arXiv:2303.04123v1 [cs.IT] 7 Mar 2023

面向存储高效且采用 Top r 稀疏化的联邦学习的可控信息泄露私有读取-更新-写入

Sajani Vithana Sennur Ulukus

电气与计算机工程系 马里兰大学帕克分校，马里兰州 20742 spallego@umd.edu
ulukus@umd.edu

摘要

在联邦学习（FL）中，大量用户利用本地设备上的私有数据共同训练机器学习（ML）模型。在联邦学习中采用 Top r稀疏化时，用户仅上传最重要的 r 比例的更新，而服务器仅向用户发送最重要的 r' 比例的参数，以降低通信成本。然而，稀疏更新的数值和索引会泄露用户私有数据的信息。本文考虑这样一种联邦学习设置：N 个非串通数据库存储待训练模型，用户从中下载并私密地更新稀疏参数，而不向数据库泄露更新的数值或索引。我们提出了四种具有不同特性的方案来完成这一任务，同时实现最低通信成本，并证明稀疏更新的数值和位置均可得到信息论隐私保障。不过，这需要付出相当高的存储成本。为降低存储成本，我们通过模型分割机制对这些方案进行推广，以一定的信息泄露为代价减少存储成本。总体而言，我们给出了采用 Top r 稀疏化的隐私联邦学习中通信成本、存储成本与信息泄露之间的权衡。

1 引言

联邦学习（FL） [1, 2] 是一种广泛使用的分布式学习技术，一组用户通过各自的设备，利用自己的本地数据远程训练机器学习模型，并仅与中央服务器共享梯度更新。这降低了数据提供者面临的隐私泄露风险，同时也分散了中央服务器的算力需求。然而，研究表明，用户共享的梯度可用于获取有关

用户隐私数据的信息 [3–9]。安全聚合 [10], 等加密协议、通过噪声添加实现的差分隐私（DP） [11] 、数据采样和数据洗牌（例如 [12–24]）都是用于减少联邦学习中此类信息泄露的方法。然而，这些方法无法保证每个用户本地数据的信息论隐私。

除了隐私泄露之外，联邦学习的另一个缺点是，在多个轮次中与数百万用户共享模型参数和更新会产生高昂的通信成本。针对这一问题的解决方案包括梯度量化 [25–28], 、联邦子模型学习（FSL） [29–37], 以及梯度稀疏化 [38–45]。在梯度量化中，梯度数值经过量化后，用更少的比特表示。在联邦子模型学习中，根据用于训练整个模型的不同类型的数据，将机器学习模型划分为多个子模型；每个用户只下载并更新可由其本地数据更新的子模型。在梯度稀疏化中，用户只通信一组选定的梯度和参数，而不通信所有梯度更新和参数。通常，稀疏化率（通信的参数/更新所占比例）约为 10^{-2} 至 10^{-3}，从而显著降低通信成本。

Top r 稀疏化是一种广泛使用的稀疏化技术，用户与中央服务器之间只共享最显著的 r 比例的参数/更新。在某些情况下，研究表明，top r 稀疏化的表现优于传统联邦学习。然而，稀疏更新的数值及其位置（索引）都会泄露用户本地数据的信息。请注意，稀疏更新的位置会泄露对于特定用户而言最重要和最不重要的参数集合，这些信息可用于推断用户的私有数据。因此，为了保障参与稀疏联邦学习过程的用户隐私，需要对以下两项进行保密：1）稀疏更新的数值；2）稀疏更新的索引。在本研究中，我们设计了多种方案，在采用 top r 稀疏化的联邦学习中实现用户-数据库通信，同时确保稀疏更新的数值和索引具有信息论隐私。

我们考虑一种联邦学习设置：多个非串通数据库存储机器学习模型，且单个用户与每个数据库通信，如图 1 所示。本文提出的方案基于置换技术：协调器对参数集合进行随机置换初始化，并将其发送给用户。随后，协调器在每个数据库中放置加噪置换逆矩阵，使数据库无法获知底层置换。用户与数据库之间的所有通信均以置换后的索引为依据，从而确保稀疏更新索引的隐私。[1] 不过，在加噪置换逆矩阵的辅助下，每个数据库中的参数仍按正确顺序更新。这种方法的主要缺点是置换逆矩阵较大，导致存储成本显著增加。为此，我们提出了一些方案，通过减小加噪置换逆矩阵的规模来降低存储成本，

[1]第4.2节提供了关于隐私/信息泄露的严格证明。

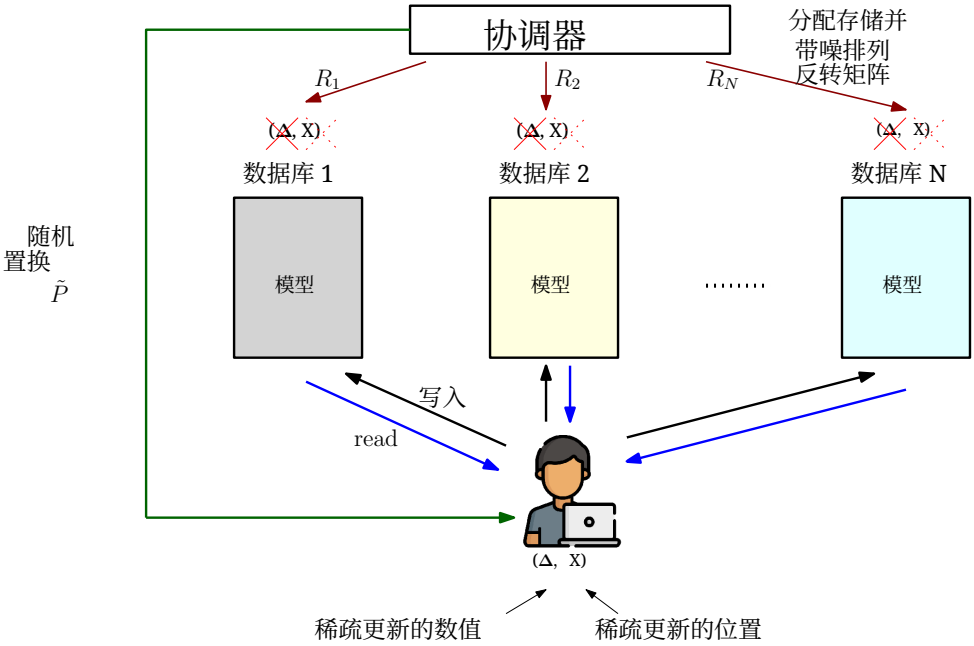


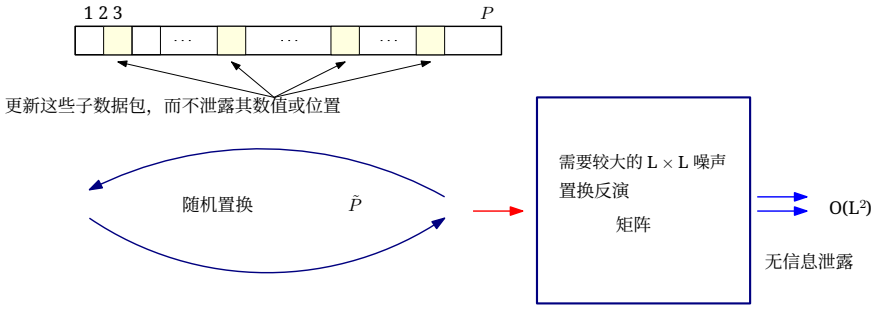
图1：系统模型：用户读取（下载）、更新并写入（上传）机器学习模型。

代价是允许一定程度的信息泄露。具体做法是将机器学习模型划分为多个分段，并在每个分段内进行置换。如图 2 所示。分段数量根据允许的信息泄露程度和数据库的存储容量确定。

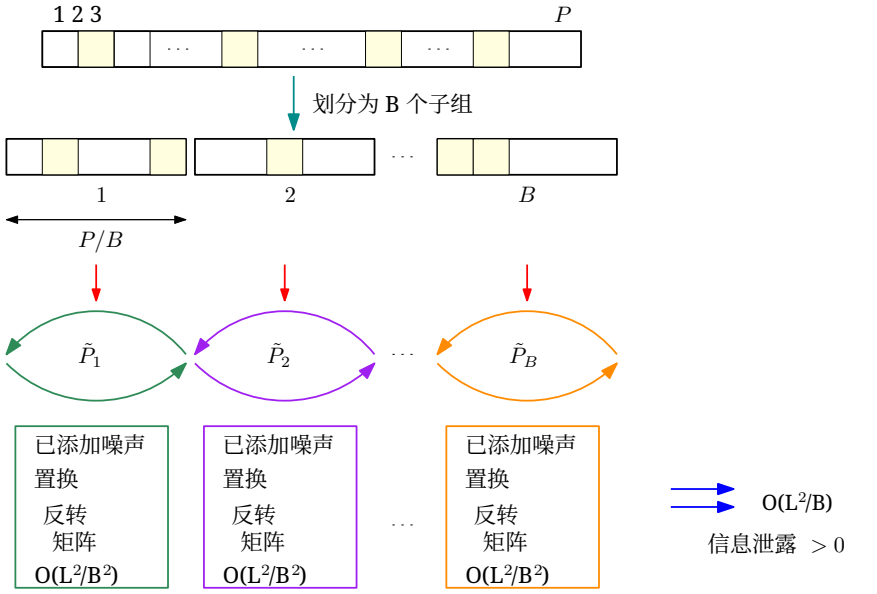
本文提出四种方案，用于在采用 top r 稀疏化的隐私联邦学习中实现用户与数据库之间的通信。这些方案具有不同特性，例如更低的通信成本、更低的存储成本或更少的信息泄露。四种方案的差异取决于所采用的存储结构（MDS编码存储或未编码存储）和置换机制（仅段内置换，或段内置换与段间置换）。MDS编码存储能够降低存储成本，但会增加通信成本；与单阶段置换（仅段内置换）相比，两阶段置换（段内置换与段间置换）可显著减少信息泄露，同时仅略微增加通信成本。根据给定联邦学习任务的具体要求和限制，可以选择最合适的方案。总体而言，我们展示了采用稀疏化的隐私联邦学习中通信成本、存储复杂度与信息泄露之间的权衡关系。

2 问题形式化

我们考虑一种联邦学习设置，其中一个由 L 个参数组成的机器学习模型存储在 N 个非串通数据库中，这些参数属于P 个子数据包。参数取自一个足够大的有限域 $\mathbb{F}_q$。在给定时间 t ，给定用户从数据库读取（下载）模型所需的参数，使用该用户的本地数据训练模型，



(a) 未进行分段。



(b) 使用分段。

图 2：置换技术中采用分段的动机：(a) 不进行分段，对整个模型进行置换。(b) 进行分段后，在各分段内进行置换。

并将更新中最重要的 r 部分写入（上传）回所有数据库。在这项工作中，我们同时考虑上行链路和下行链路的稀疏化，以降低通信成本。具体而言，读取（下行链路）和写入（上行链路）阶段的稀疏化率分别为 r' 和 r。换言之，在读取（下载）阶段，用户仅下载由数据库确定的一组选定的 Pr' 个子数据包。[2] 在本地训练模型后，每个用户在写入（上传）阶段仅向数据库上传更新中最重要的 Pr 组（对应的子数据包）。[3]

请注意，用户在读取阶段不会向数据库发送任何信息。因此，在读取阶段，用户的本地数据不会泄露给数据库。用户会在

[2]这些子数据包可以由数据库根据上一个时间步接收到的稀疏更新确定，也可以通过任何其他下行链路稀疏化协议确定。例如，数据库可以选择时间 $t-1$ 的写入阶段中更新最频繁的 Pr' 个子数据包，并在时间 t 的读取阶段将其发送给用户。[3]我们假设，写入阶段中稀疏集合内的所有 Pr 个子数据包的数值均不为零。

写入阶段向数据库发送稀疏更新及其位置（索引）以训练模型。用户的本地数据可能会通过这些更新及其索引泄露给数据库。[4] 在本研究中，我们考虑对稀疏更新的数值和索引提供以下隐私保证。

稀疏更新值的隐私：不允许向任何数据库泄露稀疏更新值的相关信息，即，

$$I(\Delta_i^{[t]}; G_n^{[t]}) = 0, \quad n \in \{1, \ldots, N\}, \quad \forall i \tag{1}$$

其中，$\Delta_i^{[t]}$ 是给定用户在时间 t 的第 i个稀疏（非零）更新值，$G_n^{[t]}$包含用户在时间 t发送给数据库 n 的所有信息。

稀疏更新位置（索引）的隐私：关于稀疏更新索引所泄露的信息量必须控制在给定的隐私泄露预算 ϵ以内，即，

$$I(X^{[t]}; G_n^{[t]}) \leq \epsilon, \quad n \in \{1, \ldots, N\}, \tag{2}$$

其中，$X^{[t]}$ 是给定用户在时间 t更新的稀疏子数据包索引集合。带有隐私约束的系统模型如图1所示。协调器用于初始化联邦学习过程。[5] 除了隐私约束外，为确保方案的可靠性，我们还要求满足以下安全性和正确性条件。

模型安全性：不允许向数据库泄露任何有关模型参数的信息，即，

$$I(W^{[t]}; S_n^{[t]}) = 0, \quad n \in \{1, \ldots, N\}, \tag{3}$$

其中，$W^{[t]}$ 是机器学习模型，$S_n^{[t]}$是数据库 n 在时间 t的数据内容。

读取阶段的正确性：用户应能够从读取阶段的下载内容中，正确解码由下行链路稀疏化协议确定的模型子数据包稀疏集合（记为 J），即

$$H(W_J^{[t-1]} | A_{1:N}^{[t]}) = 0, \tag{4}$$

其中，$W_J^{[t-1]}$是模型 W在时间 $t-1$（更新前）集合 J中的子数据包集合，$A_n^{[t]}$是从数据库 n 在时间 t下载的信息。

写入阶段的正确性：设 J'为在

[4]稀疏更新的位置（索引）会泄露有关对于给定用户而言模型中最重要和最不重要参数的信息，这可能会泄露用户本地数据的信息。[5]协调器仅在初始化阶段可用，联邦学习过程开始后将不再属于系统模型的一部分。

时间 t由给定用户更新的模型中最重要的 个子数据包。模型应正确更新为:

$$W_s^{[t]} = \begin{cases} W_s^{[t-1]} + \Delta_s^{[t]}, & \text{if } s \in J' \\ W_s^{[t-1]}, & \text{if } s \notin J' \end{cases}, \tag{5}$$

其中， $W_s^{[t-1]}$是模型在时间 $t-1$ 的子分组 s， $\Delta_s^{[t]}$是时间 t对应于子分组 s 的更新。

读写成本：读写成本分别定义为 $C_R = \frac{\mathcal{D}}{L}$ 和$C_W = \frac{\mathcal{U}}{L}$，其中 $\mathcal{D}$ 是读取阶段下载的符号总数， $\mathcal{U}$ 是写入阶段上传的符号总数， L 是模型的大小。总成本 C_T是读写成本 $C_T = C_R + C_W$之和。

