High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority

*Jun Furukawa**, *Yehuda Lindell***, ***Ariel Nof***** *and Or Weinstein***

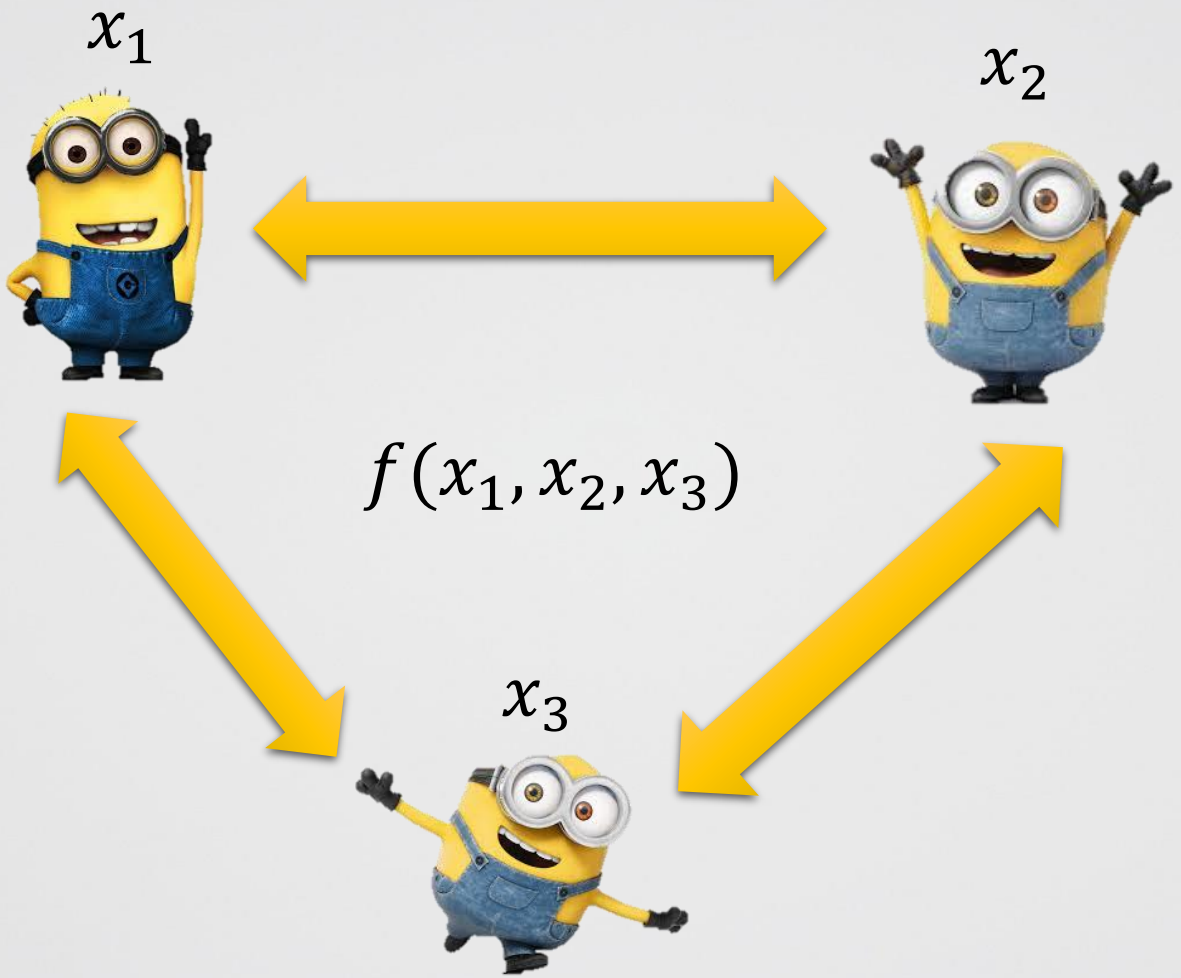
*NEC corporation, Israel

**Bar-Ilan University, Israel

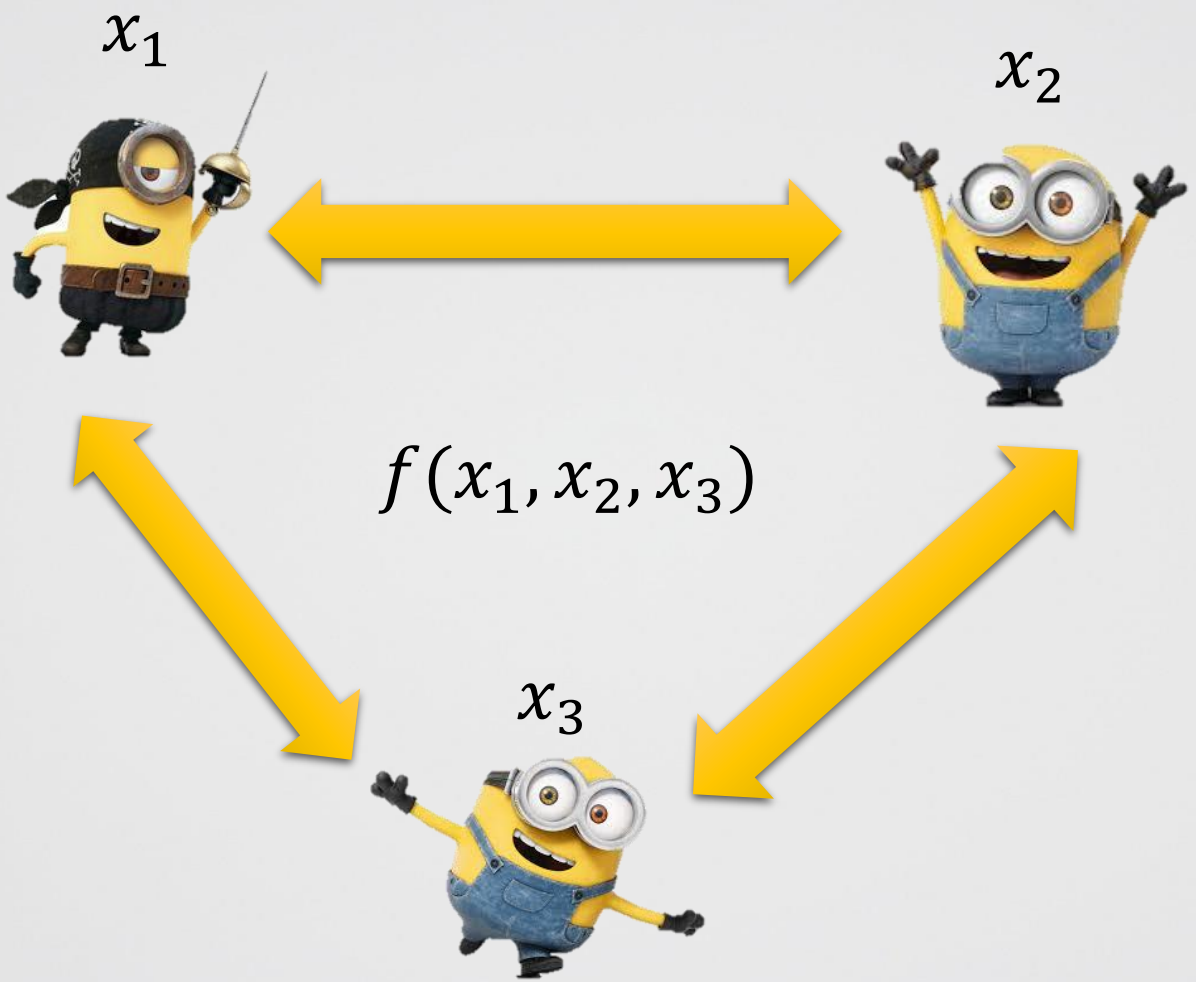
Eurocrypt 2017

BIU Center for Research in Applied Cryptography and Cyber Security

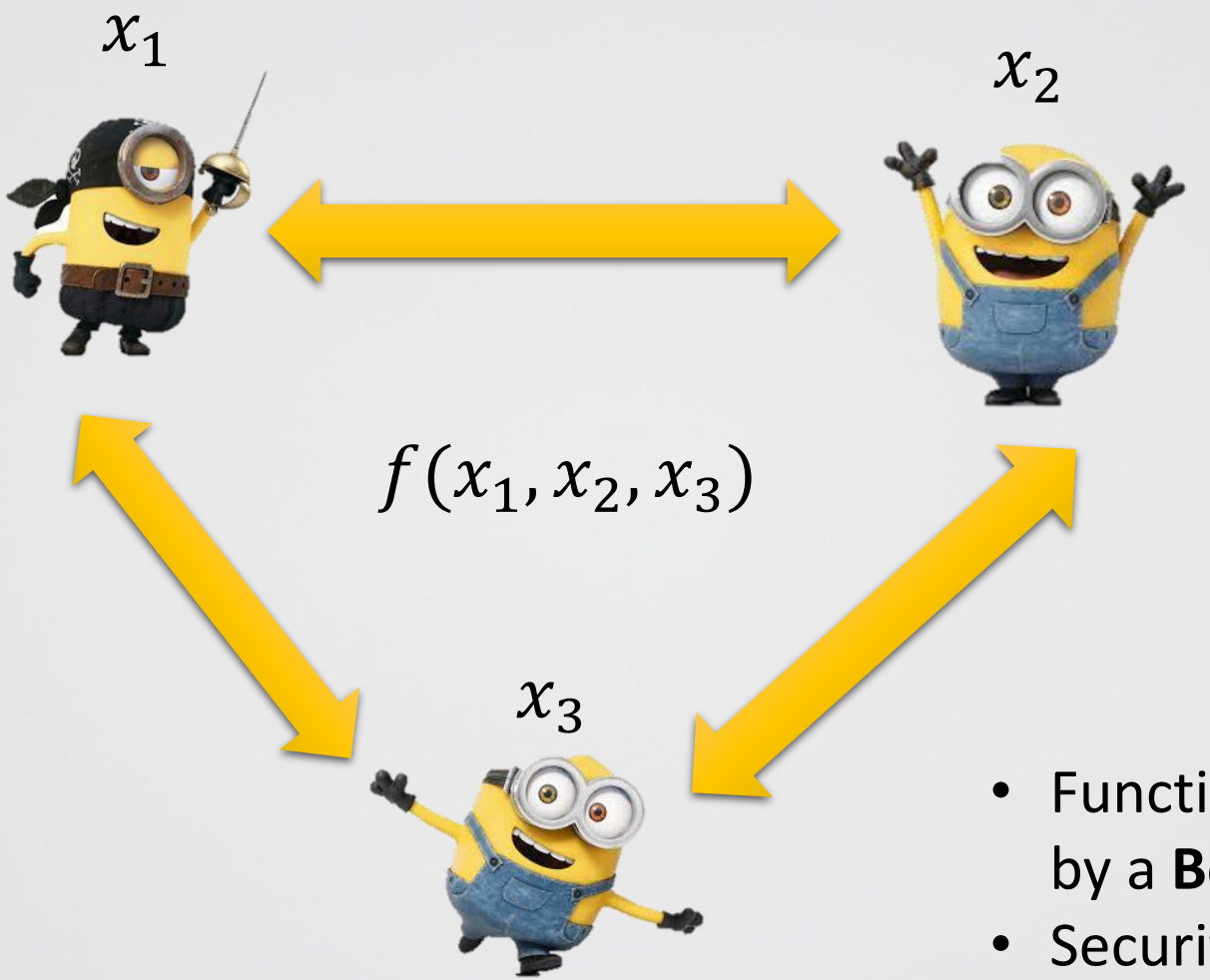
Secure Three-Party Computation with an Honest Majority



Secure Three-Party Computation with an Honest Majority

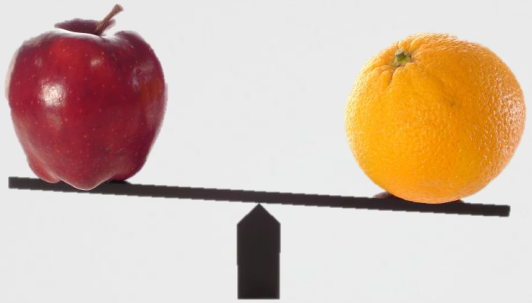


Secure Three-Party Computation with an Honest Majority

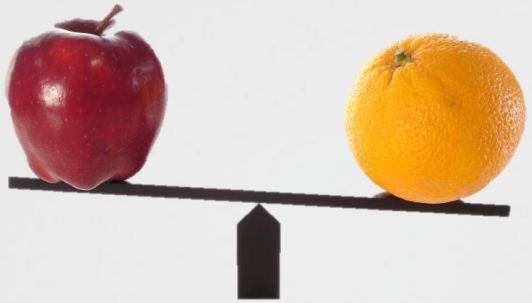


- Functionality is represented by a **Boolean circuit**
- Security with **abort**

High-Throughput Secure Three-Party Computation with an Honest Majority

