors according to the mixture probabilities, and perform the Irwin-Hall mechanism with the input and output shifted and scaled. In this section, we construct a homomorphic aggregate AINQ mechanism with a noise distribution \( Q \), called the aggregate \( Q \) mechanism, using this strategy.

Definition 7. (Mixture set) For probability distributions \( P, Q \), denote by \( \Pi_{A,B}(P, Q) \) the set of joint probability distributions \( \pi_{A,B} \) of the random variables \( A \in \mathbb{R} \) and \( B \in \mathbb{R} \) such that if \( (A, B) \sim \pi_{A,B} \) is independent of \( Z \sim P \), then \( AZ + B \sim Q \).

Definition 8. (Aggregate \( Q \) mechanism) Let \( S_1, \ldots, S_n \overset{iid}{\sim} \mathcal{U}(-1/2, 1/2) \) and \( T = (A, B) \sim \pi_{A,B} \in \Pi_{A,B}(P, Q) \). The aggregate \( Q \) mechanism is defined
By \( w := 2\sigma\sqrt{3n} \) and

\[
\begin{align*}
\mathcal{D}(x, s, a, b) & := \left[ x/(aw) + s \right], \\
\mathcal{D}((m_i)_1, (s_i)_1, a, b) & := aw \left( \sum_{i=1}^{n} m_i - \sum_{i=1}^{n} s_i \right) + b.
\end{align*}
\]

Basically, we generate \( A, B \) randomly, and then run the Irwin-Hall mechanism scaled by \( A \) and shifted by \( B \). The resultant noise distribution is \( Q \) as precise in the next Proposition.

**Proposition 3.** The aggregate mechanism of Def. 8 satisfies the AINQ property with noise distribution \( Q \), and is homomorphic.\(^2\)

### 4.4 Aggregate Gaussian Mechanism

We now study the case where \( Q \) is the Gaussian distribution, and describe the construction of \( \pi_{A,B} \) which decomposes \( Q \) into a mixture of Irwin-Hall distributions. We call this the aggregate Gaussian mechanism.

**Step 1.** First, we study how to decompose \( \mathcal{U}(-1/2, 1/2) \) (instead of \( Q \)) into a mixture of shifted and scaled versions of the Irwin-Hall distribution \( P \) with a pdf \( f \). Assume \( f \) is appropriately scaled so its support is \([-1/2, 1/2]\). For \(-1/2 \leq x \leq 1/2\), decompose \( \mathcal{U}(-1/2, 1/2) \) into the mixture \((1/f(0))f(x) + (1 - 1/f(0))\varphi(x) \) where \( \varphi(x) = (f(0) - f(x))/(f(0) - 1) \) is a bimodal distribution with modes \{-1/2, 1/2\}, and can be decomposed into uniform distributions using the strategy in Section 3.1. These uniform distributions can be recursively decomposed into a mixture of a shifted/scaled \( f \) and other uniform distributions, and so on. See Figure 3, and Algorithm DecomposeUnif in Appendix A.2.

**Step 2.** To decompose the Gaussian distribution \( Q \) with a pdf \( g \), we first decompose \( g \) into the mixture \( g(x) = \lambda f(x) + (1 - \lambda)\psi(x) \) where \( \lambda \) is as large as possible such that \( \psi \) is still unimodal. A practical choice is \( \lambda = \inf_{x > 0} g(y)/df(y) \) if \( n \geq 3 \), and \( \lambda = 0 \) if \( n \leq 2 \). We then decompose \( \psi \) into a mixture of uniform distributions, using the strategy in Section 3.1, and decompose those uniform distributions using the aforementioned strategy. Refer to Figure 3, and Algorithm DecomposeUnif in Appendix A.4.

\(^2\)Technically, \( \mathcal{D} \) is not in the form (6) due to the “+b” term, though it can be absorbed into \( s_i \). Let \( b'_i = b/n \), and treat \((s_i, b'_i)\) as the common randomness between client \( i \) and the server. We then have \( \mathcal{D}((m_i)_1, (s_i)_1, a, b) = aw \left( \sum_{i=1}^{n} m_i - \sum_{i=1}^{n} s_i - b_i \right) \), which is in the form (6) by taking \( \mathcal{D}(m, s, b', a) = aw(m - s + b') \).

### 4.5 Theoretical Analysis

We now study the communication cost. Assuming the data \( x_i \) is bounded by \( |x_i| \leq t/2 \) for \( i = 1, \ldots, n \), we have \( |\mathcal{D}(x_i, s_i, A, B)| \leq [t/(2\sigma\sqrt{3n})] \). Therefore, \( M_i \) can be encoded into \( \lceil \log(t/(w/|A|)) + 3 \rceil \) bits conditional on \( A \). The expected amount of communication per client is upper-bounded by \( \mathbb{E}[|\log(t/(w/|A|)) + 3)] \) which is approximately equal to \( \mathbb{E}[\log |A|] + \log(t/(2\sigma\sqrt{3n})) \) bits. Hence, we can construct an aggregate AINQ mechanism with a small amount of communication if we have a small \( \mathbb{E}[\log |A|] \). To study this quantity, we introduce a notion called relative mixture entropy.

**Definition 9** (Relative mixture entropy). Given probability distributions \( P, Q \) over \( \mathbb{R} \), the relative mixture entropy is \( h_M(Q|P) := \sup_{\pi_{A,B}(P,Q)} \mathbb{E}_{\pi_{A,B}}[\log |A|] \) where the supremum is over \( \Pi_{A,B}(P,Q) \).

Relative mixture entropy has several desirable properties similar to the differential entropy (see Appendix A.1). In particular, \( h_M(Q|P) \leq h(Q) - h(P) \) can be upper-bounded in terms of the difference of differential entropies. We have the following bound on the communication cost in terms of \( h_M(Q|P) \).

**Theorem 1.** (Complexity) Let \( P = \mathcal{N}(n, 0, \sigma^2) \) be an Irwin-Hall distribution. Assume \( |x_i| \leq t/2 \) for \( i = 1, \ldots, n \). There exists an aggregate AINQ mechanism for simulating a noise distribution \( Q \), with an expected amount of communication per client upper-bounded by

\[
\begin{align*}
&-h_M(Q|P) + \log \frac{t}{2\sigma\sqrt{3n}} + 6\sigma\sqrt{3n} \log e + \mathbb{E}_{Z \sim Q}[|Z|] + 1. \\
&\leq \mathbb{E}[\log |A|] - \mathbb{E}_{Z \sim P}[|Z|] + 1.
\end{align*}
\]

To give an upper bound on the expected communication cost, it remains to give a lower bound on \( h_M(Q|P) \).

\(^3\)Take log 0 = −∞, so \( \mathbb{E}[\log |A|] = -\infty \) if \( P(A = 0) > 0 \).
Theorem 2. (Lower bound) For two distributions $P, Q$ with pdfs $f, g$, respectively, that are unimodal, differentiable and symmetric around 0 with $L := 2\sup\{x : f(x) > 0\} < \infty$ and $\lambda := \inf_{x>0} d g(x)/df(x)$, we have

$$h_M(Q||P) \geq -(1-\lambda) \left( Lf(0) + \log \frac{eL(g(0)-\lambda f(0))}{2(1-\lambda)} \right).$$

We can then combine Theorems 1, 2 to give a bound on the communication cost of the aggregate Gaussian mechanism. The proofs of the theorems are given in Appendix B.6 and B.7. The pseudocode is given in Algorithms Encode and Decode in Appendix A.5. Figure 4 compares the communication cost per client of the aggregate Gaussian mechanism, individual Gaussian mechanism (direct layered quantizer) and Irwin-Hall mechanism where the bounds are computed using Theorem 1 and Theorem 2.

We can see that the aggregate Gaussian mechanism can have a smaller communication cost than the individual Gaussian mechanism for a large number of clients, though not as small as the Irwin-Hall mechanism. We also remark that aggregate Gaussian is homomorphic (unlike individual Gaussian), and has a noise distribution that is exactly Gaussian (unlike the Irwin-Hall mechanism).