存储复杂度：存储复杂度以每个数据库中存储的符号总数的阶来衡量。

在本研究中，我们提出了使用 top r 稀疏化进行联邦学习的方案，在满足上述所有隐私、安全性和正确性条件的同时，实现最低的总通信成本和存储复杂度。

3 主要结果

定理 1 考虑一个存储在 N 个非串通数据库中的联邦学习模型，该模型包含 L 个参数，其数值取自有限域 $\mathbb{F}_q$，并被划分为 P 个子数据包。模型被划分为 B 个大小相等（ $1 \le B < P$）的分段，使得每个连续的 $\frac{P}{B}$ 个子数据包构成一个分段。假设用户在每个时间点都对联邦学习模型进行更新，上行和下行稀疏化率（top r 稀疏化）分别为 r 和 r'。令 X_i 为随机变量，表示任意给定用户上传的第 i 个分段中具有稀疏~ ~（非零）更新的子数据包数量；令 $(X_1, \ldots, X_B)$ 为广义向量，表示 $(X_1, \ldots, X_B)$ 的所有不同组合，而不考虑段索引。那么，从单个用户的角度来看，在联邦学习过程的单轮中，表1所示的读取/写入成本、存储复杂度和信息泄露量均可实现。

case	读取成本	写入成本	存储复杂度	信息泄露
1	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{2}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{2}{N}}$	$O(\frac{L^2}{B})$	^ ^$H(X_1, \ldots, X_B)$
2	$\frac{3r'(1+\frac{\log_q P}{N})}{1-\frac{1}{N}}$	$\frac{3r(1+\log_q P)}{1-\frac{1}{N}}$	$O(\frac{L^2}{BN^2})$	^ ^$H(X_1, \ldots, X_B)$
3	$\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{4}{N}}$	$\frac{2r(1+\log_q P)}{1-\frac{4}{N}}$	最大值$\{O(\frac{L^2}{B})\ \ O(N^2B^2)\}$	~ ~$H(X_1, \ldots, X_B)$
4	$\frac{5r'(1+\frac{\log_q P}{N})}{1-\frac{1}{N}}$	$\frac{5r(1+\log_q P)}{1-\frac{1}{N}}$	$\max\{O(\frac{L^2}{N^2B})\ O(B^2)\}$	~ ~$H(X_1, \ldots, X_B)$

表1：可实现的通信成本、存储成本和信息泄露集合。

备注 1 表1中的信息泄露对应于稀疏更新索引泄露的信息量。[6] 对于由 ϵ 给定的稀疏更新索引隐私泄露预算，可通过最小化存储复杂度来计算最优分段数量 B ，使得 $H(X_1, \ldots, X_B) < \epsilon$ 或 $H(X_1, \ldots, X_B) < \epsilon$ 成立（取决于所考虑的情况）。这对四种情况均适用。

ˆ˜注释 2 当 $B = 1$ （不存在分段）时， $X_1 = X_1 = Pr$ 且相应的ˆ˜信息泄露为零，因为 Pr 是固定的，并且 $H(X_1) = H(X_1) = 0$，也就是说，当 $B = 1$时，对应四种情况的四个方案在产生表1所述相同通信成本的同时，实现了对稀疏更新的数值和位置的信息论隐私。然而，在这种情况下，存储成本会增加到 $O(L^2)$ 或 $O\left(\frac{L^2}{N^2}\right)$。

ˆˆ˜ ˜ ˆˆ注释 3 $H(X_1, \ldots, X_B) > H(X_1, \ldots, X_B)$，因为 $H(X_1, \ldots, X_B)$ 考虑了ˆ X_i 的所有可能取值，而 $H(\tilde{X}_1, \ldots, \tilde{X}_B)$ 仅考虑 $\{\hat{X}_i\}_{i=1}^{B}$ 的不同集合。例如，若ˆˆ˜ ˜$B = 2$， $H(X_1, X_2)$ 同时考虑置换 $\{1, 2\}$ 和 $\{2, 1\}$，而 $H(X_1, X_2)$ 只将其中˜ ˜一个纳入考虑，也就是说，与 $H(X_1, \ldots, X_B)$ 相比， $H(X_1, \ldots, X_B)$ 中考虑的概率更加密集且集中。

注4 情况1至4通过同时采用 CSA *[46]* 和置换技术来实现，具体内容详见第4节。情况1和2的方案采用单轮排列技术（仅使用段内置换），而情况3和4采用双轮排列技术（同时使用段内置换和段间置换），从而进一步降低信息泄露。情况3和4分别是在情况1和2的基础上增加一轮置换而扩展得到的。与情况1和2相比，情况3和4的通信成本更高，但信息泄露更少。

注5 四种情况（方案）各有不同的特性。情况1和3的通信成本最低，但由于采用复制存储且使用较大的带噪排列逆矩阵，存储复杂度较高。情况2和4采用MDS编码存储和紧凑排列逆矩阵，以增加通信成本为代价降低存储复杂度。

备注 6 通信成本不取决于分段数量 B .

备注 7 考虑一个示例设置，其中将 $P = 12$ 子数据包划分为 $B = 1$、2、3、4、6 个分段。假设每个子数据包被选入最重要的 $Pr = 3$ 个子数据包集合的概率相同。图3展示了信息泄露随 B的不同取值而变化的情况。

[6]如问题表述中所述，更新数值的信息论隐私得到了保障。

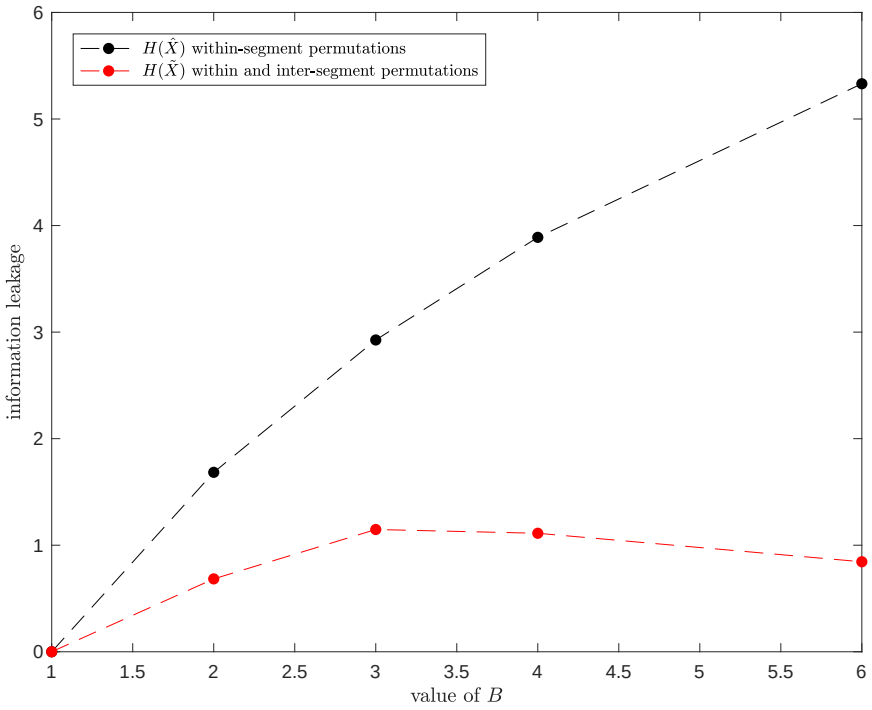


图3： $P = 12$ 与 B取不同数值时示例设置的信息泄露。

4 所提出的方案

4.1 通用方案示例

本节针对四种情况分别给出所提出的方案。在这四种方案中，我们将 P 子数据包划分为 B 个互不重叠的等大小分段，以控制存储成本和信息泄露。参数 B 是一个可根据给定的隐私泄露预算和存储容量限制进行选择的变量。本节针对任意取值的 B、 P、 r 和 r'，给出一般方案。为进一步说明，我们针对四种情况，在给出一般方案的同时也提供示例。对于情况 1 和情况 2 对应的两个示例，我们假设采用相同的设置：将 $P = 15$个子数据包（子分包化为 ℓ）划分为 $B = 3$ 个等大小分段，如图4所示。

情况1：本情况采用未编码[7] 存储和更大的置换逆矩阵，以降低通信成本，但代价是存储成本更高。

初始化：情况1中的单个子分组（子分组 s）存储在数据库 $n, n \in$ 中

[7]尽管式（6）中的模型参数和噪声符号在存储时被组合在一起（编码形式），但每个参数并未与其他参数组合，因此属于未编码存储。

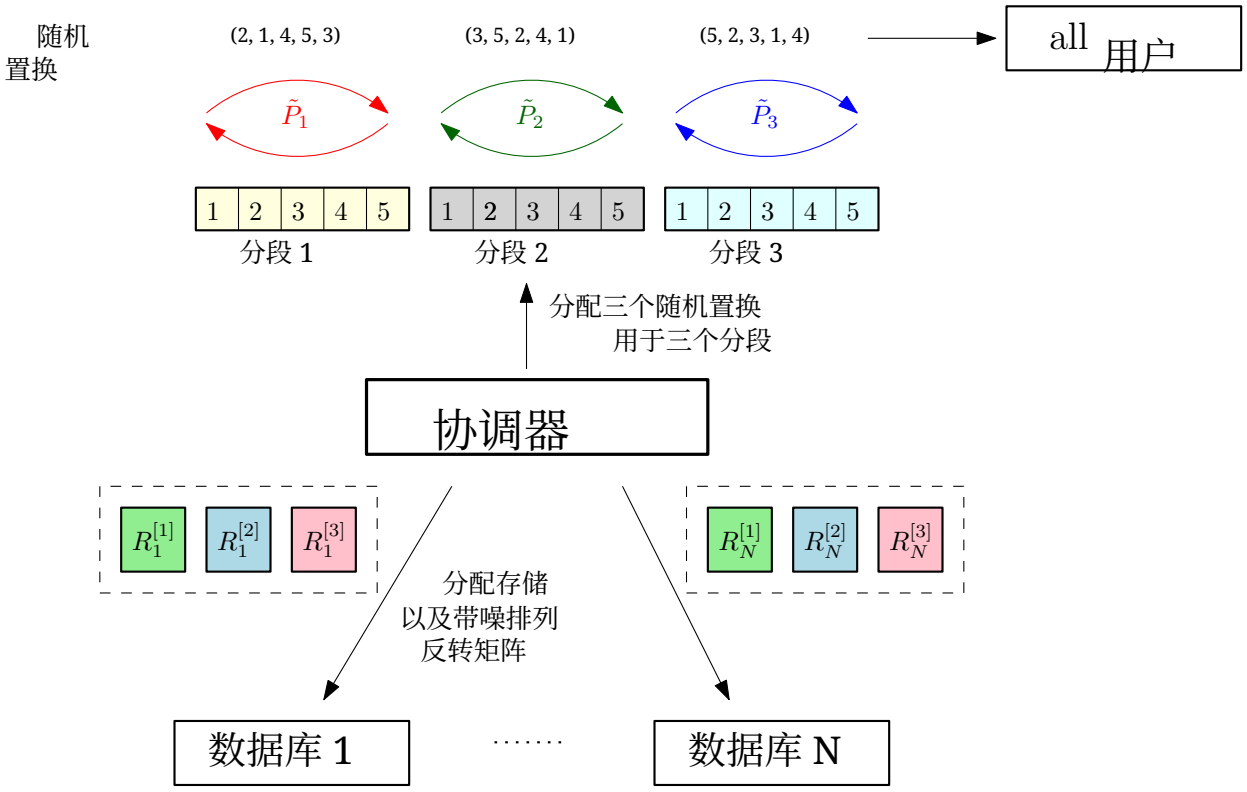


图4：方案在情形1和情形2下的初始化。

$\{1, \ldots, N\}$ 作为，

$$S_n^{[s]} = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} W_1^{[s]} + \sum_{j=0}^{\ell} \alpha_n^j Z_{1,j}^{[s]} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} W_\ell^{[s]} + \sum_{j=0}^{\ell} \alpha_n^j Z_{\ell,j}^{[s]} \end{bmatrix}, \tag{6}$$

其中， $W_i^{[s]}$ 是子数据包 s的第 i个参数， $Z_i^{[s]}$, $_j$是随机噪声符号， $\{f_i\}_{i=1}^{\ell}$和 $\{\alpha_n\}_{n=1}^{N}$是取自 $\mathbb{F}_q$ 的全局已知互异常数。每个分段中的子数据包按子数据包1至子数据包$\frac{P}{B}$ 的顺序依次堆叠。在初始化阶段，协调器向所有用户发送 B个分段中每个˜ ˜内的$\frac{P}{B}$ 个（本例中为$\frac{P}{B} = 5$ 个）子数据包的 B 个（本例中为$B = 3$ 个）随机且相互独立选取的置换。这些置换记为 P_1、.....、P_B。协调器还发送对应的 B 个加噪置换逆矩阵，如下所示：

$$R_n^{[i]} = (\tilde{R}^{[i]} \otimes \Gamma_n) + \tilde{Z}^{[i]}, \quad i = 1, \ldots, B, \tag{7}$$

如图4所示，写入数据库 n, $n \in \{1, \ldots, N\}$，其中 $\tilde{R}^{[i]}$ 是置换反演

˜与置换 P_i对应的矩阵， Γ_n 是由下式给出的对角矩阵，

$$\Gamma_n = \begin{bmatrix} \frac{1}{f_1-\alpha_n} & & \\ & \ddots & \\ & & \frac{1}{f_\ell-\alpha_n} \end{bmatrix}, \tag{8}$$

且 $\tilde{Z}^{[i]}$ 是一个大小为 $\frac{P\ell}{B} \times \frac{P\ell}{B}$ 的随机噪声矩阵。根据所考虑的示例，数据库 n、 $n \in \{1, \ldots, N\}$ 对应于第一个˜分段（置换： $P_1 =$ (2, 1, 4, 5, 3)）的置换反转矩阵为：

$$R_n^{[1]} = \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \otimes \Gamma_n \right) + \tilde{Z}^{[1]} = \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]}, \tag{9}$$

˜类似地，对于第二个分段（置换： $P_2 =$ (3,5,2,4,1)），数据库 n、 $n \in \{1, \ldots, N\}$ 的置换反转矩阵为：

$$R_n^{[2]} = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[2]}. \tag{10}$$

存储、置换和加噪置换逆转矩阵初始化完毕，系统准备好开始联邦学习过程后，协调器便会离开系统。此后，所有通信都仅在各个用户与数据库之间进行，涉及置换后的子分组索引。尽管数据库可以访问加噪置换逆转矩阵，但它们始终无法获知底层置换，因为根据香农一次一密定理，添加的噪声 $\tilde{Z}^{[i]}$ 使加噪矩阵独立于原始置换反转矩阵。

读取阶段：数据库根据在时间 $t-1$的写入阶段收到的置换后的子分组索引，决定在时间 t 的读取阶段向用户发送哪些 Pr' 稀疏子数据包的置换后索引。例如，数据库会考虑所有用户在时间 $t-1$更新的子数据包的置换后索引，并从中选出最受欢迎的 Pr' ，在时间 t的读取阶段发送给用户。请注意，数据库不知道用户在每个时间点的写入阶段更新的稀疏子数据包索引的真实索引，在两个阶段都只使用置换后的索引。我们将要发送给用户的稀疏子数据包的置换后索引表示为

˜分段 j 表示为 V_j，其中 $j \in \{1, \ldots, B\}$。在此示例中，设对应于第一段的置换后子分组索引稀疏集合为 $\tilde{V}_1 = \{1, 3\}$。[8] 一个指定的数据库会将每个分段的这些置换后索引发送给用户。随后，用户使用已知置换，针对分段$j \in \{1, \ldots, B\}$中的每个稀疏子数据包 i，找出真实索引，˜ ˜即 $V_j(i) = P_j(V_j(i))$。对于此示例（分段 1），真实索引集合为:

$$V_1(i) = \tilde{P}_1(\tilde{V}_1(i)), \quad i = 1, 2 \tag{11}$$

$$V_1 = \{2, 4\}. \tag{12}$$

˜为了发送分段 j 的第 i 个稀疏子分组 $V_j(i)$，每个数据库 $n, n \in \{1, \ldots, N\}$ 都会生成以下查询。

$$Q_n^{[\tilde{V}_j(i)]} = \sum_{k=1}^{\ell} R_n^{[j]}(:, (i-1)\ell + k). \tag{13}$$

例如，对应于第一段的第一个稀疏子分组的查询（即 ˜$V_1(1) = 1$）为:

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, k) = \begin{bmatrix} 0_\ell \\ \frac{1}{f_1 - \alpha_n} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1, \tag{14}$$

其中，Z_1 是由 $R_n^{[1]}$ 的噪声分量产生的随机噪声向量。类似地，分段 1 的第二个稀疏子分组的˜查询（即 $V_1(2) = 3$）为:

$$Q_n^{[\tilde{V}_1(2)]} = Q_n^{[3]} = \sum_{k=1}^{\ell} R_n^{[1]}(:, 2\ell + k) = \begin{bmatrix} 0_\ell \\ 0_\ell \\ 0_\ell \\ \frac{1}{f_1 - \alpha_n} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} \\ 0_\ell \end{bmatrix} + Z_2. \tag{15}$$

