术语解释

sum estimation algorithm：是在不直接访问所有数据元素的前提下，对一组数据的总和（sum）进行**快速估计**的算法。

作业部分

算法设计

> 参考5 Our Protocol中的内容
>
> 在我们的设置中，每个用户 i 持有整数 x_i，属于{0} ∪ [U]（假设 U 是一个 2 的幂）。**高级思想是将域"划分"成指数增长的区间，在每个分区上独立运行基本的私有求和（BaseSumDP 子协议）**，然后**使用带噪声的估计来确定一个在真实最大值常数因子内的剪切阈值τ。**最后，**分析器对分区求和直到τ，以产生最终估计。**这产生了实例最优误差（大约 $O(Max(D) \cdot log(logU)/\varepsilon)$），而每个用户只发送 1+o(1)条消息。

Preliminary

BaseSumDP

BaseSumDP is the fundamental sub-protocol used for private summation in the shuffle model. Its role is to enable users to contribute to a global sum with strong differential privacy guarantees while keeping communication costs extremely low (ideally, each user sends only $1 + o(1)$ messages). The key ideas underlying BaseSumDP are as follows:

1. **Input Processing and Clipping:**
   Each user holds an input value x_i and first ensures that this value lies within a known, bounded domain by applying a clipping (or randomized rounding) step. This clipping limits the sensitivity of the sum query by restricting the impact any single user can have on the output.
2. **Noise Addition:**
   To guarantee differential privacy, BaseSumDP adds noise drawn from a Laplace distribution. The noise scale is carefully calibrated to the sensitivity of the input after clipping. In practice, when a user's input lies in a subdomain with an upper bound (for example, for inputs in the interval
   $$I_j = \left[2^{j-1} + 1,\ 2^j\right],$$
   the noise added is scaled roughly as $2^j/\varepsilon$). This calibration ensures that the change in any single user's input (which is bounded by 2^j in that subdomain) does not significantly affect the output distribution.
3. **Communication Efficiency:**
   BaseSumDP is specifically designed for the shuffle model, where users send messages that are later shuffled by a trusted shuffler. The protocol is engineered so that each user sends only a small number of messages—on average $1 + o(1)$ per user. This low communication cost is achieved by carefully combining the true (clipped) value with additional noise messages, such that the aggregate noise cancels appropriately during analysis.
4. **Aggregation at the Analyzer:**
   After the shuffler collects the messages, the analyzer groups them by subdomain and sums the contributions. BaseSumDP ensures that despite the added noise, the aggregated sum in each subdomain remains an accurate approximation of the true sum over that subdomain. The noise levels are such that,

when combined across subdomains, the overall error remains low, particularly when the true inputs are much smaller than the worst-case bound U.

BaseSumDP enables private summation by combining input clipping, carefully calibrated noise addition, and efficient communication. It forms the core building block in our SumDP protocol by allowing the overall domain to be partitioned into exponentially growing intervals, with each interval being handled independently. This design is key to achieving an instance-optimal error bound—roughly

$$O\Big(\frac{\mathrm{Max}(D)\cdot \ln(\log U)}{\varepsilon}\Big)$$

—while ensuring that each user sends only $1+o(1)$ messages.

SumDP Algorithm Description

In our setting, each user i holds an integer $x_i \in \{0\}\cup[U]$ (with U assumed to be a power of 2). The high-level idea is to "partition" the domain into exponentially growing intervals, run a basic private summation (a BaseSumDP sub-protocol) on each partition independently, and then use the noisy estimates to determine a clipping threshold τ that is within a constant factor of the true maximum in the dataset. Finally, the analyzer sums over the partitions up to τ to produce the final estimate. This yields instance-optimal error (roughly

$$O\Big(\frac{\mathrm{Max}(D)\cdot \ln(\log U)}{\varepsilon}\Big)$$

) while each user sends only $1+o(1)$ messages.