5 COMPRESSION AND SUBSAMPLING FOR PRIVACY

5.1 Trusted server with subsampling strategy

Subsampling (Li et al., 2012; Balle et al., 2018) is a technique for enhancing privacy, where only a subset of clients or their data is selected for transmission. It can be leveraged to reduce communication cost while improving DP guarantees. Recently, (Chen et al., 2023) introduced coordinate-wise subsampling to derive DP schemes with optimal communication utility tradeoffs.

Table 1: Comparison of aggregate AINQ mechanisms – whether they are homomorphic, can produce a Gaussian noise, achieve Rényi DP, and have a fixed number of communication bits used.

We now describe how this scheme can be improved with an AINQ mechanism.

We are interested in the subsampled individual Gaussian mechanism (SIGM) with shifted layer quantizer. First consider the case in dimension $d = 1$ with $n$ clients and sampling variables $B_1, \ldots, B_n \sim \mathcal{B}(\gamma)$ ($\gamma \in [0, 1]$), where $B_i = 1$ if client $i$ is selected and $B_i = 0$ otherwise. Let $\tilde{n} = \sum_i B_i$ be the number of selected clients and let $\mathcal{E}, \mathcal{D}$ be the encoding and decoding functions in Def.5 for the noise distribution $\mathcal{N}(0, (\sigma \gamma n)^2)$. The mechanism works as follows: (1) generate the global shared randomness $B_1, \ldots, B_n \sim \mathcal{B}(\gamma)$; (2) generate the shared randomness $S_i$ if $B_i = 1$; (3) client $i$ sends $M_i = \mathcal{E}(x_i \sqrt{n}, S_i)$ if $B_i = 1$, or $M_i = 0$ if $B_i = 0$; and (4) server outputs $Y = (\gamma n \sqrt{n})^{-1} \sum_{i:B_i=1} \mathcal{D}(M_i, S_i)$.

It can be checked that $Y - (\gamma n)^{-1} \sum_{i:B_i=1} x_i \sim \mathcal{N}(0, \sigma^2)$. The proof, the generalization to $d$ dimensional data, and the algorithm are included in Appendix A.6. This mechanism achieves the same statistical behavior as CSGM (Chen et al., 2023), with the benefit that it directly ensures an exact Gaussian noise through quantization without the need to first incur a compression error independently of DP noise. Invoking (Chen et al., 2023, Theorem 4.1) and Proposition 2, we have the following result.

Proposition 4. If $x_1, \ldots, x_n \in [-c, c]^d$, then SIGM is $(\varepsilon, \delta)$-DP with a noise level of order

$$\sigma^2 = \Theta \left( \frac{c^2 \ln(1/\delta)}{n^2 \gamma^2} + \frac{\varepsilon^2 d \ln(d/\delta) + c \ln(d/\delta)}{n^2 \gamma^2} \right),$$

where the cost per client is $O(\gamma c \ln(2+c/(\sigma \sqrt{n\gamma})))$, and the error $E[|Y - n^{-1} \sum_i x_i|_2^2]$ is at most $\frac{d^2 \varepsilon^2}{n\gamma} + d\sigma^2$.

Remark 1. (Flattening and geometry) To extend this result for data bounded in $\ell_2$ norm, one may rely on flattening techniques to convert the $\ell_2$ geometry into an $\ell_\infty$ geometry and obtain tighter utility analysis. Using Kashin’s representation as in Chen et al. (2023) or some Walsh-Hadamard/Fourier transform as in Kairouz et al. (2021a), the SIGM mechanism enjoys the optimal utility bound of order $O(d/(n^2 \varepsilon^2))$. Observe that these flattening schemes require $O(d^2)$ or $O(d \log d)$ depending on the operation.

Subsampled ind. Gaussian (Sec. 5)  
Aggregate Gaussian (Def.8)  
Irwin-Hall (Section 4.2)

Table 1: Comparison of aggregate AINQ mechanisms – whether they are homomorphic, can produce a Gaussian noise, achieve Rényi DP, and have a fixed number of communication bits used.

We now describe how this scheme can be improved with an AINQ mechanism.

We are interested in the subsampled individual Gaussian mechanism (SIGM) with shifted layer quantizer. First consider the case in dimension $d = 1$ with $n$ clients and sampling variables $B_1, \ldots, B_n \sim \mathcal{B}(\gamma)$ ($\gamma \in [0, 1]$), where $B_i = 1$ if client $i$ is selected and $B_i = 0$ otherwise. Let $\tilde{n} = \sum_i B_i$ be the number of selected clients and let $\mathcal{E}, \mathcal{D}$ be the encoding and decoding functions in Def.5 for the noise distribution $\mathcal{N}(0, (\sigma \gamma n)^2)$. The mechanism works as follows: (1) generate the global shared randomness $B_1, \ldots, B_n \sim \mathcal{B}(\gamma)$; (2) generate the shared randomness $S_i$ if $B_i = 1$; (3) client $i$ sends $M_i = \mathcal{E}(x_i \sqrt{n}, S_i)$ if $B_i = 1$, or $M_i = 0$ if $B_i = 0$; and (4) server outputs $Y = (\gamma n \sqrt{n})^{-1} \sum_{i:B_i=1} \mathcal{D}(M_i, S_i)$.

Subsampled ind. Gaussian (Sec. 5)  
Aggregate Gaussian (Def.8)  
Irwin-Hall (Section 4.2)
Numerical comparison. We empirically evaluate our distributed mean estimation scheme SIGM with the method CSGM in a similar framework as in Chen et al. (2023). The parameter configuration is: number of clients $n \in \{1000; 2000\}$, dimension $d \in \{100; 500\}$, probability of information accidentally being leaked $\delta = 10^{-5}$, privacy budget $\varepsilon \in [0.5; 4]$ and subsampling parameter $\gamma \in \{0.3; 0.5; 1.0\}$. The data is generated by sampling uniformly over the square with a radius $1/\sqrt{d}$. Then, it is sparsified coordinate-wise by setting each coordinate to 0 with probability $0.2$. Figure 5 displays the excess mean squared errors (MSE) obtained over 100 independent runs. The number of bits used by CSGM is kept equal to the number of bits used by SIGM. For the same $\varepsilon$, $\gamma$ and the number of bits used, SIGM allows a smaller MSE compared to the one obtained with CSGM. To control the number of bits used, we calculate the communication cost of using SIGM to ensure exact Gaussian noise, then we run CSGM using the same communication budget.

5.2 Less-trusted with SecAgg

The Distributed Discrete Gaussian (DDG) mechanism (Kairouz et al., 2021a) can leverage SecAgg to achieve differential privacy guarantees against the server, which is a stronger setting than that of the less-trusted server. However, in practical implementation, it often requires a much higher number of bits per coordinate.

We adapt the experiments from Kairouz et al. (2021a) which show the utility of the DDG mechanism with different number of bits against the standard Gaussian mechanism. On the other hand, using Elias gamma coding, we calculate the number of bits needed for the aggregate Gaussian mechanism and the individual mechanism using shifted layered quantizer to match a Gaussian mechanism. The results of Figure 6 are obtained over 30 runs with $n = 500$ and $d = 75$ with data sampled uniformly from the unit sphere. They highlight the great performance of aggregate Gaussian. While DDG offers stronger privacy guarantees, i.e., DP against the server, it comes at a heavy cost in terms of bits in comparison to aggregate Gaussian, which is also compatible with SecAgg.

6 DISCUSSION

In this work, we presented multiple individual and aggregate compression mechanisms that achieve exact error distribution for federated learning applications. The good practical performance of the different mechanisms are confirmed by a theoretical analysis of their communication complexity. While the focus is on producing Gaussian noise, note that the schemes of this paper are valid for any unimodal noise. For privacy concerns, we have developed homomorphic mechanisms that are compatible with SecAgg.

The proposed schemes enable compression for free in various federated settings, i.e., compression can be viewed as a side benefit for fulfilling auxiliary requirements. For example, in differential privacy, these mechanisms allow the precise implementation of the Gaussian or Laplace mechanism using a finite number of bits. Similarly, in Langevin dynamics, AINQ mechanisms establish the relationship between the discretization of the dynamics and the number of bits required.