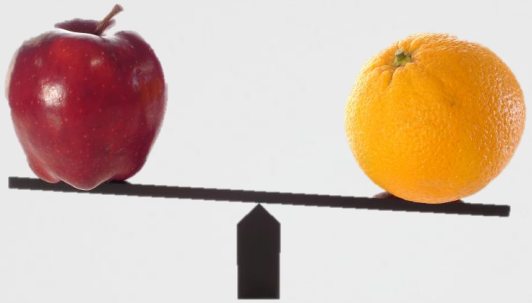


High-Throughput Secure Three-Party Computation with an Honest Majority



How much time it takes to compute a function?

High-Throughput Secure Three-Party Computation with an Honest Majority

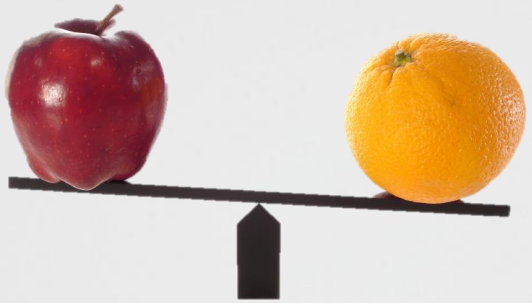


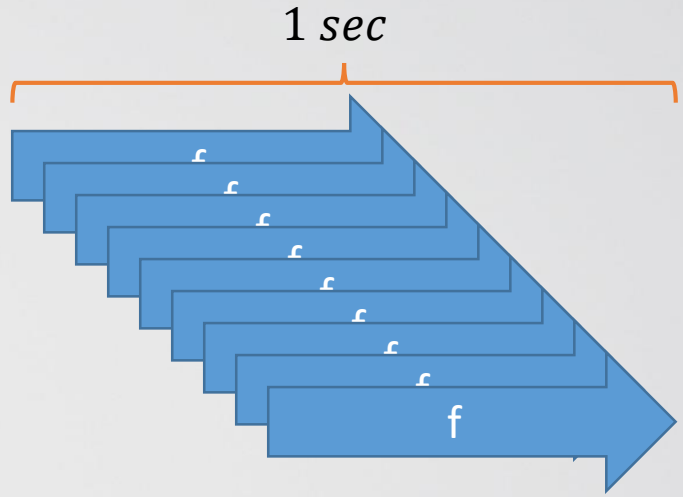
How much time it takes to compute a function?

Latency

High-Throughput Secure Three-Party Computation with an Honest Majority

How much time it takes to compute a function?





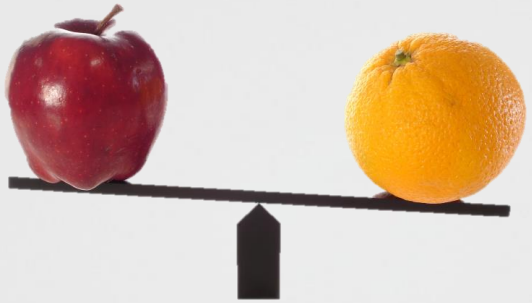
How many functions can we compute in one sec?

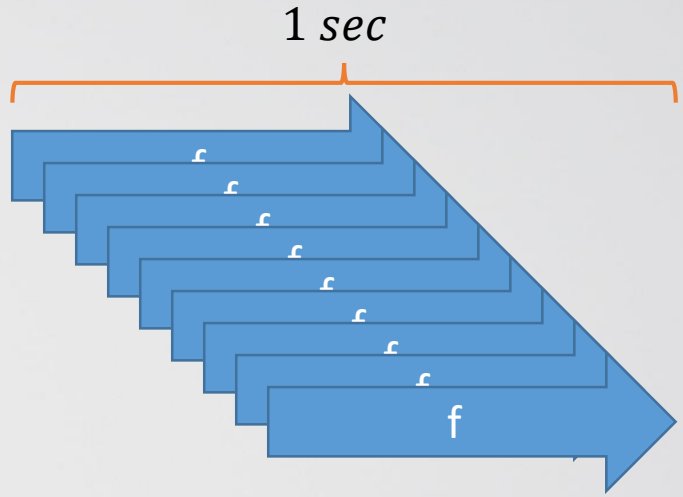
Latency

High-Throughput Secure Three-Party Computation with an Honest Majority

How much time it takes to compute a function?

Latency





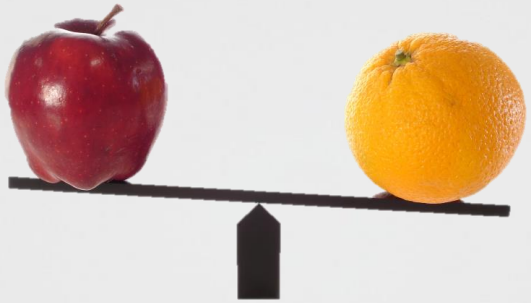
How many functions can we compute in one sec?

