Diogenes: Lightweight Scalable RSA Modulus Generation with a Dishonest Majority

Carmit Hazay
Ligero & Bar-Ilan University

Megan Chen, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, abhi shelat, Muthu Venkitasubramaniam, Ruihan Wang

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)

Ethereum 2.0 = Proof of Stake!

Why? VDF Construction

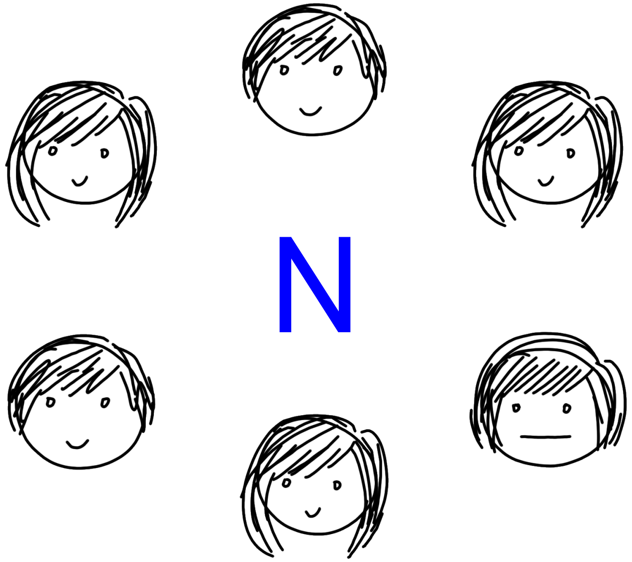
- 1996 - Rivest-Shamir-Wagner timelock puzzle