User Side (Randomizer):

1. **Domain Partitioning:**
   Partition the data domain into intervals that are powers of 2. For $j = 0, 1, 2, \ldots, \log_2 U$, define

   $$I_j = \big[2^{j-1}+1,\ 2^j\big],$$

   with the convention that I_0 covers the singleton $\{1\}$.
2. **Assigning Subdomain:**
   Each user i finds the unique j such that $x_i \in I_j$. (If $x_i = 0$, then no message is sent.)
3. **Local Randomization:**
   For the determined j, user i runs a BaseSumDP mechanism on its value (or on the indicator $x_i \cdot I\{x_i \in I_j\}$) using the full privacy budget (ε, δ). This sub-protocol randomizes the value by both clipping and adding Laplace noise scaled to 2^j. The mechanism outputs a (small) set of randomized messages along with the subdomain identifier j.
4. **Communication:**
   The user sends the messages for subdomain j to the shuffler.

Analyzer Side:

1. **Aggregation:**
   For each subdomain j, aggregate the messages coming from all users to compute a noisy estimate

   $$\widetilde{Sum}(I_j) \approx \sum_{i:x_i\in I_j} x_i.$$

   (Because the maximum value in I_j is 2^j, the Laplace noise added is roughly scaled as $2^j/\varepsilon$.)

2. **Threshold Selection:**
   To determine an appropriate clipping threshold, scan the subdomains in increasing order and let T_j be a "bar" defined by

$$T_j = 1.3 \cdot \frac{2^j \cdot \ln\left(2(\log_2 U + 1)/\beta\right)}{\varepsilon},$$

   where β is the failure probability. The analyzer finds the last index j^* such that

$$\widetilde{Sum}(I_{j^*}) > T_{j^*},$$

   and then sets $\tau = 2^{j^*}$. By the design of T_j, with high probability τ satisfies

$$\mathrm{Max}(D) \leq \tau \leq 2 \cdot \mathrm{Max}(D).$$

3. **Final Sum Estimation:**
   Output the final estimate as

$$\widetilde{Sum}(D) = \sum_{j=0}^{j^*} \widetilde{Sum}(I_j).$$

   This summation corresponds to estimating the sum over all values that are at most τ (which—by the above selection—covers almost all the true data without overshooting by too much).

Complete Differential Privacy Proof

Setup:
Let

$$D = (x_1, x_2, \ldots, x_n)$$

and

$$D' = (x_1, \ldots, x_i', \ldots, x_n)$$

be any two neighboring datasets differing only in the i-th user's value. In our protocol, each user i determines the unique subdomain I_j (for some $j \in \{0, 1, \ldots, \log_2 U\}$) in which x_i lies and then runs a randomized algorithm A_j (the BaseSumDP instance for I_j) on that value. The mechanism's overall output is produced by aggregating the messages from all users and then performing a deterministic post-processing step (threshold selection and summation).

Step 1. Privacy of the Local Randomizer

For a given subdomain I_j defined as

$$I_j = [2^{j-1} + 1, 2^j] \quad (\text{for } j \geq 1, \text{ with a suitable definition for } j = 0),$$

the randomized algorithm A_j is defined as follows:

- It first clips the user's input to ensure it lies in I_j.
- It then adds noise drawn from a Laplace distribution with scale proportional to $2^j/\varepsilon$.

Because the maximum possible change in any input in I_j is at most 2^j, adding Laplace noise with scale $2^j/\varepsilon$ ensures that for any two inputs x and x' within I_j, the probability density functions satisfy

$$\Pr[A_j(x) \in S] \leq e^{\varepsilon} \cdot \Pr[A_j(x') \in S]$$

for every measurable set S (up to an additional δ term if further randomness is used). In our construction, even if the algorithm uses a more general noise mechanism that introduces an additive slack δ, the guarantee is that

$$\Pr[A_j(x) \in S] \leq e^{\varepsilon} \cdot \Pr[A_j(x') \in S] + \delta.$$

Thus, each instance A_j is (ε, δ)-differentially private.