Acknowledgements

The work of M. Hegazy, R. Leluc and A. Dieuleveut was partially supported by Hi!Paris FLAG project. The work of Cheuk Ting Li was partially supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No.s: CUHK 24205621 (ECS), CUHK 14209823 (GRF)].
References


Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] See supplementary
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] See supplementary

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes] See supplementary
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]
Supplementary Material: Compression with Exact Error Distribution for Federated Learning

Appendix A is dedicated to additional results on aggregated mechanisms with a focus on the relative mixture entropy and the details of the algorithms used in Section 4. Appendix B gathers all the proofs of the theoretical results. Appendix C presents additional details and results for the numerical experiments. Appendix D presents how the proposed quantizers with exact error distribution can be applied to obtain optimal algorithms for non-smooth distributed optimization problems.

A Additional results on aggregated mechanisms
   A.1 Properties of relative mixture entropy ........................................... 15
   A.2 Algorithm DECOMPOSEUNIF .......................................................... 15
   A.3 Theoretical analysis of Algorithm DECOMPOSEUNIF ............................ 16
   A.4 Algorithm DECOMPOSE ............................................................... 17
   A.5 Algorithms for Aggregate Gaussian Mechanism .................................. 17
   A.6 Algorithm SIGM for Differential Privacy Applications ........................ 18

B Proofs of technical results
   B.1 Shifted Layered Quantizer satisfies AINQ ......................................... 19
   B.2 Proof of Proposition 1 ..................................................................... 19
   B.3 Proof of Proposition 2 ..................................................................... 20
   B.4 Proof of Proposition 3 ..................................................................... 20
   B.5 Proof of Proposition 4 ..................................................................... 21
   B.6 Proof of Theorem 1 ......................................................................... 21
   B.7 Proof of Theorem 2 ......................................................................... 21

C Additional Numerical Experiments
   C.1 Compression for free in Differential Privacy .................................. 22
   C.2 Langevin Dynamics ....................................................................... 24

D Compression for Randomized Smoothing in Federated Learning 25
A Additional results on aggregated mechanisms

A.1 Properties of relative mixture entropy

Proposition 5. The relative mixture entropy satisfies:

1. (Shifting and scaling) Let \(X, Y\) be random variables, and denote the distribution of \(X\) as \(P_X\). For constants \(a, c \neq 0\) and \(b, d \in \mathbb{R}\),
   \[
h_M(P_{X+a||P_{aX+b}}) = h_M(P_Y||P_X) + \log \left| \frac{c}{a} \right|.
\]

2. (Concavity in \(Q\)) For a fixed \(P\), \(h_M(Q||P)\) is concave in \(Q\).

3. (Chain rule) For distributions \(P, Q, R\),
   \[
h_M(R||P) \geq h_M(R||Q) + h_M(Q||P).
\]

4. (Bound via differential entropy) If \(P, Q\) are continuous distributions,
   \[
h_M(Q||P) \leq h(Q) - h(P),
\]
   where \(h(P)\) is the differential entropy of \(P\).

Proof. Property (1) is straightforward. To show concavity (2), consider \(\lambda \in [0,1]\) and two distributions \(Q_1, Q_2\), and consider \((A_1, B_1)\) independent of \((A_2, B_2)\) independent of \(Z \sim P\) such that \(A_1Z + B_1 \sim Q_1, A_2Z + B_2 \sim Q_2\). Take \((A, B) = (A_1, B_1)\) with probability \(\lambda\), and \((A, B) = (A_2, B_2)\) with probability \(1 - \lambda\). We have
   \[
   AZ + B \sim \lambda Q_1 + (1 - \lambda)Q_2,
   \]
   \[
   \mathbb{E}[\log |A|] = \lambda \mathbb{E}[\log |A_1|] + (1 - \lambda)\mathbb{E}[\log |A_2|].
\]
   Taking supremum gives us the desired result. To show the chain rule (3), consider \((A_1, B_1)\) independent of \((A_2, B_2)\) independent of \(Z \sim P\) such that \(Y = A_1Z + B_1 \sim Q, A_2Y + B_2 \sim R\). We have \(A_1A_2Z + A_2B_1 + B_2 \sim R\), and \(\mathbb{E}[\log |A_1A_2|] = \mathbb{E}[\log |A_1|] + \mathbb{E}[\log |A_2|]\). Taking supremum gives us the desired result. To show the bound via differential entropy (4), if \((A, B)\) is independent of \(Z \sim P\), then \(AZ + B \sim Q\) and
   \[
   h(Q) = h(AZ + B) \geq h(AZ + B|A, B) = h(Z) + \mathbb{E}[\log |A|].
\]

A.2 Algorithm DECOMPOSEUNIF

As an intermediate step, we study how we can simulate \(U(-1/2, 1/2)\) using a noise with p.d.f. function \(f\) that is unimodal, symmetric around 0, and supported over \([-1/2, 1/2]\) (e.g. a scaled version of Irwin-Hall). The idea is to recursively express uniform distributions as mixtures of shifted and scaled versions of \(f\) and other uniform distributions, and repeat ad infinitum.

We now describe the operation of decomposing \(U(-1/2, 1/2)\) into a mixture of shifted and scaled versions of \(f(x)\). For the sake of simplicity, assume \(f(x)\) is unimodal, symmetric around 0, and supported over \([-1/2, 1/2]\) (e.g. a scaled version of Irwin-Hall). First, we can express the uniform distribution \(U(-1/2, 1/2)\) as a mixture of \(f(x)\) and other uniform distributions as follows. Generate \(U \sim U(-1/2, 1/2)\) independent of \(V \sim U(0, 1)\). If \(V \leq f(U)/f(0)\), then generate \(X \sim f\). If \(V > f(U)/f(0)\) and \(U > 0\), then take \(S = f^{-1}(V f(0))\) (where \(f^{-1}(y) := \inf\{x \geq 0 : f(x) \leq y\}\)), and generate \(X \sim U(S, 1/2)\). If \(V > f(U)/f(0)\) and \(U < 0\), then take \(S = f^{-1}(V f(0))\), \(X \sim U(-1/2, -S)\). Then we can show that \(X \sim U(-1/2, 1/2)\). We can then recursively express these uniform distributions in the mixture as mixtures of shifted and scaled versions of \(f(x)\) and other uniform distributions, and repeat ad infinitum.

Algorithm DECOMPOSEUNIF takes \(f\) and a random number generator \(\Psi\) as input (where we can invoke \(\Psi()\) to obtain a \(U(0, 1)\) random number), and outputs two random variables \(A, B\) such that if we generate \(X \sim f\), then \(AX + B \sim U(-1/2, 1/2)\).
A.3 Theoretical analysis of Algorithm DECOMPOSEUNIF

We now analyze the value of $E[\log A]$ given by Algorithm DECOMPOSEUNIF, which will give a bound on $h_M(\mathcal{U}(-1/2,1/2)\|P)$. We invoke the following result which follows from (Hegazy and Li, 2022, Theorem 1).

**Theorem 3.** For any probability density function $f$, we have

$$h_M(f \| \mathcal{U}(-1/2,1/2)) \leq h_L(f)$$

and equality holds if $f$ is a unimodal distribution with a finite mean, where $h_L(f) := \int_0^\infty \lambda(L_+^+(f)) \log \lambda(L_+^+(f)) \, \mathrm{d}r$ is the layered entropy of $f$, where $L_+^+(f) := \{x \in \mathbb{R} : f(x) \geq \tau\}$ is the superlevel set, and $\lambda(L_+^+(f))$ is the Lebesgue measure.

If $(X,Y) \sim \mathcal{U}\{(x,y) : 0 \leq y \leq f(x)\}$, then $h_L(f) = -h(Y)$ (Hegazy and Li, 2022). Therefore, $h_L(f) \geq -\log \sup_{x \in \mathbb{R}} f(x)$ is lower bounded by the differential min-entropy of $f$. We have the following bound.

**Corollary 1.** For any unimodal p.d.f. $f$ with a finite mean: $h_M(f \| \mathcal{U}(-1/2,1/2)) \geq -\log \sup_{x \in \mathbb{R}} f(x)$.

We can now bound $h_M(\mathcal{U}(-1/2,1/2)\|P)$. The following lemma will be useful in the proof of Theorem 2.

