Scalable RSA Modulus Generation with a Dishonest Majority

Muthu Venkitasubramaniam

Ligero Inc. & University of Rochester

Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, abhi shelat, Ruihan Wang

IEEE S&P 2021

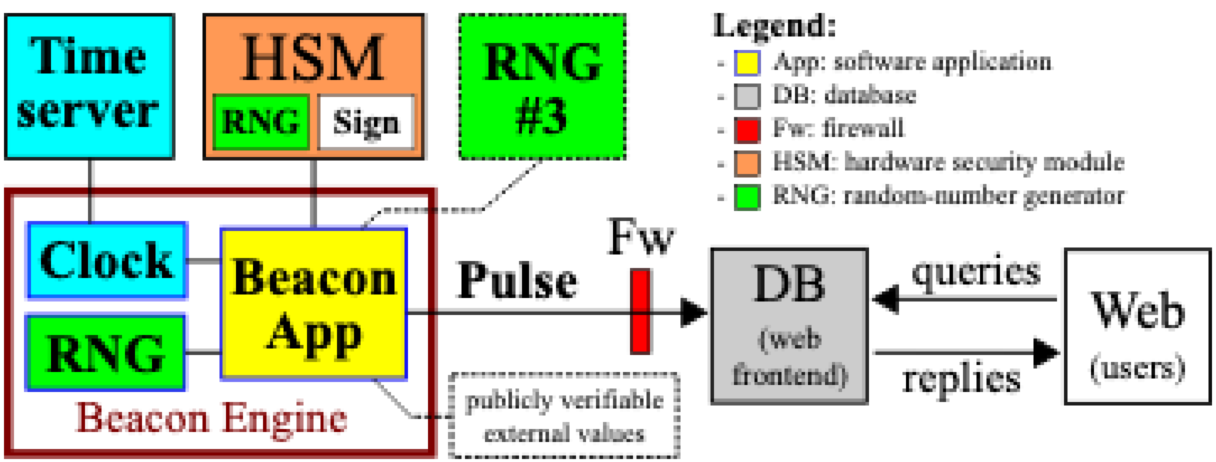
What is an RSA Modulus?

$$N = p \cdot q$$

Biprime - product of exactly two primes

Why? RSA History

- 1977 - RSA Public-Key Encryption
- 1999 - Paillier Public-Key Encryption
- 2001 - CRS for UC setting
- 2018 - Verifiable Delay Functions (VDF)



NIST Randomness Beacon

Source: https://csrc.nist.gov/projects/interoperable-randomness-beacons

Verifiable Delay Functions

- [Rivest-Shamir-Wagner96] introduced Inherently Sequential functions (ISH)

$$y = g^{2^T} \bmod N$$

- 2018 - VDF constructions by Pietrzak, Wesolowski

Goal

Sample a biprime N where factorization “hidden”

USE MPC!

Desiderata

- **Modulus size:** 2048 bits
- **Threshold:** n-1 corruption
- **# Participants:** > 1000
- **Party Spec:** "Lightweight"
- **Bandwidth:** < 5 Mbps
- Security: 60-bit statistical security
  128-bit computational security

Protocol Blueprint

Step 1: Design protocol for **PASSIVE corruptions**

Step 2: Upgrade security to tolerate **ACTIVE corruptions**

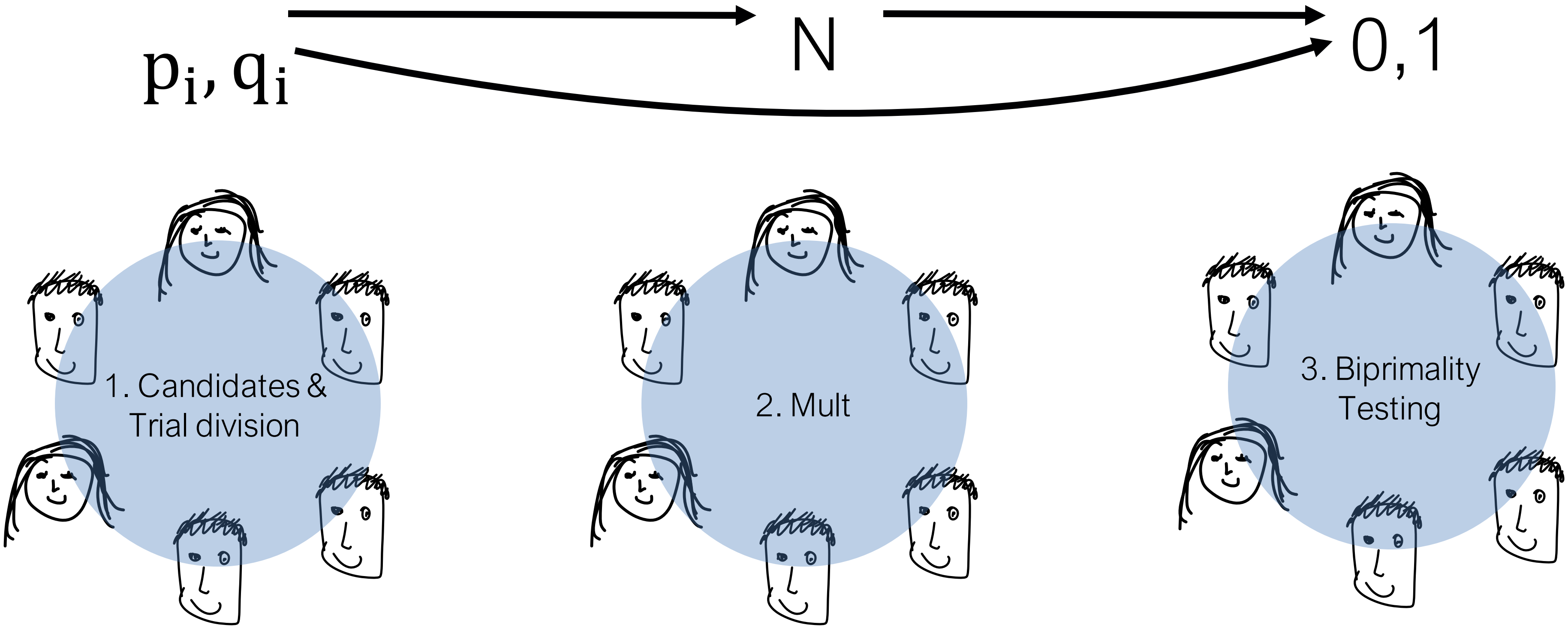
Step 1: Scalable Passive Protocol

Previous Works: Overview

Milestone	Work	Adversary	Parties	Corruption Threshold
First Work	[BF97]	Passive	n >= 3	t < n/2
	[FMY98]	Active	n	t < n/2
	[PS98]	Active	2	t = 1
Based on OT	[Gil99]	Passive	2	t = 1
	[ACS02]	Passive	n	t < n/2
	[DM10]	Active	3	t = 1
	[HMRT12]	Active	n	t < n
	[FLOP18]	Active	2	t = 1
	[CCD+20]	Active	n	t < n