请注意，随机噪声使数据库无法获知置换逆操作

[8]分段 2 和 3 也各自存在两组类似的集合（基数可以相同或不同，且三组集合的基数之和等于 Pr'）。

向量 Z_1和 Z_2。然后，数据库 $n, n \in \{1, \ldots, N\}$通过计算查询与缩放存储的点积，将对应于分段 j 中第 i个稀疏子分组的答案发送给所有用户，

$$A_n^{[\tilde{V}_j(i)]} = (D_n \times S_n)^T Q_n^{[\tilde{V}_j(i)]}, \quad j \in \{1, \ldots, B\} \tag{16}$$

$$= \frac{1}{f_1 - \alpha_n} W_1^{[V_j(i)]} + \ldots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[V_j(i)]} + P_{\alpha_n}(\ell + 1), \tag{17}$$

其中， D_n 是大小为 $\frac{P\ell}{B} \times \frac{P\ell}{B}$ 的对角矩阵，定义如下：

$$D_n = I_{\frac{P}{B}} \otimes \Gamma_n^{-1} = \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}, \tag{18}$$

其中， $I_{\frac{P}{B}}$是大小为$\frac{P}{B} \times \frac{P}{B}$的单位矩阵， $P_{\alpha_n}(\ell + 1)$是 α_n 中的一个次数为$\ell+1$的多项式。例如，数据库 n、$n \in \{1, \ldots, N\}$ 对应于第 1 段的第一个稀疏子分组（即 $V_1(1) = 1$）的答案如下：

$$A_n^{[\tilde{V}_1(1)]} = (D_n \times S_n)^T Q_n^{[\tilde{V}_1(1)]} \tag{19}$$

$$= \left(\begin{bmatrix} \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[1]} + \sum_{j=0}^{\ell} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[1]} + \sum_{j=0}^{\ell} \alpha_n^j I_{\ell,j} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[5]} + \sum_{j=0}^{\ell} \alpha_n^j I_{1,j} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} W_\ell^{[5]} + \sum_{j=0}^{\ell} \alpha_n^j I_{\ell,j} \end{bmatrix} \end{bmatrix} \right)^T \left(\begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_\ell \\ 0_\ell \\ 0_\ell \end{bmatrix} + Z_1 \right) \tag{20}$$

$$= \frac{1}{f_1 - \alpha_n} W_1^{[2]} + \ldots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell + 1). \tag{21}$$

现在，用户通过求解以下方程，获得第 1 个分段中实际子分组 2 的参数（即 $V_1(1) = \tilde{P}_1(V_1(1)) = 2$）：

$$\begin{bmatrix} A_1^{\tilde{V}_1(1)} \\ \vdots \\ A_N^{\tilde{V}_1(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \ldots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^{\ell+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \ldots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \ldots & \alpha_N^{\ell+1} \end{bmatrix} \begin{bmatrix} W_1^{[2]} \\ \vdots \\ W_\ell^{[2]} \\ R_0 \\ \vdots \\ R_{\ell+1} \end{bmatrix}, \tag{22}$$

其中， R_i 是式 (21) 中多项式 $P_{\alpha_n}(\ell+1)$ 的系数。注意，给定 $N=2\ell+2$ 时，式 (22)（以及式 (17) 中对应的一般方程组）有解，这决定了子分包化为 $\ell=\frac{N-2}{2}$。上述过程会对每个 B 分段中的所有稀疏子数据包执行。由此得到的读取成本（包括数据下载和置换索引下载）为:

$$C_R=\frac{Pr'(\log_q\frac{P}{B}+\log_q B)+Pr'N}{L}=\frac{Pr'(\log_q P+N)}{P\frac{N-2}{2}}=\frac{2r'(1+\frac{\log_q P}{N})}{1-\frac{2}{N}}. \tag{23}$$

写入阶段：在写入阶段，每个用户选择具有最重要更新的 Pr 子分组，并将相应的加噪组合更新（每个子分组一个比特）及其置换后的子分组索引发送给每个数据库。发送至数据库 n 的分段 j 的实际子分组 i 的加噪组合更新（假设该子分组属于 Pr 个选定的子分组） $n\in\{1,\dots,N\}$）为:

$$U_n^{[i,j]}=\sum_{k=1}^{\ell}\prod_{r=1,r\neq k}^{\ell}(f_r-\alpha_n)\tilde{\Delta}_k^{[i,j]}+\prod_{r=1}^{\ell}(f_r-\alpha_n)Z^{[i,j]}, \tag{24}$$

其中 $\tilde{\Delta}_k^{[i,j]}=\frac{\Delta_k^{[i,j]}}{\prod_{r=1}^{\ell},_{r\neq k}(f_r-f_k)}$， $\Delta_k^{[i,j]}$ 是稀疏子分组 i 中第 k 个比特的更新，该子分组属于分段 j，而 $Z^{[i,j]}$ 是一个随机噪声位。为了确定分段 j 中子分组 i 的置换后的子分组索引 $\tilde{}$，考虑将分段 j 所分配的置换（即 P_j）视为从集合 $\{1,\dots,\frac{P}{B}\}$ 到集合 $\tilde{P}_j$ 按确切顺序建立的一一映射。于是，分段 j 中子分组 i 所对应的置换后的子分组索引为:

$$Y^{[i,j]}=\tilde{P}_j^{-1}(i),\quad j\in\{1,\dots,B\}. \tag{25}$$

计算出所有 Pr 选定子分组对应的合并更新和置换后的子分组索引后，用户将 Pr (更新、子分组、分段) 元组上传到所有数据库。[9]

例如，假设某个用户想要更新分段 1 中的实际子分组 2 和 4、分段 2 中的子分组 2，以及分段 3 中的子分组 5。基于~本例中考虑的置换，即 $P_1=\{2,1,4,5,3\}$、 $P_2=\{3,5,2,4,1\}$ 和 $\tilde{P}_3=\{5,2,3,1,4\}$，用户生成合并更新 $U_n^{[2,1]}$、 $U_n^{[4,1]}$、 $U_n^{[2,2]}$和 $U_n^{[5,3]}$，其形式如式 (24) 所示。相应的置换后子分组索引为:

$$Y^{[2,1]}=\tilde{P}_1^{-1}(2)=1 \tag{26}$$

$$Y^{[4,1]}=\tilde{P}_1^{-1}(4)=3 \tag{27}$$

$$Y^{[2,2]}=\tilde{P}_2^{-1}(2)=3 \tag{28}$$

[9]请注意，(更新、子分组、分段)中的“子分组”和“分段”元素分别指置换后的子分组索引和实际分段索引。

$$Y^{[5,3]} = \tilde{P}_3^{-1}(5) = 1. \tag{29}$$

因此，用户发送给数据库 n、对应于分段 1 的两个置换后的（更新，子分组，分段）元组为 ($U_n^{[2,1]}$, 1, 1) 和 ($U_n^{[4,1]}$, 3, 1)。类似地，对应于分段 2 和分段 3 的置换后的（更新，子分组，分段）元组分别为 ($U_n^{[2,2]}$, 3, 2) 和 ($U_n^{[5,3]}$, 1, 3)。[10]数据库 $n, n \in \{1, \ldots, N\}$收到$Pr$（更新，子分组，分段）元组后，会为每个分段 $j \in \{1, \ldots, B\}$创建相应的置换后更新向量 $\hat{}Y_n^{[j]}$，如下所示:

$$\hat{Y}_n^{[j]} = \sum_{i=1}^{\frac{P}{B}} U_n^{[i,j]} e_{\frac{P}{B}}(Y^{[i,j]}), \tag{30}$$

其中，$e_{\frac{P}{B}}(Y^{[i,j]})$是一个大小为 $\frac{P}{B} \times 1$的全零向量，且第 $Y^{[i,j]}$个位置为 1。请注意，对于 i、$i \in \{1, \ldots, \frac{P}{B}\}$的那些取值，如果其对应的子分组未包含在 Pr 选定的子分组集合中，我们会考虑 $U_n^{[i,j]} = 0$。为了以私密方式逆转置换，每个数据库都会创建,

$$\hat{U}_n^{[j]} = \hat{Y}_n^{[j]} \otimes 1_\ell = [\hat{Y}_n^{[j]}(1) \cdot 1_\ell, \ldots, \hat{Y}_n^{[j]}(\frac{P}{B}) \cdot 1_\ell]^T, \tag{31}$$

对于每个分段 j，其中 1_ℓ是一个大小为 $\ell \times 1$的全一向量。对于所考虑的示例，数据库 n、$n \in \{1, \ldots, N\}$根据接收的信息（$U_n^{[2,1]}$, 1, 1）、（$U_n^{[4,1]}$, 3, 1）、（$U_n^{[2,2]}$, 3, 2）和（$U_n^{[5,3]}$, 1, 3）为三个分段生成的$\hat{}U_n^{[j]}$向量如下:

$$\hat{U}_n^{[1]} = \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[4,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix}, \quad \hat{U}_n^{[2]} = \begin{bmatrix} 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[2,2]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix}, \quad \hat{U}_n^{[3]} = \begin{bmatrix} U_n^{[5,3]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix}. \tag{32}$$

接下来，各数据库以私密方式按真实顺序重新排列更新，并计算每个分段的增量更新 $\bar{U}_n^{[j]} = R_n^{[j]} \hat{}U_n^{[j]}$for $j \in \{1, \ldots, B\}$，然后将其加到现有存储的第 j 个分段中，从而得到更新后的存储。以所考虑的示例为例，计算数据库 n 中分段 1 的增量更新 $n \in \{1, \ldots, N\}$,

$$\bar{U}_n^{[1]} = R_n^{[1]} \hat{U}_n^{[1]} \tag{33}$$

[10]注意，段索引没有置换，只有每个分段内的子分组索引会被置换。

$$= \left(\begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \bar{Z}_1 \right) \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[4,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \tag{34}$$

$$= \begin{bmatrix} 0_\ell \\ \frac{U_n^{[2,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{U_n^{[2,1]}}{f_\ell-\alpha_n} \\ 0_\ell \\ \frac{U_n^{[4,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{U_n^{[4,1]}}{f_\ell-\alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0_\ell \\ \frac{\Delta_1^{[2,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[2,1]}}{f_\ell-\alpha_n} \\ 0_\ell \\ \frac{\Delta_1^{[4,1]}}{f_1-\alpha_n} \\ \vdots \\ \frac{\Delta_\ell^{[4,1]}}{f_\ell-\alpha_n} \\ 0_\ell \end{bmatrix} + P_{\alpha_n}(\ell), \tag{35}$$

其中，此处的 $P_{\alpha_n}(\ell)$ 是一个大小为 $\frac{P\ell}{B}$ 的向量，其元素由 α_n 中次数为 ℓ 的多项式构成；最后一个等式是通过在 [47] 中应用引理 1 得出的。其他两个分段也执行相同的过程。由于增量更新与式 (6) 中的存储形式相同，因此时刻 t 分段 $j, j \in \{1, 2, 3\}$ 的存储可更新为：

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \quad n \in \{1, \ldots, N\}. \tag{36}$$

注意，由（35）可知，对于分段 1，两个真实稀疏子分组 2 和 4 已正确更新，同时其余子分组保持不变，且不会向任何数据库泄露实际子分组索引 2 和 4。由此得到的写入成本为：

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} = \frac{PrN(1 + \log_q P)}{P^{\frac{N-2}{2}}} = \frac{2r(1 + \log_q P)}{1 - \frac{2}{N}}. \tag{37}$$

总存储复杂度（包括数据和置换逆矩阵）为 $O(L) + O(\frac{L^2}{B^2} \times B) = O(\frac{L^2}{B})$。信息泄露的推导见第4.2节。

情况2：本情况使用 MDS编码存储和更小的置换逆矩阵，以降低存储成本，但代价是通信成本更高。由于情况1和情况2都只使用段内置换，因此两种情况下的信息泄露相同。本情况也考虑了与情况1相同的示例（如图4所示）。

初始化：情况2中的单个子分组 s 存储在数据库 n, $n \in \{1, \ldots, N\}$ 中，如下所示：

$$S_n^{[s]} = \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[s]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[s]}. \tag{38}$$

因此，分段 j, $j \in \{1, \ldots, B\}$ 的存储量为：

$$S_n^{[j]} = \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,j]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[1,j]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[\frac{P}{B},j]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[\frac{P}{B},j]} \end{bmatrix}, \tag{39}$$

其中，$W_i^{[s,j]}$ 是分段 j 中子分组 s 的第 i 个参数，$Z_i^{[s,j]}$ 是随机噪声符号。注意，对于所考虑的示例，$\frac{P}{B} = 5$ 。与情况1类似，协调~器在存储中初始化所有噪声项，为每个分段中的子分组分配 B 个置换 P_i, $i \in \{1, \ldots, B\}$，并将它们发送给用户；同时，将相应的加噪置换逆转矩阵 $R_n^{[i]}$, $i \in \{1, \ldots, B\}$ 发送到数据库 n,$n \in \{1, \ldots .N\}$，如图4所示。加噪置换逆转矩阵的形式如下：

$$R_n^{[i]} = \bar{R}^{[i]} + \alpha_n^{\ell} \bar{Z}^{[i]}, \quad i \in \{1, \ldots, B\}, \tag{40}$$

其中，$\bar{R}^{[i]}$ 是对应于第 i 个置换 $\tilde{P}_i$ 的置换逆矩阵，$\bar{Z}^{[i]}$ 是一个随机噪声矩阵，两者的大小均为 $\frac{P}{B} \times \frac{P}{B}$ 。对于图4中的示例，发送到数据库 n, $n \in \{1, \ldots, N\}$ ~ ~ ~、分别对应三个分段的加噪置换逆转矩阵如下（回顾：$P_1 = (2, 1, 4, 5, 3)$, $P_2 = (3, 5, 2, 4, 1)$, $P_3 = (5, 2, 3, 1, 4)$）：

$$R_n^{[1]} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]}, \; R_n^{[2]} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[2]}, \; R_n^{[3]} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[3]}. \tag{41}$$

如情况1中所述，联邦学习过程开始后，协调器便退出系统，之后所有通信都仅在用户与数据库之间进行，并使用置换后的子分组索引。

~读取阶段：如情况1所述，令 V_j 为从分段 j 中选取并将发送给用户用于 $j \in \{1, \ldots, B\}$的稀疏子数据包的置换后索引集合。例如，令 $V_1 = \{1, 3\}$ 为分段1中需要

在时间 t发送给用户的置换后稀疏子数据包集合。一个指定的数据库将每个分段的置换后子分组˜索引（分段1： $V_1 = \{1, 3\}$）发送给用户，用户利用已知置换，通过 $V_j(i) =_{\sim} {}_{\sim}P_j(V_j(i))$识别相应的真实稀疏子数据包索引，其中 V_j 是包含分段 j中稀疏子数据包真实索引的向量。具体而言，用户根据收到的集合 V_2 和 V_3，对分段1˜ ˜以及另外两个分段执行式（12）中的相同计算。与情况1类似，每个数据库都会生成查询，以发送选定的子数据包。分段 j 中第 i个置换后稀疏子数据包对应的查询为:

$$Q_n^{[\tilde{V}_j(i)]} = R_n^{[j]}(:, \tilde{V}_j(i)), \quad j \in \{1, \dots, B\}. \tag{42}$$

以下是数据库 n, $n \in \{1, \dots, N\}$ 生成的两个查询，用于向用户发送分段 1 的两个稀疏子分组（索引为置换后的 $V_1 = \{1, 3\}$）：

$$Q_n^{[\tilde{V}_1(1)]} = Q_n^{[1]} = R_n^{[1]}(:, 1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_1, \quad \text{and} \quad Q_n^{[\tilde{V}_1(2)]} = Q_n^{[3]} = R_n^{[1]}(:, 3) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_3, \tag{43}$$

其中，ˆ Z_1 和ˆ Z_3 是 (41) 中 $\bar{Z}^{[1]}$ 的第一列和第三列。然后，数据库 n, $n \in \{1, \dots, N\}$ 将每个分段中各稀疏子分组对应的答案发送给用户，如下所示:

$$A_n^{[\tilde{V}_j(i)]} = (S_n^{[j]})^T Q_n^{[\tilde{V}_j(i)]}, \quad j \in \{1, \dots, B\} \tag{44}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[V_j(i), j]} + P_{\alpha_n}(2\ell), \tag{45}$$

