ブラウザ乱数化型 **DP** システムにおけるシャッフラーにおけるコミットメントの具体例

スライド構成

1. システム全体の流れ
2. シャッフラーに渡す３要素
3. RAPPOR＋Shuffle モデルの具体例
4. なぜこの３要素が必要か
5. まとめ

1. システム全体の流れ

1. ブラウザ側でローカル乱数化
    - 元データに差分プライバシー用ノイズを付与
2. データ送信
    - 乱数化レポート ＋ コミットメント列 ＋ 暗号化ペイロード
3. シャッフラー処理
    - 順序のシャッフル＋再ランダム化＋ZKP で証明
4. 集計サーバ
    - 合計を集計・復号して最終統計を生成

2. シャッフラーに渡す３要素

1. 乱数化済みレポート y_i

- RAPPOR のビット列、または GRR でノイズを混ぜた整数
- 目的: プライバシー保護された「公開してよい値」

2. ペデンコミットメント C_i

$$C_i = r_i H \ + \ y_i G$$

- y_i を隠蔽しつつ整合性を担保
- 目的: シャッフル後も「並べ替えただけ」であることを証明

3. 暗号化ペイロード（任意）

$$\mathrm{Enc}_{\mathrm{agg}}(y_i)$$

- 集計サーバのみが復号
- 目的: シャッフラーに中身を見せずに二重秘匿

3. RAPPOR＋Shuffle モデルの具体例

1. ブラウザで乱数化
   - URL ID＝42 → 100ビットの RAPPOR ベクトル y_i
2. 各ビットをコミット

$$C_{i,j} = r_{i,j}H + y_{i,j}G \quad (0 \le j < 100)$$

3. 送信パケット

```
{
  "commitments": [C_{i,0}, …, C_{i,99}],
  "ciphertext" : Enc_pub(y_i)
}
```

4. シャッフラー処理
   - ランダム置換 π
   - 再ランダム化: $C' = C_\pi + r'H$

4. なぜこの３要素が必要か

- 乱数化レポートだけでは順序情報から再識別リスク
- コミットメントで「順序以外不変」をゼロ知識証明
- 暗号化ペイロードで二重ガード
    i. ブラウザ→シャッフラー：DPノイズ込みで秘匿
    ii. シャッフラー→集計サーバ：再識別防止

ブラウザ側で「ローカル乱数化 → 差分プライバシ（**DP**）→ シャッフル」の流れを採る場合、

#	シャッフラーに入るもの	具体例	目的
1	乱数化済みレポート y_i	RAPPOR のビット列、GRR で変換した整数など	DPノイズ込みの「公開してよい値」
2	ペデンコミットメント $C_i = r_i H + y_i G$	各ビット（0/1）や整数値に対し個別に作成	後段でシャッフル正当性や範囲証明をZKPで検証するため
3	（任意）暗号化ペイロード	$Enc_{\text{agg}}(y_i)$	集計サーバだけが復号できるようにし、シャッフラーには中身を見せない

フローを例で見る（**RAPPOR** ＋ **Shuffle** モデル **100** 個バケットのヒストグラム）

1. ブラウザでローカル乱数化
    - 元データ: ユーザが訪れた URL ID = 42
    - RAPPOR の「Unary Encoding＋ランダム化」で 100 bit のベクトル (y_i) を生成

2. 各ビットをコミット

$$C_{i,j} = r_{i,j}H + y_{i,j}G \quad (0 \le j < 100)$$

   - 0/1 なのでレンジプルーフはとても軽量

3. パケットを構成して送信

```
{
  "commitments": [C_{i,0}, …, C_{i,99}],
  "ciphertext" : Enc_pub(y_i)
}
```

4. シャッフラーの処理
   i. レポートの順序をランダム置換 $ \pi $
   ii. 再ランダム化：各コミットメントに追加乱数 r'_j を加えて
   $C'_j = C_{\pi(j)} + r'_j H$
   iii. 置換と r'_j が正しく行われたことを Bulletproofs の集約 ZKP で証明

なぜこの３点セットなのか？

- 乱数化済みレポートだけでは、シャッフラーがメタデータ（順序・IP など）で個人を再識別しかねません。
- ペデンコミットメントを付けることで「順序以外は何も変えていない」ことをシャッフラー自身がゼロ知識で証明できます。
- 暗号化ペイロードを併用すれば、
  - ブラウザ→シャッフラー間：データ値秘匿（DP ノイズ入りでも念のため）
  - シャッフラー→集計サーバ間：再識別リスクを最小化
    という二重ガードが張れます。

まとめ

- 値 y_i は「ブラウザで既に DP ノイズを混ぜたレポート」
- コミットメント C_i はその値を隠したまま整合性を持たせる暗号ラッパ
- シャッフラーは「順序を壊す＋再ランダム化＋ZKP で証明」という役だけを果たし、
  元データもユーザ ID も見えない。
  以上が、ブラウザ乱数化型 DP システムでシャッフラーに渡す“中身”です。