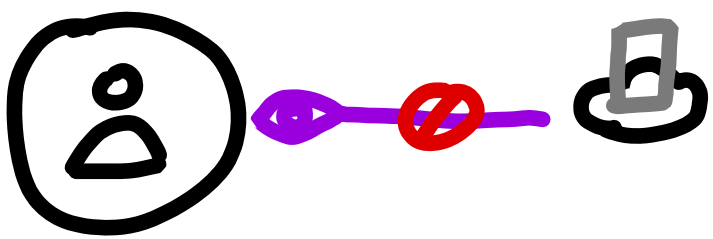
Replicated Secret Sharing

Aly Cerruti

2025-09-15

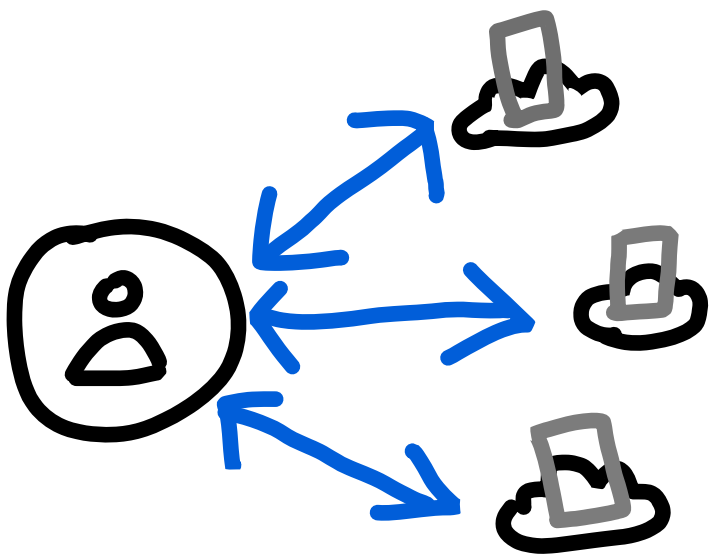
Motivation

1. We have some “secret” values



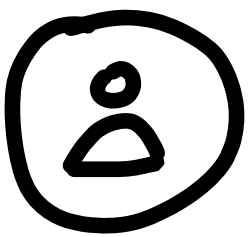
Motivation

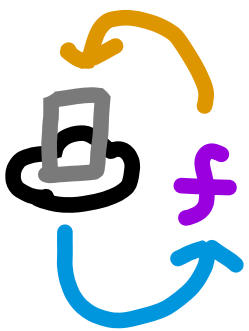
1. We have some “secret” values
2. We want these values to be “replicated”



Motivation

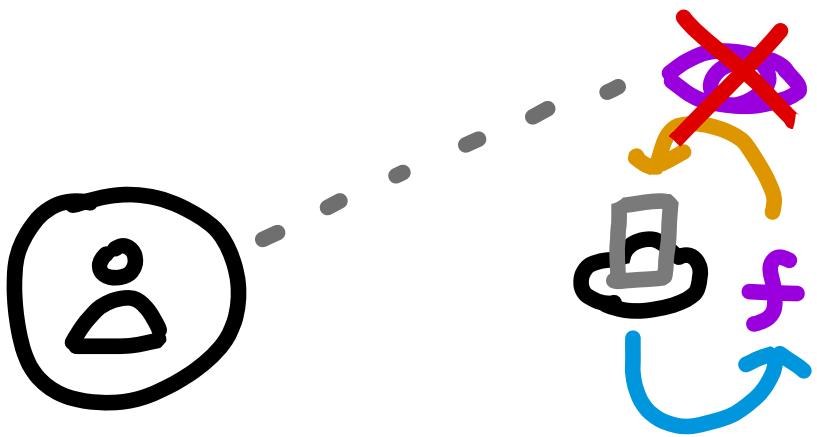
1. We have some “secret” values
2. We want these values to be “replicated”
3. We have functions that should run remotely on secrets to produce new secrets





Motivation

1. We have some “secret” values
2. We want these values to be “replicated”
3. We have functions that should run remotely on secrets to produce new secrets
4. We don’t want any machine holding a “share” of the secret to learn the secret when doing computations



Shares

s our secret value (in a finite group such as $\mathbb{Z}_r$)

n the number of parties holding a piece of s

Shares

s our secret value (in a finite group such as $\mathbb{Z}_r$)

n the number of parties holding a piece of s

Pick randomly group values $s_1, \ldots, s_n$ s.t.