Throughput

Low Latency vs. High-Throughput

Low Latency

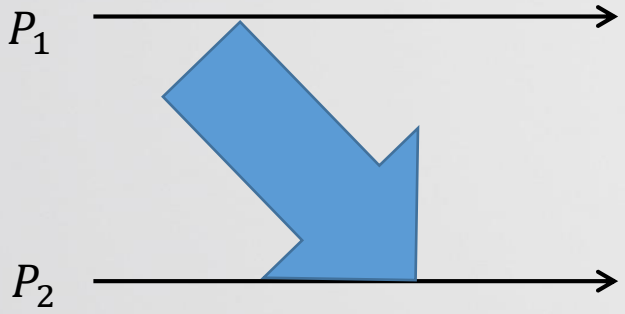
High-Throughput



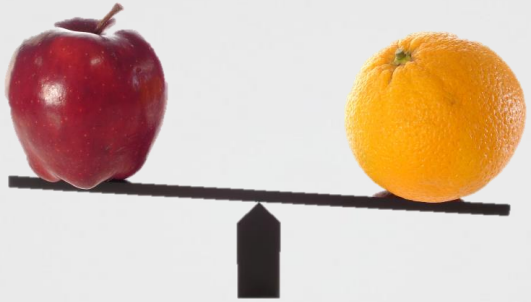
Low Latency vs. High-Throughput

Low Latency

- Constant rounds of communication



High-Throughput

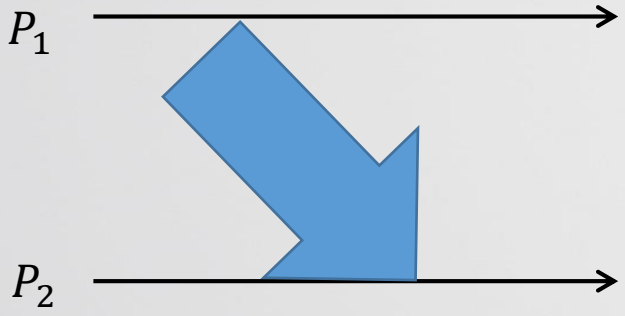


"the garbled-circuit approach"

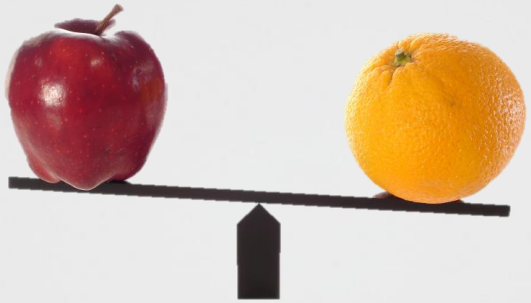
Low Latency vs. High-Throughput

Low Latency

- Constant rounds of communication

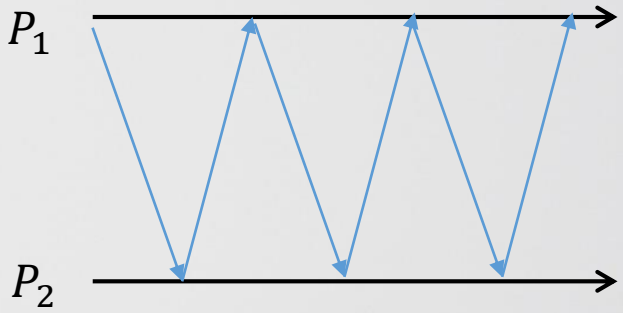


"the garbled-circuit approach"



High-Throughput

- Low bandwidth
- Simple Computations

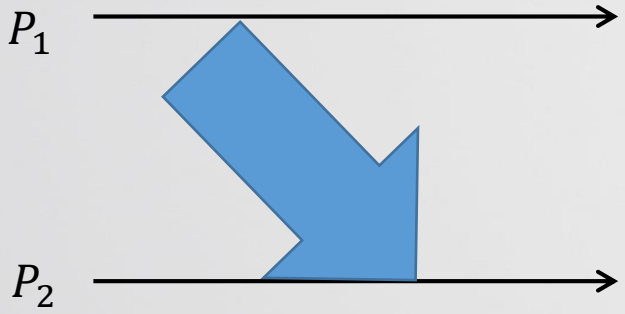


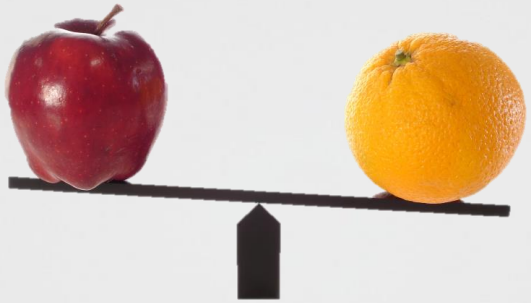
"the secret-sharing approach"

Low Latency vs. High-Throughput

Low Latency

- Constant rounds of communication

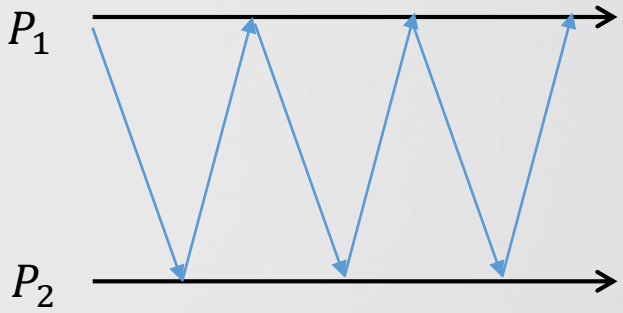




"the garbled-circuit approach"

High-Throughput

- Low bandwidth
- Simple Computations



"the secret-sharing approach"

The Starting Point: The Semi-honest protocol of [AFLNO16]

- Based on replicated secret sharing
- Requires **1 bit of communication** sent by each party per AND gate.
- Speed: compute over 7 billion AND gates per second
  - Concretely, over 1,300,000 AES operations per second

From Semi-Honest to Malicious adversary

- Sharing the inputs
- Emulating the circuit
- Output Reconstruction

From Semi-Honest to Malicious adversary

- Sharing the inputs
- Emulating the circuit
- Output Reconstruction

How to force the corrupted party to share its inputs “correctly”?

How to verify AND gates were computed correctly?

How to verify that the output was reconstructed correctly?

From Semi-Honest to Malicious adversary

- Sharing the inputs

How to force the corrupted party to share its inputs “correctly”?

- Emulating the circuit

How to verify AND gates were computed correctly?

- Output Reconstruction

How to verify that the output was reconstructed correctly?

Verification of AND Gates

A “multiplication triple” is a triple of shares $([a], [b], [c])$ such that $c = a \cdot b$

Verification of AND Gates

A “multiplication triple” is a triple of shares $([a], [b], [c])$ such that $c = a \cdot b$

Let $([x], [y], [z])$ be a triple generated by computing an AND gate

Let $([a], [b], [c])$ be a random triple

Verification of AND Gates

A “multiplication triple” is a triple of shares $([a], [b], [c])$ such that $c = a \cdot b$

Let $([x], [y], [z])$ be a triple generated by computing an AND gate

Let $([a], [b], [c])$ be a random triple

Verification of AND Gates

A "multiplication triple" is a triple of shares $([a], [b], [c])$ such that $c = a \cdot b$

Let $([x], [y], [z])$ be a triple generated by computing an AND gate

Let $([a], [b], [c])$ be a random triple

If $([a], [b], [c])$ is a "valid" triple, then we can use $([a], [b], [c])$ to detect cheating in $([x], [y], [z])$ with probability 1.

Verification of AND Gates

A "multiplication triple" is a triple of shares $([a], [b], [c])$ such that $c = a \cdot b$

Let $([x], [y], [z])$ be a triple generated by computing an AND gate

Let $([a], [b], [c])$ be a random triple

If $([a], [b], [c])$ is a "valid" triple, then we can use $([a], [b], [c])$ to detect cheating in $([x], [y], [z])$ with probability 1.

Sub-protocol "triple verification without opening"

Communication: <u>2 bits per each party</u>

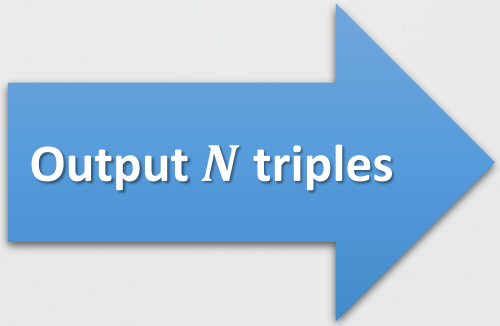
The Protocol

On-line protocol

1. Share the inputs
2. Run the Semi-honest protocol
3. Verify all ANDs gates
4. Reconstruct Output

3 bits per AND gate

The Protocol



On-line protocol

1. Share the inputs
2. Run the Semi-honest protocol
3. Verify all ANDs gates
4. Reconstruct Output

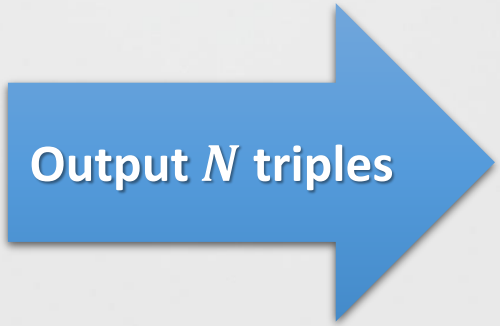
3 bits per AND gate

The Protocol

Pre-processing protocol

On-line protocol

1. Share the inputs
2. Run the Semi-honest protocol
3. Verify all ANDs gates
4. Reconstruct Output



3 bits per AND gate

The Protocol

Pre-processing protocol

Output N triples

On-line protocol

1. Share the inputs
2. Run the Semi-honest protocol
3. Verify all ANDs gates
4. Reconstruct Output

3 bits per AND gate

Generation of Random Multiplication Triples

- $[a], [b]$ are generated without any interaction!
- $[c]$ is computed using the semi-honest protocol

Generation of Random Multiplication Triples

- $[a], [b]$ are generated without any interaction!
- $[c]$ is computed using the semi-honest protocol

1 bit of communication!