Notice that when a user's value changes from x_i to x_i', one of two cases occurs:

1. **Same Subdomain:** Both x_i and x_i' lie in the same subdomain I_j. In this case, the change is bounded by 2^j, and the privacy guarantee of A_j directly ensures that the output distribution changes by at most a factor of e^{ε} (plus δ).
2. **Different Subdomains:** The value changes subdomains (say from I_j to I_k). In this case, the user's output shifts from being generated by A_j to being generated by A_k. However, since every user participates in exactly one instance and the worst-case change in output is still controlled by the mechanism's guarantee in the respective subdomain, the change in the overall output distribution is again bounded by (ε, δ).

Step 2. Parallel Composition

Each user's data is used in exactly one subdomain—there is no overlap between the subdomains. Thus, the outputs for different subdomains are produced by mechanisms that act on disjoint parts of the input. The parallel composition property of differential privacy states that if a user's data affects only one of several disjoint mechanisms (each with a guarantee of (ε, δ)-DP), then the combined mechanism still satisfies (ε, δ)-DP. Therefore, the collection of outputs $\{A_j(x_i)\}$ for all i (with each user contributing only to one A_j) is (ε, δ)-differentially private.

Step 3. Post-Processing Invariance

After the local randomization, the analyzer aggregates the messages to compute a noisy sum for each subdomain and then applies a deterministic rule to select a threshold τ and sum the estimates for subdomains up to τ. Because differential privacy is immune to post-processing, any further computation on the (ε, δ)-DP outputs cannot degrade the privacy guarantee. Hence, the final output of the analyzer remains (ε, δ)-differentially private.

Conclusion

For every pair of neighboring datasets D and D', the change in the distribution over the final output of our protocol is governed solely by the single BaseSumDP instance in which the differing user participates. By the privacy guarantees of that instance, along with the disjoint nature of the subdomains (parallel composition) and the post-processing invariance, we conclude that

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

for every measurable set S. Thus, the overall SumDP protocol satisfies (ε, δ)-differential privacy.

Error Bound Analysis

The protocol partitions the domain into intervals that grow exponentially. For each interval indexed by j, the mechanism runs a BaseSumDP instance on the subdomain

$$I_j = [2^{j-1} + 1, 2^j]$$

so that every nonzero x_i falls into exactly one such interval. In each BaseSumDP instance, the value is clipped (if needed) and noise drawn from a Laplace distribution with scale proportional to $2^j/\varepsilon$ is added. In our analysis, we can show that with probability at least $1 - \beta$ (over the randomness in the noise and the union over all $O(\log U)$ intervals), the following holds for every j:

$$\left| \widetilde{Sum}(I_j) - \sum_{i:\, x_i \in I_j} x_i \right| \leq 1.3 \cdot \frac{2^j \cdot \ln\big(2(\log U + 1)/\beta\big)}{\varepsilon}.$$

Let j^* be the largest index such that the noisy sum $\widetilde{Sum}(I_{j^*})$ exceeds the threshold

$$T_j = 1.3 \cdot \frac{2^j \cdot \ln\big(2(\log U + 1)/\beta\big)}{\varepsilon}.$$

Then τ is set to 2^{j^*}. Because the true maximum value $\mathrm{Max}(D)$ lies in the interval I_{j^*} (or possibly in I_{j^*+1}), it follows that τ is within a constant factor of $\mathrm{Max}(D)$; in fact, one can show that

$$\mathrm{Max}(D) \leq \tau \leq 2 \cdot \mathrm{Max}(D).$$

Since the final estimate is formed by summing the noisy sums over all intervals $j \leq j^*$, the total error is dominated by the noise added in each interval. The noise levels form a geometric series; therefore, their sum is bounded by a constant multiple of the noise in the highest interval. In particular, the overall error is bounded by

$$\mathrm{Error} \leq O\Big(\frac{2^{j^*} \cdot \ln\big(2(\log U + 1)/\beta\big)}{\varepsilon}\Big).$$

Since 2^{j^*} is $\Theta(\mathrm{Max}(D))$, we obtain an error bound of

$$\mathrm{Error} = O\Big(\frac{\mathrm{Max}(D) \cdot \ln(\log U/\beta)}{\varepsilon}\Big).$$