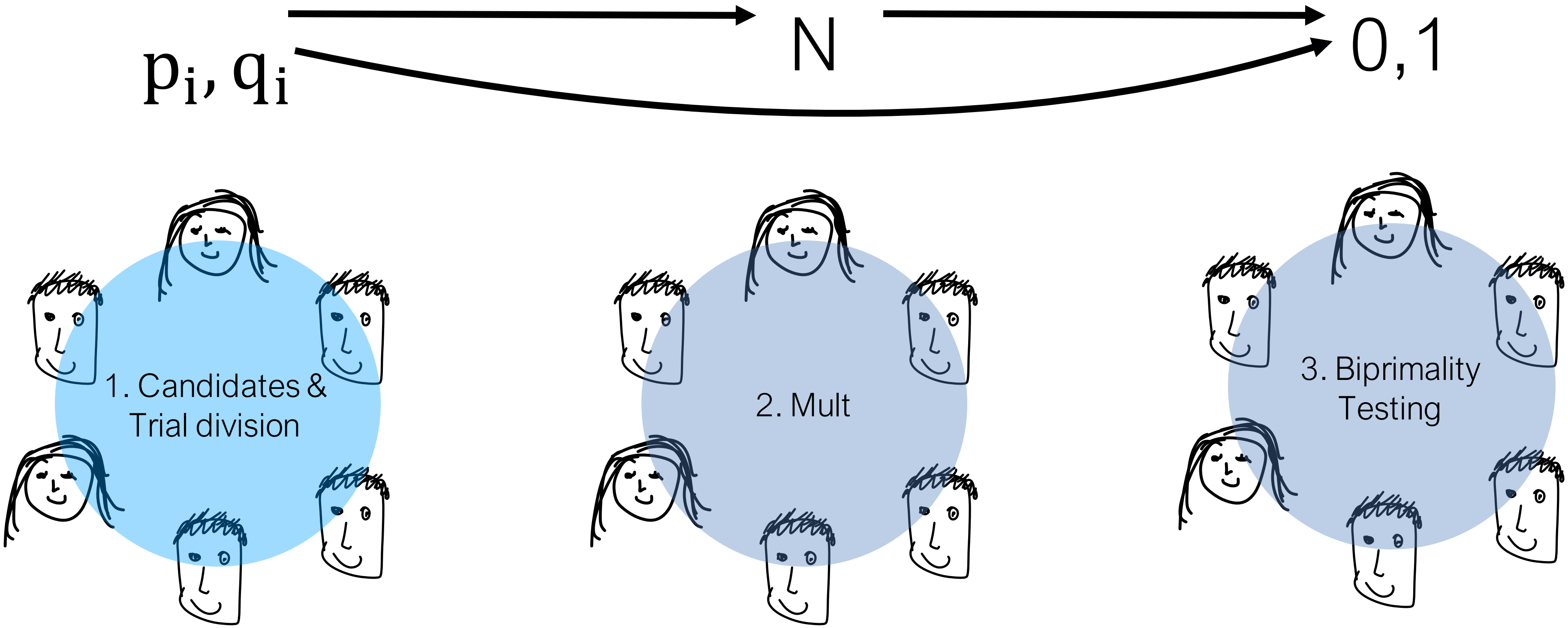
Boneh-Franklin Framework

[BF97]



Boneh-Franklin Framework

[BF97]

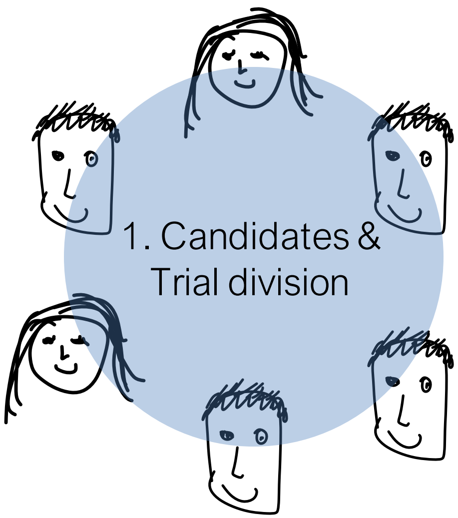


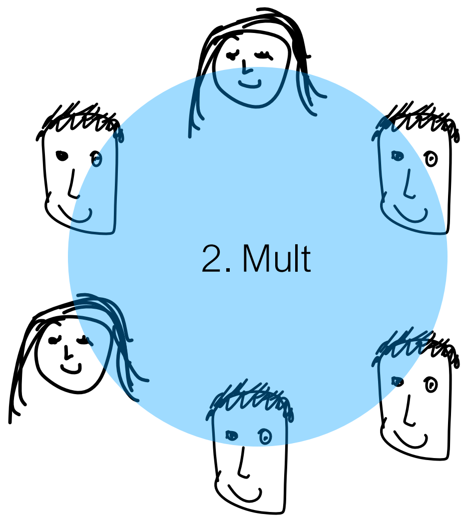
Parties choose p_i, q_i randomly

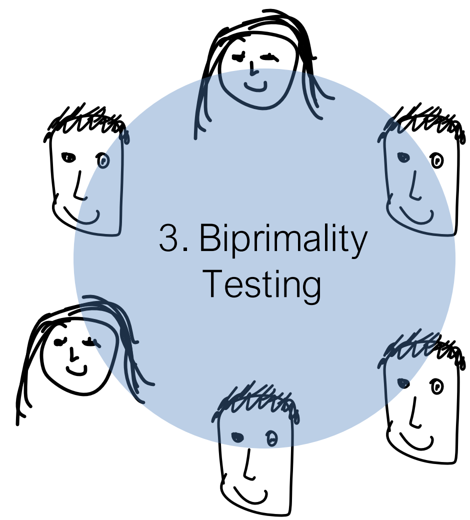
Boneh-Franklin Framework

[BF97]

$$\begin{tikzcd} p_i, q_i \arrow[r] \arrow[rr, bend right] & N \arrow[r] & 0,1 \end{tikzcd}$$







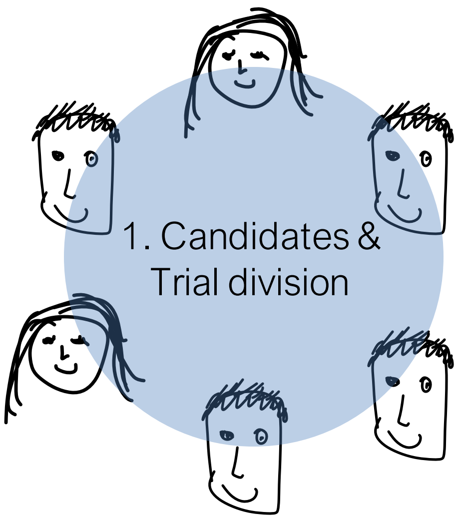
Parties choose p_i, q_i randomly

$$N = \left(\sum_i p_i\right) \cdot \left(\sum_i q_i\right)$$

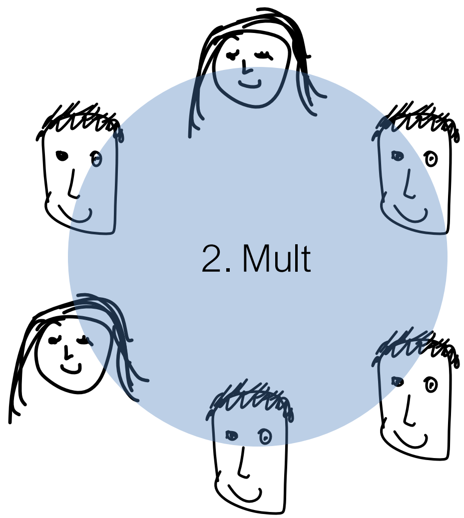
Boneh-Franklin Framework

[BF97]

