I. SUPPLEMENTARY

A. Horizontally-Federated XGBoost (Homogeneous SecureBoost)

In the scenario of horizontal federation learning, participants are divided into a central server and several clients. All the clients hold different training samples but share the same set of features and class labels, which is considered homogeneous. All the clients train the same tree model. The central server does not hold any dataset, it is only responsible for aggregate the gradient passed by all the clients and broadcast to the clients what the best global split point is now.

As can be observed from XGBoost algorithm principle and formulas, it is easy for XGBoost to be adapted to the setting of federated learning for the following reasons.

- The evaluation of split candidates and the calculation of the optimal weight of leaf only depends on the g_i and h_i.
- The calculation of g_i and h_i only relies on each sample's true labels and predicted labels.
- Each party has its own labels, so it is easy for each party to compute its own gradients histogram for the feature bins.

So clients only need to send their feature bins and each bin's encrypted histogram information to the central server. And the central server only need to tranverse all the bins from all the clients, decrypt the histograms and calculate the global best split point. The algorithm 1 and Fig. 1 visualize some crucial procedures.

B. Vertically-Federated XGBoost (Heterogeneous SecureBoost)

In the scenario of vertical federation learning, there is no central server. All participants are clients and each client has its own training dataset. Opposite to the data partition method of horizontal federated scenario, all the parties in vertically-federated setting share the same samples set but hold different features space. Different from the dataset schema in horizontal federation, only one client owns the labels, with the rest of the clients just having the features. The client which owns the labels is called $guest$ party, while other clients are called $host$ parties. $Guest$ party plays a similar role as central server in vertically-ferderated learning.

For the $guest$ party, since it holds the class labels, so it is responsible for calculating the gradient values of all samples, encrypt them and send them to all $host$ parties. Besides, the $guest$ party also has the duty to aggregate all the feature bins from the $host$ parties, decipher the gradient histograms, tranverse them, then figure out the best split point and the corresponding feature. For $host$ parties, its main function is to compute their own feature bins and local gradient histogram based on the encrypted gradient values of all samples sent by the $guest$ party (as shown in Algorithm 2). When broadcasted by the $guest$ party about the best splitting feature, the $host$ party which hold the best splitting feature should find the corresponding threshold value. The framework of the function of the $host$ and $guest$ parties are presented in Fig. 2.

Algorithm 1: Implementation of XGBoost-based horizontal FL

Input: N, the number of the clients, with the i^{th} client holds n_i instance spaces
Input: d, feature dimension
Input: x, the dataset matrix
Output: the best split point for the current instance space

/*On clients*/
for *each client* $i = 1$ **to** $N - 1$ **do**
  Propose each feature's values by percentiles to form feature bins
  for *each feature bin* **do**
    Accumulate the g, h of all sample spaces in this feature bin to get G, H
  end
end
/*On central server */
for *each client* $i = 1$ **to** $N - 1$ **do**
  for *each feature* $m = 1$ **to** $d - 1$ **do**
    g_l = g_l + Decrypt(G feature $bins$)
    h_l = h_l + Decrypt(H feature $bins$)
    g_r = $g - g_l$, $h_r = h - h_l$
    Score = Max(Score, $\frac{1}{2}[\frac{g_l^2}{h_l+\lambda} + \frac{g_r^2}{h_r+\lambda} - \frac{g^2}{h+\lambda}] - \gamma$)
  end
end
Broadcast the m_{opt} and the corresponding threshold value to all clients to split

