# Combinatorial privacy: Packing splinters in polytopes at scale for private bit sums via SecureHull

## Abstract

We present a scheme to obtain counts of 0's and 1's at a server based on private bit streams hosted by multiple clients. The goal is to obtain this solution at the server while maintaining privacy of client data. The bit sums need to be obtained with respect to data from all clients; and not at a per client granularity. In our scheme called *SecureHull*, we hide the private data encoded as permutations amidst publicly shareable permutation matrices and form a secret doubly stochastic matrix via a convex combination with secret coefficients. We exploit the non-uniqueness of the Birkhoff-von Neumann decomposition and use some remnants of the splintering scheme to provide an unconventional secure computation method to this private bitsum problem. This scheme does not require any private data-dependent communication with the server as is ideal. We also provide lower bounds to quantify the probability of a successful attack. We show that the lower bound can be quadratically reduced with a linear increase in communication upto a constant. Our solution also involves a cryptographic shuffling routine that scales linearly with number of clients as against to the size of the datasets. The rest of the operations do not require a cryptographic approach and are secured through our scheme thereby benefiting its scalability.

## 1 Introduction and Problem Statement

Given the digital nature of mainstream computation, the bit sum is quite a fundamental operation and its applications are in plenty. We walkthrough an example use-case below without loss of generality. **Motivating problem of COVID-19 private test statistics:** Release of private client level statistics of symptoms such as fever/no-fever, test-results (positive/negative) and so forth for analysis and monitoring by a central server during the COVID-19 pandemic while maintaining the privacy of the clients data is a motivating use-case for example. This enables data-driven decision making at the central authority while not having to forego the critical privacy constraints and legal regulations of the clients data.

We now state the exact problem statement we consider in this paper followed by related work, preliminaries and our proposed solution.

### 1.1 Problem Statement

**Private Boolean Sum:** Given $k$ client entities that each holds a private binary bitstream $M_i \in \{0,1\}^n$ of length $n$, the problem statement is to provide an algorithm for a server to compute the total number of 0's and 1's in $\cup_i^k M_i$ without access to any $M_i$ or any statistics of $M_i$'s.

### 1.2 Related work

**Differentially private Boolean sum** The works in [1] provides a method for private estimation of Boolean bit sums where the privacy guarantees are based on differential privacy. The work in [2] provides differential privacy mechanisms for linear queries and hence can be adapted for this problem. Differential privacy solutions typically have a trade-off of accuracy of the solution Vs. privacy guarantees and are not geared for sharing exact solutions. Our work instead focuses on a hybrid approach between cryptographic schemes for shuffling and a combinatorial variant of a recent unconventional private computation approach called splintering to scalably obtain exact solutions at the server while protecting the privacy of individual client data.

**Splintering:** [3] A recent method called splintering allows for private computation in distributed settings via privatized stochastic shares of sensitive data called splinters. This enables client devices to receive services from server without directly sharing any sensitive data. The server performs computations over these splinters and the corresponding results are sent back to the client. The client has required private coefficients to perform specific operations referred to as unsplintering over these intermediate results in order to obtain the required final result as if the sensitive data itself was sent across to the server instead of the communicated splinters. Their work gives splintering and unsplintering schemes for operations of inner-product, sigmoid, softmax and matrix inverse. Their privacy guarantees are based on stochastic guarantees such as lower bounds on sample complexity for distribution testing and estimating distributions. Some of the splinters sent in this scheme are private data dependent while the rest are private data independent. In contrast our approach ensures any communication sent to the server via splinters that are permutations are completely independent of the private data.

**Instance mixing schemes:** Another line of work that mixes the private data records with public datasets, and random noise for the purpose of security is InstaHide [4, 5]. This approach also utilizes cryptographic computation for some operations while ensuring greater scalability as is the case in our scheme. One key difference is that our approach focuses on the Boolean bit sum problem while the instance mixing schemes focus on the task of private prediction with machine learning models.

### 1.3 Preliminaries for Splintering

For a $d$ dimensional input query vector $\mathbf{x}$, the client device creates $d$ shares corresponding to $\mathbf{x}$ as $\{\mathbf{z_1}, \mathbf{z_2} \dots \mathbf{z_d}\}$ so that $\mathbf{x} = Splint(\mathbf{z_1}, \mathbf{z_2} \dots \mathbf{z_d}), \forall i \in 1..d$. The most basic splint function that allows for such a representation is a linear combination using coefficients $\alpha_i$ as $\mathbf{x} = \sum_i^d \alpha_i \mathbf{z_i}, \forall i \in 1..d$. The $\alpha_i's$ are private to the client and not shared with any other entity, be it another client or a server. The splinters $\mathbf{z_i}$ are shared with the server. The server performs a set of application dependent operations on the splinters $\mathbf{z}_i, \forall i \in 1 \dots d$ and sends results $\{\beta_i\}$ back to client on either all or a subset of the $d$ shares. The client performs a local computation called $UnSplint$ using original shares $\mathbf{z_i}$, its corresponding $\alpha_i$'s that are known only to the client and received $\beta_i's$ obtained from the server. This unsplinting operation reveals the true result $l$ of the intended application to the client as $l = UnSplint(\alpha_i, \mathbf{z_i}, \beta_i), \forall i \in 1..d$. Note that although $\mathbf{x}$ is represented via a linear combination, the computation of $\{\beta_i\}$ and $UnSplint$ is not necessarily linear.