其中， $P_{\alpha_n}(2\ell)$ 是一个关于 α_n 的 2ℓ 次多项式。例如，数据库 n 发送给用户的、对应于分段 1 的第一个稀疏子分组（$V_1(1) = 1$）的答案为， $n \in \{1, \dots, N\}$

$$A_n^{[\tilde{V}_1(1)]} = (S_n^{[1]})^T Q_n^{[\tilde{V}_1(1)]} \tag{46}$$

$$= \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,1]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[1,1]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[5,1]} + \sum_{i=0}^{\ell} \alpha_n^i Z_i^{[5,1]} \end{bmatrix}^T \left(\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_n^{\ell} \hat{Z}_1 \right) \tag{47}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[2,1]} + P_{\alpha_n}(2\ell). \tag{48}$$

用户通过求解以下方程，获得分段 1 的实际子分组 2 的参数（即 $V_1(1) =\sim \sim P_1(V_1(1)) = 2$），

$$\begin{bmatrix} A_1^{\tilde{V}_1(1)} \\ \vdots \\ A_N^{\tilde{V}_1(1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_1^\ell} & \dots & \frac{1}{\alpha_1} & 1 & \alpha_1 & \dots & \alpha_1^{2\ell} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\alpha_N^\ell} & \dots & \frac{1}{\alpha_N} & 1 & \alpha_N & \dots & \alpha_N^{2\ell} \end{bmatrix} \begin{bmatrix} W_\ell^{[2,1]} \\ \vdots \\ W_1^{[2,1]} \\ R_0 \\ \vdots \\ R_{2\ell} \end{bmatrix}, \tag{49}$$

其中， R_i 是式 (48) 中多项式的系数。注意，若满足 $N = 3\ell + 1$，则式 (49)（以及式 (45) 中的一般答案）可解，而这决定了子分包化为 $\ell = \frac{N-1}{3}$。上述相同过程会应用于每个分段中的所有稀疏子分组。由此得到的读取成本为:

$$C_R = \frac{Pr' \log_q P + Pr'N}{L} = \frac{Pr'(\log_q P + N)}{P^{\frac{N-1}{3}}} = \frac{3r'(1 + \frac{\log_q P}{N})}{1 - \frac{1}{N}}. \tag{50}$$

写入阶段：在写入阶段，用户针对更新最显著的 Pr 个子分组生成 Pr 个合并更新。分段 j 的第 i 个子分组的合并更新定义为（假设该子分组属于 Pr 个选定的子分组之一）：

$$U_n^{[i,j]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} \Delta_k^{[i,j]} + Z^{[i,j]}, \tag{51}$$

其中， $\Delta_k^{[i,j]}$ 是分段 j 中第 i个子分组的第 k位的更新， $Z^{[i,j]}$ 是一个随机噪声符号。与情况1类似，用户使用式（25），为每个分段 j中所选的 Pr 个子分组对应的实际子分组索引 i 生成置换后的子分组索引。生成置换后的子分组索引后，用户与情况1类似，将所选的 Pr 个子分组对应的置换后的（更新，子分组，分段）元组发送到所有数据库。对于情况1中的同一示例，用户希望更新分段1中的实际子分组2和4、分段2中的子分组2以及分段3中的子分组5。用户向数据库 n发送与情况1中相同的置换后的（更新，子分组，分段）元组，即（$U_n^{[2,1]}$, 1, 1）、（$U_n^{[4,1]}$, 3, 1）、（$U_n^{[2,2]}$, 3, 2）、（$U_n^{[5,3]}$, 1, 3）；这里假设三个分段分别采用相同的三个置换 $P_1 = \{2, 1, 4, 5, 3\}$、 $P_2 = \{3, 5, 2, 4, 1\}$ 和 $P_3 = \{5, 2, 3, 1, 4\}$。数据库收到这些 Pr（更新，子分组，分段）元组后，会为每个分段创建置换后的更新向量$\hat{Y}_n^{[j]}$

$j \in \{1,\ldots,B\}$ 使用式（30）。对于所考虑的示例，数据库 n创建的三个置换后的更新向量为：

$$\hat{Y}_n^{[1]} = \begin{bmatrix} U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \\ 0 \end{bmatrix}, \quad \hat{Y}_n^{[2]} = \begin{bmatrix} 0 \\ 0 \\ U_n^{[2,2]} \\ 0 \\ 0 \end{bmatrix}, \quad \hat{Y}_n^{[3]} = \begin{bmatrix} U_n^{[5,3]} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{52}$$

根据接收的信息 $(U_n^{[2,1]}, 1, 1)$、$(U_n^{[4,1]}, 3, 1)$、$(U_n^{[2,2]}, 3, 2)$、$(U_n^{[5,3]}, 1, 3)$，数据库 n，$n \in \{1,\ldots,N\}$ 利用置换后的更新向量 $\hat{Y}_n^{[j]}$, $j \in \{1,\ldots,B\}$，以保护隐私的方式计算每个分段经过正确重排后的增量更新向量 $\bar{U}_n^{[j]} = R_n^{[j]} \hat{Y}_n^{[j]}$, $j \in \{1,\ldots,B\}$。得到的增量更新与式 (39) 中的存储形式相同，因此可以将其添加到现有存储中，从而得到每个分段更新后的存储。为通过示例说明上述过程，考虑迄今为止同一示例中的分段 1 的增量更新，

$$\bar{U}_n^{[1]} = R_n^{[1]} \hat{Y}_n^{[1]} \tag{53}$$

$$= \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]} \right) \begin{bmatrix} U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \\ 0 \end{bmatrix} \tag{54}$$

$$= \begin{bmatrix} 0 \\ U_n^{[2,1]} \\ 0 \\ U_n^{[4,1]} \\ 0 \end{bmatrix} + P_{\alpha_n}(\ell) = \begin{bmatrix} 0 \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,1]} \\ 0 \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[4,1]} \\ 0 \end{bmatrix} + P_{\alpha_n}(\ell), \tag{55}$$

其中，此处的 $P_{\alpha_n}(\ell)$ 是一个大小为 5×1 的向量，由 α_n 中次数为 ℓ 的多项式构成。由于每个分段的增量更新（即式 (55)）与式 (39) 中的存储形式相同，因此可将增量更新直接加到现有存储中，得到其更新后的版本，即：

$$S_n^{[j]}(t) = S_n^{[j]}(t-1) + \bar{U}_n^{[j]}, \quad j \in \{1,\ldots,B\},\ n \in \{1,\ldots,N\}. \tag{56}$$

情形 2 的写入成本为：

$$C_W = \frac{PrN(1 + \log_q B + \log_q \frac{P}{B})}{L} = \frac{PrN(1 + \log_q P)}{P^{\frac{N-1}{3}}} = \frac{3r(1 + \log_q P)}{1 - \frac{1}{N}}. \tag{57}$$

总存储复杂度（包括数据和置换逆矩阵）为 $O(P) + O(\frac{P^2}{B^2} \times B) = O(\frac{P^2}{B}) = O(\frac{L^2}{BN^2})$。信息泄露的推导见第4.2节。

情况3：在此情况下，我们采用具有大型置换反转矩阵的未编码存储。请注意，在情况1和情况2中，仅对每个分段内的子分组索引进行置换，用户将真实的段索引上传至数据库。在此情况下，我们不仅置换分段内的子分组索引，也置换段索引，以进一步减少信息泄露。然而，这会增加存储成本，因为段索引的置换要求数据库额外存储一个加噪置换逆矩阵。与情况1（迄今通信成本最低）相比，额外一轮置换不会显著影响通信成本。

对于情况3和情况4，我们给出一般方案，并结合示例设置，介绍具有$P = 12$ 个子数据包（子分包化为 ℓ）的情形，这些子数据包被划分为 $B = 3$ 个相等的分段，如图5所示。

初始化 ：情况3中单个子分组（子分组 s）的存储方式如下： ,

$$S_n^{[s]} = \begin{bmatrix} \frac{1}{f_1 - \alpha_n} W_1^{[s]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[s]} \\ \vdots \\ \frac{1}{f_\ell - \alpha_n} W_\ell^{[s]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[s]} \end{bmatrix}, \tag{58}$$

与情况1中的记号相同。子数据包按照分段1至分段 B的顺序，在每个分段内依次堆叠（从子数据包1到子数据包$\frac{P}{B}$）。在初始化阶段，协调器向用户发送每个 B 分段中 $\frac{P}{B}$个子数据包的 B 个随机且独立选取的置换，即 $\tilde{P}_1$、...、 $\tilde{P}_B$，以及对 B 个分段 P 随机且独立选取的置换ˆ。协调器还在每个数据库 n、 $n \in \{1, \ldots, N\}$中放置相应的加噪置换逆矩阵（对应于 B段内置换： $R_n^{[1]}$、...、$R_n^{[B]}$，以及一个段间置换：ˆ R_n）。与存储在数据库 n、$n \in \{1, \ldots, N\}$ 中的第 i个分段 $i \in \{1, \ldots, B\}$ 相对应的加噪置换逆矩阵为：

$$R_n^{[i]} = (\tilde{R}^{[i]} \otimes \Gamma_n) + \tilde{Z}^{[i]}, \tag{59}$$

与情况1中的记号相同。数据库 n 中存储的段间置换 P $n \in \{1, \ldots, N\}$ 对应的加噪置换逆矩阵为

by,

$$\hat{R}_n = (\bar{R} \otimes I_\ell) + (I_B \otimes \Gamma_n^{-1})\hat{Z} = \begin{bmatrix} b_{1,1}^{[n]} & \dots & b_{1,B}^{[n]} \\ \vdots & \ddots & \vdots \\ b_{B,1}^{[n]} & \dots & b_{B,B}^{[n]} \end{bmatrix}, \tag{60}$$

¯其中， R 是与段间置换 P 对应的置换反转矩阵， I_k 是大小为 $k \times k$ 的单位矩阵， Γ_n^{-1} 是如下所示的对角矩阵：

$$\Gamma_n^{-1} = \begin{bmatrix} f_1 - \alpha_n & & \\ & \ddots & \\ & & f_\ell - \alpha_n \end{bmatrix} \tag{61}$$

ˆˆ，且 Z 是一个大小为 $B\ell \times B\ell$的随机噪声矩阵。每个矩阵 R_n 均表示为大小为 $\ell \times \ell$的分块形式，如式 (60) 中最后一个等式所示。根据图5中给出的置换，与第一个分段内置换（$\tilde{P}_1 =$ (2, 4, 3, 1)）对应的加噪置换反转矩阵，即 $R_n^{[1]}$，可表示为：

$$R_n^{[1]} = \left(\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \otimes \Gamma_n \right) + \tilde{Z}^{[1]} = \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]}. \tag{62}$$

类似地， $R_n^{[2]}$和 $R_n^{[3]}$可表示为：

$$R_n^{[2]} = \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} + \tilde{Z}^{[2]}, \quad R_n^{[3]} = \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[3]}, \tag{63}$$

对于同一示例，数据库 n、 $n \in \{1, \dots, N\}$ 对应的分段间加噪置换逆矩阵可表示为：

$$\hat{R}_n = \left(\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \otimes \Phi \right) + \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \Gamma_n^{-1} \right) \hat{Z} \tag{64}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi \\ \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1} \end{bmatrix} \hat{Z} = \begin{bmatrix} b_{1,1}^{[n]} & b_{1,2}^{[n]} & b_{1,3}^{[n]} \\ b_{2,1}^{[n]} & b_{2,2}^{[n]} & b_{2,3}^{[n]} \\ b_{3,1}^{[n]} & b_{3,2}^{[n]} & b_{3,3}^{[n]} \end{bmatrix}, \tag{65}$$

ˆ，其中 $\Phi = I_{\ell\circ}$ 每个矩阵 R_n 都按大小为 $\ell \times \ell$ 的块表示，如最后

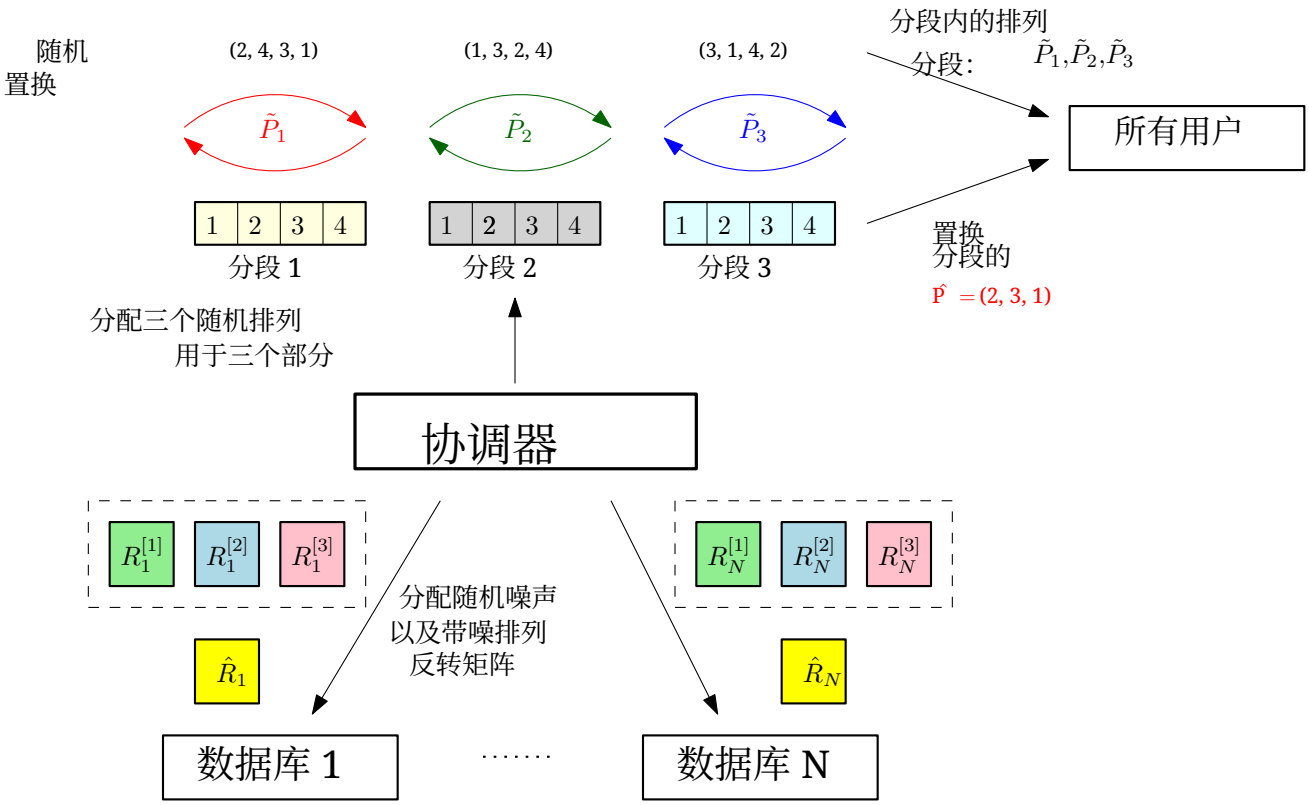


图5：情况3和情况4的方案初始化。

个等式（65）所示，这对后续计算很有帮助。存储和置换初始化后，协调器便退出系统；在联邦学习过程开始之前，加噪置换逆矩阵被放置在各数据库中。联邦学习过程中的所有通信均由用户与数据库之间通过置换后的子分组和段索引进行。