**Lemma 1.** For any unimodal probability density function $f$ with $f^* := \sup_x f(x)$, with a bounded support $L := \sup\{x : f(x) > 0\} - \inf\{x : f(x) > 0\}$, we have $h_M(\mathcal{U}(-1/2,1/2)\|f) \geq -L f^* - \log(eL/2)$.

**Proof.** Write $U := \text{Unif}(-1/2,1/2)$. By the shifting and scaling property in Proposition 5, we can assume $L = 1$ and $f$ is supported over $[-1/2,1/2]$ without loss of generality. Assume the maximum of $f$ is attained at $x_0$. We can express $\text{Unif}(-1/2,1/2)$ as a mixture

$$\frac{1}{f^*} f(x) + \nu_1 g_1(x) + \nu_2 g_2(x),$$

where $\nu_1 := \int_{-1/2}^{x_0} \left(1 - \frac{f(x)}{f^*}\right) \, \mathrm{d}x$, $\nu_2 := \int_{x_0}^{1/2} \left(1 - \frac{f(x)}{f^*}\right) \, \mathrm{d}x$,

and $g_1(x) := \frac{1 - f(x)/f^*}{\nu_1}$ for $-\frac{1}{2} \leq x \leq x_0$, $g_2(x) := \frac{1 - f(x)/f^*}{\nu_2}$ for $x_0 \leq x \leq \frac{1}{2}$.

Note that both $g_1(x)$ and $g_2(x)$ are unimodal, and $\nu_1 + \nu_2 = 1 - 1/f^*$. We have

$$h_M(U\|f) \overset{(a)}{\geq} \nu_1 h_M(g_1\|f) + \nu_2 h_M(g_2\|f) \overset{(b)}{\geq} \nu_1 (h_M(g_1\|U) + h_M(U\|f)) + \nu_2 (h_M(g_2\|U) + h_M(U\|f)) = \nu_1 h_M(g_1\|U) + \nu_2 h_M(g_2\|U) + \left(1 - \frac{1}{f^*}\right) h_M(U\|f).$$

---

A probability density function $f$ is unimodal if there exists $x_0 \in \mathbb{R}$ such that $f(x)$ is nondecreasing for $x \leq x_0$, and nonincreasing for $x \geq x_0$. 
where (a) is by concavity, and (b) is by the chain rule (Proposition 5). Hence,

$$h_M(U\|f) \geq f^* \nu_1 h_M(g_1\|U) + f^* \nu_2 h_M(g_2\|U)$$

(c)

$$\geq f^* \nu_1 \log f^* + f^* \nu_2 \log \frac{f^*}{2}$$

$$= (f^* - 1) \log \frac{f^* - 1}{2f^*}$$

$$= -(f^* - 1) - (f^* - 1) \log \left(1 + \frac{1}{f^* - 1}\right)$$

$$\geq -(f^* - 1) - (f^* - 1) \log e$$

$$= -f^* - \log e$$

where (c) is by Corollary 1.

A.4 Algorithm DECOMPOSE

The goal is to simulate an additive noise channel with a noise p.d.f. $g$ that is unimodal, symmetric around 0 and differentiable knowing that we can simulate an additive noise channel with a noise p.d.f. $f$ with same properties. The naive and quite inefficient way is to decompose $g$ into a mixture of uniform distributions, and run the DECOMPOSEUNIF algorithm on those uniform distributions. Instead, we first decompose $g$ into the mixture $g(x) = \lambda f(x) + (1 - \lambda) \psi(x)$ where $\lambda$ is as large as possible such that $\psi$ is still unimodal. A practical choice is $\lambda = \inf_{x>0} \frac{dg(x)}{df(x)}$ if $n \geq 3$, and $\lambda = 0$ if $n \leq 2$. We then decompose $\psi$ into a mixture of uniform distributions, and run DECOMPOSEUNIF. Algorithm DECOMPOSE below computes the decomposition of $g$ into a mixture of shifted and scaled versions of $f$. It takes $f, g$ and a random number generator $\mathfrak{P}$ as input, and outputs two random variables $A, B$ such that if we generate $X \sim f$, then $AX + B \sim g$.

**Algorithm 2 DECOMPOSE($f, g, \mathfrak{P}$)**

<table>
<thead>
<tr>
<th>Input:</th>
<th>pdfs $f, g$, random number generator $\mathfrak{P}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>scale $a \in (0, \infty)$ and shift $b \in \mathbb{R}$</td>
</tr>
<tr>
<td>1:</td>
<td>$\lambda \leftarrow \inf_{x&gt;0} \frac{dg(x)}{df(x)}$</td>
</tr>
<tr>
<td>2:</td>
<td>Sample $x \sim g$ using $\mathfrak{P}$</td>
</tr>
<tr>
<td>3:</td>
<td>$v \leftarrow g(x) \cdot \mathfrak{P}()$</td>
</tr>
<tr>
<td>4: if</td>
<td>$v &gt; g(x) - \lambda f(x)$ then</td>
</tr>
<tr>
<td>5:</td>
<td>return $(1, 0)$</td>
</tr>
<tr>
<td>6: else</td>
<td></td>
</tr>
<tr>
<td>7:</td>
<td>$s \leftarrow \sup{x' \geq 0 : v \leq g(x') - \lambda f(x')}$</td>
</tr>
<tr>
<td>8:</td>
<td>$L \leftarrow 2 \sup{x : f(x) &gt; 0}$</td>
</tr>
<tr>
<td>9:</td>
<td>$\bar{f} \leftarrow (x \mapsto f(x/L))/L$ (support of $\bar{f}$ is $[-1/2, 1/2]$)</td>
</tr>
<tr>
<td>10:</td>
<td>$(a, b) \leftarrow \text{DECOMPOSEUNIF}(\bar{f}, \mathfrak{P})$</td>
</tr>
<tr>
<td>11:</td>
<td>return $(2as/L, 2bs)$</td>
</tr>
<tr>
<td>12: end if</td>
<td></td>
</tr>
</tbody>
</table>

A.5 Algorithms for Aggregate Gaussian Mechanism

We describe the encoding and decoding algorithms for the aggregate Gaussian mechanism. We assume the server and all the clients share a common random seed, which they use to initialize their pseudorandom number generators (PRNG) $\mathfrak{P}$. Since each PRNG is initialized with the same seed, they are guaranteed to produce the same common randomness $(S_i)_i, A, B$ at the clients and server.
Algorithm 3 `ENCODE(x, n, i, σ, Ψ)`

1: **Input:** data \( x \in \mathbb{R} \), number of clients \( n \), client id \( i \), standard deviation \( σ \), RNG \( Ψ \)
2: **Output:** description \( m \in \mathbb{Z} \)
3: \((a, b) \leftarrow \text{DECOMPOSE}(\text{IH}(n, 0, 1), \mathcal{N}(0, 1), Ψ)\)
4: for \( j = 1, \ldots, n \) do
5: \( s_j \leftarrow Ψ() - 1/2 \)
6: end for
7: \( w \leftarrow 2\sigma\sqrt{3n} \)
8: return \[ \frac{x}{aw} + s_i \]

A.6 Algorithm SIGM for Differential Privacy Applications

Condition on any \( B_1, \ldots, B_n \). Definition 5 ensures that \( \mathcal{D}(M_i, S_i) - x_i\sqrt{n} \sim \mathcal{N}(0, (\sigma n)^2) \). Therefore

\[
Y - (\gamma n)^{-1} \sum_{i : B_i = 1} x_i = (\gamma n\sqrt{n})^{-1} \sum_{i : B_i = 1} (\mathcal{D}(M_i, S_i) - x_i\sqrt{n})
\]

\[
\sim \mathcal{N} \left( 0, (\sigma \gamma n)^2 (\gamma n\sqrt{n})^{-2} \right) = \mathcal{N}(0, \sigma^2).
\]

To generalize to \( d \) dimensional data \( x_1, \ldots, x_n \in \mathbb{R}^d \), we can simply apply SIGM on each coordinate individually. Note that each coordinate \( j \in [d] \) has \( B_1(j), \ldots, B_n(j) \) sampled independently, indicating whether client \( i = 1, \ldots, n \) should send the coordinate \( x_i(j) \), similar to the coordinate subsampling strategy in (Chen et al., 2023). The algorithm is given below.

