Bulletproofs を用いたシャッフラーのコード概要

- Rust で実装したサンプルコードの紹介
- ペデンコミットメントとレンジプルーフの生成
- コミットメントのシャッフルとその準同型性の検証
- 今後の応用展開について

コードの主要機能

- コミットメント生成
  各値に対し、ランダムなブラインディングを用いてペデンコミットメントを計算
  $$C = rH + vG$$
- レンジプルーフ生成
  秘密の値が 0 ～ $2^{64} - 1$ の範囲内にあることをゼロ知識で証明
- シャッフル処理
  複数のコミットメントをランダムに並び替え、全体の合計が保存される（加法的性質）

処理フロー図

```
%%{init: {'theme': 'neutral', 'fontSize': 14}}%%

sequenceDiagram
    participant App as アプリ
    participant Gen as コミット生成
    participant Shuffle as シャッフル処理
    participant Sum as 合計計算

    App->>Gen: 各値のコミットメントを生成する
    Gen-->>App: コミットメントリストを返す
    App->>Sum: コミットメントリストの合計を計算（original sum）
    App->>Shuffle: コミットメントリストをシャッフルする
    Shuffle-->>App: シャッフル済みリストを返す
    App->>Sum: シャッフル済みリストの合計を計算（shuffled sum）
    App->>App: original sum と shuffled sum を比較
    App->>App: 結果を出力
```

- 加法的性質の検証
  シャッフル前後で合計が同じであれば、シャッフル処理に問題はないと確認

今後の展開

- 検証可能なシャッフル証明の統合
  出力が入力の順序変更であることをゼロ知識証明で保証
- 大規模ミキシングプロトコルの構築
  Bulletproofs の集約機能を活かし、複数の取引に対する証明を効率的に管理
- セキュリティとプライバシーの強化

まとめ

- Bulletproofs を活用することで、
  - ペデンコミットメントに基づくレンジプルーフが生成可能
  - 秘密の値の範囲を非対話型ゼロ知識で証明
  - コミットメントの加法性を利用してシャッフルの正当性を検証
- 今回のサンプルコードは、これらの基礎処理（コミットメント生成、シャッフル、合計検証）を実装