#### 1.3.1 Terminology: Types of splinters

**Decoy Splinters:** Data independent splinters $z_i$ whose corresponding coefficients $\alpha_i$ are zero.
**Minor Splinters:** Data independent splinters $z_i$ whose corresponding coefficients $\alpha_i$ are non-zero.
**Major Splinters:** Data dependent splinters $z_i$ whose corresponding coefficients $\alpha_i$ are non-zero.
Attackers would need to be able to distinguish decoy splinters from the rest of splinters, while estimating their corresponding coefficients in order to reconstruct the raw data. A good scheme for generating the splinters would need to prevent that attack.

#### 1.3.2 Motivation for combinatorial splintering

**Combinatorial splintering with SecureHulls:** We present a modification to this scheme where a convex combination of the splinters is shared as against to sharing the individual splinters themselves as done in the original splintering scheme. It is ideal for such a combination to be non-unique (many to one) for security purposes; in addition to having the security benefit of private coefficients. Our scheme is based on exploiting the non-uniqueness of the popular Birkhoff-von Neumann decomposition [6, 7, 8, 9] and also borrows from some remnants of the splintering scheme. In addition to that our scheme only depends on data independent splinters; and thereby negates leakage of information about the raw private dataset via these splinters. We refer to our algorithm as *SecureHull*.

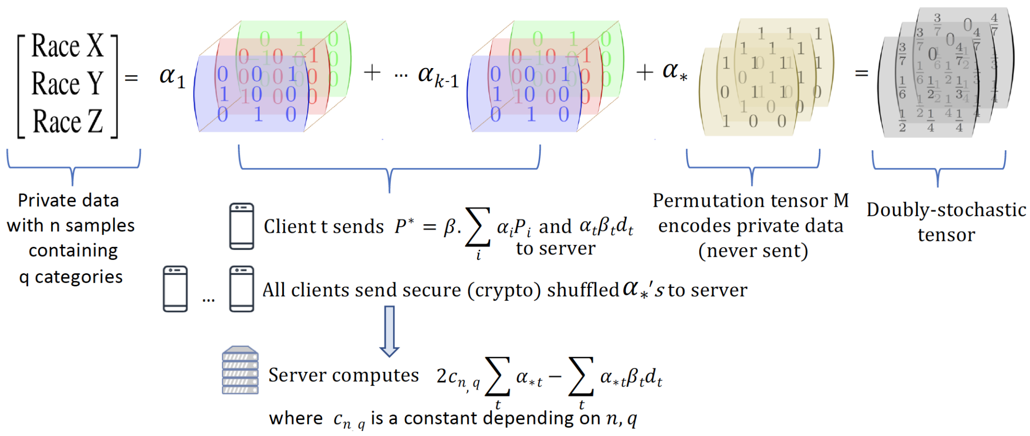
## 2 Preliminaries for SecureHull: Polyhedral Combinatorics

**Definition 1 (Doubly stochastic matric)** *A square matrix $A = (a_{ij})$ of nonnegative real numbers, each of whose rows and columns sums satisfy the condition $\sum_i a_{ij} = \sum_j a_{ij} = 1$ is doubly stochastic.*

**Definition 2 (Sub-permutation matrix)** *A doubly-stochastic matrix is a sub-permutation matrix if every row and every column has at most one non-zero entry with value 1.*

**Theorem 2.1 (Birkhoff–von Neumann theorem)** *The Birkhoff–von Neumann theorem states that the polytope of $n \times n$ doubly $B_n$ is the convex hull of the set of $n \times n$ permutation matrices, and furthermore that the vertices of $B_n$ are precisely the permutation matrices. In other words, if $A$ is doubly stochastic matrix, then there exist $\theta_1, \ldots, \theta_k \geq 0, \sum_{i=1}^{k} \theta_i = 1$ and permutation matrices $P_1, \ldots, P_k$ such that $A = \theta_1 P_1 + \cdots + \theta_k P_k$. This representation is also known as the Birkhoff–von Neumann decomposition [6, 7, 8, 9].*