Algorithm 5 Subsampled Individual Gaussian Mechanism (SIGM)

**Parameters:** Noise variance \( σ^2 > 0 \), Subsampling Bernoulli parameter \( γ \in [0, 1] \)

(Shared Randomness)

\( B_1, \ldots, B_n \in \{0, 1\}^d \) with \( B_i(j) \sim B(γ) \)

\( S_1, \ldots, S_n : S_i(1, j) \sim U(0, 1), S_i(2, j) \sim W_\mathcal{N}(0, (\sigma γ n)^2) \)

\( \tilde{n}(j) \leftarrow \sum_{i=1}^n B_i(j) \quad \text{for} \ j \in [d] \)

(Client side)

**input:** private \( x_i \in \mathbb{R}^d \) with \( \|x_i\|_2 \leq c \)

**Encode:** For \( j \in [d] \)
- If \( B_i(j) = 1 \) then \( M_i(j) \leftarrow \mathcal{D}(x_i(j)\sqrt{n}(j), S_i(\cdot, j)) \)
- else \( M_i(j) \leftarrow 0 \)

**Send:** \( M_i \) to the server

(Server side)

**input:** \( M_1, \ldots, M_n \in \mathbb{Z}^d \)

**Decode:** For \( j \in [d] \)
- \( \bar{\mu}(j) \leftarrow (\gamma n\sqrt{n})^{-1} \sum_{i : B_i(j) = 1} \mathcal{D}(M_i(j), S_i(\cdot, j)) \)

**Return:** \( \bar{\mu} \)

Algorithm 4 `DECODE(Σ_m, n, σ, Ψ)`

1: **Input:** sum of descriptions \( Σ_m = \sum_{i=1}^n m_i \in \mathbb{Z} \), number of clients \( n \), standard deviation \( σ \), RNG \( Ψ \)
2: **Output:** estimated mean \( y \in \mathbb{R} \)
3: \((a, b) \leftarrow \text{DECOMPOSE}(\text{IH}(n, 0, 1), \mathcal{N}(0, 1), Ψ)\)
4: for \( j = 1, \ldots, n \) do
5: \( s_j \leftarrow Ψ() - 1/2 \)
6: end for
7: \( w \leftarrow 2\sigma\sqrt{3n} \)
8: return \( \frac{aw}{n} (Σ_m - \sum_{i=1}^n s_i) + bσ \)
B Proofs of technical results

B.1 Shifted Layered Quantizer satisfies AINQ

Let us show that the shifted layered quantizer yields the same distribution of error scheme as the direct layered quantizer (Definition 4). Let $\Delta = Y - X$, then $\Delta|V \sim \mathcal{U}(b_Z(\bar{Z} - V), b_Z^+(V))$. Denote $M = \arg\max f_Z(x)$ and $m = \text{median}(M)$. It follows that

$$f_\Delta(x) = \int \frac{1\{x \in (b_Z^{-1}(\bar{Z} - v), b_Z^+(v))\}}{W_Z(v)}W_Z(v)dv$$

\[(a) = \int \frac{1\{x \in (b_Z^{-1}(\bar{Z} - v), m)\}}{W_Z(v)}dv + \int \frac{1\{x \in (m, b_Z^+(v))\}}{W_Z(v)}dv\]

\[(b) = \int \frac{1\{x \in (b_Z^{-1}(\bar{Z} - v), m)\}}{W_Z(v)}dv + \int \frac{1\{x \in (m, b_Z^+(v))\}}{W_Z(v)}dv\]

\[(c) = \int \frac{1\{x \in L_v(f_Z)\}}{W_Z(v)}dv = f_Z(x)\]

where (a) is due to the fact that $m \in L_\tau(f_Z)$ for any $\tau$, (b) by the change of variable $v = \bar{Z} - v$, and (c) by unimodality of $f_Z$.

B.2 Proof of Proposition 1

The conditional entropy $\mathcal{H}(M_s|S_s)$ may be decomposed as follows

$$\mathcal{H}(M_s|S_s) - \log(t) = - \log(t) + \int_0^1 \int_0^{\bar{Z}} \mathcal{H}(M_s|S_s = (u, \tau))f_W(\tau)d\tau ds$$

$$\leq - \log(t) + \int_0^1 \int_0^{\bar{Z}} \log \left( \frac{t}{f_W(\tau)} + 2 \right) f_W(\tau)d\tau$$

$$= h(W_Z) + \int_0^{\bar{Z}} \log \left( 1 + \frac{2f_W(\tau)}{t} \right) f_W(\tau)d\tau$$

$$\leq h(W_Z) + \frac{2 \log(e)}{t} \int_0^{\bar{Z}} f_W(\tau)^2 d\tau.$$

The last integral on the right-hand side may be upper bounded by considering the random variables $(Z, S) \sim \mathcal{U}\{(z, s)|s \in [a(z), b(z)]\}$ with $\gamma \in \arg\max f_Z$ and

$$a(x) = 1\{x < \gamma\}\bar{Z} - f_Z(x)$$

$$b(x) = 1\{x < \gamma\}f_Z(x) + 1\{x \geq \gamma\}\bar{Z}$$

This coupling satisfies $S \sim W_Z$, $Z \sim f_Z$, $Z|S \sim \mathcal{U}[b_Z^{-1}(\bar{Z} - S), b_Z^+(S)]$. Then if follows that

$$\int_0^{\bar{Z}} f_W(\tau)^2 d\tau = 4 \int_0^{\bar{Z}} E \left[ \left| Z - \frac{b_Z^+(\tau) + b_Z^{-1}(\bar{Z} - \tau)}{2} \right| S = \tau \right] f_W(\tau)d\tau.$$

Since $Z$ is uniformly distributed over an interval conditional on $S = \tau$, we have that $E\|Z - m\|S = \tau$ is minimized when $m$ is the midpoint of the interval so it holds that

$$\int_0^{\bar{Z}} f_W(\tau)^2 d\tau \leq 4 \int_0^{\bar{Z}} E \|Z - \text{median}(Z)\| S = \tau f_W(\tau)d\tau$$

$$= 4 E \|Z - \text{median}(Z)\|$$
and the last term may be upper bounded using that $\mathbb{E} [\| Z - \text{median}(Z) \|] \leq \mathbb{E} [\| Z \|]$ combined with Jensen inequality to finally obtain $\int_0^Z f_W(\tau)^2 d\tau \leq 4 \sqrt{\mathbb{V}[Z]}$ which gives the final bound

$$\mathcal{H}(M|S) \leq \log(t) + \frac{8 \log(e)}{t} \sqrt{\mathbb{V}[Z]} + h(W_Z).$$

Observe that the optimality gap is

$$\frac{8 \log(e)}{t} \sqrt{\mathbb{V}[Z]} + h(W_Z) - h(D_Z).$$

Thus it is sufficient to bound $h(W_Z) - h(D_Z)$. As $f_W$ and $f_D$ are translation invariant, then we assume without loss of generality that the mode of $f_Z$ is 0. Using the symmetry of $f_Z$, we have $b_Z^+(x) = -b_Z^-(x)$. Then we recover

$$h(D_Z) = -\int_0^Z 2b_Z^+(x) \log(2b_Z^+(x)) \, dx,$$

$$h(W_Z) = -\int_0^Z 2b_Z^+(x) \log(b_Z^+(x) + b_Z^+(Z - x)) \, dx.$$

As $b_Z^+(x) \geq 0$, then

$$h(W_Z) - h(D_Z) \leq \int_0^Z 2b_Z^+(x) \log \left( \frac{2b_Z^+(x)}{b_Z^+(x) + b_Z^+(Z - x)} \right) \, dx$$

$$= 2 \int_0^Z D_Z(X) \, dx = 2.$$

Similarly as $f_W(X) \geq \eta_Z$,

$$h(W_Z) \leq \int_0^Z 2b_Z^+(x) \log \left( \frac{1}{\eta_Z} \right) \, dx$$

$$= -\log(\eta_Z)$$

**B.3 Proof of Proposition 2**

For the Gaussian case, refer to [Wilson (2000, Page 22)](#). We prove the case for Laplace random variable with a similar strategy. Note that $f_W$ is translation invariant w.r.t to $f_Z$, so we consider zero mean Laplace. For $Z \sim \text{Laplace}(0, b)$, we have $f_W(x) = -b \ln(2bx) - b \ln(1 - 2bx)$. Solving $df_W(x^*)/dx = 0$ gives $x^* = 1/4b$ so that $f_W(x^*) = 2b \ln 2$ and plug the value $b = \sigma/\sqrt{2}$ to conclude.