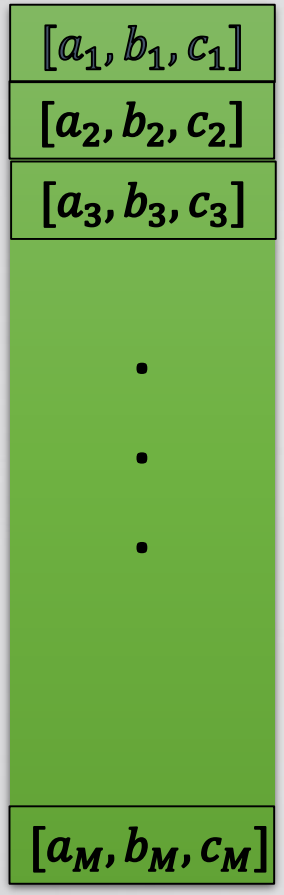
Generation of Random Multiplication Triples

- $[a], [b]$ are generated without any interaction!
- $[c]$ is computed using the semi-honest protocol

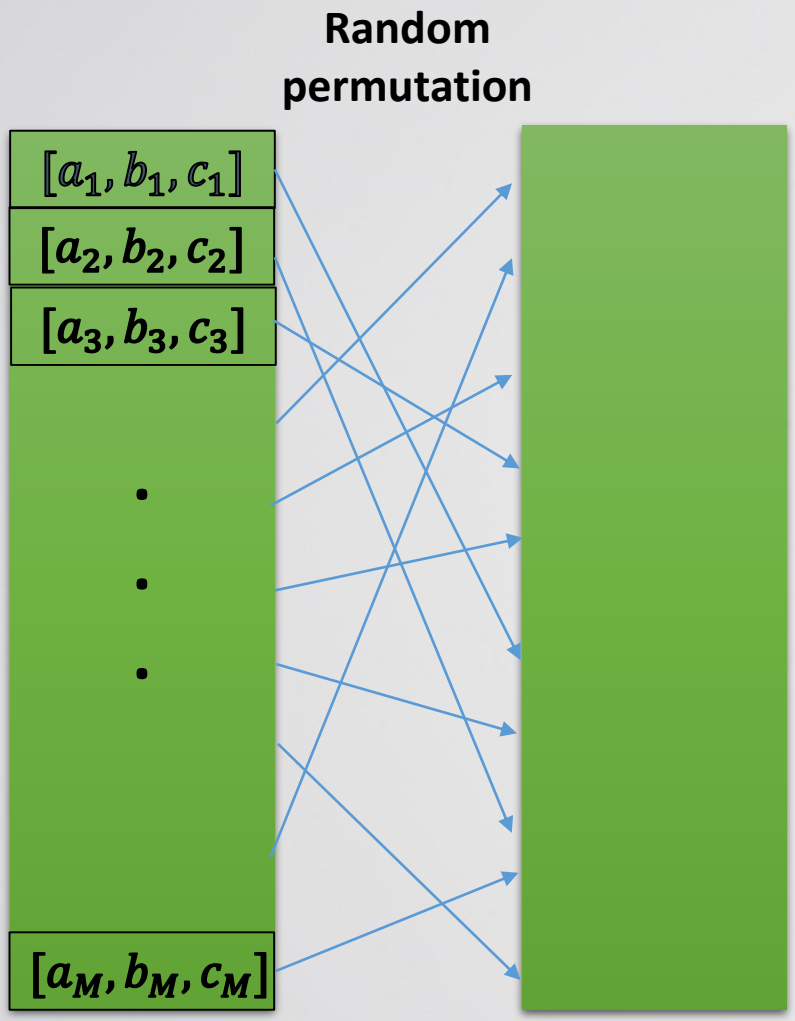
1 bit of communication!

How to verify that the triple is valid?

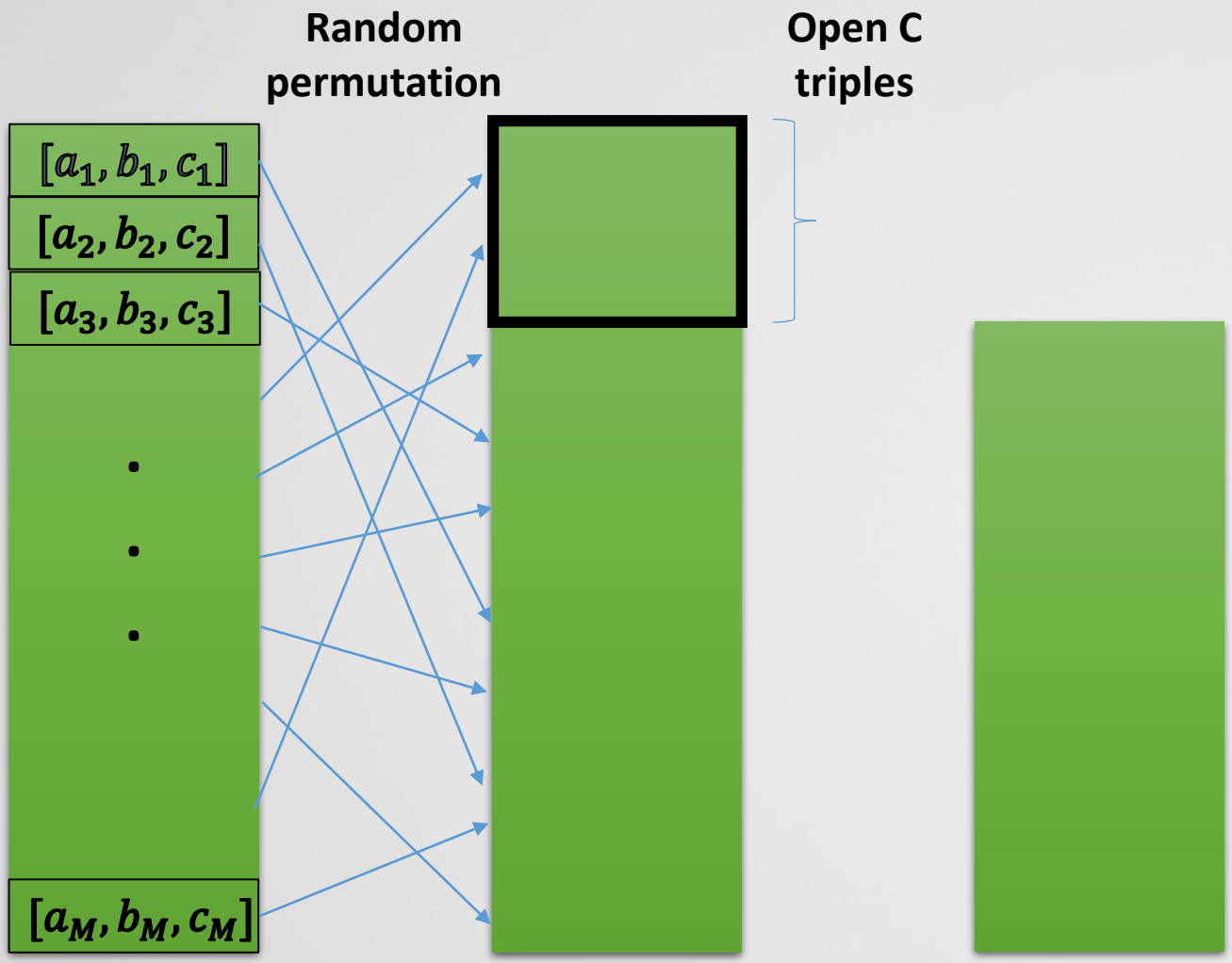
Generation of Random Multiplication Triples



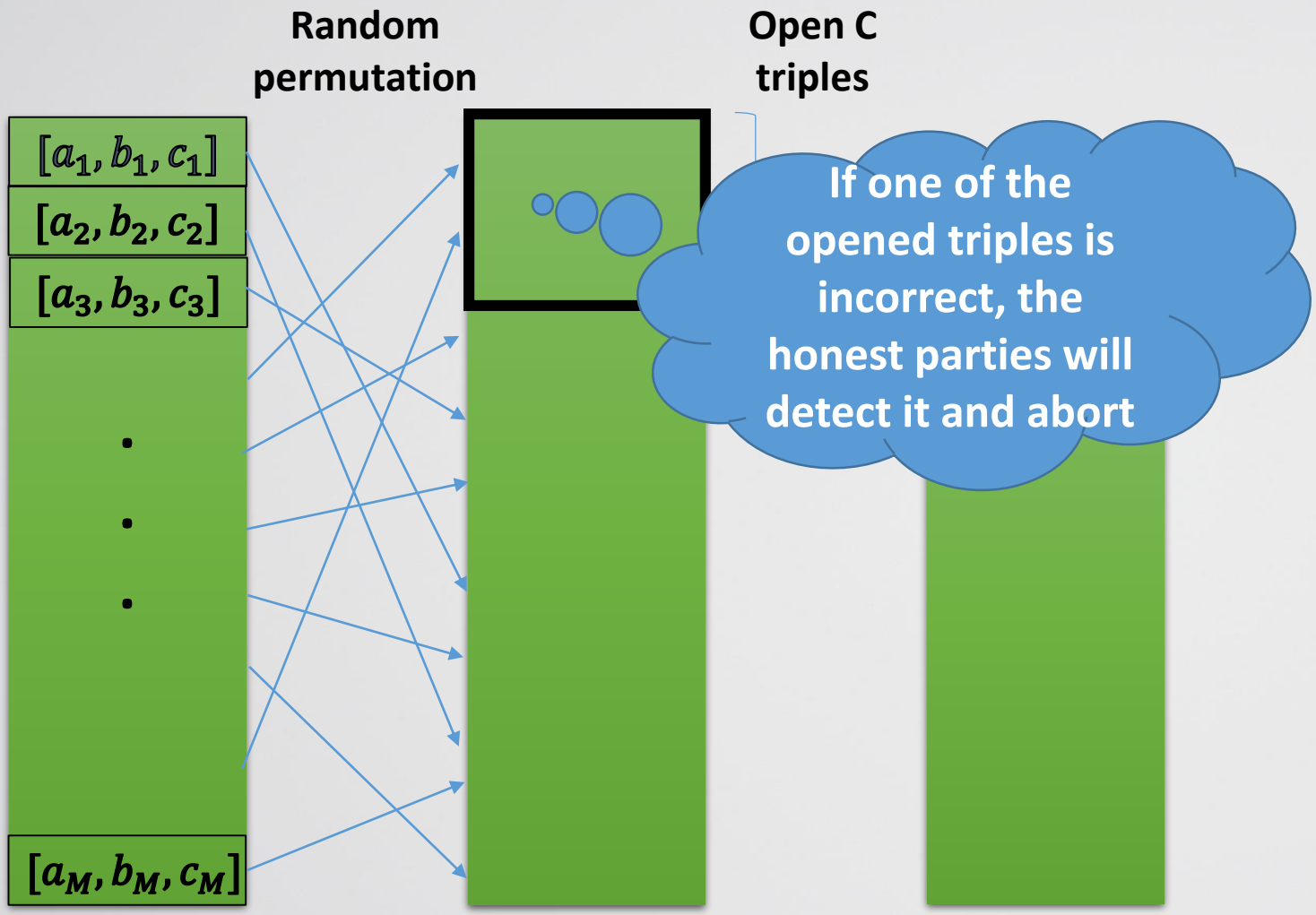
Generation of Random Multiplication Triples