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

- 2018 - VDF constructions by Pietrzak, Wesolowski

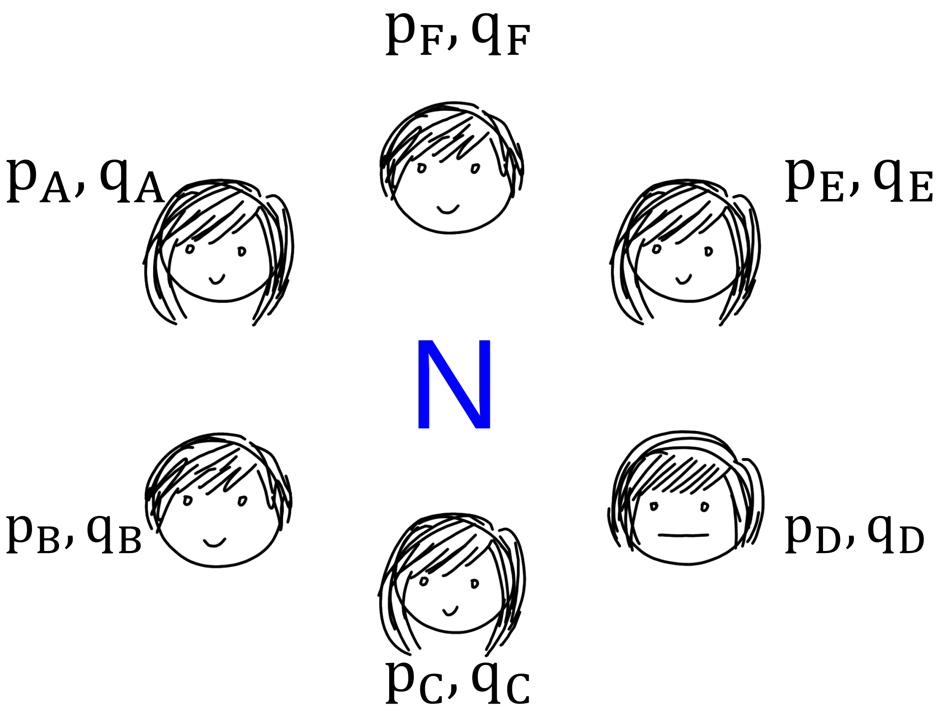
Goal

Parties interact to jointly sample a bi-prime modulus N



Goal

Parties interact to jointly sample a bi-prime modulus N



Goal

1024 parties

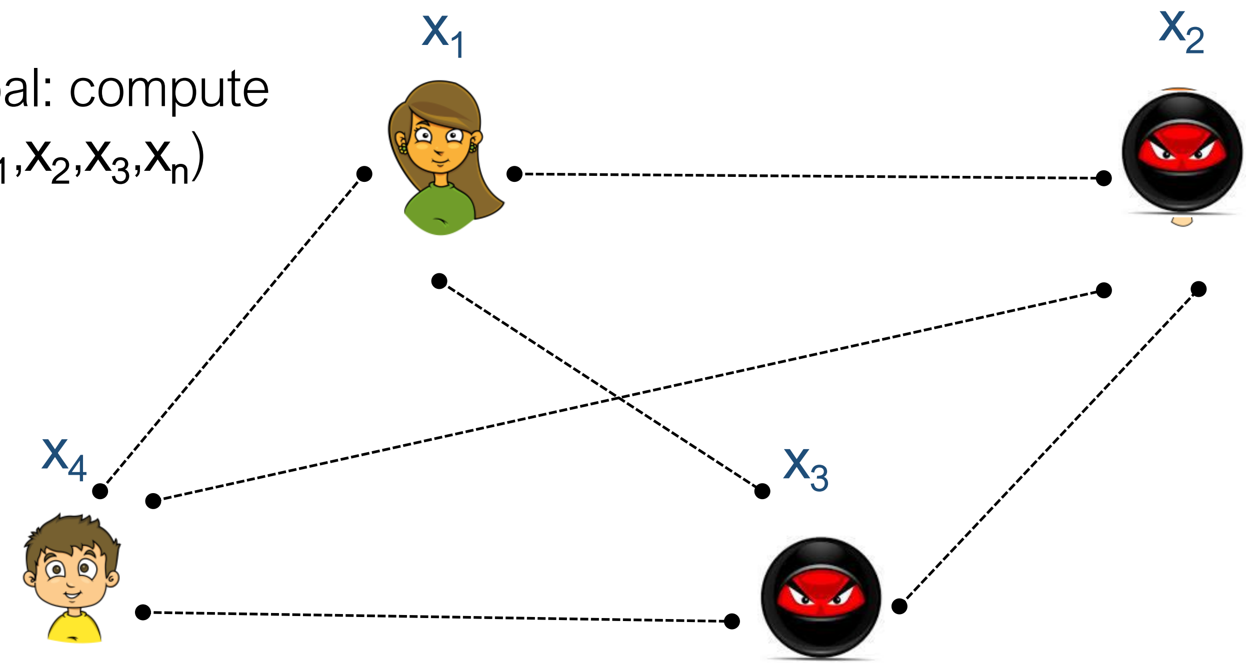
+

(n-1) active security

Need just 1 honest participant….

Secure Multi-Party Computation (MPC)

Goal: compute $f(x_1,x_2,x_3,x_n)$



MPC is useful for many applications

- Auctions with private bids
- Privacy-preserving data mining
- Private health records
- Cryptographic key protection
- Secure statistical analyses
- Smart city research – gender inequity
- Private blockchains
- …

Passive vs. Active

Honest majority vs. Dishonest majority

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	$t < n$

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
	[HMRTN12]	**Active**	**n**	**t < n**
	[FLOP18]	Active	2	t = 1
	[CCD+20]	**Active**	**n**	**t < n**

Previous Works: Implementations

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
Passive impl. only	**[HMRTN12]**	**Active**	**n**	**t < n**
Passive impl. only	**[FLOP18]**	**Active**	**2**	**t = 1**
	[CCD+20]	Active	n	t < n

State-of-the-Art

	[FLOP18]
RSA Modulus Size	2048 bits
Implementation	Passive
Num Parties	2
Party Spec	8 GB RAM 8 cores CPU
Bandwidth	40 Gbps
Comm. (Per-Party)	>1.9 GB
Time	35 sec (8 threads)