$$s = \sum_{i=1}^{n} s_i$$

Shares

s our secret value (in a finite group such as $\mathbb{Z}_r$)

n the number of parties holding a piece of s

Pick randomly group values $s_1, \ldots, s_n$ s.t.

$$s = \sum_{i=1}^{n} s_i$$

For the ith participant, give it multiple shares $[s_i, \ldots, s_{i+t}]$:

- at most t participants can "fail" while leaving the secret recoverable
- at most $t - 1$ participants can lie while leaving the secret recoverable
- at least $n - t$ participants have to collude to learn the secret s

Sharing Example

For $s = 5 \pmod{\mathbb{Z}_r}$, $n = 3$, we can describe s as the sum

$$s = 9 + (-7) + 3 \pmod{\mathbb{Z}_r}$$

Sharing Example

For $s = 5 \pmod{\mathbb{Z}_r}$, $n = 3$, we can describe s as the sum

$$s = 9 + (-7) + 3 \pmod{\mathbb{Z}_r}$$

If we choose $t = 1$,

Participant 1 receives $[9, -7]$

Participant 2 receives $[-7, 3]$

Participant 3 receives $[3, 9]$

Reconstruction

$$p_1 \leftarrow [9, -7] \qquad p_2 \leftarrow [-7, 3] \qquad p_3 \leftarrow [3, 9]$$

What happens if participants 1 and 2 collude?

- They see that they both have -7...

Reconstruction

$$p_1 \leftarrow [9, -7] \qquad p_2 \leftarrow [-7, 3] \qquad p_3 \leftarrow [3, 9]$$

What happens if participants 1 and 2 collude?

- They see that they both have -7...
- ...so they can recover $9 + (-7) + 3 = 5$

Reconstruction

$$p_1 \leftarrow [9, -7] \qquad p_2 \leftarrow [-7, 3] \qquad p_3 \leftarrow [3, 9]$$

What happens if participants 1 and 2 collude?

- They see that they both have -7...
- ...so they can recover $9 + (-7) + 3 = 5$

What happens if participant 3 fails?

Reconstruction

$$p_1 \leftarrow [9, -7] \qquad p_2 \leftarrow [-7, 3] \qquad p_3 \leftarrow [3, 9]$$

What happens if participants 1 and 2 collude?

- They see that they both have -7...
- ...so they can recover $9 + (-7) + 3 = 5$

What happens if participant 3 fails?

- The reconstruction of values from p_1 and p_2 is still $9 + (-7) + 3 = 5$, recovering the secret

Addition on Secrets

Addition on secrets is just addition on their replicated shares:

Addition on Secrets

Addition on secrets is just addition on their replicated shares:

- Our target value is $c = a + b = c_1 + \ldots + c_n$.

Addition on Secrets

Addition on secrets is just addition on their replicated shares:

- Our target value is $c = a + b = c_1 + \ldots + c_n$.
- If we say that $c_i = a_i + b_i$, then...

Addition on Secrets

Addition on secrets is just addition on their replicated shares:

- Our target value is $c = a + b = c_1 + \ldots + c_n$.
- If we say that $c_i = a_i + b_i$, then...
- ...each participant can calculate their share of c without any extra information.

Addition on Secrets

Addition on secrets is just addition on their replicated shares:

- Our target value is $c = a + b = c_1 + \ldots + c_n$.
- If we say that $c_i = a_i + b_i$, then...
- ...each participant can calculate their share of c without any extra information.

For example:

$$a = 5 = 9 + (-7) + 3 \qquad b = 8 = 2 + 1 + 5$$

Addition on Secrets

Addition on secrets is just addition on their replicated shares:

- Our target value is $c = a + b = c_1 + \ldots + c_n$.
- If we say that $c_i = a_i + b_i$, then...
- ...each participant can calculate their share of c without any extra information.

For example:

$$a = 5 = 9 + (-7) + 3 \qquad b = 8 = 2 + 1 + 5$$

We can calculate $c = a + b = 5 + 8$ as:

$$\begin{aligned} c &= (9 + 2) + (-7 + 1) + (3 + 5) \\ &= 11 + (-6) + 8 \\ &= 13 \end{aligned}$$

Multiplication on Secrets: Basic Concept

- Our target value is $c = ab = c_1 + \ldots + c_n$.