Algorithm 2: A $host$ party calculates local feature bins and gradient histogram

Input: I, instance space of the current node
Input: d, feature dimension
Input: L, the number of bins of features, or the reciprocal of the quantile.
Input: $x \in R^{n*d}$, the dataset matrix for the party, n is the number of samples
Input: $\{< g >, < h >\} \in R^n$, obtaining from the active party through its true labels and predicted labels
Output: $\{G, H\} \in R^{d*L}$, the gradient histogram information

for *each client* $k = 1$ **to** $d - 1$ **do**
  Propose S_k=$s_{k1}, s_{k2}, ..., s_{kL}$ by percentiles on feature k.
end
for *each feature* k **to** $d_i - 1$ **do**
  $s_{k(-1)} = -\infty$, $h_{KL} = -\infty$
  for *each feature bins* v **to** L **do**
    $G_{kv} = \sum_{i \in i | s_{kv} >= x_{ik} > s_{k(v-1)}} {}^{< g_i >, i \in I}$,
    $H_{kv} = \sum_{i \in i | s_{kv} >= x_{ik} > s_{k(v-1)}} {}^{< h_i >, i \in I}$,
  end
end
Send G, H to the active party for aggregation

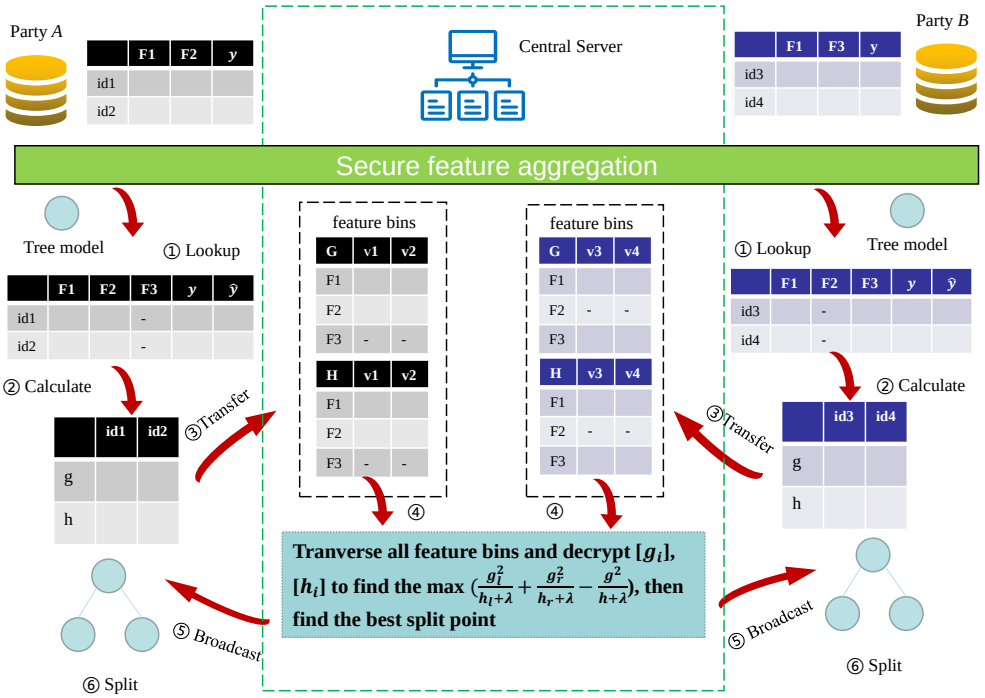

Fig. 1. An overview of how horizontal-SecureBoost algorithm works based on secure aggregation of feature spaces. Our first goal is to align the data feature space cross all participants to build a HFL federated model. When the data are extracted separately for heart sound features at each institution, different parties hold different but partially overlapping feature spaces that can be identified by the ID of the feature. The problem is how to align the data feature spaces of each party without revealing the non-shared parts. To achieve this goal, we aligned the data feature space according to a privacy-preserving protocol across databases [58]. Our protocols are asymmetric, making them especially applicable to applications in which a low-powered client interacts with a server in a privacy-preserving manner. **Note:** The feature bins tables store the histogram of first derivative $\langle \sum_i g_i \rangle$ and second derivative $\langle \sum_i h_i \rangle$, $\langle \ldots \rangle$ means the value is encrypted.