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	
Implementation	Passive	
Num Parties	2	
Party Spec	8 GB RAM 8 cores CPU	
Bandwidth	40 Gbps	
Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	2048 bits
Implementation	Passive	
Num Parties	2	
Party Spec	8 GB RAM 8 cores CPU	
Bandwidth	40 Gbps	
Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	**2048 bits**
Implementation	Passive	**Active (Id-A)**
Num Parties	2	
Party Spec	8 GB RAM 8 cores CPU	
Bandwidth	40 Gbps	
Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	2048 bits
Implementation	Passive	Active (Id-A)
Num Parties	2	1024
Party Spec	8 GB RAM 8 cores CPU	
Bandwidth	40 Gbps	
Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	2048 bits
Implementation	Passive	Active (Id-A)
Num Parties	2	1024
Party Spec	8 GB RAM 8 cores CPU	2 GB RAM single-core CPU
Bandwidth	40 Gbps	
Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	2048 bits
Implementation	Passive	Active (Id-A)
Num Parties	2	1024
Party Spec	8 GB RAM 8 cores CPU	2 GB RAM single-core CPU
Bandwidth	40 Gbps	1 Mbps 100 ms latency
Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	**2048 bits**
Implementation	Passive	**Active (Id-A)**
Num Parties	2	**1024**
Party Spec	8 GB RAM 8 cores CPU	**2 GB RAM single-core CPU**
Bandwidth	40 Gbps	**1 Mbps 100 ms latency**
Comm. (Per-Party)	>1.9 GB	**< 100 MB**
Time	35 sec (8 threads)	

State-of-the-Art vs. Our Target

	[FLOP18]	Our Challenge
RSA Modulus Size	2048 bits	2048 bits
Implementation	Passive	Active (Id-A)
Num Parties	2	1024
Party Spec	8 GB RAM 8 cores CPU	2 GB RAM single-core CPU
Bandwidth	40 Gbps	1 Mbps 100 ms latency
Comm. (Per-Party)	>1.9 GB	< 100 MB
Time	35 sec (8 threads)	< 20 mins