## 3 SecureHull: Dual splintering for categorical/label data



Race X
Race Y
Race Z
=
$\alpha_1$
+ ··· $\alpha_{k\text{-}1}$
+ $\alpha_*$
=
Private data with n samples containing q categories
Client t sends $P^* = \beta.\sum_i \alpha_i P_i$ and $\alpha_t \beta_t d_t$ to server
Permutation tensor M encodes private data (never sent)
Doubly-stochastic tensor
...
All clients send secure (crypto) shuffled $\alpha_*'s$ to server
Server computes $2c_{n,q}\sum_t \alpha_{*t} - \sum_t \alpha_{*t}\beta_t d_t$
where $c_{n,q}$ is a constant depending on $n, q$

Figure 1: Illustration of our scalable private bitsum computation algorithm (*SecureHull*) where the goal is for the server to obtain a grand sum total count of all 0's and $1's$ across private bitstreams of all clients while maintaining privacy of clients data.

We now provide a modified version of splintering that is suitable for data that is one-hot encoded (i.e, in the form of permutation matrices). Any binary bit can be encoded as a permutation matrix. This can be generalized to any label/categorical data in machine learning which can also be encoded as permutation matrices.

### 3.1 Data representation

As an example if the input message from client 1 was $[0, 1, 1, 0]$, we can stack 4 permutation matrices of size $2 \times 2$ on top of each other and pad 0's everywhere else to represent it as a square matrix of size $8 \times 8$. Every 0 bit can be encoded as $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and 1 as $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. This may seem overdone but is quite necessary because for the convex combination of permutations to be doubly stochastic it is required that the permutations have to be represented as matrices. In this representation under the non-private case, in order to get sum of 1's in a message, one needs to trivially count the number of 1's in first column and to get number of 0's, I need to count number of 0's in second column; when the permutation matrices are stacked to form a tall matrix. They could as well be stacked as a tensor; and an analogous approach to count would follow. In the following sections we show how the permutation matrix that represents any given data record is mixed with data independent permutation matrices; followed by a specific computation protocol to solve the Boolean bitsum problem.

## 3.2 Approach: SecureHull

In this approach, we share a specific combination of splinters instead of the splinters themselves as typically done in the splintering technique.
**Desirable primitives of Birkhoff-von Neumann decomposition:** Typically decompositions such as eigen-decompositions are unique (when all eigenvalues are unique). But the relationship between doubly stochastic matrices to permutation matrices is non-unique, i.e there are multiple ways in which convex combination of permutation matrices can be used to represent a doubly stochastic matrix. We exploit this in our proposed scheme as follows.

The raw one-hot encoded matrix $\mathbf{M}$ that is generated using the data representation discussed in previous section, is first rescaled with a coefficient $\alpha_* \in (0,1)$ to get $\mathbf{M}_*$. This scaled matrix is combined (in a convex combination) with several secret permutation matrices $\mathbf{P_1}, \mathbf{P_2}, \ldots, \mathbf{P_k}$ and secret coefficients $\alpha_1, \ldots, \alpha_k$ to obtain a doubly stochastic matrix $\mathbf{D}$. The client then computes $\mathbf{P} = \sum_{i=1}^{k} \alpha_i \mathbf{P_i}$ and shares $\mathbf{P}_*$ with the server upon a rescaling $\mathbf{P}_* = \beta_* \mathbf{P}$. The server cannot reconstruct $\mathbf{M}$ due to the lack of secret coefficients and the fact that the decomposition from doubly stochastic $\mathbf{D}$ to the permutations is non-unique. Server computes $\mathbf{P}_*.y = d$ and sends the result to client. All the clients send a secure shuffled set of $\alpha_{*i}$'s to server along with each client sending one product (unshuffled) $\alpha_{*i}\beta_i d_i$ to the server. Server then computes the following to get the final bit sums:

$$2n \sum_i \alpha_{*i} - \sum_i \alpha_{*i}\beta_i d_i$$

The $2n$ in the final step can be replaced by a known constant, in order to generalize to data with ternary or more categories instead of the Boolean bitstream. The benefit of knowing this constant happens due to the stacked doubly-stochastic property of $\mathbf{D}$ as described in the data representation section. This gives a huge advantage in not having to either send the raw message or the $\mathbf{D}$ to ther server; while only having to send data independent permutation matrices to the server. We refer to this proposed algorithm as *SecureHull* and share it in the algorithmic block below.