Multiplication on Secrets: Basic Concept

- Our target value is $c = ab = c_1 + \ldots + c_n$.
- We can break ab into an n^2-sized sum of products:

$$c = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j$$

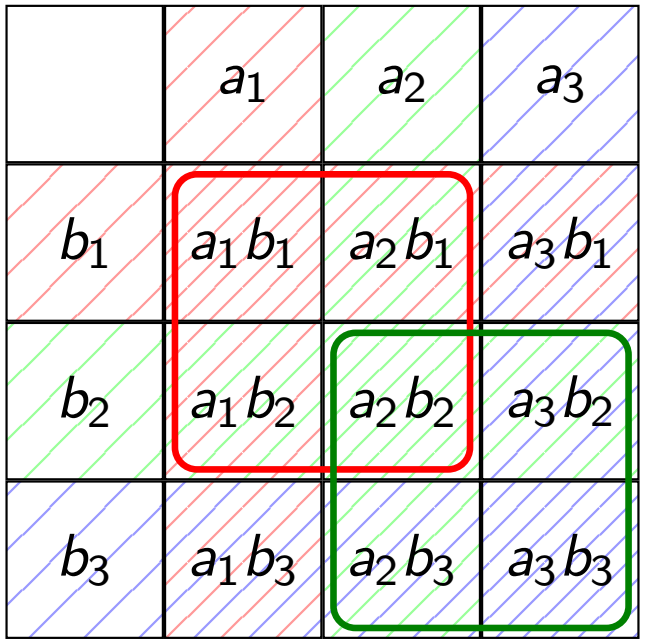
Multiplication on Secrets: Basic Concept

- Our target value is $c = ab = c_1 + \ldots + c_n$.
- We can break ab into an n^2-sized sum of products:

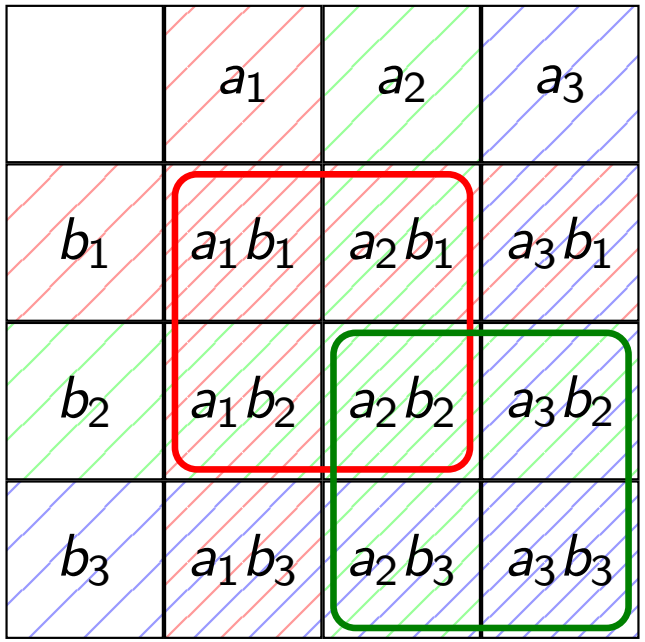
$$c = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i b_j$$

- We would like some definition of c_i s.t. each c_i is an n-sized sum of these products.