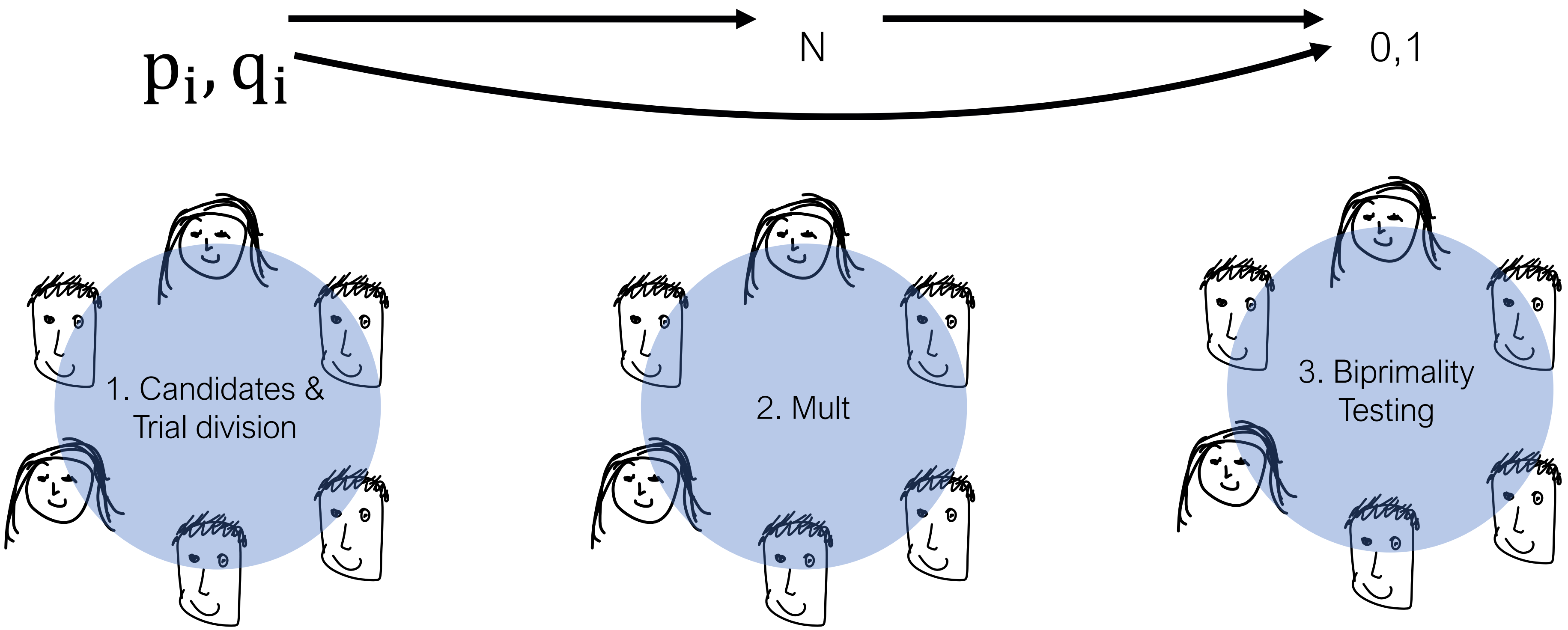
Protocol Blueprint

Step 1: Design protocol secure against passive adversary

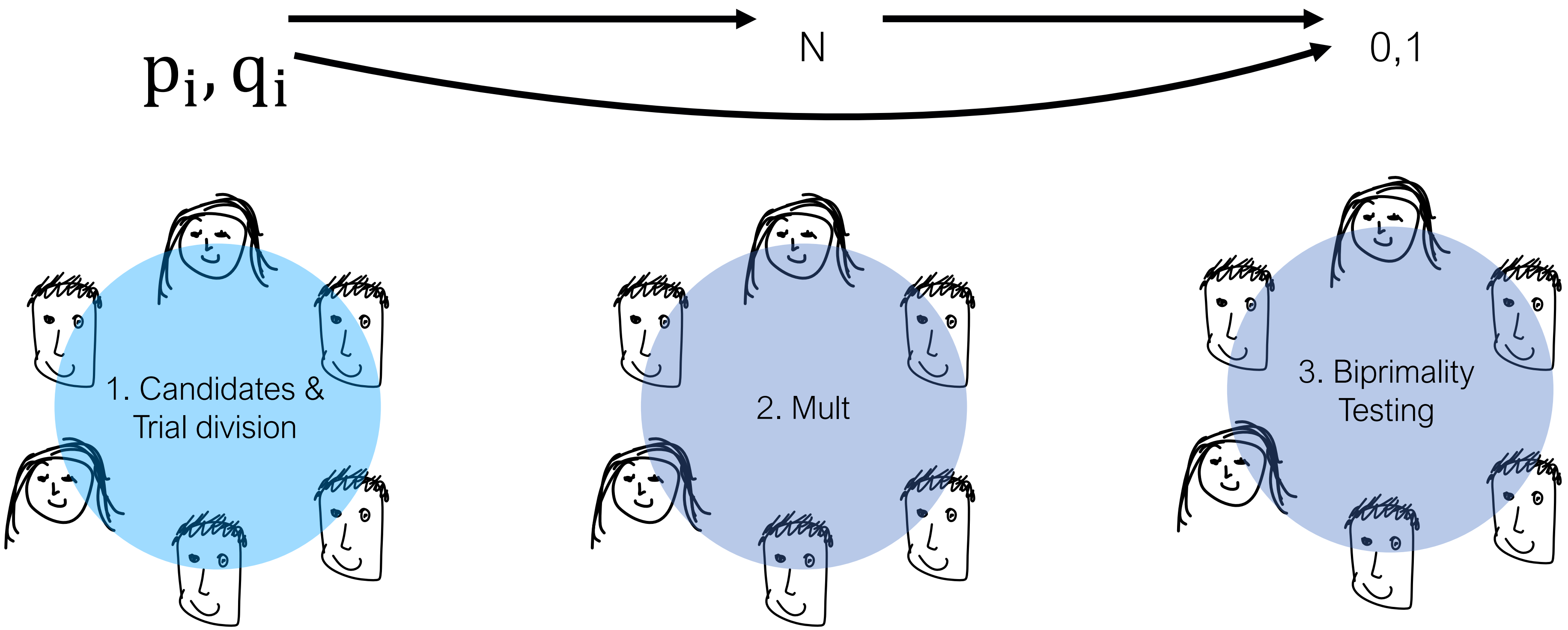
Step 2: Compile to security against active adversary