**Algorithm 1** SecureHull

```
for Each of $k$ clients perform do
    Input: One-hot encoded data $\mathbf{M}$ is scaled down as $\mathbf{M}_* = \frac{1}{\alpha_*}\mathbf{M}$ with $\alpha_* \in (0,1)$
    Generate: Client generates;
            Secret coefficients $\{\alpha_1, \ldots, \alpha_k\} \in \mathbb{R}^+$ and Permutation matrices $\mathbf{P_1}, \mathbf{P_3} \ldots \mathbf{P_k}$.
    Create: Client creates doubly stochastic matrix $\mathbf{D} = \sum_{i=1}^{k} \alpha_i \mathbf{P_i} + \mathbf{M}_*$
    Compute: Client computes $\mathbf{P} = \sum_{i=1}^{k} \alpha_i \mathbf{P_i}$
    Rescale: Client rescales $\mathbf{P}_* = \frac{1}{\beta}\mathbf{P}$ and sends $\mathbf{P}_*$ to server.
    Compute: Server computes $\mathbf{P}_*.y = d$ and sends the result to client.
    Triplets: Each client computes one product $\alpha_{*i}\beta_i d_i$ and sends to server.
    End of for loop
Secure Shuffle: All clients perform a cryptographic shuffle of $\alpha_{*1} \ldots \alpha_{*k}$, if there are $k$ clients
and send to server (one $\alpha_*$ per client).
Solution: Server computes $2n\sum_i \alpha_{*i} - \sum_i \alpha_{*i}\beta_i d_i$ to to get the final bit sums.
```

## 3.3 Lower and upper bounds on # of possible Birkhoff von Neumann decompolinksitions of a doubly stochastic matrix

The # of possible Birkhoff von Neumann decompsitions of a doubly stochastic matrix is lower bounded by the square of # of positive elements in the doubly stochastic matrix $\mathbf{D}_n \times n$ and upper bounded by $n^2 - 2n + 2$ as in [10, 11, 12, 13]. The upper bound originally is based on the Marcus-Ree theorem in [14]. The lower bound in the context of *SecureHull* can be inflated by adding fake rows into the original permutation matrix; thereby increasing the security. This comes with a trade-off of increase in communication. It is also known that finding a Birkhoff-von Neumann decomposition with the least possible number of permutations is NP-Hard. As part of future work we would like to expand the scheme to other operations in addition to studying polytopes based on alternating sign matrices with $\{-1, 0, 1\}$ [15, 16, 17] or the transportation polytope [18].

## References

[1] Albert Cheu et al. "Distributed differential privacy via shuffling". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 375–403.

[2] Chao Li et al. "Optimizing linear counting queries under differential privacy". In: *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2010, pp. 123–134.

[3] Praneeth Vepakomma, Julia Balla, and Ramesh Raskar. "Splintering with distributions: A stochastic decoy scheme for private computation". In: *arXiv preprint arXiv:2007.02719* (2020).

[4] Yangsibo Huang et al. "InstaHide: Instance-hiding Schemes for Private Distributed Learning". In: *Proceedings of the 37'th International Conference on Machine Learning* (2020).

[5] Zhijian Liu et al. "DataMix: Efficient Privacy-Preserving Edge-Cloud Inference". In: ().

[6] Günter M Ziegler. *Lectures on polytopes*. Vol. 152. Springer Science & Business Media, 2012.

[7] Branko Grünbaum and Geoffrey C Shephard. "Convex polytopes". In: *Bulletin of the London Mathematical Society* 1.3 (1969), pp. 257–300.

[8] Peter R Cromwell. *Polyhedra*. Cambridge University Press, 1999.

[9] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003.

[10] Richard A Brualdi. "Notes on the Birkhoff algorithm for doubly stochastic matrices". In: *Canadian Mathematical Bulletin* 25.2 (1982), pp. 191–199.

[11] Fanny Dufossé and Bora Uçar. "Notes on Birkhoff–von Neumann decomposition of doubly stochastic matrices". In: *Linear Algebra and its Applications* 497 (2016), pp. 108–115.

[12] Richard A Brualdi and Geir Dahl. "An extension of the polytope of doubly stochastic matrices". In: *Linear and Multilinear Algebra* 61.3 (2013), pp. 393–408.

[13] Lu-Bin Cui, Wen Li, and Michael K Ng. "Birkhoff–von Neumann theorem for multistochastic tensors". In: *SIAM Journal on Matrix Analysis and Applications* 35.3 (2014), pp. 956–973.

[14] Marvin Marcus and Rimhak Ree. "Diagonals of doubly stochastic matrices". In: *The Quarterly Journal of Mathematics* 10.1 (1959), pp. 296–302.

[15] Bahman Kalantari. "Alternating sign matrices and polynomiography". In: *the electronic journal of combinatorics* (2011), P24–P24.

[16] Jessica Striker. "The alternating sign matrix polytope". In: *arXiv preprint arXiv:0705.0998* (2007).

[17] Roger E Behrend and Vincent A Knight. "Higher spin alternating sign matrices". In: *arXiv preprint arXiv:0708.2522* (2007).

[18] Edward D Kim. "Geometric combinatorics of transportation polytopes and the behavior of the simplex method". In: *arXiv preprint arXiv:1006.2416* (2010).