Multiplication on Secrets: $n = 3, t = 1$

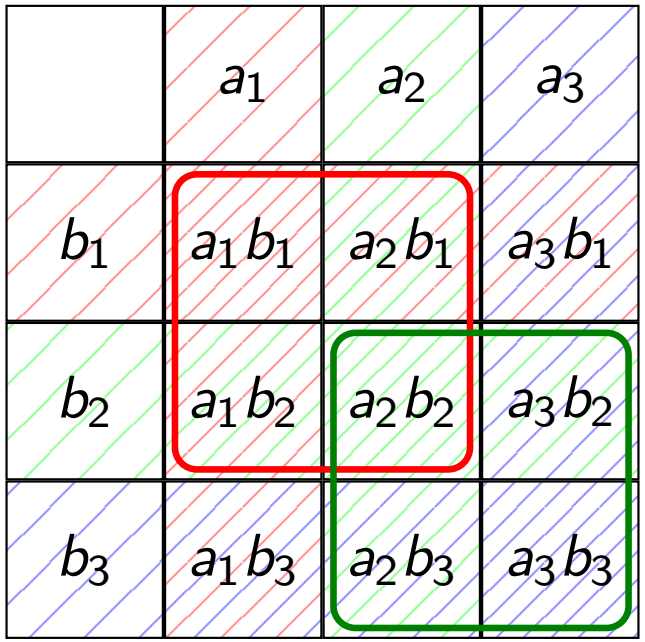


Multiplication on Secrets: $n = 3, t = 1$



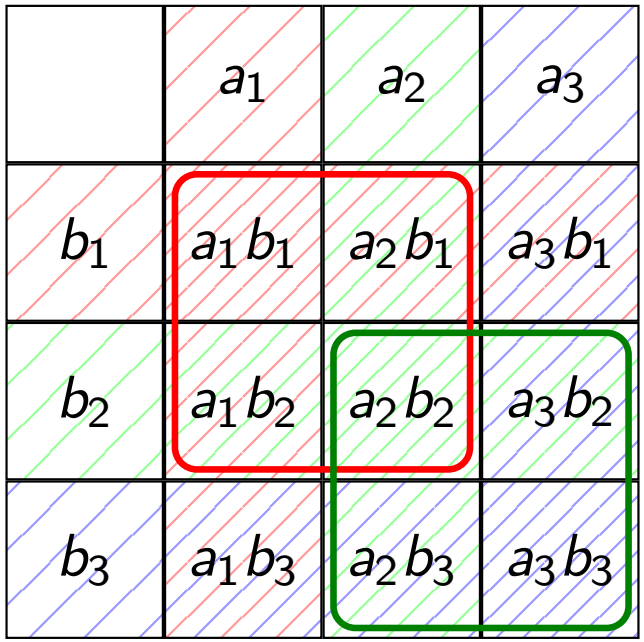
- Each participant can calculate 4 products

Multiplication on Secrets: $n = 3, t = 1$



- Each participant can calculate 4 products
- Extra product is case $i = j$: resolve by giving $a_i b_i$ to participant i

Multiplication on Secrets: $n = 3, t = 1$



- Each participant can calculate 4 products
- Extra product is case $i = j$: resolve by giving $a_i b_i$ to participant i
- So $c_i = a_i b_i + a_i b_{i+1} + a_{i+1} b_i$

Multiplication on Secrets: Reintroducing Replication

- This process only allows a participant to calculate one share of the secret, not a replicated share

Multiplication on Secrets: Reintroducing Replication

- This process only allows a participant to calculate one share of the secret, not a replicated share
- We can fix this by just sending c_1 to participant 3 so it has $[c_3, c_1]$, c_2 to participant 1 so it has $[c_1, c_2]$, etc.

Multiplication on Secrets: Reintroducing Replication

- This process only allows a participant to calculate one share of the secret, not a replicated share
- We can fix this by just sending c_1 to participant 3 so it has $[c_3, c_1]$, c_2 to participant 1 so it has $[c_1, c_2]$, etc.

One big issue:

Raw multiplication results aren't secure!

Multiplication on Secrets: Leaky Replication

Sending a raw share c_i to participant $i-1$ allows that participant to learn about a_{i+1} and b_{i+1}, which can teach it the secret values of a and b:

Multiplication on Secrets: Leaky Replication

Sending a raw share c_i to participant $i-1$ allows that participant to learn about a_{i+1} and b_{i+1}, which can teach it the secret values of a and b:

1. Participant 1 sends c_1 to participant 3, so participant 3 holds the replicated share $[c_3, c_1]$

Multiplication on Secrets: Leaky Replication

Sending a raw share c_i to participant $i-1$ allows that participant to learn about a_{i+1} and b_{i+1}, which can teach it the secret values of a and b:

1. Participant 1 sends c_1 to participant 3, so participant 3 holds the replicated share $[c_3, c_1]$
2. Participant 3 knows:
   - a_1, a_3, b_1, b_3, and c_3, as it is participant 3
   - $c_1 = a_1 b_1 + a_1 b_2 + a_2 b_1$ as it received c_1 as part of multiplication

Multiplication on Secrets: Leaky Replication

Sending a raw share c_i to participant $i-1$ allows that participant to learn about a_{i+1} and b_{i+1}, which can teach it the secret values of a and b:

1. Participant 1 sends c_1 to participant 3, so participant 3 holds the replicated share $[c_3, c_1]$
2. Participant 3 knows:
   - a_1, a_3, b_1, b_3, and c_3, as it is participant 3
   - $c_1 = a_1 b_1 + a_1 b_2 + a_2 b_1$ as it received c_1 as part of multiplication
3. This teaches participant 3 about $a_1 b_2 + a_2 b_1$