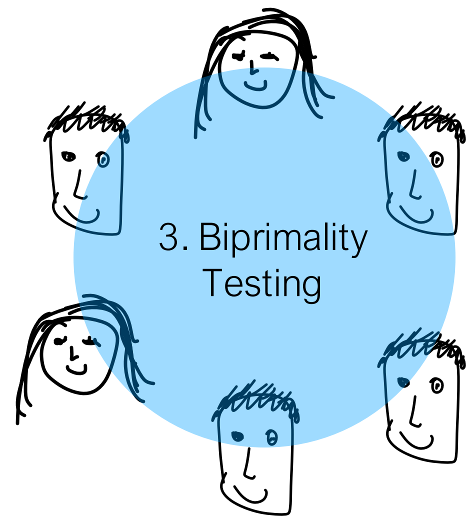
$p_i, q_i \longrightarrow N \longrightarrow 0,1$



Parties choose p_i, q_i randomly



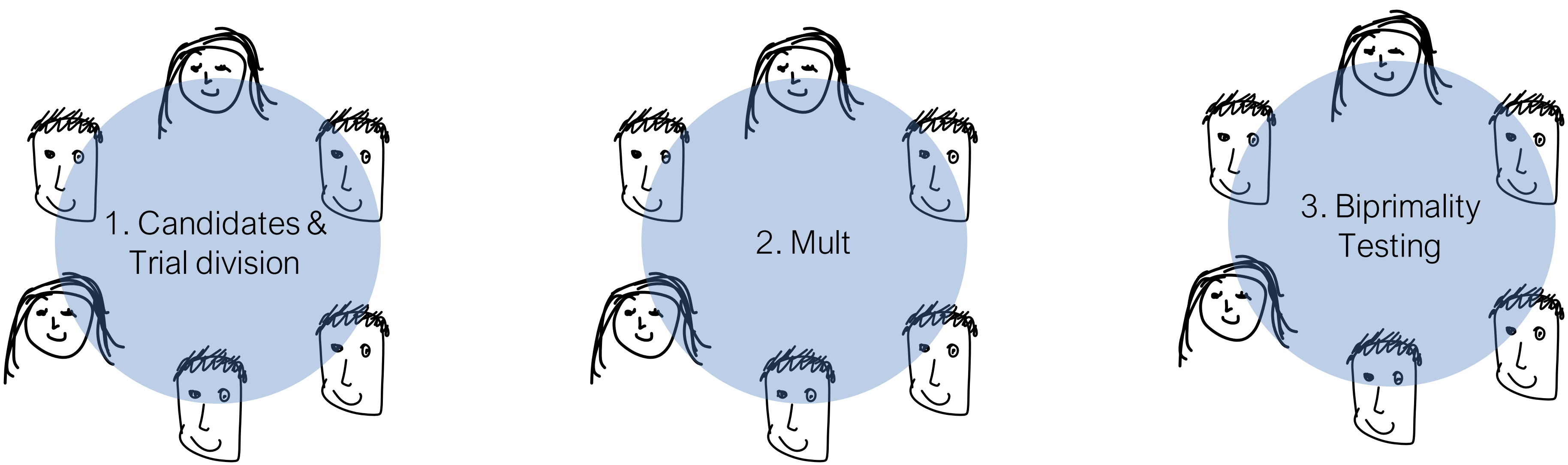
$$N = \left(\sum_i p_i\right) \cdot \left(\sum_i q_i\right)$$



Is N the product of two primes?

[CCD+20] Passive Protocol

$$p_i, q_i \longrightarrow N \longrightarrow 0,1$$

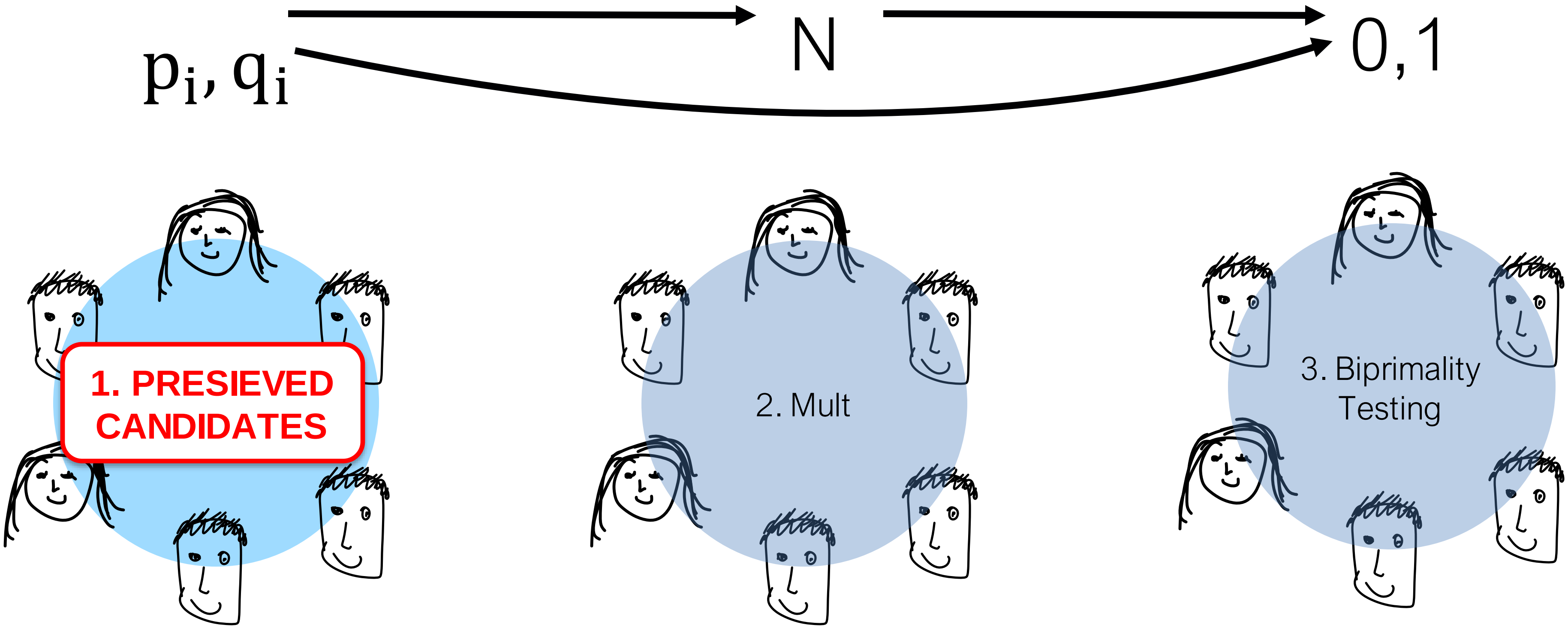


Parties choose p_i, q_i randomly

$$N = \left(\sum_i p_i\right) \cdot \left(\sum_i q_i\right)$$

Is N the product of two primes?

