JuLenny Data Privacy

# Joint Record Overlap

## How private record matching works

### The scenario

A retail chain (ShopMart) and a payment processor (PayFlow) want to know how many customers they share. ShopMart has a loyalty program with 50,000 members. PayFlow processes cards for 200,000 merchants. Neither will hand over their customer list to the other.

### The problem

Knowing the overlap size is valuable for both sides. ShopMart wants to understand what percentage of its loyalty members also use PayFlow cards (potential co-branded offer). PayFlow wants to know how many of its cardholders shop at ShopMart (targeted cashback campaign). But sharing raw customer lists is a privacy and compliance violation.

### How it works

#### *Step 1: Each side encodes their records*

ShopMart takes its customer list (names, emails, or hashed IDs) and turns it into a binary vector. Think of it as a long row of zeros and ones, where a 1 means "we have this record" and a 0 means "we don't."

```
ShopMart:  [0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, ...]
```

PayFlow does the same with its cardholder list:

```
PayFlow:   [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, ...]
```

Both vectors use the same slot assignment, so a customer that appears at position 5 in ShopMart's vector also appears at position 5 in PayFlow's vector.

#### *Step 2: Encrypt and upload*

Each side encrypts their binary vector locally using the JuLenny toolkit. The raw lists never leave their premises. Only encrypted data is uploaded to the platform.

### Step 3: The platform computes the overlap

JuLenny multiplies the two encrypted vectors element by element and sums the result. All math happens on encrypted data.

```
ShopMart:   [0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, ...]
PayFlow:    [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, ...]
Product:    [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, ...]
Sum = 3
```

### Step 4: Decrypt the result

The result (3 in this example) is released after both parties approve. Both sides now know they share 3 customers.

## Two variants

**Count variant:** returns a single number (how many records overlap). Neither side learns WHICH specific records matched. Good for sizing the opportunity before committing to a deeper collaboration.

**Itemized variant:** returns the full result vector showing exactly which positions matched. The viewing party can look up which of their own records overlapped. Good for building a co-marketing list or a joint campaign.

## What nobody learns

- ShopMart never sees PayFlow's full cardholder list
- PayFlow never sees ShopMart's full loyalty member list
- Records that did NOT match are completely hidden
- JuLenny never sees either list, even during processing

## Use cases beyond retail

This function works for any "how many do we share?" question between two organizations:

- **Cybersecurity:** Two firms compare their threat indicator databases to find common IOCs (indicators of compromise) without exposing their full intelligence
- **Manufacturing:** Two companies compare parts catalogs to find overlapping SKUs for joint procurement
- **Genomics:** Two biobanks compare gene variant databases to find shared research subjects
- **Finance:** Two portfolio managers compare holdings to find co-invested assets
- **HR / Recruiting:** Two companies compare candidate pools to find shared applicants (with consent)

## Technical details

- **Scheme:** BFV (exact integer arithmetic, no rounding)
- **Multiplicative depth:** 1
- **Max records:** 8,192 per ciphertext (larger datasets require chunking)
- **Result:** exact integer count (count variant) or exact binary vector (itemized variant)