More concretely, if we expand the constant factors from each step, the error is at most roughly

$$\Big(1.3 \cdot \ln\big(2(\log U + 1)/\beta\big) + 1.3 \cdot C_{\mathrm{geo}}\Big) \cdot \frac{\mathrm{Max}(D)}{\varepsilon},$$

where C_{geo} is the (small) constant from summing the geometric series of noise contributions (typically at most 1–2). In many settings (with β constant), this error can be summarized as

$$\mathrm{Error} \approx \frac{2.6 \cdot \ln(2 \log U) \cdot \mathrm{Max}(D)}{\varepsilon}.$$

Thus, the protocol achieves an instance-optimal error that scales with $\mathrm{Max}(D)$ and grows only double-logarithmically in U (up to a logarithmic factor in $1/\beta$), rather than with U itself.

Computational Complexity Analysis

User-Side Computation:

1. **Interval Determination:**
   Each user must determine in which interval I_j their input lies. Since the intervals are of the form
   $$[2^{j-1}+1, 2^j],$$
   this requires computing $\log_2(x_i)$ (or performing a binary search over $j \in \{0, \ldots, \log U\}$), which takes $O(\log U)$ time.
2. **Running BaseSumDP:**
   In the chosen subdomain, the user executes the BaseSumDP randomizer. This step involves (i) possibly clipping the value and (ii) adding noise sampled from a Laplace distribution with scale proportional to $2^j/\varepsilon$. In practice, the running time of the BaseSumDP randomizer is $O(\min(U, \sqrt{n}))$; in many common regimes U is very large but $\sqrt{n}$ is much smaller, so one may expect $O(\sqrt{n})$ time per user in the worst case. However, since each user only participates in one instance, the overall user-side cost is
   $$O(\log U + \min(U, \sqrt{n})).$$
3. **Message Encoding:**
   Each message consists of a few $O(1)$-sized numbers (e.g., the randomized value and a subdomain label). The encoding of each message takes $O(1)$ time, and in expectation each user sends $1 + o(1)$ messages.

Analyzer-Side Computation:

1. **Aggregation:**
   The analyzer receives messages tagged with subdomain indices. There are $O(\log U)$ intervals. The analyzer aggregates the messages for each subdomain by summing the contributions. With n users, the total number of messages is $O(n)$ (since each user sends $1 + o(1)$ messages in expectation). Thus, the summation step runs in $O(n)$ time.
2. **Threshold Selection and Final Summation:**
   The analyzer scans through the $O(\log U)$ noisy sums to select the threshold τ and then sums over the selected intervals. This process takes $O(\log U)$ time, which is negligible compared to the $O(n)$ aggregation.
3. **Total Analyzer Complexity:**
   Overall, the analyzer's computational complexity is $O(n + \log U) = O(n)$.

Result

- **Error Bound:**
  The error is at most
  $$O\Big(\frac{\mathrm{Max}(D) \cdot \ln(\log U/\beta)}{\varepsilon}\Big).$$
  In concrete terms, one may expect a constant factor around 2.6 (including the 1.3 factor from each subdomain and the geometric series sum) multiplied by $\ln\big(2(\log U + 1)/\beta\big)$, so that the final error is approximately

$$\frac{2.6 \cdot \ln(2 \log U) \cdot \mathrm{Max}(D)}{\varepsilon}$$

when β is a constant.

- **User-Side Complexity:**
  Each user performs $O(\log U)$ work to determine the interval and runs the BaseSumDP randomizer in $O(\min(U, \sqrt{n}))$ time. In typical regimes, this results in an overall user-side cost of $O(\sqrt{n})$ (assuming U is very large relative to $\sqrt{n}$) plus $O(\log U)$ overhead.
- **Analyzer-Side Complexity:**
  The analyzer aggregates $O(n)$ messages and processes $O(\log U)$ intervals, resulting in an overall time of $O(n)$.