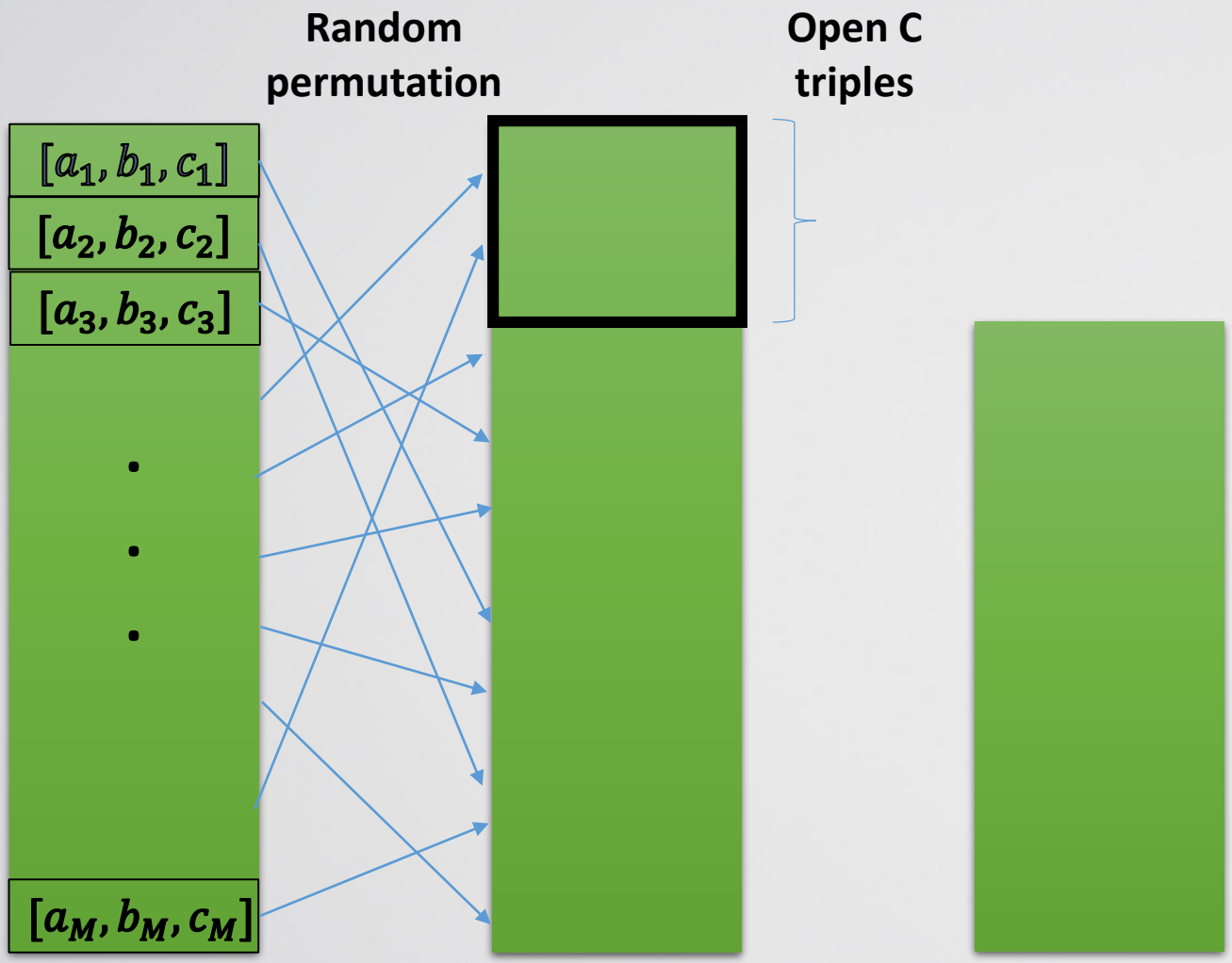
Generation of Random Multiplication Triples



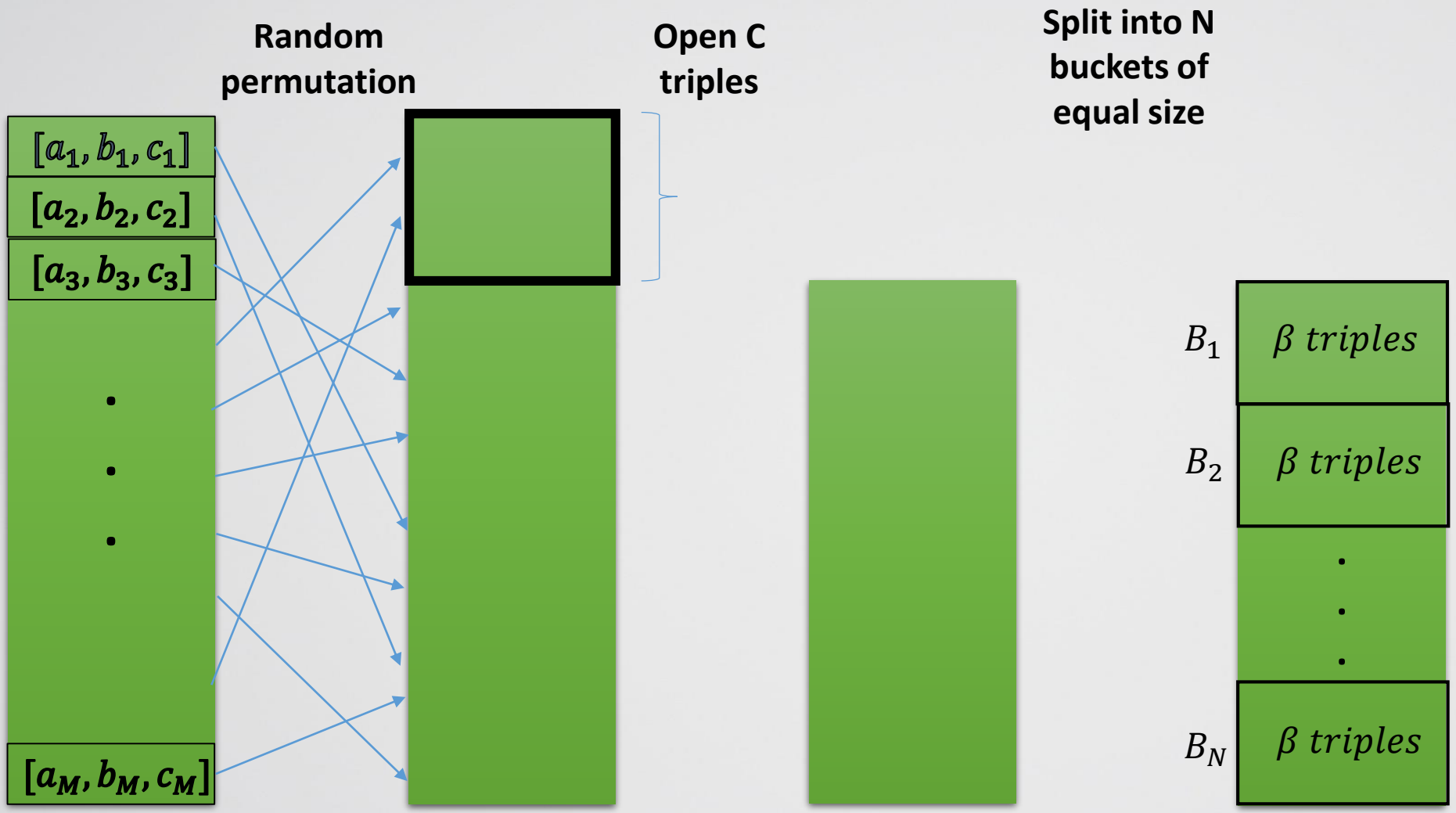
Generation of Random Multiplication Triples