读取阶段：数据库根据在时间 $t-1$ 的写入阶段从用户处收到的置换后信息，确定在时间 t 要发送给用户的 Pr' 个子分组集合。例如，数据库会考虑所有用户在时间 $t-1$ 更新的子分组的置换后索引，并从中选出最受欢迎的 Pr' 个（按置换后索引计算），在时间 t 的读取阶段发送给用户。在情况3中，数据库不知道用户更新的每个分段内子分组的真实索引，也不知道相应分段的真实索引。不过，数据库可以用置换后的版本向用户传送稀疏子分组和分段索引（与在时间 $t-1$ 的写入阶段收到的信息相同）。由于用户知道所有置换，因此能够将置换后的索引转换为真实索引。令每个 Pr' 子分组的置换后（子分组，分段）信息表示为（η_p， ϕ_p）。这些信息由一个指定数据库在时间 t 发送给所有用户。用户可以使用.....将每个置换后的（η_p， ϕ_p）转换为其真实版本（η_r， ϕ_r），

$$\phi_r = \hat{P}(\phi_p) \tag{66}$$

$$\eta_r = \tilde{P}_{\phi_r}(\eta_p). \tag{67}$$

对于图5中的示例，假设某个用户从指定数据库收到以下置换后的（子分组，分段）对，

$$(\eta_p, \phi_p) = \{(1,3),(1,1),(1,2)\}. \tag{68}$$

回顾一下，本例中考虑的置换为 $P_1 = (2,4,3,1)$、$P_2 = (1,3,2,4)$、$P_3 = (3,1,4,2)$ 和 $P = (2,3,1)$。为了解实际子分组和段索引是如何得到的，考虑第一个置换后的对 $(1,3)$。由于置换后的段索引为 $\phi_p = 3$，对应的实际段索引为 $\phi_r = P(3) = 1$。然后，用户可以解码第一段内的子分组索引，即 $\eta_r = P_1(1) = 2$。因此，与置换后的（子分组，分段）对 $(\eta_p, \phi_p) = (1,3)$ 对应的实际（子分组，分段）对为 $(\eta_r, \phi_r) = (2,1)$。同理，与这三个置换后的对对应的实际（子分组，分段）对集合为：

$$(\eta_r, \phi_r) = \{(2,1),(1,2),(3,3)\}. \tag{69}$$

获得稀疏子分组和分段的真实索引后，用户逐一下载相应的子分组。为执行读取阶段的计算，每个数据库首先生成一个组合加噪置换逆矩阵，将段内和段间的加噪置换逆矩阵合并为一个加噪置换逆矩阵，以便进行后续计算。数据库 n、$n \in \{1,\dots,N\}$ 的组合加噪置换逆矩阵表示为：

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times \begin{bmatrix} I_{\frac{P}{B}} \otimes b_{1,1}^{[n]} & \dots & I_{\frac{P}{B}} \otimes b_{1,B}^{[n]} \\ \vdots & \vdots & \vdots \\ I_{\frac{P}{B}} \otimes b_{B,1}^{[n]} & \dots & I_{\frac{P}{B}} \otimes b_{B,B}^{[n]} \end{bmatrix} \tag{70}$$

$$= \begin{bmatrix} R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} & \dots & R_n^{[1]} \begin{bmatrix} b_{1,B}^{[n]} & & \\ & \ddots & \\ & & b_{1,B}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} \\ & \ddots & \\ R_n^{[B]} \begin{bmatrix} b_{B,1}^{[n]} & & \\ & \ddots & \\ & & b_{B,1}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} & \dots & R_n^{[B]} \begin{bmatrix} b_{B,B}^{[n]} & & \\ & \ddots & \\ & & b_{B,B}^{[n]} \end{bmatrix}_{\frac{P\ell}{B}\times\frac{P\ell}{B}} \end{bmatrix} \tag{71}$$

$$= \dot{R}_n + P_{\alpha_n}(1), \tag{72}$$

其中，$\dot{R}_n$ 是通过将式 (60) 中 R_n 的

第i行中的 1 替换为式 (59) 中的 $R^{[i]} \otimes \Gamma_n$，并将零替换为大小为 $\frac{P\ell}{B} \times \frac{P\ell}{B}$的全零矩阵而得到的组合置换反转矩阵。$P_{\alpha_n}(1)$ 是一个大小为 $L \times L$ 的矩阵，其元素是关于 α_n的一次多项式。

例如，考虑在图5所示的示例设置中，为数据库 n、 $n \in \{1, \ldots, N\}$ 生成添加了组合噪声的置换逆矩阵，

$$R_n = \begin{bmatrix} R_n^{[1]} & 0 & 0 \\ 0 & R_n^{[2]} & 0 \\ 0 & 0 & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} I_4 \otimes b_{1,1}^{[n]} & \ldots & I_4 \otimes b_{1,3}^{[n]} \\ \vdots & \vdots & \vdots \\ I_4 \otimes b_{3,1}^{[n]} & \ldots & I_4 \otimes b_{3,3}^{[n]} \end{bmatrix} \tag{73}$$

$$= \begin{bmatrix} R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[1]} \begin{bmatrix} b_{1,2}^{[n]} & & \\ & \ddots & \\ & & b_{1,2}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[1]} \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \\ R_n^{[2]} \begin{bmatrix} b_{2,1}^{[n]} & & \\ & \ddots & \\ & & b_{2,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[2]} \begin{bmatrix} b_{2,2}^{[n]} & & \\ & \ddots & \\ & & b_{2,2}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[2]} \begin{bmatrix} b_{2,3}^{[n]} & & \\ & \ddots & \\ & & b_{2,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \\ R_n^{[3]} \begin{bmatrix} b_{3,1}^{[n]} & & \\ & \ddots & \\ & & b_{3,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[3]} \begin{bmatrix} b_{3,2}^{[n]} & & \\ & \ddots & \\ & & b_{3,2}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} & R_n^{[3]} \begin{bmatrix} b_{3,3}^{[n]} & & \\ & \ddots & \\ & & b_{3,3}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} \end{bmatrix}. \tag{74}$$

请注意，

$$R_n^{[1]} \begin{bmatrix} b_{1,1}^{[n]} & & \\ & \ddots & \\ & & b_{1,1}^{[n]} \end{bmatrix}_{4\ell \times 4\ell} = \left(\begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right)$$

$$\times \begin{bmatrix} \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,1} \end{bmatrix} \tag{75}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \hat{Z}_{1,1} \\ \hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \hat{Z}_{1,1} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \hat{Z}_{1,1} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + P_{\alpha_n}(1) = P_{\alpha_n}(1) \tag{76}$$

^^其中，$Z_{1,1}$ 是（65）中 Z 的子矩阵，由前 ℓ 行和前 ℓ 列组成；$P_{\alpha_n}(1)$ 是大小为 $4\ell \times 4\ell$ 的矩阵，其元素是 α_n 中次数为 1 的多项式，以及 $0_{\ell\times\ell}$

是大小为 $\ell \times \ell$ 的全零矩阵。接下来，考虑 , 的计算。

$$R_n^{[1]} \begin{bmatrix} b_{1,3}^{[n]} & & \\ & \ddots & \\ & & b_{1,3}^{[n]} \end{bmatrix}_{4\ell\times 4\ell} = \left(\begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + \tilde{Z}^{[1]} \right)$$

$$\times \left(\begin{bmatrix} \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Phi \end{bmatrix} + \begin{bmatrix} \Gamma_n^{-1}\hat{Z}_{1,3} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,3} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,3} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n^{-1}\hat{Z}_{1,3} \end{bmatrix} \right) \tag{77}$$

$$= \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} + P_{\alpha_n}(1), \tag{78}$$

ˆ ˆ 其中，$Z_{1,\ 3}$ 是（65）中 Z 的子矩阵，由前 ℓ 行以及从 $2\ell+1$ 到 3ℓ 和 P_{α_n}给出的列索引组成；(1)是一个大小为 $4\ell \times 4\ell$ 的矩阵，其中包含次数为 1 的 α_n 多项式。基于类似的计算，我们可以将（74）中的组合置换反转矩阵 R_n 写为:

$$R_n = \begin{bmatrix} 0_{4\ell\times 4\ell} & 0_{4\ell\times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} \\ \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} & 0_{4\ell\times 4\ell} & 0_{4\ell\times 4\ell} \\ 0_{4\ell\times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} & 0_{4\ell\times 4\ell} \end{bmatrix} + P_{\alpha_n}(1), \tag{79}$$

其中， P_{α_n}(1) 是一个大小为 $12\ell \times 12\ell$ 的矩阵，其元素是 α_n 中次数为 1 的多项式。注意，组合加噪置换逆矩阵_R_n 的第一部分，只是将 R 中第 i 行的 1（对应于段间置换 ˆP 的置换反转矩阵）替换为 $\tilde{R}^{[i]} \otimes \Gamma_n$ ；其中，对于每个 $i \in \{1,2,3\}$， $\tilde{R}^{[i]}$ 是对应于段内置换 P_i 的置换反转˜矩阵。

为了下载与置换后的对 (η_p, ϕ_p) 对应的子分组，每个数据库都会生成如下查询：

$$Q_n^{[\eta_p,\phi_p]} = (I_P \otimes \Gamma_n^{-1}) \times \sum_{k=1}^{\ell} R_n(:, (\phi_p - 1)\frac{P}{B}\ell + (\eta_p - 1)\ell + k), \tag{80}$$

其中， I_P是大小为 $P \times P$的单位矩阵。然后，数据库 n, $n \in \{1, \ldots, N\}$ 发送与置换后的子分组（η_p, ϕ_p）相对应的答案，如下：

$$A_n^{[\eta_p,\phi_p]} = S_n^T Q_n^{[\eta_p,\phi_p]} = \frac{1}{f_1 - \alpha_n} W_1^{[\eta_r,\phi_r]} + \cdots + \frac{1}{f_\ell - \alpha_n} W_\ell^{[\eta_r,\phi_r]} + P_{\alpha_n}(\ell + 3), \tag{81}$$

其中， $P_{\alpha_n}(\ell + 3)$是 α_n 中的一个次数为 $\ell + 3$的多项式。然后，用户可以利用从 N 个数据库收到的所有答案，获得由（η_r, ϕ_r）指示的相应实际子分组的数值，即 $W_1^{[\eta_r,\phi_r]}, \ldots,$ $W_\ell^{[\eta_r,\phi_r]}$，如下：

$$\begin{bmatrix} A_1^{[\eta_p,\phi_p]} \\ \vdots \\ A_N^{[\eta_p,\phi_p]} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1-\alpha_1} & \cdots & \frac{1}{f_\ell-\alpha_1} & 1 & \alpha_1 & \cdots & \alpha_1^{\ell+3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{f_1-\alpha_N} & \cdots & \frac{1}{f_\ell-\alpha_N} & 1 & \alpha_N & \cdots & \alpha_N^{\ell+3} \end{bmatrix} \begin{bmatrix} W_1^{[\eta_r,\phi_r]} \\ \vdots \\ W_\ell^{[\eta_r,\phi_r]} \\ R_{0:\ell+3} \end{bmatrix}, \tag{82}$$

其中，每个 R_i 对应于式（81）中多项式 $P_{\alpha_n}(\ell + 3)$的第 i个系数。当 $N = 2\ell + 4$时，方程（82）有解，这决定了子分包化为$\ell = \frac{N-4}{2}$。

例如，考虑从图5中的相同示例设置下载由（η_p, ϕ_p） =(1, 3)指示的置换后的子分组。数据库 n, $n \in \{1, \ldots, N\}$ 生成如下查询：

$$Q_n^{[1,3]} = (I_{12} \otimes \Gamma_n^{-1}) \times \sum_{k=1}^{\ell} R_n(:, 8\ell + k) \tag{83}$$

$$= \begin{bmatrix} \Gamma_n^{-1} & & \\ & \ddots & \\ & & \Gamma_n^{-1} \end{bmatrix}_{12\ell \times 12\ell} \times \left(\begin{bmatrix} 0_\ell \\ \frac{1}{f_1-\alpha_n} \\ \vdots \\ \frac{1}{f_\ell-\alpha_n} \\ 0_{10\ell} \end{bmatrix} + \dot{P}_{\alpha_n}(1) \right) = \begin{bmatrix} 0_\ell \\ 1_\ell \\ 0_{10\ell} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1 - \alpha_n)P_{\alpha_n}(1) \\ \vdots \\ (f_\ell - \alpha_n)P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_1 - \alpha_n)P_{\alpha_n}(1) \\ \vdots \\ (f_\ell - \alpha_n)P_{\alpha_n}(1) \end{bmatrix} \end{bmatrix}_{L \times 1}, \tag{84}$$

其中， $\dot{P}_{\alpha_n}(1)$ 是一个大小为 $12\ell \times 1$ 的向量，由 α_n 中次数为 1 的多项式构成，而 $P_{\alpha_n}(1)$ 是 α_n 中次数为 1 的多项式。注意， 0_k 和 1_k 分别表示大小为 $k \times 1$ 的全零向量和全一向量。对于上述查询，数据库发送给

用户的答案 n 由 $n \in \{1, \ldots, N\}$ 给出:

$$A_n^{[1,3]} = S_n^T Q_n^{[1,3]} = \begin{bmatrix} \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[1]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_\ell-\alpha_n} W_\ell^{[1]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[1]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_1-\alpha_n} W_1^{[12]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{1,j}^{[12]} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \frac{1}{f_\ell-\alpha_n} W_\ell^{[12]} + \sum_{j=0}^{\ell+1} \alpha_n^j Z_{\ell,j}^{[12]} \end{bmatrix} \end{bmatrix}^T \times \left(\begin{bmatrix} 0_\ell \\ 1_\ell \\ 0_{10\ell} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} (f_1-\alpha_n) P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_\ell-\alpha_n) P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_1-\alpha_n) P_{\alpha_n}(1) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} (f_\ell-\alpha_n) P_{\alpha_n}(1) \end{bmatrix} \end{bmatrix}_{L\times 1} \right) \tag{85}$$

$$= \frac{1}{f_1-\alpha_n} W_1^{[2]} + \cdots + \frac{1}{f_\ell-\alpha_n} W_\ell^{[2]} + P_{\alpha_n}(\ell+3), \tag{86}$$

由此，如果满足 $N = 2\ell + 4$，则可以利用 N 个数据库的所有答案，得到第 1 段的（实际）第二个子分组，即 $(\eta_r, \phi_r) = (2, 1)$，这决定了子分包化为 $\ell = \frac{N-4}{2}$。由此得到的读取成本为:

$$C_R = \frac{Pr'(N + \log_q B + \log_q \frac{P}{B})}{L} = \frac{Pr'(N + \log_q P)}{P^{\frac{N-4}{2}}} = \frac{2r'(1 + \frac{\log_q P}{N})}{1 - \frac{4}{N}}. \tag{87}$$

写入阶段：在写入阶段，用户将更新幅度最大的 Pr 个子分组的合并更新、置换后的子分组索引和置换后的段索引发送给所有数据库。设 $(\eta_r^{[i]}, \phi_r^{[i]})$, $i \in \{1, \ldots, Pr\}$ 为对应于第 i 个选定子分组的实际（子分组、段）对。对于每个 Pr 个选定子分组，用户生成一个组合更新比特，其定义如下:

$$U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]} = \sum_{k=1}^{\ell} \prod_{j=1, j\neq k}^{\ell} (f_j - \alpha_n) \tilde{\Delta}_k^{[\eta_r^{[i]}, \phi_r^{[i]}]} + \prod_{j=1}^{\ell} (f_j - \alpha_n) Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \quad i \in \{1, \ldots, Pr\}, \tag{88}$$

