Encrypted Negotiation Matrix

How private B2B deal-matching works on JuLenny

The scenario

Acme (buyer) wants to buy laptop screens from BrightScreen (supplier). They need to agree on three things: price per unit, order quantity, and delivery timeline.

Neither side wants to reveal their real constraints. Acme doesn't want BrightScreen to know their budget ceiling. BrightScreen doesn't want Acme to know their cost floor or capacity limits.

Step 1: Agree on the option grid

Before anything encrypted happens, both sides agree on the possible combinations. Think of it as a spreadsheet of every option on the table:

#	Price	Quantity	Delivery
0	$80	1,000	30 days
1	$80	1,000	60 days
2	$80	5,000	30 days
3	$80	5,000	60 days
4	$95	1,000	30 days
5	$95	1,000	60 days
6	$95	5,000	30 days
7	$95	5,000	60 days

This grid is public. Both sides see it. It's just the menu of options, not anyone's private position.

Step 2: Each side marks their acceptable terms (privately)

Acme (buyer) reviews the grid internally. Their budget ceiling is $90/unit and they need at least 5,000 units within 30 days. So they mark:

Position	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Acme	0	0	1	0	0	0	0	0

Only combination 2 ($80, 5,000 units, 30 days) works for them.

BrightScreen (supplier) checks their margins and capacity. They can do 5,000 units at $80 but need 60 days for production. So they mark:

Position	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
BrightScreen	0	0	0	1	0	0	0	0

Combination 3 ($80, 5,000, 60 days) is their offer.

Step 3: Encrypt and upload

Each side encrypts their binary vector using the JuLenny toolkit on their own machine. The raw vectors never leave their premises. Only encrypted ciphertext is uploaded to the JuLenny platform.

Step 4: The platform checks for overlap

JuLenny multiplies the two encrypted vectors element by element and sums the result. All math happens on encrypted data. The platform never sees either side's actual positions.

```
Acme:              [0, 0, 1, 0, 0, 0, 0, 0]
BrightScreen:      [0, 0, 0, 1, 0, 0, 0, 0]
Product:           [0, 0, 0, 0, 0, 0, 0, 0]
Sum = 0
```

Result: 0. No match. Neither side learns WHY it failed. Acme doesn't know BrightScreen needed 60 days. BrightScreen doesn't know Acme's budget ceiling was $90.

Step 5: Adjust and try again

Both sides go back to their teams privately. Maybe Acme's procurement lead decides 60 days is workable if the price is right. They update their acceptable terms:

Position	0	1	2	3	4	5	6	7
Acme (round 2)	0	0	1	1	0	0	0	0

Now combinations 2 and 3 are both acceptable. They re-encrypt and run again:

```
Acme:              [0, 0, 1, 1, 0, 0, 0, 0]
BrightScreen:      [0, 0, 0, 1, 0, 0, 0, 0]
Product:           [0, 0, 0, 1, 0, 0, 0, 0]
Sum = 1
```

Result: 1. Match found! Deal at $80, 5,000 units, 60 days.

What each side learns

With the Count variant: both sides learn "there is 1 matching combination." If a party marked multiple positions as acceptable, they know the match is one of those, but not which specific one.

With the Itemized variant: both sides learn exactly which position(s) matched. In this case, position 3 ($80, 5,000, 60 days). This is typically what procurement teams want.

What nobody learns

- Acme never learns BrightScreen's cost floor or capacity constraints.
- BrightScreen never learns Acme's budget ceiling or minimum volume.
- Failed rounds reveal nothing about how far apart the parties were.
- JuLenny never sees either side's positions, even during processing.

Why this is powerful

In a normal negotiation, every rejected counter-offer leaks information. "They said no to $95, so their ceiling must be lower." "They countered with 60 days, so they have a production bottleneck." Over multiple rounds, both sides can reverse-engineer each other's constraints.

With the encrypted negotiation matrix:
- A buyer can run 50 suppliers through this simultaneously, without any supplier learning what the others offered.
- Failed rounds leak zero information.
- Each side can adjust their positions between rounds without the other side seeing the change.
- The only information revealed is: match or no match (count variant), or which specific combination matched (itemized variant).

Scaling up

The grid can be as large as needed. 10 price points, 10 quantities, 10 delivery windows, 5 payment terms = 5,000 combinations. All fit in a single encrypted vector (the platform supports up to 8,192 slots per ciphertext).

For ongoing supplier relationships, the grid can be updated and new rounds run at any time. New combinations only append to the end of the grid, preserving compatibility with previous rounds.