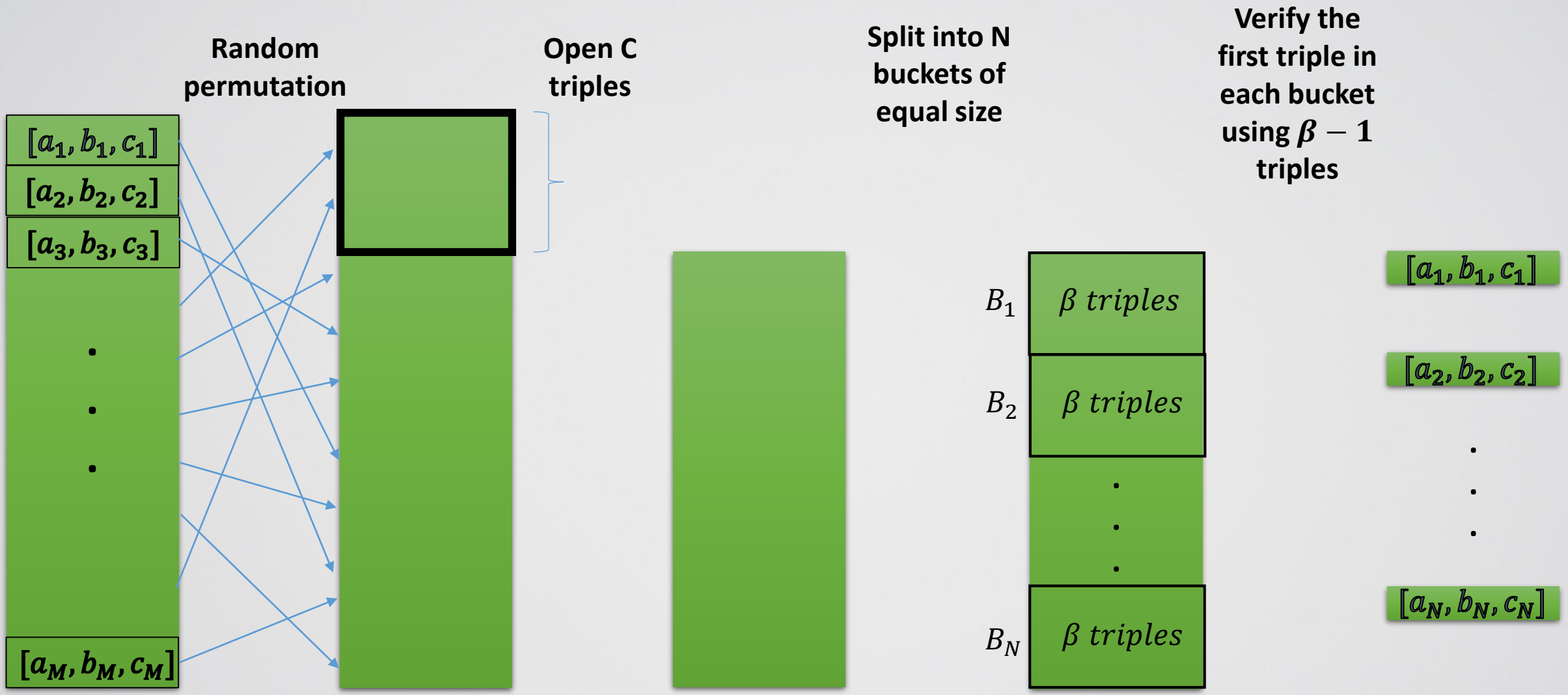
Generation of Random Multiplication Triples



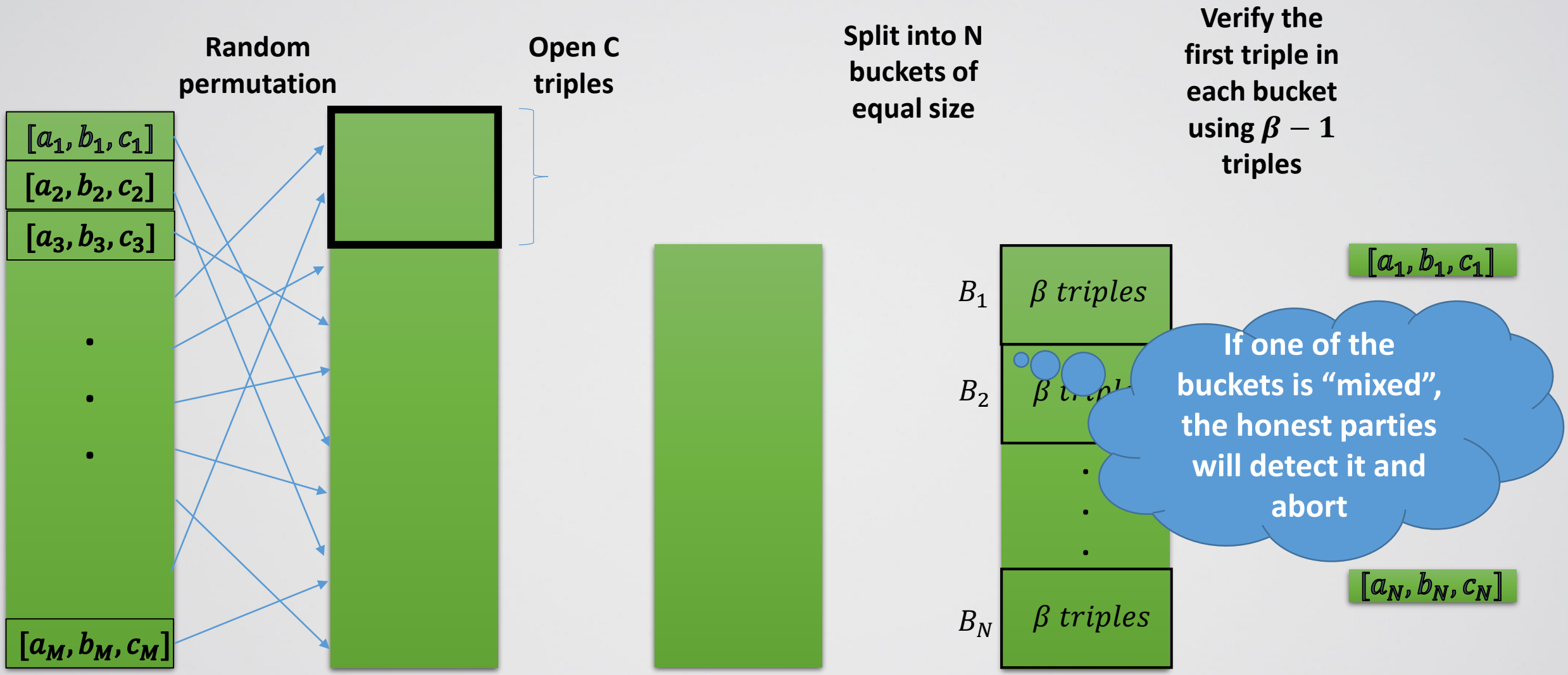
Generation of Random Multiplication Triples



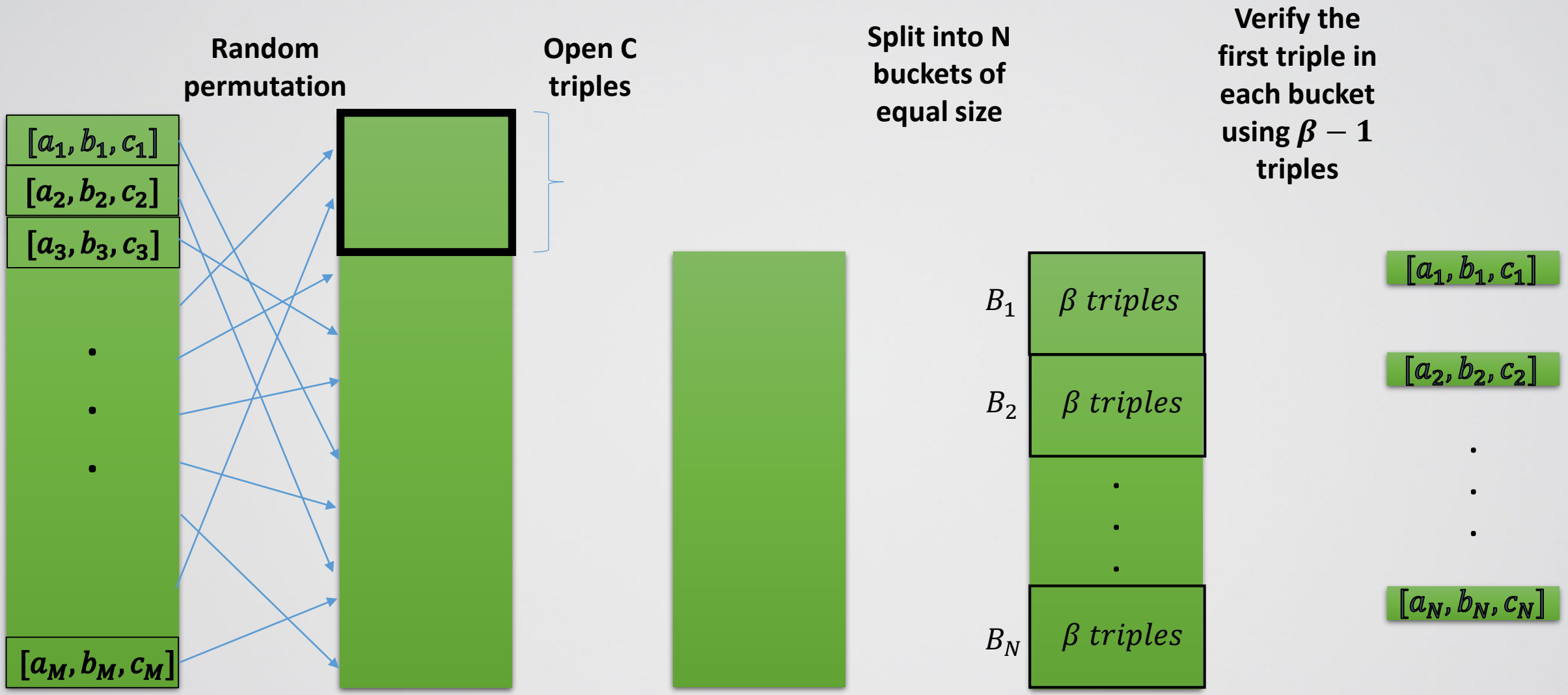
Generation of Random Multiplication Triples