其中 $\tilde{\Delta}_k^{[\eta_r^{[i]}, \phi_r^{[i]}]} = \frac{\Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]}}{\prod_{j=1, j\neq k}^{\ell} (f_j - f_k)}$，$\Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ 是分段 $\phi_r^{[i]}$的子分组 $\eta_r^{[i]}$中第 k个参数的更新，$Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}$是一个随机噪声符号。对于第 i个选定的子分组，用户发送由（$U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}$, $\eta_p^{[i]}$, $\phi_p^{[i]}$）给出的置换后的（更新、子分组、分段）元组，其中 $U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}$是形如（88）的子分组合并更新，$\eta_p^{[i]}$是通过以下方式获得的置换后的子分组索引，

$$\eta_p^{[i]} = \tilde{P}^{-1}_{\phi_r^{[i]}}(\eta_r^{[i]}) \tag{89}$$

而 $\phi_p^{[i]}$是通过以下方式获得的置换后的分段索引，

$$\phi_p^{[i]} = \hat{P}^{-1}(\phi_r^{[i]}) \tag{90}$$

~^ ~其中， $P_{\phi_r^{[i]}}$和 P 分别被视为从 j 到 $P_{\phi_r^{[i]}}$（j）的$j = 1, ..., \frac{P}{B}$以及从 i到ˆP（i）的 $i = 1, ..., B$的一一映射。对于图5中的示例，假设某个用户想要更新由以下实际（子分组、段）对标识的 Pr 稀疏子分组：$(\eta_r, \phi_r) = \{(2, 1), (2, 2), (3, 3)\}$。根据给定的段内置换 $P_1 = (2, 4, 3, 1)$、 $P_2 = (1, 3, 2, 4)$、 $P_3 = (3, 1, 4, 2)$，以及给定的ˆ逐段置换 $P = (2, 3, 1)$，用户向数据库 $n, n \in \{1, \ldots, N\}$发送以下（置换后的）信息，

$$(U_n^{[\eta_r,\phi_r]}, \eta_p, \phi_p) = \{(U_n^{[2,1]}, \tilde{P}_1^{-1}(2), \hat{P}^{-1}(1)), (U_n^{[2,2]}, \tilde{P}_2^{-1}(2), \hat{P}^{-1}(2)), (U_n^{[3,3]}, \tilde{P}_3^{-1}(3), \hat{P}^{-1}(3))\} \tag{91}$$

$$= \{(U_n^{[2,1]}, 1, 3), (U_n^{[2,2]}, 3, 1), (U_n^{[3,3]}, 1, 2)\}, \tag{92}$$

其中三个 $U_n^{[\eta_r,\phi_r]}$术语按照（88）式生成。作为示例，给定的实际（子分组、段）对（2, 1）按如下方式转换为置换后的对（1，3）。注意，在逐段置换 $P =$ （2，3，1）中，实际分段索引1位于第三个位置。因此， $\phi_r = 1$ 被转换为 $\phi_p = 3$。接下来，在对应于段1的置换˜$P_1 =$ （2，4，3，1）中，子分组索引2位于第一个位置。因此，$\eta_r = 2$ 被转换为 $\eta_p = 1$。

数据库接收到所有置换后的（更新、子分组、段）元组后，将构造置换后的更新向量，表示为：

$$\tilde{U}_n = \sum_{i=1}^{Pr} U_n^{[\eta_r^{[i]},\phi_r^{[i]}]} e_P((\phi_p^{[i]} - 1)\frac{P}{B} + \eta_p^{[i]}), \tag{93}$$

其中， e_p（$\phi_p^{[i]} - 1$）$\frac{P}{B} + \eta_p^{[i]}$是大小为 P的全零向量，其第（$\phi_p^{[i]} - 1$）$\frac{P}{B} + \eta_p^{[i]}$个位置为1。随后，将该向量与大小为 $\ell \times 1$ 的全一向量（即 1_ℓ）进行缩放，以便于后续计算。缩放置换更新向量表示为：

$$\hat{U}_n = \tilde{U}_n \otimes 1_\ell = \begin{bmatrix} \tilde{U}_n(1) \cdot 1_\ell \\ \vdots \\ \tilde{U}_n(P) \cdot 1_\ell \end{bmatrix}. \tag{94}$$

然后，数据库 n、 $n \in \{1, \ldots, N\}$ 使用式（72）中的组合加噪置换逆矩阵计算增量更新，如下：

$$\bar{U}_n = R_n \times \hat{U}_n, \tag{95}$$

其形式与式（58）中的存储相同。因此，时刻 t、 $S_n^{[t]}$的存储可更新为:

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n. \tag{96}$$

对于所考虑的示例，基于数据库在（92）中接收到的信息得到的经缩放的置换后的更新向量为:

$$\hat{U}_n = (U_n^{[2,1]} e_{12}(9) + U_n^{[2,2]} e_{12}(3) + U_n^{[3,3]} e_{12}(5)) \otimes 1_\ell = \begin{bmatrix} \begin{bmatrix} 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[2,2]} \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[3,3]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \end{bmatrix} \tag{97}$$

接着，数据库 n, $n \in \{1, \ldots, N\}$ 使用式 (79) 中的组合加噪置换逆矩阵计算增量更新，得到:

$$\bar{U}_n = R_n \times \hat{U}_n \tag{98}$$

$$= \begin{bmatrix} 0_{4\ell \times 4\ell} & 0_{4\ell \times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \end{bmatrix} \\ \begin{bmatrix} \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \end{bmatrix} & 0_{4\ell \times 4\ell} & 0_{4\ell \times 4\ell} \\ 0_{4\ell \times 4\ell} & \begin{bmatrix} 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n \\ \Gamma_n & 0_{\ell\times\ell} & 0_{\ell\times\ell} & 0_{\ell\times\ell} \\ 0_{\ell\times\ell} & 0_{\ell\times\ell} & \Gamma_n & 0_{\ell\times\ell} \end{bmatrix} & 0_{4\ell \times 4\ell} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ U_n^{[2,2]} \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[3,3]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \\ \begin{bmatrix} U_n^{[2,1]} \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \\ 0 \cdot 1_\ell \end{bmatrix} \end{bmatrix} \tag{99}$$

$$+ P_{\alpha_n}(\ell + 1)$$

$$= \left[0_\ell, \frac{U_n^{[2,1]}}{f_1-\alpha_n}, \ldots, \frac{U_n^{[2,1]}}{f_\ell-\alpha_n}, 0_{3\ell}, \frac{U_n^{[2,2]}}{f_1-\alpha_n}, \ldots, \frac{U_n^{[2,2]}}{f_\ell-\alpha_n}, 0_{4\ell}, \frac{U_n^{[3,3]}}{f_1-\alpha_n}, \ldots, \frac{U_n^{[3,3]}}{f_\ell-\alpha_n}, 0_\ell, \right]^T + P_{\alpha_n}(\ell+1) \tag{100}$$

$$= \left[0_\ell, \frac{\Delta_1^{[2,1]}}{f_1-\alpha_n}, \ldots, \frac{\Delta_\ell^{[2,1]}}{f_\ell-\alpha_n}, 0_{3\ell}, \frac{\Delta_1^{[2,2]}}{f_1-\alpha_n}, \ldots, \frac{\Delta_\ell^{[2,2]}}{f_\ell-\alpha_n}, 0_{4\ell}, \frac{\Delta_1^{[3,3]}}{f_1-\alpha_n}, \ldots, \frac{\Delta_\ell^{[3,3]}}{f_\ell-\alpha_n}, 0_\ell, \right]^T + P_{\alpha_n}(\ell+1), \tag{101}$$

其中， $P_{\alpha_n}(\ell+1)$ 是一个大小为 $L\times 1$ 的向量，其元素是 α_n 中次数为 $\ell+1$ 的多项式，式 (101) 由 [47] 中的引理 1 得出。注意，从式 (101) 可知，分段 1 的（真实）子分组 2、分段 2 的子分组 2 以及分段 3 的子分组 3 均已正确更新，同时未向任何数据库泄露这些真实索引。[11] 由于 $\bar{U}_n$ 与式 (58) 中的存储形式相同，因此可通过将 U_n 加到现有存储中来更新时刻 t 的存储。由此得到的写入成本为:

$$C_W = \frac{PrN(1+\log_q B+\log_q \frac{P}{B})}{L} = \frac{PrN(1+\log_q P)}{P^{\frac{N-4}{2}}} = \frac{2r(1+\log_q P)}{1-\frac{4}{N}}. \tag{102}$$

数据、分段内加噪置换反转矩阵和段间加噪置换反转矩阵的存储复杂度分别为 $O(L)$、 $O(\frac{L^2}{B})$ 和 $O(\ell^2B^2) = O(N^2B^2)$。因此，存储复杂度为 $\max\{O(\frac{L^2}{B}) O(N^2B^2)\}$。稀疏更新索引的信息泄露将在第4.2节中推导。

情况 4：在这种情况下，我们考虑采用编码存储和更小的置换反转矩阵来降低存储成本。此处同时考虑分段内置换和段间置换，以减少信息泄露。

初始化：数据库 n 中单个子分组 s、 $n\in\{1,\ldots,N\}$ 的存储方式如下:

$$S_n^{[s]} = \sum_{i=1}^{\ell}\frac{1}{\alpha_n^i}W_i^{[s]} + \sum_{i=0}^{2\ell}\alpha_n^i Z_i^{[s]}, \tag{103}$$

与情况 2 中使用的符号相同。因此，给定分段 j、 $j\in\{1,\ldots,B\}$ 的存储方式如下:

$$S_{n,j} = \begin{bmatrix} \sum_{i=1}^{\ell}\frac{1}{\alpha_n^i}W_i^{[1,j]} + \sum_{i=0}^{2\ell}\alpha_n^i Z_i^{[1,j]} \\ \vdots \\ \sum_{i=1}^{\ell}\frac{1}{\alpha_n^i}W_i^{[\frac{P}{B},j]} + \sum_{i=0}^{2\ell}\alpha_n^i Z_i^{[\frac{P}{B},j]} \end{bmatrix}, \tag{104}$$

其中 $W_i^{[k,j]}$ 是第 e i 个参数，属于分段 j 的子分组 k，而 $Z_i^{[k,j]}$ 是随机噪 e

[11] 注意 式 (101) 中的向量 $P_{\alpha_n}(\ell+1)$ 向数据库隐藏了这些非零更新。

符号。各分段按照从分段 1 到分段 B 的顺序依次堆叠。如情况3所述，协调器将分段内置换和段间ˆ置换 $P_1, \ldots, P_B$ 以及 P发送给用户，并将相应的加噪置换逆矩阵（由 $R_n^{[1]}, \ldots, R_n^{[B]}$ 和 ˆ R_n 给出）发送给数据库 $n, n \in \{1, \ldots, N\}$.˜。分段内置换 P_i 对应的加噪置换逆矩阵具有形式 (40)，而ˆ段间置换 P 对应的加噪置换逆矩阵具有如下形式，

$$\hat{R}_n = \hat{R} + \alpha_n^{\ell} Z, \tag{105}$$

ˆ ˆ 其中， R 是与 P 对应的置换反转矩阵， Z 是一个随机噪声矩阵，两者的大小均为 $B \times B$。对于图5中所考虑的示例，与 $P_1 = \{2, 4, 3, 1\}$、 $P_2 = \{1, 3, 2, 4\}$、 $P_3 =$ˆ $\{3, 1, 4, 2\}$ 和 $P = \{2, 3, 1\}$对应的加噪置换反转矩阵˜ ˜ ˜存储在数据库 n、 $n \in \{1, \ldots, N\}$ 中，具体如下：

$$R_n^{[1]} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[1]}, \quad R_n^{[2]} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[2]}, \quad R_n^{[3]} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} \bar{Z}^{[3]} \tag{106}$$

and

$$\hat{R}_n = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \alpha_n^{\ell} Z. \tag{107}$$

存储、置换和加噪置换逆矩阵初始化完成后，初始化阶段结束，协调器退出系统。

为便于进行下文读写阶段所述的计算，数据库按照情况3所述的方式计算组合加噪置换逆矩阵。数据库 n、 $n \in \{1, \ldots, N\}$ 对应的该矩阵如下：

$$R_n = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times (\hat{R}_n \otimes I_{\frac{P}{B}}) = \begin{bmatrix} R_n^{[1]} & & \\ & \ddots & \\ & & R_n^{[B]} \end{bmatrix} \times \begin{bmatrix} \hat{R}_n(1,1) I_{\frac{P}{B}} & \ldots & \hat{R}_n(1,B) I_{\frac{P}{B}} \\ \vdots & \vdots & \vdots \\ \hat{R}_n(B,1) I_{\frac{P}{B}} & \ldots & \hat{R}_n(B,B) I_{\frac{P}{B}} \end{bmatrix} \tag{108}$$

其中， $I_{\frac{P}{B}}$是大小为$\frac{P}{B} \times \frac{P}{B}$的单位矩阵。组合矩阵 R_n将 $R_n^{[i]}$置于（105）中 R 的第 i行中ˆ 1 所在的位置，并添加噪声。对于数据库 n，图5所示示例中的组合加噪置换逆矩阵为：

$n \in \{1, \ldots, N\}$ 为：

$$R_n = \begin{bmatrix} R_n^{[1]} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & R_n^{[2]} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & R_n^{[3]} \end{bmatrix} \times \begin{bmatrix} \hat{R}_n(1,1)I_4 & \hat{R}_n(1,2)I_4 & \hat{R}_n(1,3)I_4 \\ \hat{R}_n(2,1)I_4 & \hat{R}_n(2,2)I_4 & \hat{R}_n(2,3)I_4 \\ \hat{R}_n(3,1)I_4 & \hat{R}_n(3,2)I_4 & \hat{R}_n(3,3)I_4 \end{bmatrix} \tag{109}$$

$$= \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[1]} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[2]} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \alpha_n^\ell \bar{Z}^{[3]} \end{bmatrix}$$

$$\times \left(\begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{4\times4} \\ \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{4\times4} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}_{4\times4} & 0_{4\times4} \end{bmatrix} + \alpha_n^\ell \tilde{Z} \right) \tag{110}$$

$$= \begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} & 0_{4\times4} \end{bmatrix} + \alpha_n^\ell P_{\alpha_n}(\ell), \tag{111}$$

˜其中，(110) 中的 Z 是一个大小为 12×12 的随机噪声矩阵，由 R_n 的噪声分量ˆ产生；(111) 中的 $P_{\alpha_n}(\ell)$ 是一个大小为 12×12 的矩阵，其元素由 α_n 中次数为 ℓ 的多项式组成。注意，(111) 中的组合加噪置换逆矩阵会将每个 $R_n^{[i]}$ 放置在 (107) 中置换反转矩阵第 i 行中 1 所在的位置，适用于 $i = 1$、2、3，并添加了噪声。

读取阶段：如情况3中所述，数据库确定由 (η_p, ϕ_p) 给出的置换后的（子分组、分段）元组，这些元组对应于 Pr' 稀疏子数据包，并将其发送给用户。用户根据 (66) 和 (67) 获得 (η_p, ϕ_p) 元组的实际（子数据包、段）信息。对于图5中考虑的示例，情况3中考虑的 (68) 和 (69) 中的 (η_p, ϕ_p) 和 (η_r, ϕ_r) 对，对于情况 4 同样有效。

为了发送与置换后的（子分组、分段）对（η_p, ϕ_p）对应的子分组，数据库 n，$n \in \{1, \ldots, N\}$ 创建了如下查询：

$$Q_n^{[\eta_p,\phi_p]} = R_n(:, (\phi_p - 1)\frac{P}{B} + \eta_p). \tag{112}$$

对于（η_p, ϕ_p）$=(1, 3)$，相应的查询如下：

$$Q_n^{[1,3]} = R_n(:, 9) = \begin{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ 0_8 \end{bmatrix} + \alpha_n^{\ell} P_{\alpha_n}(\ell), \tag{113}$$

其中，$P_{\alpha_n}(\ell)$ 是一个大小为 12×1 的向量，其元素由关于 α_n、次数为 ℓ 的多项式构成。数据库将对应于（η_p, ϕ_p）的每个查询的答案发送如下：

$$A_n^{[\eta_p,\phi_p]} = S_n^T Q_n^{[\eta_p,\phi_p]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} W_k^{[\eta_r,\phi_r]} + P_{\alpha_n}(4\ell), \tag{114}$$

其中，$P_{\alpha_n}(4\ell)$ 是关于 α_n、次数为 4ℓ 的多项式。用户可以使用以下方式获取对应实际（子分组、分段）对（η_r, ϕ_r）的参数：

$$\begin{bmatrix} A_1^{[\eta_p,\phi_p]} \\ \vdots \\ A_N^{[\eta_p,\phi_p]} \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_1^{\ell}} & \ldots & \frac{1}{\alpha_1} & 1 & \alpha_1 & \ldots & \alpha_1^{4\ell} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\alpha_N^{\ell}} & \ldots & \frac{1}{\alpha_N} & 1 & \alpha_N & \ldots & \alpha_N^{4\ell} \end{bmatrix} \begin{bmatrix} W_{\ell}^{[\eta_r,\phi_r]} \\ \vdots \\ W_1^{[\eta_r,\phi_r]} \\ R_{0:4\ell} \end{bmatrix}, \tag{115}$$