[CCD+20] Passive Protocol



Parties choose p_i, q_i randomly

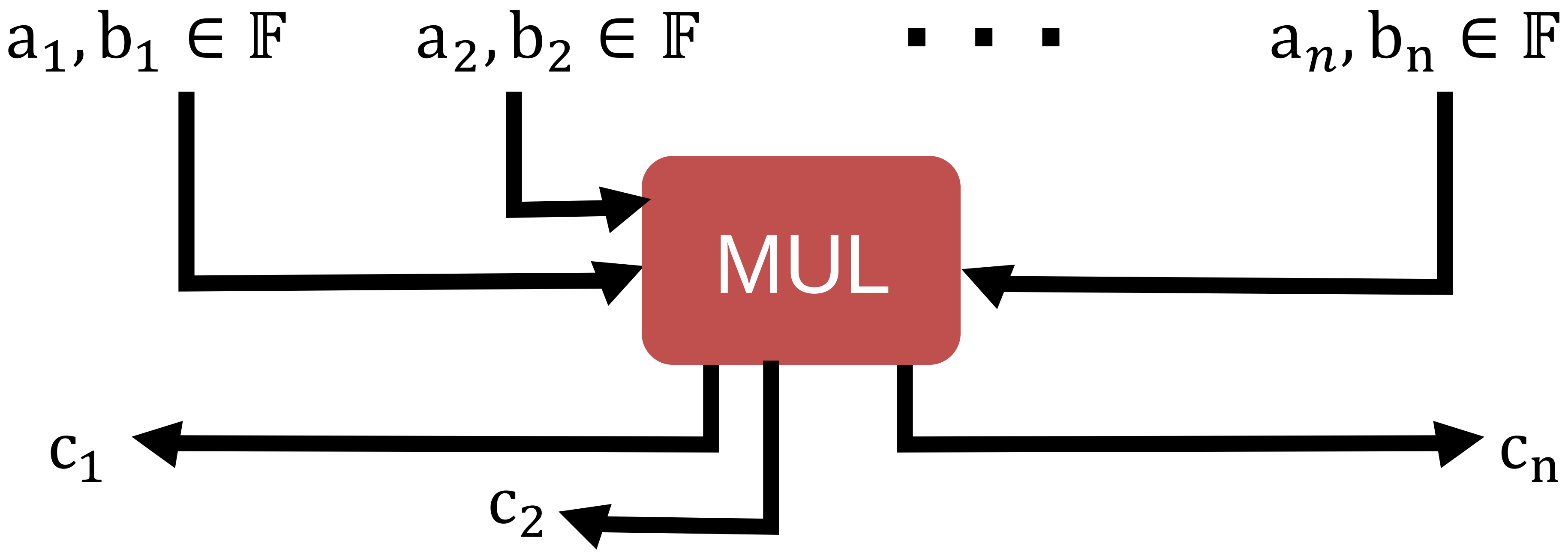
$$N = \left(\sum_i p_i\right) \cdot \left(\sum_i q_i\right)$$

Is N the product of two primes?

[CCD+20] Passive Protocol

1. Pre-sieving candidates	Secure Multiplication
2. Mult	Secure Multiplication
3. Biprimality testing	Secure Multiplication Jacobi test [BF97]

Secure Multiplication



$$\sum c_i = \left(\sum a_i\right) \cdot \left(\sum b_i\right)$$

Implementing Secure Multiplication

- Oblivious Linear Evaluation (OLE)
  - Scales quadratic in # parties
- Threshold Additively Homomorphic Encryption (TAHE) [CDN01]
  - Scales linearly in # parties
- Our Approach: TAHE with verifiable coordinator
  - per-party comm. scales logarithmically in # parties

Threshold AHE with a coordinator

	P_i	C
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$\mathrm{Enc}_{\mathrm{PK}}(p_i)$	
Coord. adds		$\sum \mathrm{Enc}_{\mathrm{PK}}(p_i)$
Receive Enc(p) from Coord.	$\mathrm{Enc}_{\mathrm{PK}}(p)$	
Multiply by q_i	$q_i \cdot \mathrm{Enc}_{\mathrm{PK}}(p)$	
Coord. adds		$\sum q_i \cdot \mathrm{Enc}_{\mathrm{PK}}(p)$
Receive Enc(pq) from Coord.	$\mathrm{Enc}_{\mathrm{PK}}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$\mathrm{Enc}_{PK}(p_i)$	
Coord. adds		$\sum \mathrm{Enc}_{PK}(p_i)$
Receive Enc(p) from Coord.	$\mathrm{Enc}_{PK}(p)$	
Multiply by q_i	$q_i \cdot \mathrm{Enc}_{PK}(p)$	
Coord. adds		$\sum q_i \cdot \mathrm{Enc}_{PK}(p)$
Receive Enc(pq) from Coord.	$\mathrm{Enc}_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

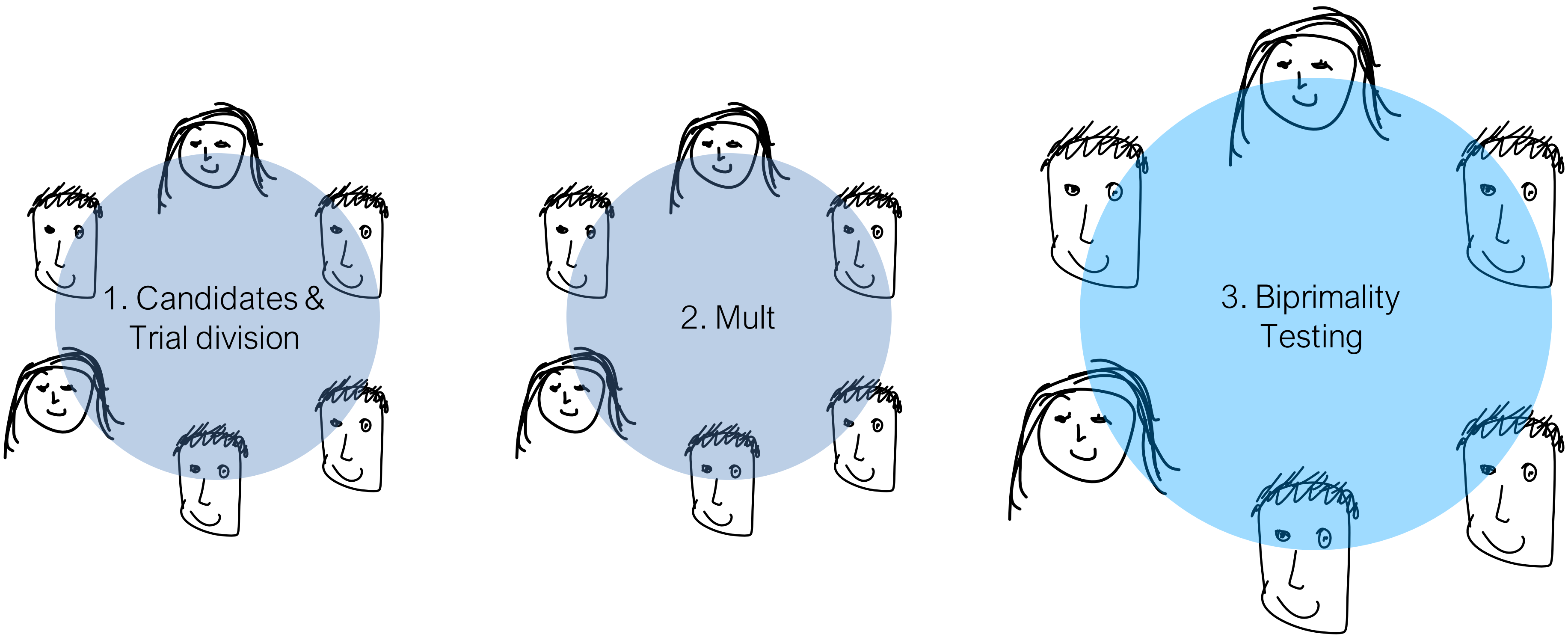
Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