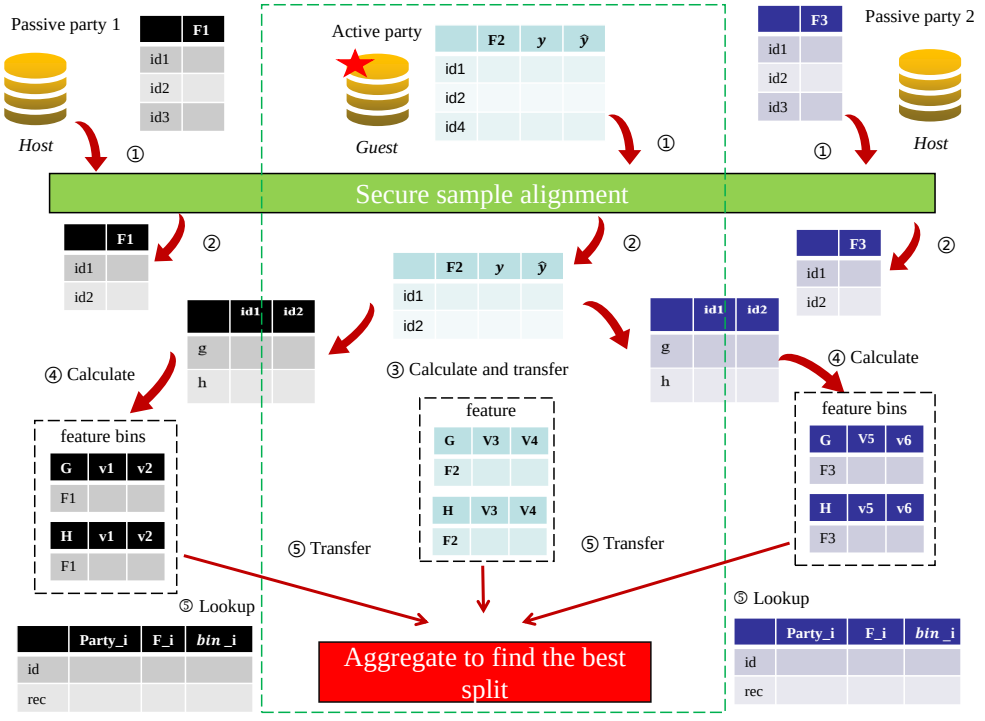

Fig. 2. Framework of the function of $guest$ party and $host$ parties in the VFL with interpretability. Heterogeneous-SecureBoost guarantee the privacy and security in the process when multiple parties jointly build the tree model. When the $guest$ party figures out the best split feature, it will notify the party which holds the feature, denoted as B. Then B will search for its threshold value, split the local model and get the left children and right children. After splitting the local model, B will tranfer its party id and the sample space in left children node to the $guest$ party, since the sample space in the right children can be inferred from the left childern. this study divides each feature space X_i^H into multiple quantile bins ($Max\_bin = 16$) as federated feature bins (X_{bins}) according to the procedure of the histogram approximate algorithm in XGBoost. Using bins instead of the original data is equivalent to a generalisation of that feature space, which loses the feature details of the data to protect the privacy of the host party. Notably, this method is beneficial to balance the VFL model interpretability and data privacy.