**B.4 Proof of Proposition 3**

Condition on $A = a, B = b$. Subtractive dithering gives

$$\frac{aw}{n} \left( \left\lfloor \frac{x_i}{aw} + S_i \right\rfloor - S_i \right) - \frac{x_i}{n} \sim \mathcal{U} \left( -\frac{aw}{2n}, \frac{aw}{2n} \right).$$

Hence,

$$Y - \frac{1}{n} \sum_{i=1}^n x_i = \frac{aw}{n} \left( \sum_{i=1}^n \left\lfloor \frac{x_i}{aw} + S_i \right\rfloor - \sum_{i=1}^n S_i \right) - \frac{1}{n} \sum_{i=1}^n x_i + b$$

$$\sim \text{IH} \left( n, b, \frac{n}{12} \left( \frac{aw}{n} \right)^2 \right)$$

$$= \text{IH} \left( n, b, \sigma^2 \right)$$

has the same distribution as $aZ + b$ where $Z \sim \mathcal{P} = \text{IH}(n, 0, \sigma^2)$. If we randomize over $A, B$, since $(A, B) \sim \pi_{A,B} \in \Pi_{A,B}(P, Q)$, we have $Y - \frac{1}{n} \sum_{i=1}^n x_i \sim Q$ by the definition of mixture set.
We then demonstrate that the mechanism is homomorphic. Since \( \mathcal{D} \) only depends on \( m_1, \ldots, m_n \) through \( \sum_i m_i \), it allows the server to decode using only \( \sum_i m_i \) and the shared randomness. While \( \mathcal{D} \) is technically not in the form (6) due to the “+b” term, that term can be absorbed into \( s_i \). Let \( b'_i = b/n \), and treat \((s_i, b'_i)\) as the common randomness between client \( i \) and the server. We then have \( \mathcal{D}((m_i), (s_i, b'_i), a) = \frac{aw}{n} (\sum_{i=1}^n m_i - \sum_{i=1}^n (s_i - b'_i)) \), which is in the form (6) by taking \( \mathcal{D}(m, (s, b'), a) = aw(m - s + b') \).

### B.5 Proof of Proposition 4

The SIGM algorithm returns the mean estimate \( \bar{\mu} = (\bar{\mu}_1, \ldots, \bar{\mu}_d) \) where for \( j \in [d] \) we have

\[
\bar{\mu}(j) = (\gamma n \sqrt{n})^{-1} \sum_{i: B_i(j) = 1} \mathcal{D}(M_i(j), S_i(\cdot, j)).
\]

Similarly to the proof in A.6, we have \( \bar{\mu}_j - (n\gamma)^{-1} \sum_{i: B_i(j) = 1} x_i(j) \sim \mathcal{N}(0, \sigma^2) \) and conclude by invoking the proof of (Chen et al., 2023, Theorem 4.1) to obtain the bound on \( \sigma^2 \).

For the communication cost per client, Proposition 2 gives

\[ |\text{Supp} M| \leq 2 + \frac{t}{2\sqrt{n}\sigma \sqrt{\ln 4}}. \]

Since the data \( x_1, \ldots, x_n \) belong to \([-c; c]^d\) and the clients encode \( x_i \sqrt{n} \), we have \( t = 2c \sqrt{n} \). Since \( \mathbb{E}[\bar{t}] = \gamma n \) and only \( \gamma d \) coordinates are selected on average, taking the log on both sides gives the desired result.

### B.6 Proof of Theorem 1

We have

\[
\mathbb{E} \left[ \left\lfloor \log \left( \frac{t}{|A||d|} + 3 \right) \right\rfloor \right] \leq \mathbb{E} \left[ \log \left( 1 + \frac{3d|A|}{t} \right) \right] + \log \frac{t}{d} + \mathbb{E}[-\log |A|] + 1
\]

\[ \leq -\mathbb{E} \log |A| + \frac{3d \log e}{t} \mathbb{E} |A| + \log \frac{t}{d} + 1. \tag{7} \]

Recall that if \( Z \sim P \) is independent of \( A, B \), then \( AZ + B \sim Q \). We have

\[
\mathbb{E}[\|AZ + B\|] = \mathbb{E} \left[ \mathbb{E}[\|AZ + B\| \mid A, B] \right] \\
\quad \geq \mathbb{E} \left[ \mathbb{E}[\|AZ\| \mid A, B] \right] \\
\quad = \mathbb{E}[|A|] \mathbb{E}[|Z|],
\]

where \( (a) \) is because the median of \( aZ \) is 0 for any \( a \in \mathbb{R} \), so \( \mathbb{E}[|aZ + b|] \) is minimized when \( b = 0 \). Combining this with (7) gives the desired result.

### B.7 Proof of Theorem 2

Write \( U := \mathcal{U}(-1/2, 1/2) \). Express \( g(x) \) as a mixture \( g(x) = \lambda f(x) + (1 - \lambda)\psi(x) \). We have

\[
h_M(g \bowtie f) \overset{(a)}{\geq} (1 - \lambda) h_M(\psi \bowtie f) \\
\overset{(b)}{\geq} (1 - \lambda) (h_M(\psi \bowtie U) + h_M(U \bowtie f)) \\
\overset{(c)}{\geq} -(1 - \lambda) \left( \log \frac{g(0) - \lambda f(0)}{1 - \lambda} + Lf(0) + \log \frac{eL}{2} \right),
\]

where \( (a) \) is by concavity, \( (b) \) is by the chain rule (Proposition 5), and \( (c) \) is by Corollary 1 and Lemma 1.
C Additional Numerical Experiments

Comparing AINQ mechanism with Quantization. Applying the shifted layered quantizer with a required compression error \( N(0, \sigma^2 I_d) \) coordinate-wise to an input \( x \in \mathbb{R}^d \) that is unbounded does not necessarily lead to a bounded encoding \( M \) (alternatively bounded number of bits to send \( M \)). However, a classical implementation of quantization starts by specifying the number of bits, then normalizing \( x \) by \( \|x\|_p \) to scale it to the \( \ell_p \) unit sphere and finally perform quantization on each coordinate using \( b \) bits. For decompresion, \( \|x\|_p \) is shared with the decoder to rescale the quantized vector. The communication cost of sharing \( \|x\|_p \) may be ignored as it is often negligible after normalizing by the dimension \( d \). To have a fair comparison with classical quantization schemes, in DP experiments, shifted layered quantization is used to reproduce the desired error distribution. Then, the number of bits used by the shifted layered quantizer is measured on the one hand and the number of bits for quantization is adjusted on the other hand such that it is always equal or higher than the number of bits used for shifted layered quantizer.

For Langevin dynamics, the number of bits \( b \) is specified first. Each client \( i \in [n] \) scales its vector \( x_i \) by \( \|x_i\|_\infty \). The variance \( \sigma_b^2 \) used for the compression is calculated using Proposition 2 with \( t = 2 \). Finally, decoding achieves a compression error of variance \( \sigma_b^2 \|x\|_\infty^2 \). As \( \sigma_b^2 \) is known and the different norms \( \{\|x_i\|_\infty\}_{i \in [n]} \) are shared, the server does some accounting to calculate the error variance of the distributed mean operation and adjusts the added error to reproduce the desired Markov chain. Refer to Algorithm 6 for more details.

Software/Hardware details. The code to reproduce the curves in the Figures below is available in the supplementary material. Other experiments were reproduced through the original implementation of Kairouz et al. (2021b) and Vono et al. (2022). The experiments on differential privacy applications (see Appendix C.1) were performed on a laptop Intel Core i7-10510U CPU 1.80GHz \( \times 8 \). The experiments on stochastic Langevin dynamics (see Appendix C.2) with toy gaussian data were performed on 4 cores of slurm-based cluster with 2.1 Ghz frequency. The computations for Langevin experiments required around 72 hours.