Threshold AHE with a coordinator

	P_i	C
	PK	
Parties' secret shares	p_i, q_i	
Key Generation	sk_i	
Encrypt p_i	$Enc_{PK}(p_i)$	
Coord. adds		$\sum Enc_{PK}(p_i)$
Receive Enc(p) from Coord.	$Enc_{PK}(p)$	
Multiply by q_i	$q_i \cdot Enc_{PK}(p)$	
Coord. adds		$\sum q_i \cdot Enc_{PK}(p)$
Receive Enc(pq) from Coord.	$Enc_{PK}(p \cdot q)$	
Decrypted product	$p \cdot q$	

[BF97]'s Distributed Biprimality Test



Test whether N is the product of two primes [BF97]

- Jacobi Test (Dist "Miller-Rabin" test)
- GCD Test

[BF97]'s Distributed Biprimality Test

$$\gamma^{\frac{(p-1)(q-1)}{4}} \pmod N$$

Test whether N is the product of two primes [BF97]

- Jacobi Test (Dist "Miller-Rabin" test)
- GCD Test

[BF97]'s Distributed Biprimality Test

$$\gamma^{\frac{N-\sum p_i-\sum q_i+1}{4}} \pmod{N}$$

Test whether N is the product of two primes [BF97]

- Jacobi Test (Dist “Miller-Rabin" test)
- GCD Test

[BF97]'s Distributed Biprimality Test

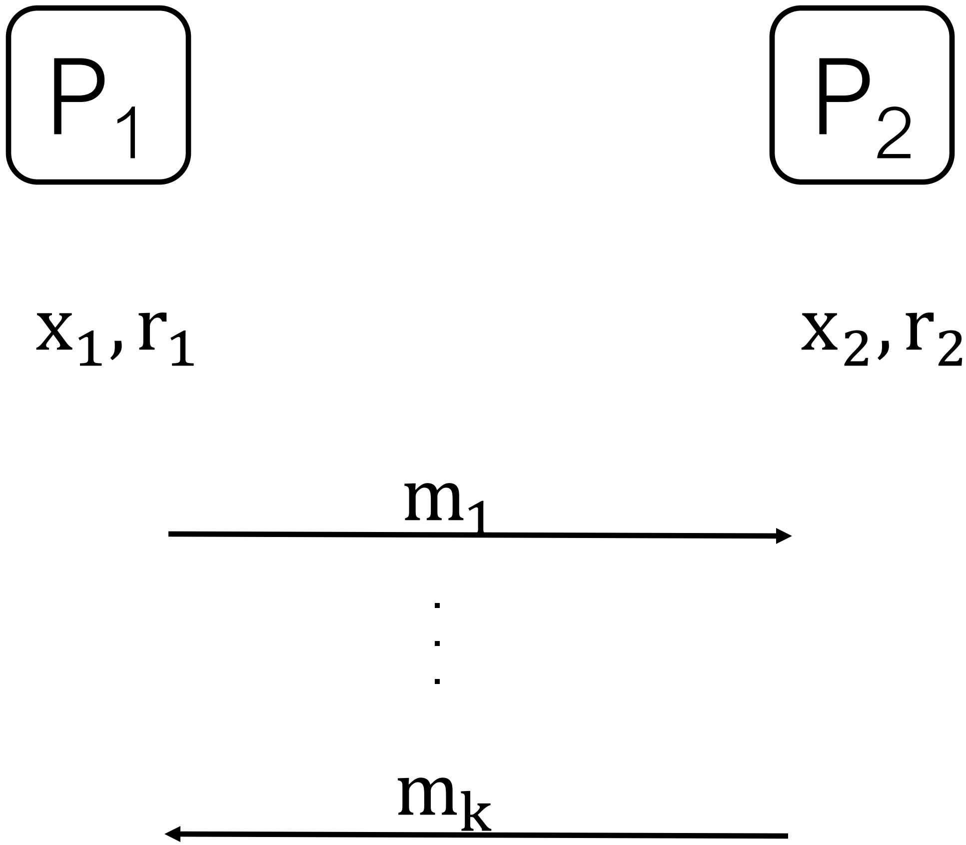
$$\left(\gamma^{\frac{N-p_1-q_1+1}{4}}\right)\left(\gamma^{\frac{-p_2-q_2}{4}}\right)\cdots\left(\gamma^{\frac{-p_n-q_n}{4}}\right)(\bmod N)$$

Test whether N is the product of two primes [BF97]