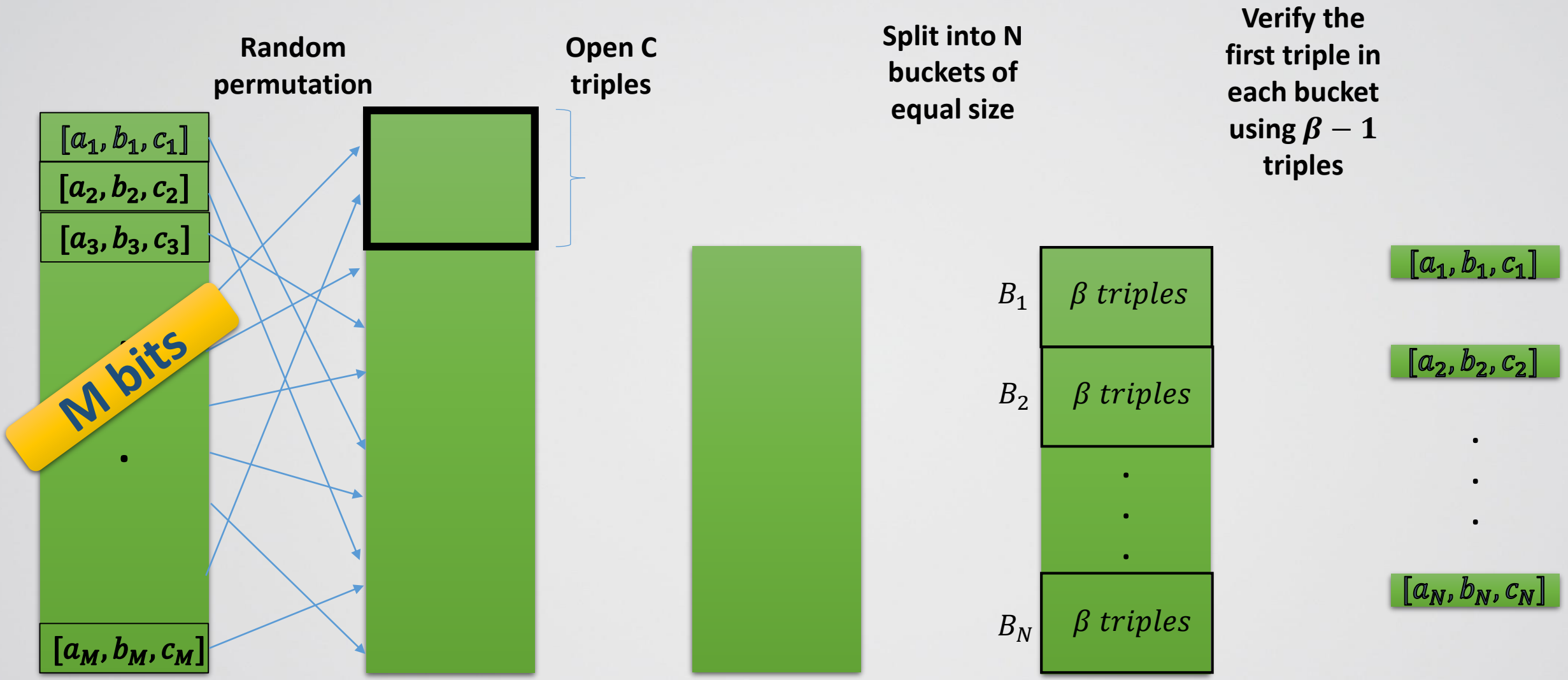
Generation of Random Multiplication Triples



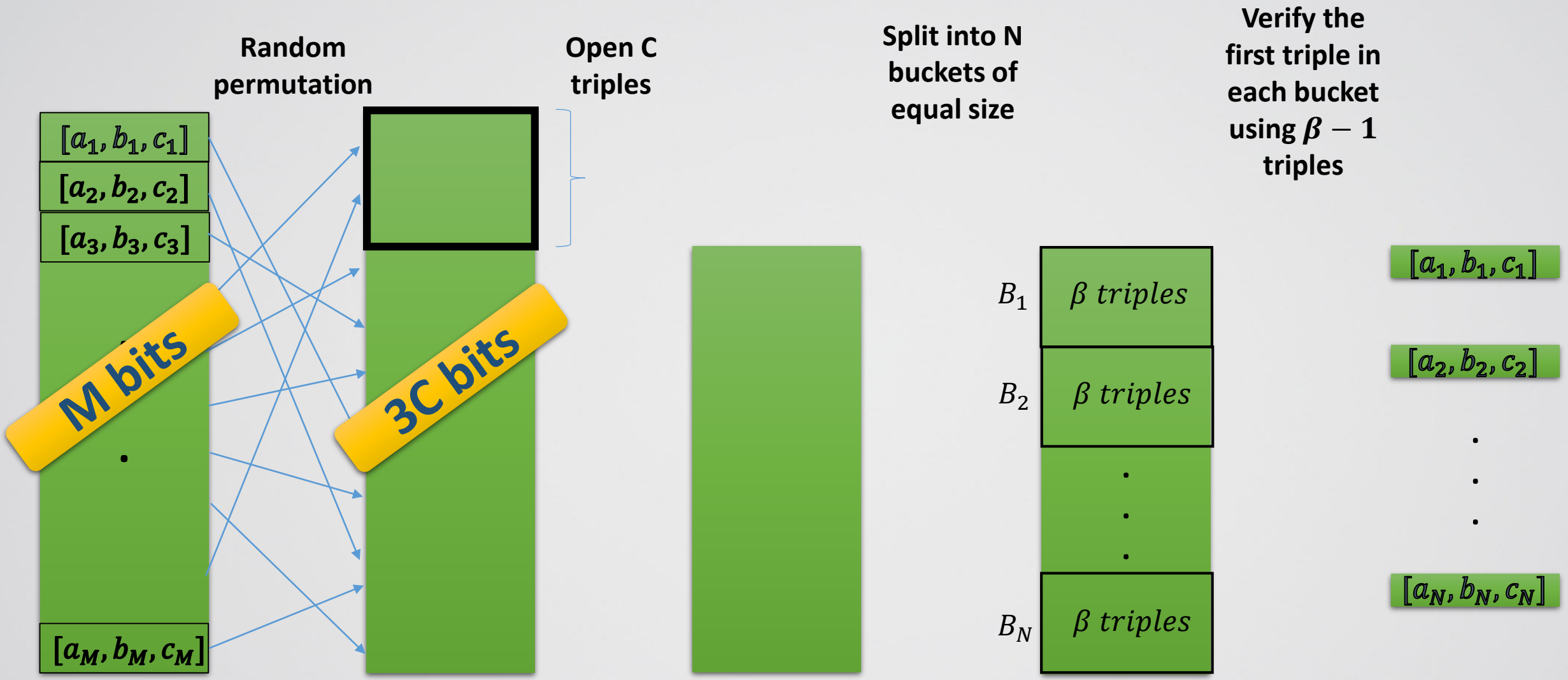
Generation of Random Multiplication Triples