Multiplication on Secrets: Leaky Replication

Sending a raw share c_i to participant $i-1$ allows that participant to learn about a_{i+1} and b_{i+1}, which can teach it the secret values of a and b:

1. Participant 1 sends c_1 to participant 3, so participant 3 holds the replicated share $[c_3, c_1]$
2. Participant 3 knows:
   - a_1, a_3, b_1, b_3, and c_3, as it is participant 3
   - $c_1 = a_1 b_1 + a_1 b_2 + a_2 b_1$ as it received c_1 as part of multiplication
3. This teaches participant 3 about $a_1 b_2 + a_2 b_1$
4. Which allows participant 3 to learn:
   - $(a_1 b_2 + a_2 b_1) \equiv a_2 b_1 \pmod{a_1} \Rightarrow a_2 b_1 b_1^{-1} \equiv a_2 \pmod{a_1}$
   - Likewise for $b_2 \pmod{b_1}$

Multiplication on Secrets: Fixing the Leak

Each participant should choose a random number to publish with the result of its multiplication. Participant 1, for example:

Multiplication on Secrets: Fixing the Leak

Each participant should choose a random number to publish with the result of its multiplication. Participant 1, for example:

1. picks a random r_1
2. instead of sending c_1 to participant 3, sends $c_1 + r_1$
3. additionally, sends r_1 to participant 2

Multiplication on Secrets: Fixing the Leak

Each participant should choose a random number to publish with the result of its multiplication. Participant 1, for example:

1. picks a random r_1
2. instead of sending c_1 to participant 3, sends $c_1 + r_1$
3. additionally, sends r_1 to participant 2
4. receives $c_2 + r_2$ from participant 2
5. receives r_3 from participant 3

Multiplication on Secrets: Fixing the Leak

Each participant should choose a random number to publish with the result of its multiplication. Participant 1, for example:

1. picks a random r_1
2. instead of sending c_1 to participant 3, sends $c_1 + r_1$
3. additionally, sends r_1 to participant 2
4. receives $c_2 + r_2$ from participant 2
5. receives r_3 from participant 3
6. calculates $c_1' = c_1 + r_1 - r_3, c_2' = c_2 + r_2 - r_1$
7. assumes the replicated share $[c_1', c_2']$ for the secret c

Multiplication on Secrets: Fixing the Leak

Each participant should choose a random number to publish with the result of its multiplication. Participant 1, for example:

1. picks a random r_1
2. instead of sending c_1 to participant 3, sends $c_1 + r_1$
3. additionally, sends r_1 to participant 2
4. receives $c_2 + r_2$ from participant 2
5. receives r_3 from participant 3
6. calculates $c_1' = c_1 + r_1 - r_3, c_2' = c_2 + r_2 - r_1$
7. assumes the replicated share $[c_1', c_2']$ for the secret c

In the final sum, the random values cancel:

$$c' = c_1 + r_1 - r_3 + c_2 + r_2 - r_1 + c_3 + r_3 - r_2 = c$$

Larger n and Optimal t

- The requirements for failure or successful collusion when $n = 3$ are dangerously lax

Larger n and Optimal t

- The requirements for failure or successful collusion when $n = 3$ are dangerously lax
- Ideally: a large number of things have to go wrong for our protocol to fail

Larger n and Optimal t

- The requirements for failure or successful collusion when $n = 3$ are dangerously lax
- Ideally: a large number of things have to go wrong for our protocol to fail

Question

Can we scale to "honest-majority MPC"?

Larger n and Optimal t

- The requirements for failure or successful collusion when $n = 3$ are dangerously lax
- Ideally: a large number of things have to go wrong for our protocol to fail

Question

Can we scale to "honest-majority MPC"?

- $z < n - t$ collusion requirement, $z \leq t - 1$ liar requirement

Larger n and Optimal t

- The requirements for failure or successful collusion when $n = 3$ are dangerously lax
- Ideally: a large number of things have to go wrong for our protocol to fail

Question

Can we scale to "honest-majority MPC"?