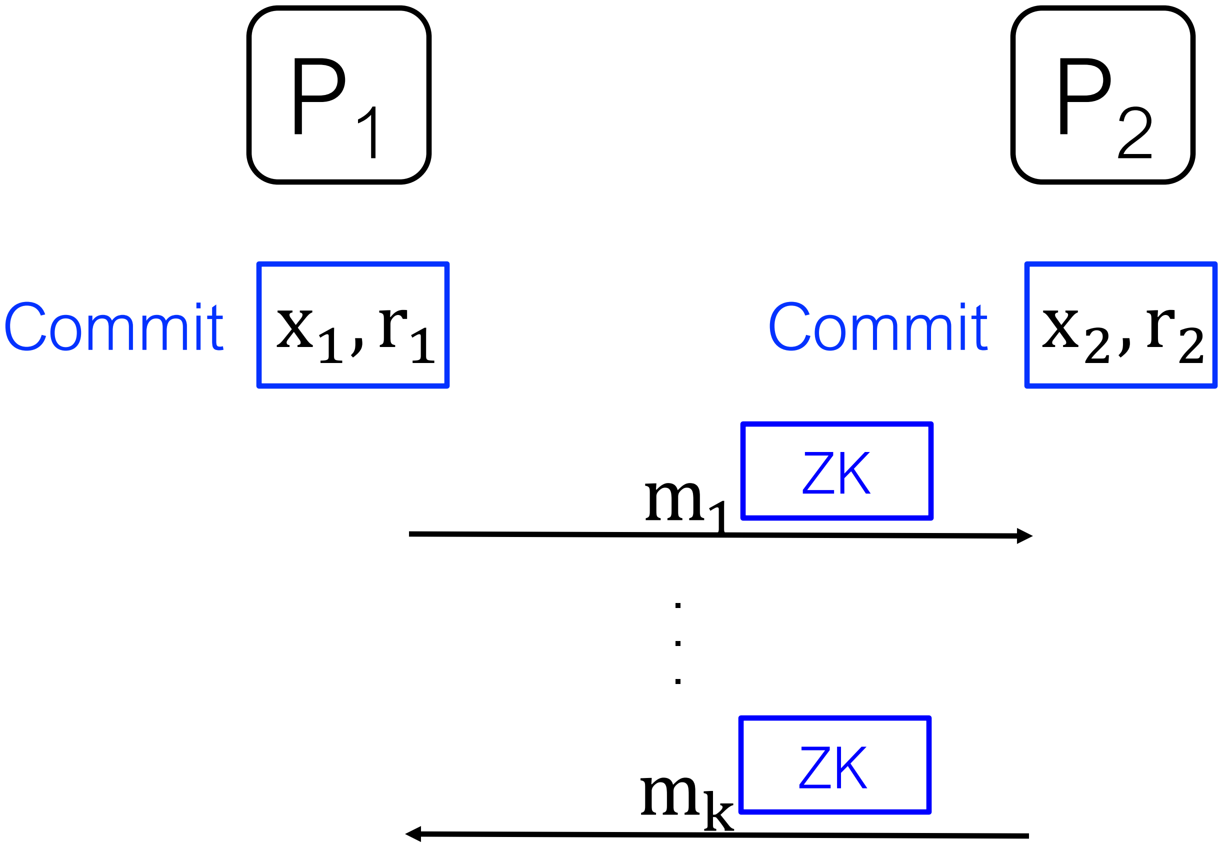
- Jacobi Test (Dist "Miller-Rabin" test)
- GCD Test