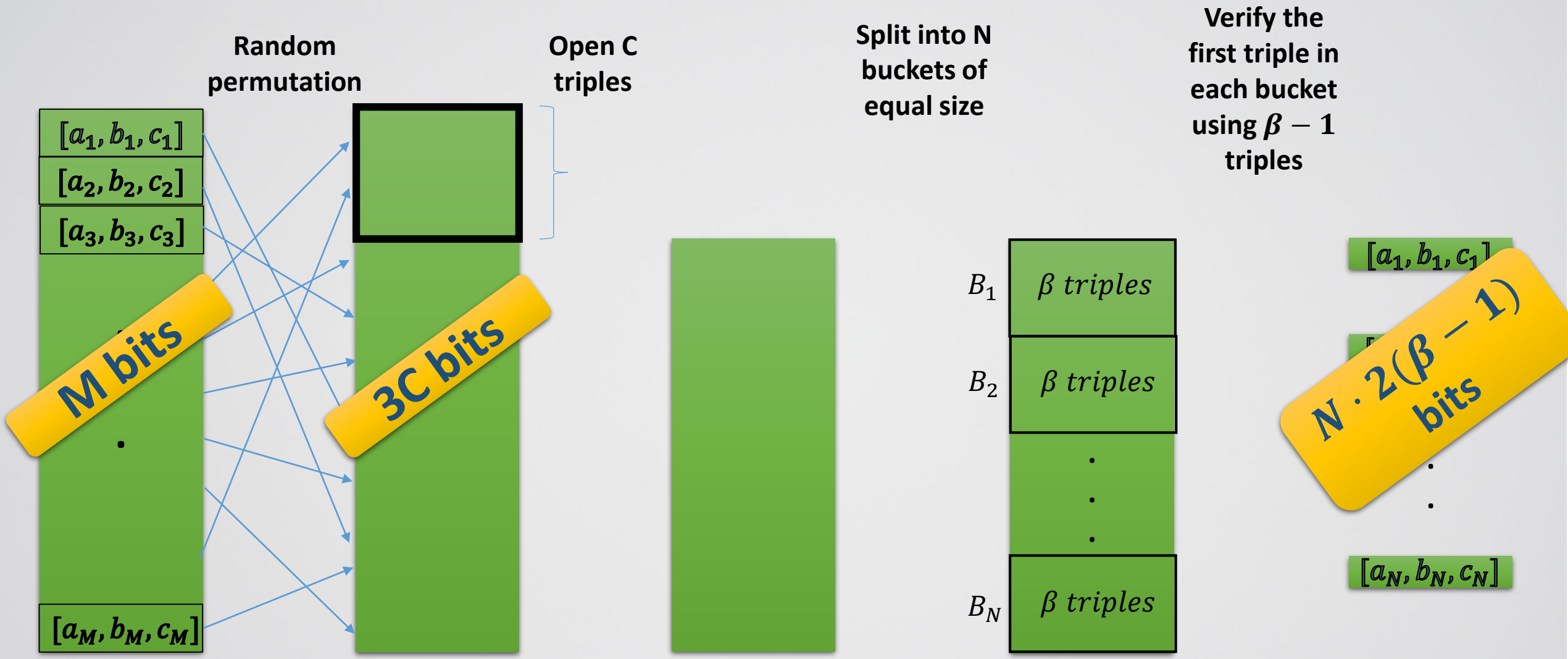
Generation of Random Multiplication Triples



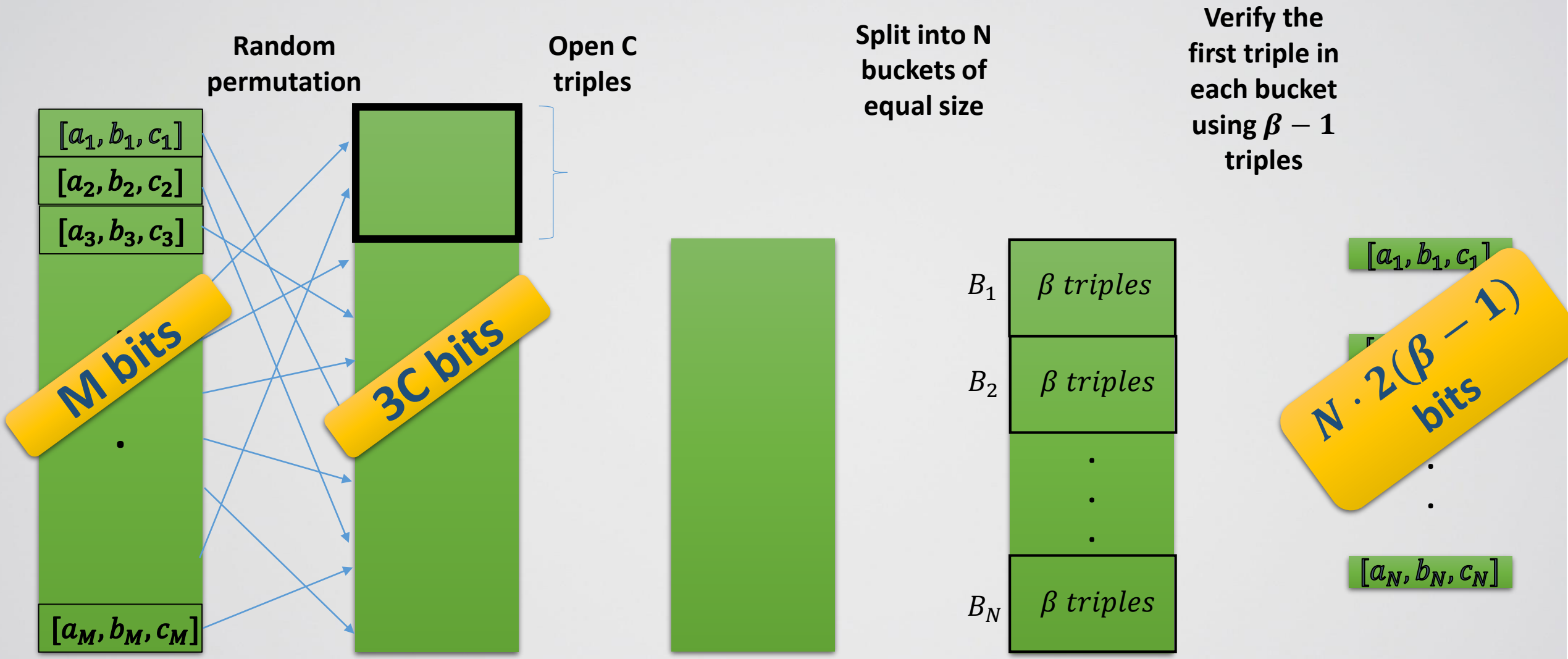
Generation of Random Multiplication Triples



Generation of Random Multiplication Triples



Generation of Random Multiplication Triples



Overall: $M + 3C + 2N(\beta - 1) = N(3\beta - 2) + 4C$ bits

The Balls & Buckets Game

Cost of the pre-processing: $N(3\beta - 2) + 2C$

$\underline{ame(A, N, B, C)}$

1. chooses $N\beta + C$ balls, where each ball can be **bad** or **good**

1.

The Balls & Buckets Game

Cost of the pre-processing: $N(3\beta - 2) + 2C$

$\underline{CC), BBNN, , AAGame(e(A, N, B, C)}$

1. chooses $N\beta + C$ balls, where each ball can be **bad** or **good**

1.

The Balls & Buckets Game

Cost of the pre-processing: $N(3\beta - 2) + 2C$

CC balls, where each ball can be bad or good

1. The adversary A chooses $N\beta + C$ balls, where each ball can be **bad** or **good**
2. chooses $N\beta + C$ balls, where each ball can be **bad** or **good**

1.

The Balls & Buckets Game

Cost of the pre-processing: $N(3\beta - 2) + 2C$

CC balls, where each ball can be bad or good

1. C balls are randomly chosen and opened.
   - If a bad ball was opened – the adversary loses

1.

The Balls & Buckets Game

Cost of the pre-processing: $N(3\beta - 2) + 2C$

CC balls, where each ball can be bad or good

1. C balls are randomly chosen and opened.
   - If a bad ball was opened – the adversary loses
2. The balls are being randomly thrown into N buckets of size β
   - If all the buckets are **fully good** or **fully bad**, then the adversary wins

The Balls & Buckets Game

Cost of the pre-processing: $N(3\beta - 2) + 2C$

CC balls, where each ball can be bad or good

1. C balls are randomly chosen and opened.
   - If a bad ball was opened – the adversary loses
2. The balls are being randomly thrown into N buckets of size β
   - If all the buckets are **fully good** or **fully bad**, then the adversary wins

<u>The goal</u>: Given a statistical parameter σ and number of triples N to generate, find β, C of minimal size such that:

$$Pr[A\ wins] \leq 2^{-\sigma}$$

The Balls & Buckets Game

The goal: Given a statistical parameter σ and number of triples N to generate, find β, C of minimal size such that:

$$Pr[A\ wins] \leq 2^{-\sigma}$$

	Tiny OT	Our Work
β	4	3
C	65,536	3
$M = N\beta + C$	4,259,840	3,154,731

$$N = 2^{20}, \sigma = 40$$

Summary & Efficiency

Parameters: $N = 2^{20}, \beta = 3, C = 3$

Pre-processing protocol

1. Generate $N\beta + C$ triples
2. Open C triples
3. Split into buckets and use $\beta - 1$ triples to verify one triple

7 bits per AND gate

Output N triples

On-line protocol

1. Share the inputs
2. Run the Semi-honest protocol
3. Verify all ANDs gates
4. Reconstruct Output

3 bits per AND gate

Efficiency Comparison

	Communication (bits)	Number of AES Computations
MRZ[15]	85N	3N
Our protocol	10N	N/5

N = number of AND gates

Efficiency Comparison

Three-party with honest majority at the cost of semi-honest Yao

	Communication (bits)	Number of AES Computations
MRZ[15]	85N	3N
Our protocol	10N	N/5

N = number of AND gates

More Improvements & Implementation

"Optimized Honest-Majority MPC for Malicious Adversaries – Breaking the 1-Billion-Gate Barrier" *(To appear in IEEE S&P 2017)*

- Reduction of communication from 10 bits to 7 bits only!
  - This is achieved by better combinatorics that allows to use a bucket of size 2 (instead of 3)
- Cache-efficient shuffling
- Implementation
  - Basic implementation: **503,766,615 AND gates/sec**
  - With optimizations: **1,152,751,967 AND gates/sec**

THANK-YOU!

nofarie@cs.biu.ac.il