- $z < n - t$ collusion requirement, $z \leq t - 1$ liar requirement
- Maximum z is $z = \frac{n}{2} - 1$, with $t = \frac{n}{2}$

Larger n and Optimal t

- The requirements for failure or successful collusion when $n = 3$ are dangerously lax
- Ideally: a large number of things have to go wrong for our protocol to fail

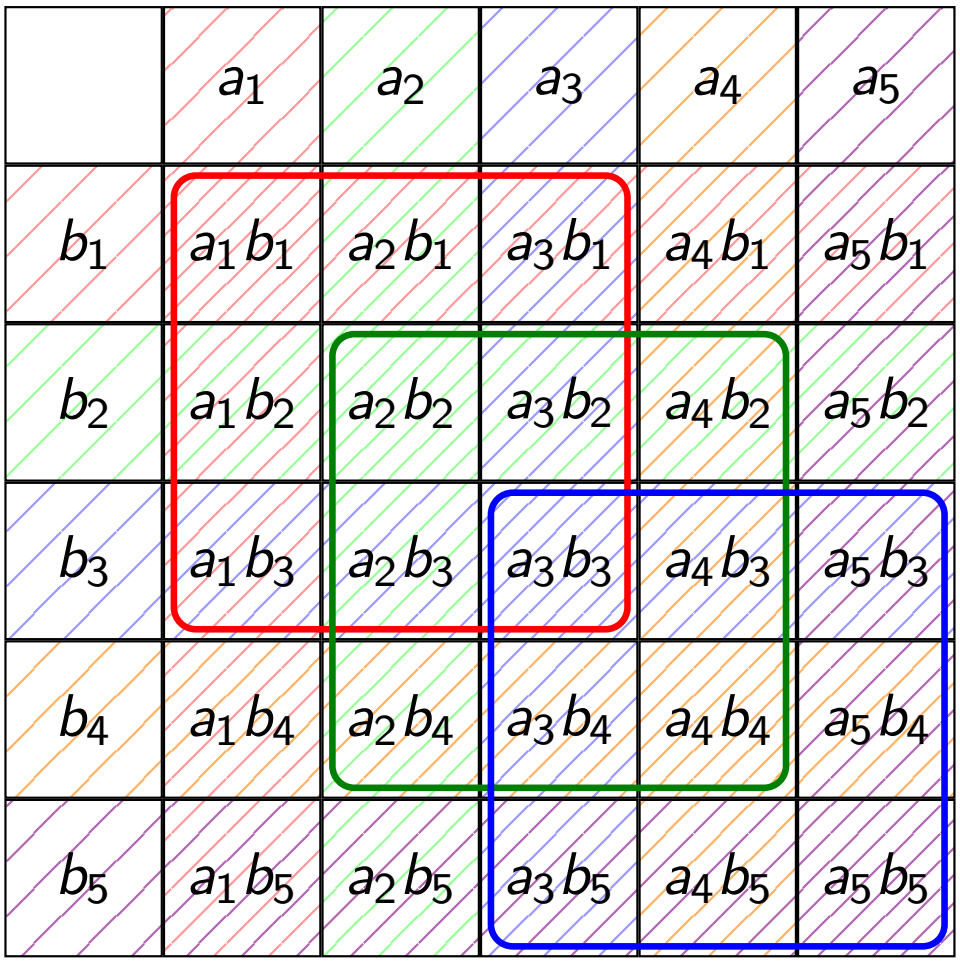
Question

Can we scale to "honest-majority MPC"?

- $z < n - t$ collusion requirement, $z \leq t - 1$ liar requirement
- Maximum z is $z = \frac{n}{2} - 1$, with $t = \frac{n}{2}$

Needs one more honest participant than simple majority: $z < \frac{n-1}{2}$

Multiplication for Larger t



	a_1	a_2	a_3	a_4	a_5
b_1	$a_1 b_1$	$a_2 b_1$	$a_3 b_1$	$a_4 b_1$	$a_5 b_1$
b_2	$a_1 b_2$	$a_2 b_2$	$a_3 b_2$	$a_4 b_2$	$a_5 b_2$
b_3	$a_1 b_3$	$a_2 b_3$	$a_3 b_3$	$a_4 b_3$	$a_5 b_3$
b_4	$a_1 b_4$	$a_2 b_4$	$a_3 b_4$	$a_4 b_4$	$a_5 b_4$
b_5	$a_1 b_5$	$a_2 b_5$	$a_3 b_5$	$a_4 b_5$	$a_5 b_5$

(Table for $n = 5, t = 2$.)

A participant's c_i is the sum of the products on the top-left border of their square. Each participant still generates just one random number, and sends $c_i + r_i$ to the t participants to its left and r_i to the t participants to its right.

The End :)