其中 R_i 是式（114）中多项式的系数，前提是满足 $N = 5\ell + 1$。这将子分包化确定为 $\ell = \frac{N-1}{5}$。对于所考虑的示例，

$(\eta_p, \phi_p) = (1, 3)$ 的答案由数据库 $n, n \in \{1, \ldots, N\}$ 给出，

$$A_n^{[1,3]} = S_n^T Q_n^{[1,3]} = \begin{bmatrix} \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,1]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,1]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,1]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,1]} \end{bmatrix} \\ \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,2]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,2]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,2]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,2]} \end{bmatrix} \\ \begin{bmatrix} \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[1,3]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[1,3]} \\ \vdots \\ \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[4,3]} + \sum_{i=0}^{2\ell} \alpha_n^i Z_i^{[4,3]} \end{bmatrix} \end{bmatrix}^T \left(\begin{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ 0_8 \end{bmatrix} + \alpha_n^{\ell} P_{\alpha_n}(\ell) \right) \tag{116}$$

$$= \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} W_i^{[2,1]} + P_{\alpha_n}(4\ell). \tag{117}$$

用户可以利用在满足 $N = 5\ell + 1$ 时收到的 N 答案，获取分段 1 的（实际）第二个子数据包的参数（因为根据 (68) 和 (69)，与置换后的 $(\eta_p, \phi_p) = (1, 3)$ 对应的真实索引为 $(\eta_r, \phi_r) = (2, 1)$。）。这定义了情况 4 的子数据包划分方式，即 $\ell = \frac{N-1}{5}$。由此得到的读取成本为：

$$C_R = \frac{Pr'(N + \log_q B + \log_q \frac{P}{B})}{L} = \frac{Pr'(N + \log_q P)}{P^{\frac{N-1}{5}}} = \frac{5r'(1 + \frac{\log_q P}{N})}{1 - \frac{1}{N}}. \tag{118}$$

写入阶段：与情况3类似，用户选择更新量最大的 Pr 个子数据包，并令 $(\eta_r^{[i]}, \phi_r^{[i]})$，$i \in \{1, \ldots, Pr\}$ 表示第 i 个所选子数据包的实际（子数据包、段）信息。对于每个这样的子数据包，用户生成一个合并更新（单个符号），其定义如下：

$$U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]} = \sum_{k=1}^{\ell} \frac{1}{\alpha_n^k} \Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]} + Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \tag{119}$$

其中，$\Delta_k^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ 是分段 $\phi_r^{[i]}$的子分组 $\eta_r^{[i]}$中第 k个参数的更新，$Z^{[\eta_r^{[i]}, \phi_r^{[i]}]}$ 是一个随机噪声符号。对于 $i \in \{1, \ldots, Pr\}$，用户针对第 i个稀疏子分组发送由$(U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}, \eta_p^{[i]}, \phi_p^{[i]})$给出的置换后的（更新，子分组，分段）元组，其中 $U_n^{[\eta_r^{[i]}, \phi_r^{[i]}]}$是式（119）形式的子分组的合并更新，$\eta_p^{[i]}$是通过 $\eta_p^{[i]} = \tilde{P}_{\phi_r^{[i]}}^{-1}(\eta_r^{[i]})$得到的置换后的子分组索引，$\phi_p^{[i]}$是通过 $\phi_p^{[i]} = \hat{P}^{-1}(\phi_r^{[i]})$得到的置换后的分段索引；所用符号与情况3的描述相同。对于所考虑的示例，假设某用户希望更新实际（子分组，分段）对$\eta_r(2, 1)$、$(2, 2)$、$(3, 3)\}$。根据给定的段内置换 $\tilde{P}_1 = (2, 4, 3, 1)$、$\tilde{P}_2 = (1, 3, 2, 4)$、$\tilde{P}_3 = (3, 1, 4, 2)$以及段间置换

ˆ给定为 $P = (2,3,1)$，用户向数据库 n,$n \in \{1,\ldots,N\}$发送以下（置换后的）信息，如情况3所述，

$$(U_n, \eta_p, \phi_p) = \{(U_n^{[2,1]}, 1, 3), (U_n^{[2,2]}, 3, 1), (U_n^{[3,3]}, 1, 2)\}. \tag{120}$$

根据置换稀疏更新元组（$U_n^{[\eta_r^{[i]},\phi_r^{[i]}]}$、$\eta_p^{[i]}$、$\phi_p^{[i]}$），$i \in \{1,\ldots,Pr\}$ 收到后，数据库 n、$n \in \{1,\ldots,N\}$ 构造出式（93）所示的置换后的更新向量。然后，数据库 n、$n \in \{1,\ldots,N\}$ 计算得到逆置换增量更新：

$$\bar{U}_n = R_n \tilde{U}_n, \tag{121}$$

其形式与式（104）中的存储相同。因此，时刻 t 的存储可更新为：

$$S_n^{[t]} = S_n^{[t-1]} + \bar{U}_n. \tag{122}$$

对于 t 所考虑的示例，式（93）中的置换后的更新向量为 ，

$$\tilde{U}_n = [0, 0, U_n^{[2,2]}, 0, U_n^{[3,3]}, 0, 0, 0, U_n^{[2,1]}, 0, 0, 0]^T. \tag{123}$$

然后，每个数据库计算得到逆置换增量更新：

$$\bar{U}_n = R_n \tilde{U}_n = \left(\begin{bmatrix} 0_{4\times4} & 0_{4\times4} & \begin{bmatrix} 0&0&0&1\\1&0&0&0\\0&0&1&0\\0&1&0&0 \end{bmatrix} \\ \begin{bmatrix} 1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1 \end{bmatrix} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & \begin{bmatrix} 0&1&0&0\\0&0&0&1\\1&0&0&0\\0&0&1&0 \end{bmatrix} & 0_{4\times4} \end{bmatrix} + \alpha_n^{\ell} P_{\alpha_n}(\ell) \right) \times \begin{bmatrix} 0\\0\\U_n^{[2,2]}\\0\\U_n^{[3,3]}\\0\\0\\0\\U_n^{[2,1]}\\0\\0\\0 \end{bmatrix} \tag{124}$$

$$= [0, U_n^{[2,1]}, 0, 0, 0, U_n^{[2,2]}, 0, 0, 0, 0, U_n^{[3,3]}, 0]^T + P_{\alpha_n}(2\ell) \tag{125}$$

$$= \left[0, \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,1]}, 0, 0, 0, \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[2,2]}, 0, 0, 0, 0, \sum_{i=1}^{\ell} \frac{1}{\alpha_n^i} \Delta_i^{[3,3]}, 0\right]^T + P_{\alpha_n}(2\ell) \tag{126}$$

其中 $P_{\alpha_n}(2\ell)$ 为 一个大小为 12×1 的向量，由 α_n 中次数为 2ℓ 的多项式组成。注意 e

（真实）第 1 段的第 2 个子分组、第 2 段的第 2 个子分组和第 3 段的第 3 个子分组（$(\eta_r, \phi_r) = \{(2,1),(2,2),(3,3)\}$）已正确放置在 (126) 中，且未向数据库泄露真实索引。[12] 由于 (126) 中的增量更新与 (104) 形式相同，因此可直接将其添加到现有存储中，以获得更新后的存储。写入成本为：

$$C_W = \frac{PrN(1+\log_q B+\log_q \frac{P}{B})}{L} = \frac{PrN(1+\log_q P)}{P^{\frac{N-1}{5}}} = \frac{5r(1+\log_q P)}{1-\frac{1}{N}}. \tag{127}$$

数据、所添加的噪声以及段间置换逆矩阵的存储复杂度分别为 $O(P) = O(\frac{L}{N})$、$O(\frac{P^2}{B}) = O(\frac{L^2}{N^2B})$和 $O(B^2)$。因此，存储复杂度为 $\max\{O(\frac{L^2}{N^2B}) O(B^2)\}$。稀疏更新索引的信息泄露将在第4.2节中推导。

4.2 信息泄露

本节针对给定的联邦学习设置，量化四种情况下的信息泄露。该设置包含 N 个数据库、P 个子数据包、 B 个分段，以及分别由 r、 r'给定的上行链路和下行链路稀疏化率。在所提出的方案中，用户在读取阶段不会向数据库发送任何信息，而由数据库确定稀疏集合中的子数据包并发送给用户。因此，读取阶段不存在信息泄露。在写入阶段，用户向数据库发送 Pr 个置换后的（更新、子数据包、分段）元组，由此允许泄露有关某个用户更新的稀疏子数据包索引的特定数量的信息。

给定用户上传的内容中有两类信息会泄露用户本地数据的信息，即：1）稀疏更新的数值；2）稀疏更新的位置。请注意，在所提出的方案中，所有四种情况下发送给数据库的所有合并稀疏更新都添加了一个随机噪声符号。因此，根据香农一次一密定理，用户发送给所有数据库的合并更新数值都是随机噪声符号，且与其中包含的稀疏更新数值相互独立。因此，在本工作中，稀疏更新数值所泄露的信息量为零。

在时间 t，给定用户发送的置换后的（更新，子分组，分段）元组集合中，只有（子分组，分段）对可能包含有关稀疏更新实际位置的信息，因为更新分量只是与稀疏更新实际位置无关的随机噪声。令 $Y^{[t]}$ 为给定用户在时间 t发送的置换后的（子分组，分段）对所对应的 Pr 子分组索引集合。令 $X^{[t]}$ 为用户在时间 t选择用于更新的模型 Pr 稀疏子分组的真实索引集合。因此，在时间 t 泄露给数据库的有关稀疏更新实际位置的信息量可通过以下方式量化：

[12]稀疏更　　新在式 (126) 第一部分中的内容通过噪声向量 $P_{\alpha_n}(2\ell)$ 对数据库隐藏　　.

$X^{[t]}$ 与 $Y^{[t]}$之间的互信息，即 $I(X^{[t]};Y^{[t]})$。本节将量化四种情况中的这种互信息。

情况 1 和 2：在情况 1 和 2 中，置换仅发生在各分段内部，不会发生在分段之间。为简化符号，以下计算中省略时间索引，并假设每个 X 和 Y 分别对应时间 t 的真实量和置换后的量。为了量化 $I(X;Y)$，我们首先推导以下条件概率。

$$P(X=x|Y=y) = \frac{\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} P(X=x,Y=y,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)}{P(Y=y)} \tag{128}$$

$$= \frac{\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} P(Y=y|X=x,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)P(X=x)\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i)}{P(Y=y)} \tag{129}$$

$$= \frac{P(X=x)\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \mathbf{1}_{\{Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}}\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i)}{P(Y=y)} \tag{130}$$

$$= \begin{cases} \frac{P(X=x)\prod_{i=1}^{B}\hat{y}_i!\prod_{i=1}^{B}(\frac{P}{B}-\hat{y}_i)!}{(\frac{P}{B})!^B P(Y=y)}, & \text{for } x,y:\hat{x}_i=\hat{y}_i,\ \forall\ i \\ 0, & \text{otherwise} \end{cases} \tag{131}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\prod_{i=1}^{B}\frac{\hat{y}_i!(\frac{P}{B}-\hat{y}_i)!}{(\frac{P}{B})!}, & \text{对于 } x,y:\hat{x}_i=\hat{y}_i,\ \forall\ i \\ 0, & \text{否则} \end{cases} \tag{132}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\prod_{i=1}^{B}\frac{1}{(P/B\ \cdot y_i)}, & \text{对于 } x、\ y:\hat{x}_i=\hat{y}_i、\ \forall\ i \\ 0, & \text{否则} \end{cases} \tag{133}$$

其中，$\hat{x}_i$ 和$\hat{y}_i$ 分别表示分段 i中实际稀疏子数据包和置换后的稀疏子数据包的数量；式（129）是根据B 个分段的置换与稀疏更新的实际位置相互独立这一性质得到的。式（131）通过计算从给定的 $X=x$中得到给定 $Y=y$ 的所有可能置换的数量推导得出。接下来，我们计算概率，

$$P(Y=y) = \sum_{\tilde{p}_1,\ldots,\tilde{p}_B}\sum_{x} P(Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B) \tag{134}$$

$$= \sum_{\tilde{p}_1,\ldots,\tilde{p}_B}\sum_{x} P(Y=y|X=x,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)P(X=x)\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i) \tag{135}$$

$$= \sum_{x} P(X=x)\sum_{\tilde{p}_1,\ldots,\tilde{p}_B} \mathbf{1}_{\{Y=y,X=x,\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}}\prod_{i=1}^{B} P(\tilde{P}_i=\tilde{p}_i) \tag{136}$$

$$= \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)\frac{\prod_{i=1}^{B}\hat{y}_i!\prod_{i=1}^{B}(\frac{P}{B}-\hat{y}_i)!}{\frac{1}{(\frac{P}{B})!^B}} \tag{137}$$

$$= \prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x) \tag{138}$$

对于每个满足 $\sum_{i=1}^{B} \hat{}y_i = Pr$的 y。因此，由式（133）可得，

$$P(X=x|Y=y) = \begin{cases} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)}, & \text{for } x,y:\hat{x}_i=\hat{y}_i,\ \forall i \\ 0, & \text{otherwise} \end{cases} \tag{139}$$

。然后，

$$H(X|Y=y) = -\sum_{x} P(X=x|Y=y)\log P(X=x|Y=y) \tag{140}$$

$$= -\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)}, \tag{141}$$

由此可得，

$$H(X|Y) = \sum_{y} P(Y=y)H(X|Y=y) \tag{142}$$

$$= -\sum_{y} \left(\prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x) \right) \times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \tag{143}$$

$$= -\sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \prod_{i=1}^{B} \binom{P/B}{\hat{y}_i} \left(\prod_{i=1}^{B} \frac{1}{\binom{P/B}{\hat{y}_i}} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x) \right) \times \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \tag{144}$$

$$= -\sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr} \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x) \sum_{x:\hat{x}_i=\hat{y}_i,\forall i} \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \log \frac{P(X=x)}{\sum_{x:\hat{x}_i=\hat{y}_i,\forall i} P(X=x)} \tag{145}$$

$$= -\sum_{\hat{x}:\sum_{i=1}^{B}\hat{x}_i=Pr} \sum_{x\in\hat{x}} P(X=x) \sum_{x\in\hat{x}} \frac{P(X=x)}{\sum_{x\in\hat{x}} P(X=x)} \log \frac{P(X=x)}{\sum_{x\in\hat{x}} P(X=x)}, \tag{146}$$

其中，$\hat{}y = (\hat{}y_1, \ldots, \hat{}y_B)$ 和 $\hat{}x = (\hat{}x_1, \ldots, \hat{}x_B)$ 分别表示每个 B 分段中 $\hat{}$置换后的稀疏子数据包数量和真实稀疏子分组数量的特定实现值，而 $x \in X$对应于 X 的每个实现值，这些实现值会导致 B 分段中出现 $\hat{}x_1, \ldots, \hat{}x_B$ 个稀疏子数据包。一般而言，随机变量 $X = (X_1, \ldots, X_B)$ 表示用户在每个 B 分段中更新的（真实）稀疏子数据包数量，这些数量的总和为 Pr。请注意，在此计算中，我们不假设 X 具有任何特定分布。

注意到 $\sum_{x\in\hat{x}} P(X=x) = P(X=\hat{x})$，式 (146) 中的条件熵可简化为：

$$H(X|Y) = \sum_{\hat{x}:\sum_{i=1}^{B}\hat{x}_i=Pr} P(\hat{X}=\hat{x})\log P(\hat{X}=\hat{x}) - \sum_{\hat{x}:\sum_{i=1}^{B}\hat{x}_i=Pr}\sum_{x\in\hat{x}} P(X=x)\log P(X=x) \tag{147}$$

$$= -H(\hat{X}) + H(X), \tag{148}$$

由于基于给定的上行稀疏化率，X 的所有实现值都满足 $\sum_{i=1}^{B}\hat{x}_i = Pr$。因此，

$$I(X;Y) = H(X) - H(X|Y) = H(\hat{X}) = H(\hat{X}_1,\ldots,\hat{X}_B). \tag{149}$$