C.1 Compression for free in Differential Privacy

Subsampling and Trusted Server. The parameter configuration for the experiments in Section 5 are: number of clients \( n \in \{1000; 2000\} \), dimension \( d \in \{100; 500\} \). The numerical results of Figure 7 below consider the setting \( d = 500 \) and \( n \in \{250; 500; 1000\} \). For a subsampling parameter \( \gamma \in \{0.3; 0.5; 1.0\} \), we set \( \delta = 10^{-5} \) and privacy budget \( \varepsilon \in [0.5; 4] \). The data is generated in a similar spirit to Chen et al. (2023). Denote by \( X_i(j) \) the \( j \)-th coordinate of the \( i \)-th client data. Then, for \( i \in [n] \) and for \( j \in [d] \), \( X_i(j) \sim 2B(p) \) \( - 1 \) \( U/(\sqrt{d}) \) where \( B(p) \) is a Bernoulli variable with parameter \( p = 0.8 \) and \( U \sim U(0, 1) \) is a standard uniform variable. The multiplication by this uniform variable allows to work on continuous data \( X \) whereas the experiments in (Chen et al., 2023) consider discrete values. Figure 7 below reports the results of additional experiments obtained over 100 independent runs, where we plot the mean squared error curves against the privacy budget \( \varepsilon \) in dimension \( d = 500 \) with different number of clients \( n \in \{250; 500; 1000\} \).

![Figure 7: Mean Squared Error for CSGM and SIGM with \( n \in \{250; 500; 1000\} \) and \( d = 500 \).](image-url)
**Less-Trusted Server.** The Distributed Discrete Gaussian (DDG) mechanism (Kairouz et al., 2021a) can leverage SecAgg to achieve DP guarantees against the server, which is a stronger setting than that of the less-trusted server. However, in practical implementation, it often requires a much higher number of bits per coordinate. The DDG mechanism relies on the discrete Gaussian distribution (Canonne et al., 2020) $\mathcal{N}_\mathbb{Z}(\mu, \sigma^2)$ with $\mu, \sigma \in \mathbb{R}$ and $\sigma > 0$. For any $x \in \mathbb{Z}$, $\mathbb{P}(X = x)$ is proportional to $\exp(- (x - \mu)^2 / (2\sigma^2))$. Using this discrete Gaussian noise, the DDG mechanism works as follows. The input parameters are: scaling factor $\gamma$; clipping threshold $c > 0$; bias $\beta \geq 0$; modulus $m \in \mathbb{N}$ and noise scale $\sigma^2$. The shared randomness is composed of a random unitary matrix $U \in \mathbb{C}^{d \times d}$. Each client $i \in [n]$ first scales and clips the data as $x'_i = \gamma^{-1} \min \{1, c / ||x_i||_2\} : x_i$ then rotate $x''_i = Ux'_i$ and quantize $x''_i$ to $M_i$ such that $||M_i||_2 \leq C(\gamma, \beta, c)$. The worker finally sends to the server $\tilde{M}_i = M_i + G_i \mod m$, where $G_i \sim \mathcal{N}_\mathbb{Z}(0, \sigma^2)$ using SecAgg. The server receives the results of SecAgg $\tilde{M} = \sum_i \tilde{M}_i \mod m$ and outputs $(\gamma/n)U^*M$ (rescale and inverse the rotation). For more details, refer to Algorithms 1 and 2 in Kairouz et al. (2021a). Note that some recent work of Chen et al. (2022) aims at improving communication efficiency of DDG by applying projections to reduce the dimension of the data transmitted. However, it still uses DDG as a subroutine so we restrict our comparison to DDG.

To compare with the utility-communication trade-off of DDG, we adapt the experiments from Kairouz et al. (2021a). On the one hand, we reproduce the figures comparing the utility of DDG with different number of bits against the standard Gaussian mechanism. On the other hand, using Elias gamma coding, we measure over 50 runs the average number of bits needed for the aggregate Gaussian mechanism and the shifted layered quantizer to match the Gaussian mechanism. Note that only the aggregate Gaussian mechanism is compatible with SecAgg. Figures 8 and 9 below presents the mean squared error curves for different number of clients $n \in \{100; 500; 1000\}$ and highlights that AINQ mechanisms requires much lower number of bits to realize the Gaussian mechanism. The graphs in Figure 9 compare the number of bits per client for the shifted layered quantizer (fixed or variable length) and the aggregate Gaussian mechanism, for a privacy budget $\varepsilon \in [1, 10]$.

![Figure 8: Mean Squared Error for the DDG mechanism and Aggregate Gaussian mechanism for $n \in \{100; 500; 1000\}$, $\delta = 10^{-5}$ and $d = 75$. Data samples drawn from the $\ell_2$ sphere with radius $c = 10$.](image)

![Figure 9: Number of bits per clients for the Aggregation Gaussian mechanism (left), shifted layered quantizer with fixed (center) or variable (right) length and different client size $n \in \{20; 100; 500; 2000; 5000\}$.](image)
C.2 Langevin Dynamics

C.2.1 Quantized Langevin Stochastic Dynamics

We consider the same FL framework as Vono et al. (2022) where the goal is to perform Bayesian inference on a parameter \( \theta \in \mathbb{R}^d \) with respect to a dataset \( D \). The posterior distribution is assumed to admit a product-form density with respect to the \( d \)-dimensional Lebesgue measure in the form

\[
\pi(\theta|D) = C_{\pi}^{-1} \prod_{i=1}^{n} e^{-U_i(\theta)}, \quad C_{\pi} = \int_{\mathbb{R}^d} \prod_{i=1}^{n} e^{-U_i(\theta)} d\theta
\]

where \( U_i \) are clients’ potential functions. Using a sequence \((H_k)_{k \in \mathbb{N}}\) of unbiased estimates of \( \nabla U = \sum_{i=1}^{n} U_i \), the Langevin dynamics with stochastic gradient aims at sampling from a target distribution with density \( \pi \). Starting from \( \theta_0 \in \mathbb{R}^d \), it is a Markov chain defined by the update rule:

\[
\theta_{k+1} = \theta_k - \gamma H_{k+1}(\theta_k) + \sqrt{2\gamma} Z_{k+1}
\]

where \( \gamma > 0 \) is a discretization stepsize and \((Z_k)_{k \in \mathbb{N}}\) is a sequence of i.i.d. standard Gaussian random variables.

In the framework of Vono et al. (2022), at iteration \( k \), a subset \( A_{k+1} \subset [n] \) of clients is selected. Each client \( i \in A_{k+1} \) computes an estimate of \( H_{k+1,i} \) and sends \( \mathcal{C}(H_{k+1,i} - \hat{H}_{k+1,i}) \) to the server where \( \mathcal{C} \) is a compression operator and \( \hat{H}_{k+1,i} \) is a variance reduction term. The server aggregates the clients gradients estimators, potentially compensates for the variance reduction, and updates the process. Throughout the remaining of this subsection, we consider \( \mathcal{C} \) to be a shifted layered quantizer with a fixed number of bits as described at the beginning of Appendix C and returning \( x + \mathcal{N}(0, \sigma_b^2 \|x\|_\infty^2) \), \( \sigma_b^2 \|x\|_\infty^2 \leftarrow \mathcal{C}(x) \).

C.2.2 Experiments on Gaussian

We adapt this experiment from Vono et al. (2022). We use \( n = 20 \) clients with data dimension \( d = 50 \) and local potentials \( U_i(\theta) = \sum_{j=1}^{N_i} \|\theta - y_{i,j}\|^2/2 \) where \( (y_{i,j}) \) is a set of synthetic independent but not identically distributed observations across clients and \( N_i = 50 \). The data is generated by sampling \( y_{i,j} \sim \mathcal{N}(\mu_i, I_d) \) with \( \mu_i \sim \mathcal{N}(0, 25I_d) \). We set the discretization stepsize \( \gamma = 5 \times 10^{-4} \) and use full participation and full batch. We adapt the QLSD* algorithm from Vono et al. (2022) to leverage the error of compression in the Langevin dynamics. If the error of compression is smaller than the required error, then the server adds additional noise. Otherwise, the server adds no additional noise to the dynamics. The theoretical analysis in Vono et al. (2022) still holds with this adaptation as it still satisfies the required assumptions on the compression operator (refer to their assumption H3).