Step 2: Compile to full security

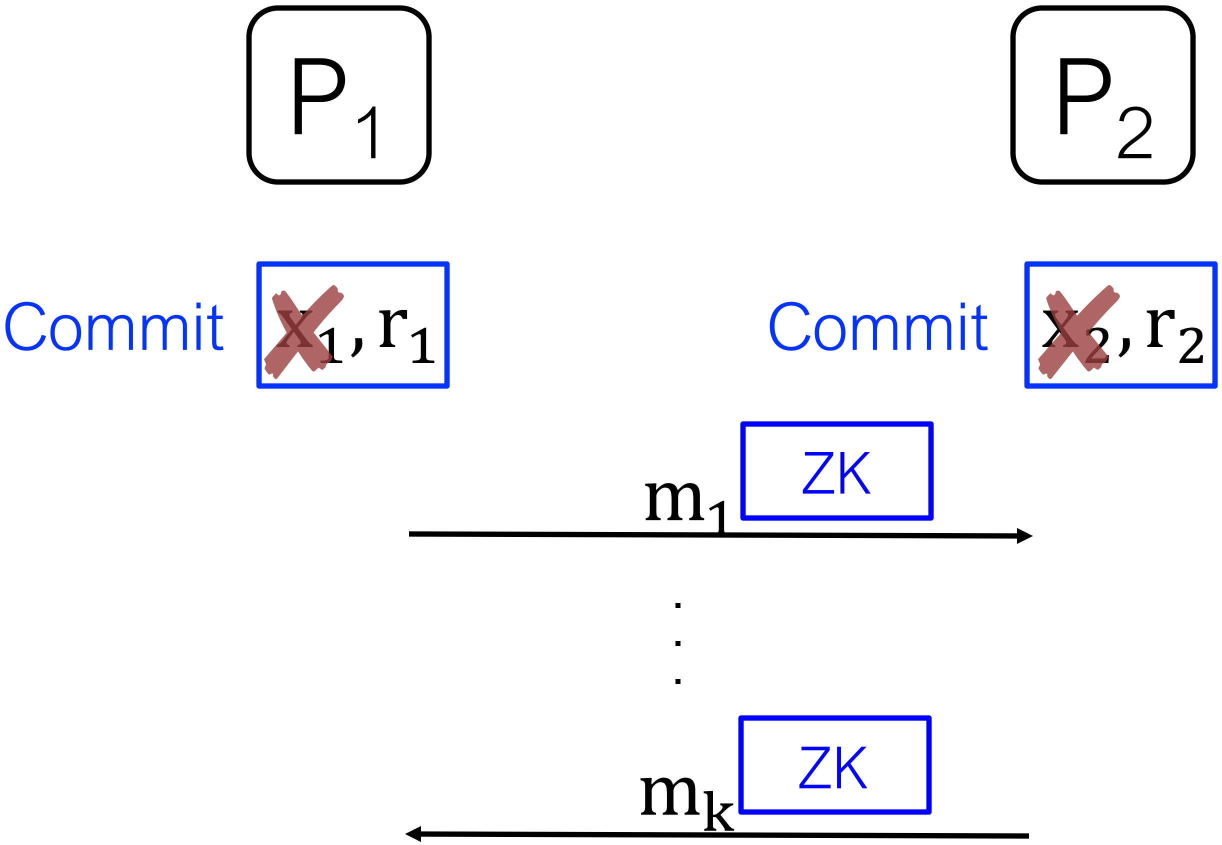
GMW Paradigm



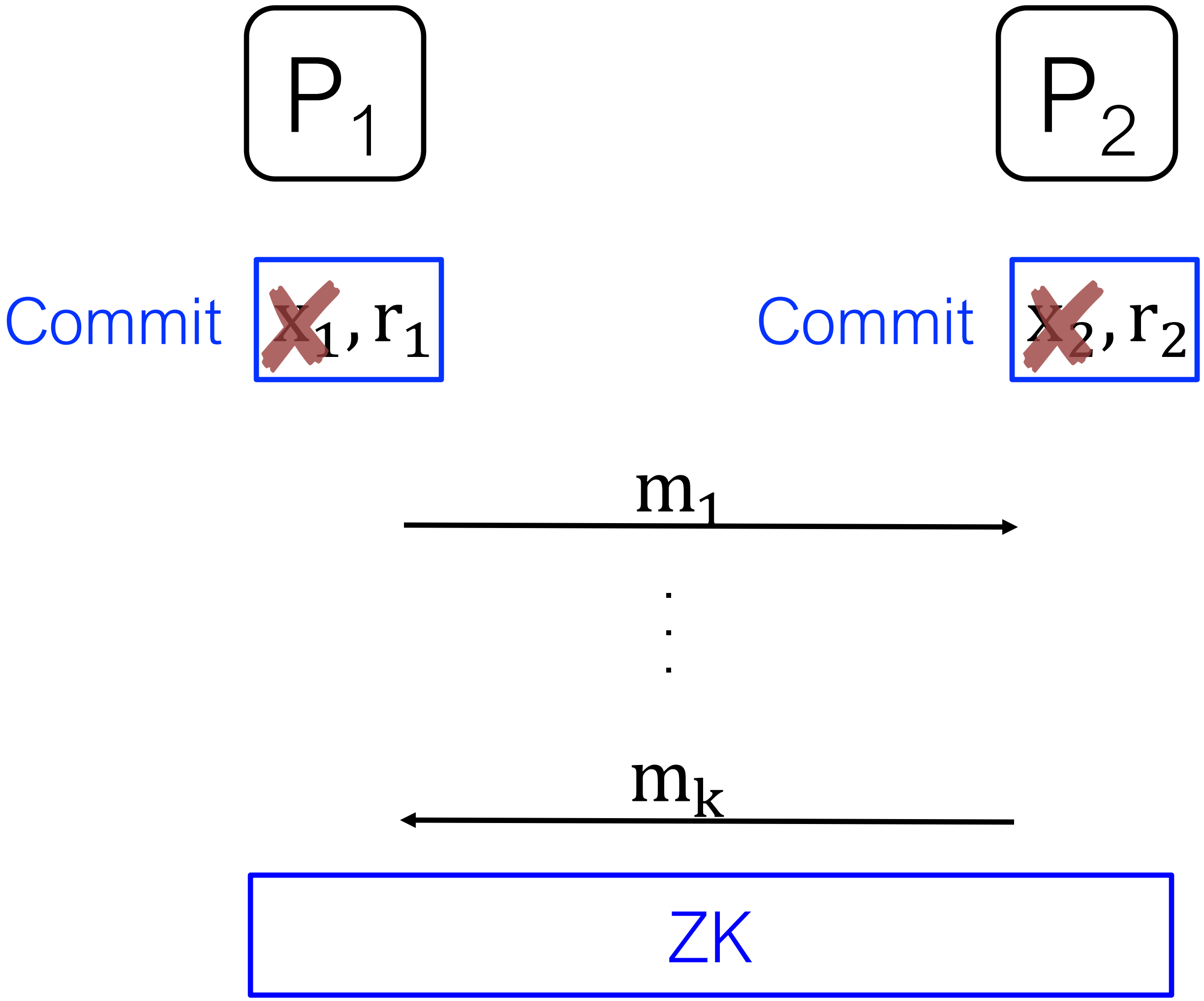
GMW Paradigm



Our Approach



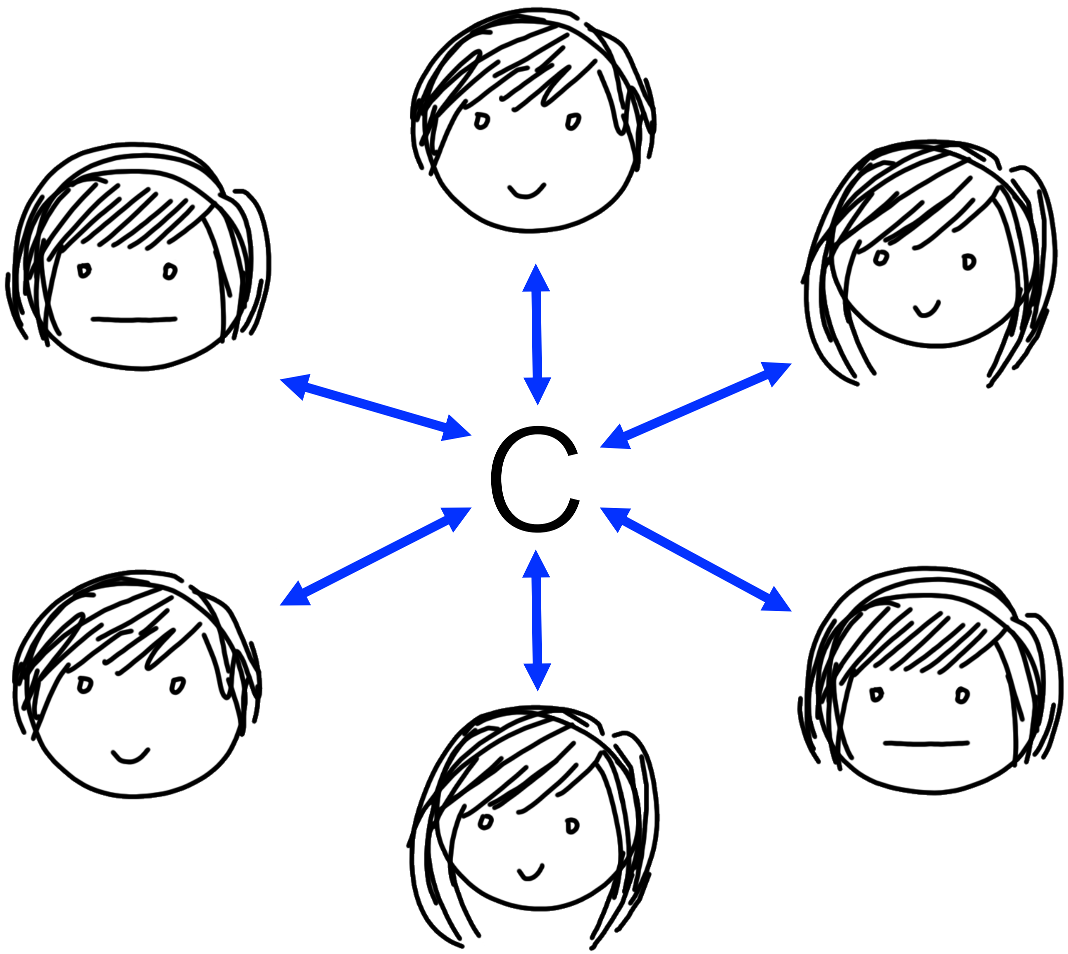
Our Approach



Our Protocol

Commitment	Commit to randomness
Key Setup	Generate threshold keys
Generate Candidates	Sample pre-sieved primes
Compute Products	Use TAHE to compute candidates
Biprimality test	Jacobi test
Certification	Zero-knowledge proof

Verifiable Coordinator



- Coordinator performs only public operations
- Sign every message
- Post message on bulletin board

Modular Proof (UC-security)

Generate Beaver triples

Passive Protocol
(with triples)