情况3和情况4：在情况3和情况4中，我们同时考虑分段内及分段之间的置换，以进一步减少信息泄露。回顾一下，B 分段内的置换记为 $\{\tilde{P}_i\}_{i=1}^{B}$，而分段之间的置换记为 $\hat{P}$。与上述计算类似，为了计算信息泄露 $I(X;Y)$，我们首先计算如下条件分布，

$$P(X=x|Y=y) = \Bigg(\sum_{\tilde{p}_1,\ldots,\tilde{p}_B}\sum_{\hat{p}} P(Y=y|X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B) \times P(X=x)P(\hat{P}=\hat{p})\prod_{i=1}^{B}P(\tilde{P}_i=\tilde{p}_i)\Bigg)/P(Y=y) \tag{150}$$

$$= \frac{P(X=x)}{P(Y=y)}\frac{1}{B!}\frac{1}{(P/B)!^B}\sum_{\tilde{p}_1,\ldots,\tilde{p}_B}\sum_{\hat{p}}\mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}} \tag{151}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\frac{1}{B!}\frac{1}{(P/B)!^B}\prod_{i=1}^{B}\hat{y}_i!(P/B-\hat{y}_i)!\prod_{i=1}^{B}K_{\hat{y}_i}!, & \text{for } x,y:\{\hat{x}\}=\{\hat{y}\} \\ 0, & \text{otherwise} \end{cases} \tag{152}$$

$$= \begin{cases} \frac{P(X=x)}{P(Y=y)}\frac{1}{B!}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}K_{\hat{y}_i}!, & \text{for } x,y:\{\hat{x}\}=\{\hat{y}\} \\ 0, & \text{otherwise} \end{cases} \tag{153}$$

其中 $K_{\hat{y}_i} = \sum_{j=1}^{B}\mathbf{1}_{\hat{y}_j=\hat{y}_i}$（即稀疏子数据包数量与分段 i 相同的分段数），符号 $\{\hat{x}\} = \{\hat{y}\}$ 表示两个集合 $\hat{x}$ 和 $\hat{y}$ 相同，而不论其顺序如何。也就是说，若 $\{\hat{x}\} = \{\hat{y}\}$，则对每个 $\hat{x}_i \in \hat{x}$，都存在某个 $\hat{y}_j \in \hat{y}$，使得 $\hat{x}_i = \hat{y}_j$，反之亦然。接下来，对于任意满足 $\sum_{i=1}^{Pr}\hat{y}_i = Pr$ 的 y，我们计算 $P(Y=y)$，

$$P(Y=y) = \sum_{x}\sum_{\tilde{p}_1,\ldots,\tilde{p}_B}\sum_{\hat{p}} P(Y=y|X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B)$$

$$\times P(X=x)P(\hat{P}=\hat{p})\prod_{i=1}^{B}P(\tilde{P}_i=\tilde{p}_i) \tag{154}$$

$$=\sum_x P(X=x)\frac{1}{B!}\frac{1}{(P/B)!^B}\sum_{\tilde{p}_1,\ldots,\tilde{p}_B}\sum_{\hat{p}}\mathbf{1}_{\{Y=y,X=x,\hat{P}=\hat{p},\tilde{P}_1=\tilde{p}_1,\ldots,\tilde{P}_B=\tilde{p}_B\}} \tag{155}$$

$$=\frac{1}{B!}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}K_{\hat{y}_i}!\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x) \tag{156}$$

因此，由式（153）可得，

$$P(X=x|Y=y)=\begin{cases}\frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)}, & \text{for } x,y:\{\hat{x}\}=\{\hat{y}\}\\ 0, & \text{otherwise}\end{cases}. \tag{157}$$

然后，

$$H(X|Y=y)=-\sum_x P(X=x|Y=y)\log P(X=x|Y=y) \tag{158}$$

$$=-\sum_{x:\{\hat{x}\}=\{\hat{y}\}}\frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)}\log\frac{P(X=x)}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)} \tag{159}$$

$$=\log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\right)-\frac{1}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)}\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\log P(X=x), \tag{160}$$

由此可得，

$$H(X|Y)=\sum_y P(Y=y)H(X|Y=y) \tag{161}$$

$$=\sum_y\frac{1}{B!}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}K_{\hat{y}_i}!\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\right)$$
$$-\sum_y\frac{1}{B!}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}K_{\hat{y}_i}!\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\frac{1}{\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)}$$
$$\times\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\log P(X=x) \tag{162}$$

$$=\frac{1}{B!}\sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr}\prod_{i=1}^{B}\binom{P/B}{\hat{y}_i}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}K_{\hat{y}_i}!\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\log\left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\right)$$
$$-\frac{1}{B!}\sum_{\hat{y}:\sum_{i=1}^{B}\hat{y}_i=Pr}\prod_{i=1}^{B}\binom{P/B}{\hat{y}_i}\prod_{i=1}^{B}\frac{1}{\binom{P/B}{\hat{y}_i}}K_{\hat{y}_i}!\sum_{x:\{\hat{x}\}=\{\hat{y}\}}P(X=x)\log P(X=x) \tag{163}$$

$$= \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \prod_{i=1}^{B} K_{\hat{y}_i}! \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$

$$- \frac{1}{B!} \sum_{\hat{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \prod_{i=1}^{B} K_{\hat{y}_i}! \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{164}$$

$$= \frac{1}{B!} \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \frac{B!}{\prod_{i=1}^{B} K_{\hat{y}_i}!} \prod_{i=1}^{B} K_{\hat{y}_i}! \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$

$$- \frac{1}{B!} \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \frac{B!}{\prod_{i=1}^{B} K_{\hat{y}_i}!} \prod_{i=1}^{B} K_{\hat{y}_i}! \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{165}$$

$$= \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$

$$- \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x), \tag{166}$$

其中，$\tilde{y}$ 在式（165）中引入，$\tilde{x}$ 是相应随机变量 $\tilde{Y}$ 和 X 的实现值，分别表示 $\hat{y}$ 和 $\hat{x}$ 的所有不同集合。例如，如果 $B = 2$，则 (1, 2) 和 (2,1) 被视为 $\hat{y}$ （或 $\hat{x}$）的两个不同实现值，而对于 $\tilde{y}$ （或 $\tilde{x}$），它们是相同的实现值。此外，

$$P(\tilde{X} = \tilde{x}) = \sum_{x \in \tilde{x}} P(X=x) \tag{167}$$

$$= \sum_{\text{all permutations of } x \in \hat{x}} P(X=x) \tag{168}$$

采用此符号，上述熵可进一步简化为：

$$H(X|Y) = \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log \left(\sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \right)$$

$$- \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} \sum_{x:\{\hat{x}\}=\{\hat{y}\}} P(X=x) \log P(X=x) \tag{169}$$

$$= \sum_{\tilde{y}:\sum_{i=1}^{B} \hat{y}_i = Pr} P(\tilde{X}=\tilde{y}) \log P(\tilde{X}=\tilde{y}) + H(X) \tag{170}$$

$$= -H(\tilde{X}) + H(X). \tag{171}$$

因此，信息泄露量为：

$$I(X;Y) = H(X) - H(X|Y) = H(\tilde{X}) = H(\tilde{X}_1, \ldots, \tilde{X}_B). \tag{172}$$

5 结论

在这项工作中，我们研究了采用 Top r 稀疏化的隐私联邦学习问题。在采用 Top r 稀疏化的联邦学习中，稀疏更新的数值和位置会泄露有关用户隐私数据的信息。我们提出了四种具有不同特性的方案，以在采用 Top r 稀疏化的联邦学习中，避免向数据库泄露稀疏更新的数值或位置。这些方案采用置换技术，但需要较高的存储成本。为此，我们利用模型分割机制对这些方案进行了推广，以一定程度的信息泄露为代价降低存储成本。总的来说，这项工作展示了采用 Top r 稀疏化的隐私联邦学习中，通信成本、存储复杂度与信息泄露之间的权衡。

参考文献

[1] H. B. McMahan、E. Moore、D. Ramage、S. Hampson 和 B. A. y Arcas。从去中心化数据中高效通信地学习深度网络。AISTATS，2017年4月。[2] Q. Yang、Y. Liu、T. Chen 和 Y. Tong。联邦机器学习：概念与应用。ACM智能系统与技术汇刊，10(2):1–19，2019年1月。[3] M. Nasr、R. Shokri 和 A. Houmansadr。深度学习的全面隐私分析：针对集中式和联邦学习的被动与主动白盒推理攻击。发表于 IEEE安全与隐私研讨会，2019年5月。[4] R. Shokri、M. Stronati、C. Song 和 V. Shmatikov。针对机器学习模型的成员推理攻击。发表于 IEEE安全与隐私研讨会，2017年5月。[5] L. Melis、C. Song、E. De Cristofaro 和 V. Shmatikov。利用协作学习中意外泄露的特征。发表于 IEEE安全与隐私研讨会，2019年5月。[6] N. Carlini、C. Liu、U. Erlingsson、J. Kos 和 D. Song。《秘密共享者》：评估和测试神经网络中的意外记忆。发表于 USENIX，2019年4月。[7] J. Geiping、H. Bauermeister、H. Droge 和 M. Moeller。反演梯度——在联邦学习中破坏隐私有多容易？在线发表于 arXiv:2003.14053。[8] L. Zhu、Z. Liu 和 S. Han。梯度导致的深度信息泄露。发表于 NeurIPS，2019年12月。[9] Z. Wang、M. Song、Z. Zhang、Y. Song、Q. Wang 和 H. Qi。超越类别代表：联邦学习中的用户级隐私泄露。发表于 IEEE计算机通信大会，2019年4月至5月。

[10] K. Bonawitz、V. Ivanov、B. Kreuter、A. Marcedone、H. B. McMahan、S. Patel、D. Ram-age、A. Segal 和 K. Seth。面向隐私保护机器学习的实用安全聚合。发表于 CCS，2017年10月。[11] C. Dwork 和 A. Roth。差分隐私的算法基础。《理论计算机科学基础与趋势》，9(3-4):211–407，2014年8月。[12] H. Ono 和 T. Takahashi。本地隐私分布式强化学习。在线发表于 arXiv:2001.11718。[13] Y. Li、T. Chang 和 C. Chi。具有差分隐私的安全联邦平均算法。IEEE MLSE，2020年9月。[14] N. Agarwal、A. Suresh、F. Yu、S. Kumar 和 H. B. McMahan。cpSGD：高通信效率且具有差分隐私的分布式SGD。发表于 NeurIPS，2018年12月。[15] B. Balle、G. Barthe 和 M. Gaboardi。子采样隐私放大：通过耦合和散度进行紧致分析。发表于 NeurIPS，2018年12月。[16] H. Ono 和 T. Takahashi。差分隐私跨孤岛联邦学习。在线发表于 arXiv:2007.05553。[17] H. B. McMahan、D. Ramage、K. Talwar 和 L. Zhang。学习差分隐私循环语言模型。发表于 ICLR，2018年5月。[18] S. Asoodeh 和 F. Calmon。差分隐私联邦学习：信息论视角。发表于 ICML-FL，2020年7月。[19] U. Erlingsson、V. Feldman、I. Mironov、A. Raghunathan、K. Talwar 和 A. Thakurta。洗牌放大：通过匿名性从本地差分隐私走向中心化差分隐私。在线发表于 arXiv:1811.12469。[20] B. Balle、J. Bell、A. Gascon、 和 K. Nissim。洗牌模型的隐私毯。发表于 CRYPTO，2019年8月。[21] A. Girgis、D. Data、S. Diggavi、P. Kairouz 和 A. Suresh。联邦学习中的差分隐私洗牌模型。发表于 AISTAT，2021年4月。[22] R. C. Geyer、T. Klein 和 M. Nabi。差分隐私联邦学习：客户端级视角。发表于 NeurIPS，2017年12月。[23] S. Ulukus、S. Avestimehr、M. Gastpar、S. A. Jafar、R. Tandon 和 C. Tian。私有检索、计算与学习：近期进展与未来挑战。《IEEE通信选定领域期刊》，40(3):729–748，2022年3月。

[24] C. Naim、R. D'Oliveira 和 S. El Rouayheb。私有多组聚合。IEEE ISIT，2021年7月。[25] D. Basu、D. Data、C. Karakus 和 S. Diggavi。Qsparse-local-SGD：结合量化、稀疏化和本地计算的分布式SGD。IEEE JSAIT，1(1):217–226，2020年5月。[26] A. Reisizadeh、A. Mokhtari、H. Hassani、A. Jadbabaie 和 R. Pedarsani。Fedpaq：一种采用周期性平均和量化的高通信效率联邦学习方法。发表于AISTATS，2020年8月。[27] D. Alistarh、D. Grubic、J. Li、R. Tomioka 和 M. Vojnovic。QSGD：通过梯度量化和编码实现高通信效率的SGD。发表于NeurIPS，2017年12月。[28] N. Shlezinger、M. Chen、Y. Eldar、H. Poor 和 S. Cui。具有量化约束的联邦学习。发表于IEEE ICASSP，2020年5月。[29] C. Niu、F. Wu、S. Tang、L. Hua、R. Jia、C. Lv、Z. Wu 和 G. Chen。移动客户端上的十亿规模联邦学习：一种具有可调隐私的子模型设计。发表于MobiCom，2020年4月。[30] M. Kim 和 J. Lee。联邦子模型学习中的信息论隐私。ICT Express，2022年2月。[31] C. Niu、F. Wu、S. Tang、L. Hua、R. Jia、C. Lv、Z. Wu 和 G. Chen。安全联邦子模型学习。在线发表于arXiv:1911.02254。[32] Z. Jia 和 S. A. Jafar。 X-安全、 T-私有联邦子模型学习，具有弹性丢弃容错能力。IEEE信息论汇刊，68(8):5418–5439，2022年8月。[33] S. Vithana 和 S. Ulukus。高效私有联邦子模型学习。发表于IEEE ICC，2022年5月。[34] Z. Jia 和 S. A. Jafar。 X-安全、 T-私有联邦子模型学习。发表于IEEE ICC，2021年6月。[35] S. Vithana 和 S. Ulukus。具有存储受限数据库的私有读取更新写入（PRUW）。发表于IEEE ISIT，2022年6月。[36] S. Vithana 和 S. Ulukus。带稀疏化的私有联邦子模型学习。发表于IEEE ITW，2022年11月。[37] S. Vithana 和 S. Ulukus。联邦子模型学习中私有读取更新写入的率失真折衷。发表于Asilomar会议，2022年10月。

[38] J. Wangni、J. Wang 等。用于高通信效率分布式优化的梯度稀疏化。发表于NeurIPS，2018年12月。[39] S. Li、Q. Qi 等。GGS：用于边缘计算中联邦学习的通用梯度稀疏化。发表于IEEE ICC，2020年6月。[40] P. Han、S. Wang 和 K. Leung。用于高效联邦学习的自适应梯度稀疏化：一种在线学习方法。发表于IEEE ICDCS，2020年11月。[41] S. Shi、K. Zhao、Q. Wang、Z. Tang 和 X. Chu。分布式SGD中高通信效率梯度稀疏化的收敛性分析。发表于IJCAI，2019年8月。[42] D. Alistarh、T. Hoefler、M. Johansson、S. Khirirat、N. Konstantinov 和 C. Renggli。稀疏化梯度方法的收敛性。发表于NeurIPS，2018年12月。[43]Y. Sun、S. Zhou、Z. Niu 和 D. Gunduz。用于无线联邦学习中高效空中模型聚合的时间相关稀疏化。在线发表于arXiv:2202.08420。[44] L. Barnes、H. Inan、B. Isik 和 A. Ozgur。rTop-k：一种分布式SGD的统计估计方法。IEEE JSAIT，1(3):897–907，2020年11月。[45]E. Ozfatura、K. Ozfatura 和 D. Gunduz。用于高通信效率联邦学习的时间相关稀疏化。发表于IEEE ISIT，2021年7月。[46] Z. Jia、H. Sun 和 S. A. Jafar。 X-安全、 T-私有信息检索的跨子空间对齐与渐近容量。IEEE信息论汇刊，65(9):5783–5798，2019年9月。[47] S. Vithana 和 S. Ulukus。联邦子模型学习（FSL）中的私有读取更新写入（PRUW）：有无稀疏化的高通信效率方案。在线发表于arXiv:2209.04421。