![Figure 10](https://via.placeholder.com/150)

**Figure 10:** MSE of different methods, b refers to the number of bits, QLSD* to QLSD* with unbiased quantization, QLSD*-MS to the algorithm with shifted layered quantization, LSD to the algorithm with no compression

Figure 10 above shows the behavior of the mean squared error (MSE) between the true parameter and the sampling based estimation. We start sampling after \( 4.5 \times 10^5 \) iteration to be sure the chain converge to a stationary distribution. The results were computed using 30 independent runs of each algorithm. Observe the great performance of shifted layered quantizers schemes. Indeed, all schemes using standard unbiased quantization performs worse than all schemes with shifted layered quantizers.
Algorithm 6 QLSD$^\star$ with Shifted Layered Quantizer

<table>
<thead>
<tr>
<th>Input:</th>
<th>minibatch sizes ${n_i}_{i \in [b]}$, number of iterations $K$, compression operators $\mathcal{C}$, step-size $\gamma \in (0,\bar{\gamma}]$ with $\bar{\gamma} &gt; 0$ and initial point $\theta_0$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $k = 0$ to $K - 1$ do</td>
<td></td>
</tr>
<tr>
<td>for $i \in A_{k+1}$ do</td>
<td></td>
</tr>
<tr>
<td>Draw $\sigma_{k+1}^{(i)} \sim \text{Uniform}\left(\mathcal{P}_{N_i,n_i}\right)$.</td>
<td></td>
</tr>
<tr>
<td>Set $H_{k+1}^{(i)}(\theta_k) = (N_i/n_i) \sum_{j \in S_{k+1}^{(i)}} \left[\nabla U_{i,j}(\theta_k) - \nabla U_{i,j}(\theta^*)\right]$.</td>
<td></td>
</tr>
<tr>
<td>Compute $g_{i,k+1}, v_{i,k+1} \leftarrow \mathcal{C}\left(H_{k+1}^{(i)}(\theta_k)\right)$.</td>
<td></td>
</tr>
<tr>
<td>Send $g_{i,k+1}, v_{i,k+1}$ to the central server.</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>Compute $g_{k+1} = \frac{b}{</td>
<td>A_{k+1}</td>
</tr>
<tr>
<td>Draw $Z_{k+1} \sim \mathcal{N}(0,d, I_d)$.</td>
<td></td>
</tr>
<tr>
<td>Compute $\beta^2 = \max\left(0, 2\gamma - \frac{b^2 \sigma^2}{</td>
<td>A_{k+1}</td>
</tr>
<tr>
<td>Compute $\theta_{k+1} = \theta_k - \gamma g_{k+1} + \beta Z_{k+1}$.</td>
<td></td>
</tr>
<tr>
<td>Send $\theta_{k+1}$ to the $b$ clients.</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>Output:</td>
<td>samples ${\theta_k}_{k=0}^K$.</td>
</tr>
</tbody>
</table>

D Compression for Randomized Smoothing in Federated Learning

In this section, we show how the proposed quantizers with exact error distribution can be applied to obtain optimal algorithms for non-smooth distributed optimization problems (Scaman et al., 2018). In particular, we highlight the link between compression and randomized smoothing (Duchi et al., 2012). In the framework of federated learning with bi-directional compression (Philippenko and Dieuleveut, 2021), it allows us to derive fast convergence rates for non-smooth objectives. To the best of found knowledge, such rates for federated learning with client compression on non-smooth objectives are novel.

Federated Learning with double compression. Consider some optimization problems of the form

$$\min_{\theta \in \mathbb{R}^d} \left\{ f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta) \right\}$$

where $f_i$ are local losses and $f$ is a convex and potentially non-smooth objective function. This distributed setting includes FL problems with $\ell_1$ norms used as regularizers to ensure sparsity, e.g. $f(\theta) = \frac{1}{n} \|A\theta - b\|_1 = \frac{1}{n} \sum_{i=1}^n |a_i^\top \theta - b_i|$ with $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$ but also neural networks with ReLU activation function.

In the standard setting of Federated Averaging with bi-directional compression, the model parameter $\theta_k$ at time step $k \in \mathbb{N}$ is updated as

$$\theta_{k+1} = \theta_k - \gamma g_{i,k} \left(\frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(g_i(\theta_k))\right)$$

where $\gamma > 0$ is the learning rate, $\mathcal{C}_i$ (resp. $\mathcal{C}_i^\dagger$) is the up-link/client compressor (resp. the down-link/server) and $g_i(\theta_k)$ is a subgradient of the local loss $f_i$ such that $\mathbb{E}[g_i(\theta_k)] \in \partial f_i(\theta_k)$.

Distributed Randomized Smoothing. Fast rates for non-smooth optimization problems can be attained using the smoothing approach of Duchi et al. (2012); Scaman et al. (2018). For $\sigma > 0$, denote by $f_\sigma$ the smoothed version of $f$ defined by

$$f_\sigma(\theta) = \mathbb{E}_\xi[f(\theta + \sigma \xi)]$$

where $\xi \sim \mathcal{N}(0, I_d)$ is a standard Gaussian random variable. If $f$ is $L$-Lipschitz then $f_\sigma$ is $(L/\sigma)$ smooth and it holds (see Lemma E.3 in Duchi et al. (2012))

$$\forall \theta \in \mathbb{R}^d, \quad f(\theta) \leq f_\sigma(\theta) \leq f(\theta) + \sigma L \sqrt{d}.$$
Thus, accelerated optimization algorithms such as Distributed Randomized Smoothing (DRS) can be applied by using the \textit{smoothed} version of $f$. These algorithms rely on approximating the smoothed gradient $\nabla f_\sigma = \frac{1}{n} \sum_{i=1}^{n} \nabla f_{i,\sigma}$ with sampled subgradients of the form $g_i(\theta + \sigma \xi_j)$ with $j = 1, \ldots, m$. Each local client $i = 1, \ldots, n$ can sample standard Gaussian variables then compute $\hat{g}_i(\theta) = \frac{1}{m} \sum_{j=1}^{m} g_i(\theta + \sigma \xi_j)$ and send it to the server which aggregates the subgradients. Interestingly, the sampling steps may be replaced with compressors that produce exact error distribution.

\textbf{Quantizers act as randomized smoothing.} The idea of sampling a random perturbation $\xi$ to evaluate the subgradients as $g_i(\theta + \sigma \xi)$ can be replaced by first compressing the model parameter $\theta$ with a Gaussian error distribution as $\mathcal{F}(\theta) = \theta + \sigma \xi$ and then evaluating the subgradients at compressed point as $g_i(\mathcal{F}(\theta))$. Similarly to Philippenko and Dieuleveut (2021), let us consider the update rules where the subgradients are evaluated at \textit{perturbed} point as

$$
\theta_{k+1} = \theta_k - \frac{\gamma}{n} \sum_{i=1}^{n} \hat{g}_i(\bar{\theta}_k); \quad \bar{\theta}_k = \mathcal{F}(\theta_k) = \theta_k + \xi_k,
$$

with a Gaussian error $\xi_k = \mathcal{F}(\theta_k) - \theta_k \sim \mathcal{N}(0, \sigma^2 I_d)$. In view of approximating the sampling scheme of DRS, the local clients can perform $m$ local compressions so that $\hat{g}_i(\bar{\theta}_k) = \frac{1}{m} \sum_{j=1}^{m} g_i(\theta_k + \sigma \xi_j)$ which gives an unbiased estimate of the smoothed gradient as $E[\hat{g}_i(\bar{\theta}_k)|\theta_k] = \nabla f_{i,\sigma}$. Thus, one can exactly recover the Distributed Randomized Smoothing algorithm of Scaman et al. (2018) and the optimal convergence rates of Theorem 1 therein.