Certify triples

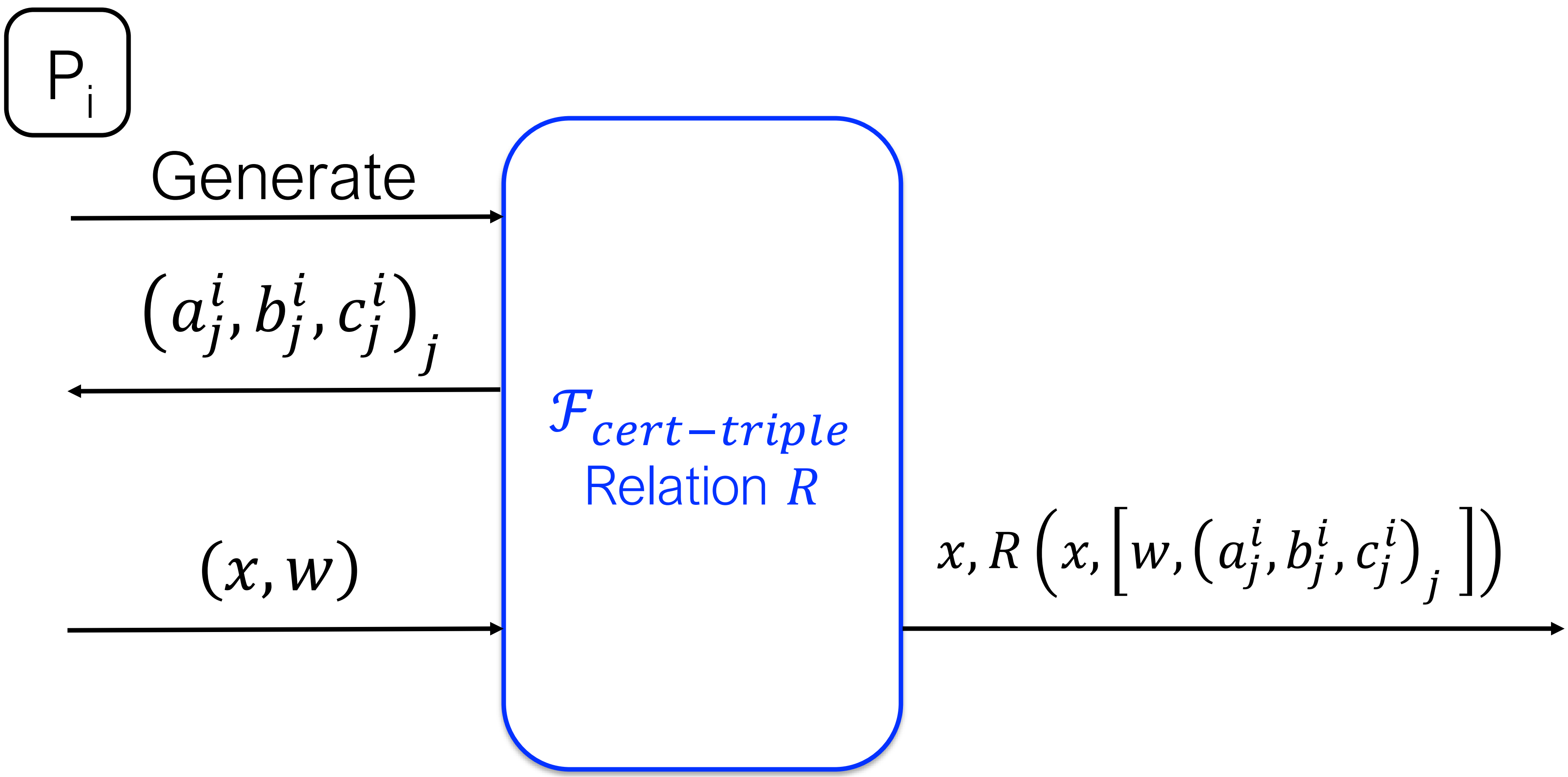
Modular Proof (UC-security)

$\mathcal{F}_{cert-triple}$

Passive Protocol
(with triples)

$\mathcal{F}_{cert-triple}$

Certified Beaver Triples Functionality



Realizing Certified Beaver Triples Functionality

Commit and Prove

$\mathcal{F}_{cp}$ (commit)

Semi-malicious security

Generate triples
using TAHE

$\mathcal{F}_{cp}$ (prove)

Which TAHE to choose?

Paillier?

- Circular choice

El Gamal?

- Inefficient decryption (discrete log)

LWE?

- Does not support all AHE operations

Ring-LWE → **more efficient, flexible**

- Supports AHE, better parameters, packing

ZK Constraints

- Triples generation - Operations in Ring $\mathbb{Z}_Q$ where $Q = p_1 \times p_2 \times \cdots \times p_n$ and each p_i is a 62-bit prime.
- Triples consumption - Linear operations modulo τ that is a product of (a different set of) primes
- Jacobi test - Operations modulo $\mathbb{Z}_N^*$ where N is the 2048-bit candidate modulus

What ZK Protocol to Use?

Needs:

- Memory efficient (2GB RAM for prover)
- Communication efficient (sublinear)
- Transparent

Our Approach

Ligero [AHIV17] + Sigma [Sho00]

The Proofs

Ligero

- Triples generation via Ring-LWE (Range Proofs)
- Triples consumption (modular arithmetic)

Sigma

- Jacobi test (knowledge of exponent)

Our Protocol

- Security w/ abort upto n-1 party corruptions and the coordinator by an active adversary
  - Verifiable coordinator
- Identifiable abort
- Public-verifiability [BDO14,BDD20]

Implementation

Setup

- Parties
  - AWS t3.small (2 vcpu, 2GB RAM)
- Coordinator
  - AWS r5dn24x.large (96 vcpu, 768 GB RAM)
- Ring LWE Parameter Selection
  - FHE Standardization (based on best attacks)
- PKI
  - Sign every message

Threshold AHE with Ring-LWE: Parameters

Parameter	**Notation**	**Value**
Security parameter	κ	128
Number of parties	N	1024
Gaussian parameter	σ	8
Degree/Packing Factor	n	2^{16}
Ciphertext Modulus Size	$\|Q\|$	1302 bits
Plaintext Modulus Size	$\|P\|$	558 bits
Maximum number of bits for τ	$\mathsf{max\_bits}(\tau)$	175 bits

Table 1: Ring-LWE choice of parameters.

Practical Considerations

- Bandwidth filtering
  - Run a throughput test and deny entry for parties with insufficient bandwidth
- Restart with kickout
  - If protocol aborts, identify and kickout failing party
  - What does n-1 security imply here?
- Distributed verification
- Benchmarking

Performance Metrics

Parties	Passive ($\mu \pm \sigma$ s)	Active ($\mu \pm \sigma$ s)	Registration (s)	# Runs (passive/active)
2	20.5 ± 0.9	594.3 ± 1.1	0.3	20 / 10
5	52.4 ± 3.7	785.9 ± 5.5	0.8	20 / 10
10	53.3 ± 1.9	788.5 ± 3.3	0.8	20 / 10
20	56.6 ± 2.3	797.7 ± 6.6	0.8	20 / 11
50	67.9 ± 6.6	808.8 ± 8.6	1.0	20 / 16
100	91.4 ± 5.3	832.3 ± 5.5	3.9	20 / 9
200	133.5 ± 12.2	884.4 ± 14.2	1.0	15 / 9
500	219.8 ± 5.9	970.0 ± 6.1	0.9	9 / 6
700	279.7 ± 4.9	1069.8 ± 9.8	61.4	5 / 5
1000	352.0 ± 14.0	1429.2 ± 0.0	1.6	3 / 1
2000	817.8 ± 0.0	2966.8 ± 0.0	2.0	1 / 1
4046	684.2 ± 0.0	4580.7 ± 0.0	158.7	1 / 1

Summary

- First scalable MPC with dishonest majority
- A practical implementation of the **generic GMW paradigm**
  - 4-8x computation overhead
  - <2x communication overhead
  - Bottleneck is coordinator spec
- Modular proof

Thank You