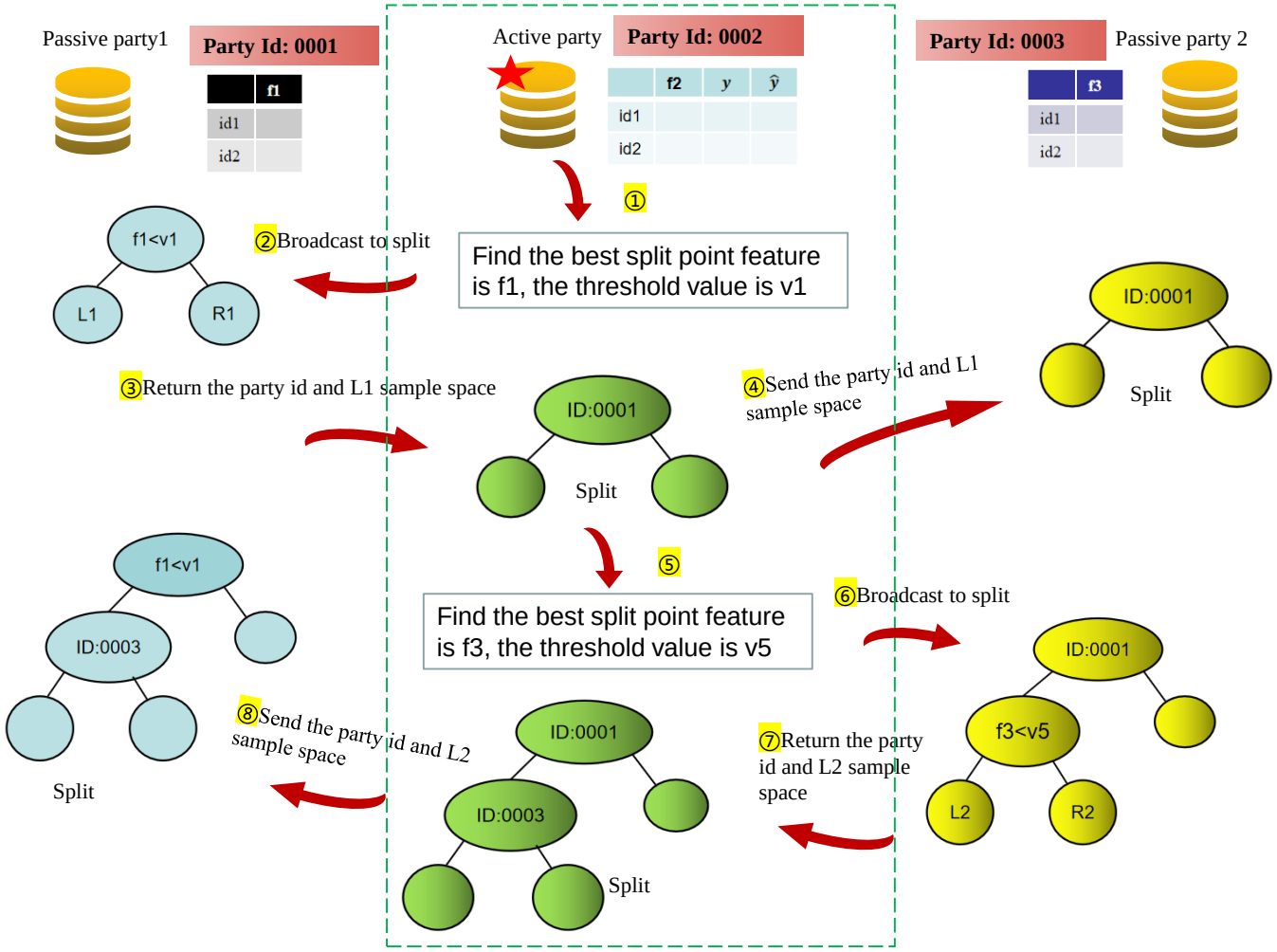

Fig. 3. The splitting mechanism for privacy preserving. Heterogeneous SecureBoost guarantee the privacy and security in the process when multiple parties jointly build the tree model. When the $\boldsymbol{guest}$ party figures out the best split feature, it will notify the party which holds the feature, denoted as $\boldsymbol{B}$. Then $\boldsymbol{B}$ will search for its threshold value, split the local model and get the left children and right children. After splitting the local model, $\boldsymbol{B}$ will tranfer its party id and the sample space in left children node to the $\boldsymbol{guest}$ party, since the sample space in the right children can be inferred from the left childern. The $\boldsymbol{guest}$ party then records the party id in the current node, and split the local tree model. In this way, although all the parties share the same tree model, the recorded information of each node of each party's tree model may be different. Each party can only have the authority to see its own data information. This splitting mechanism is visualized in Fig. 3.