Step 1: Scalable Passive Protocol

Boneh-Franklin Framework[BF97]

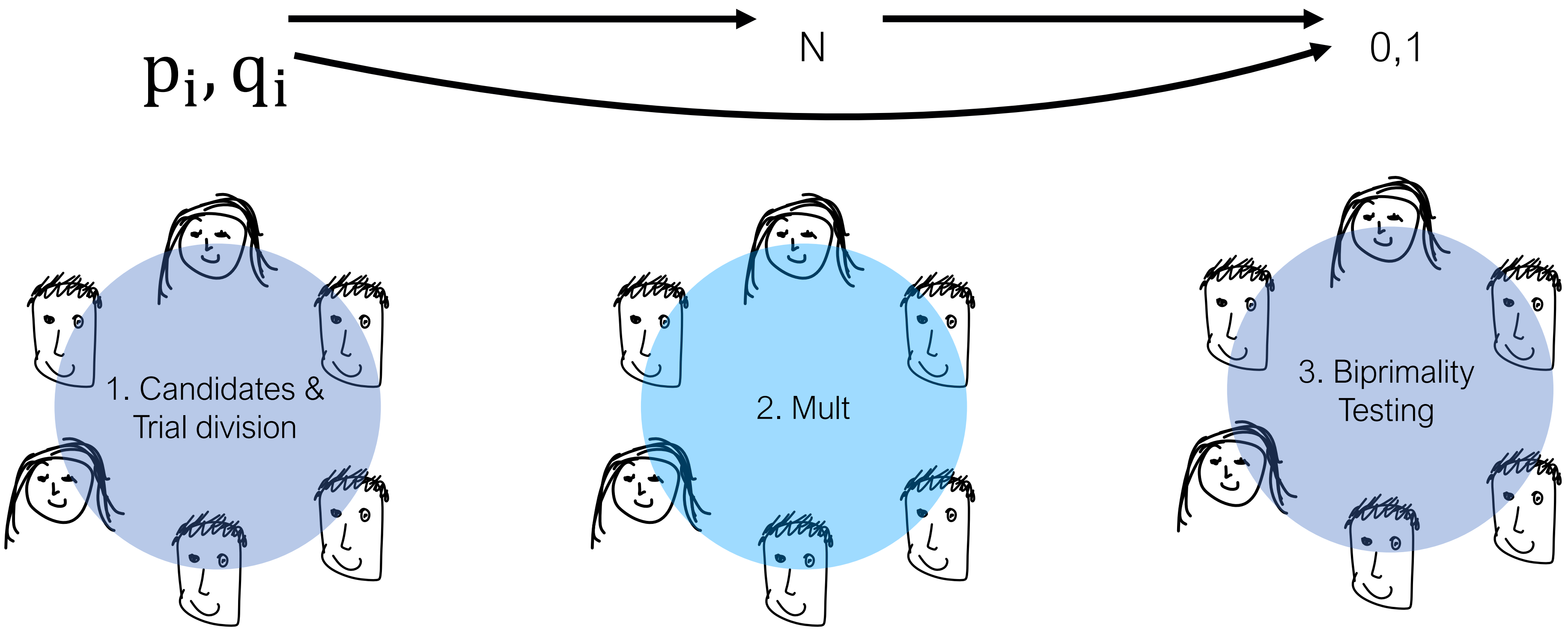


Boneh-Franklin Framework [BF97]



Parties choose p_i, q_i randomly

Boneh-Franklin Framework[BF97]

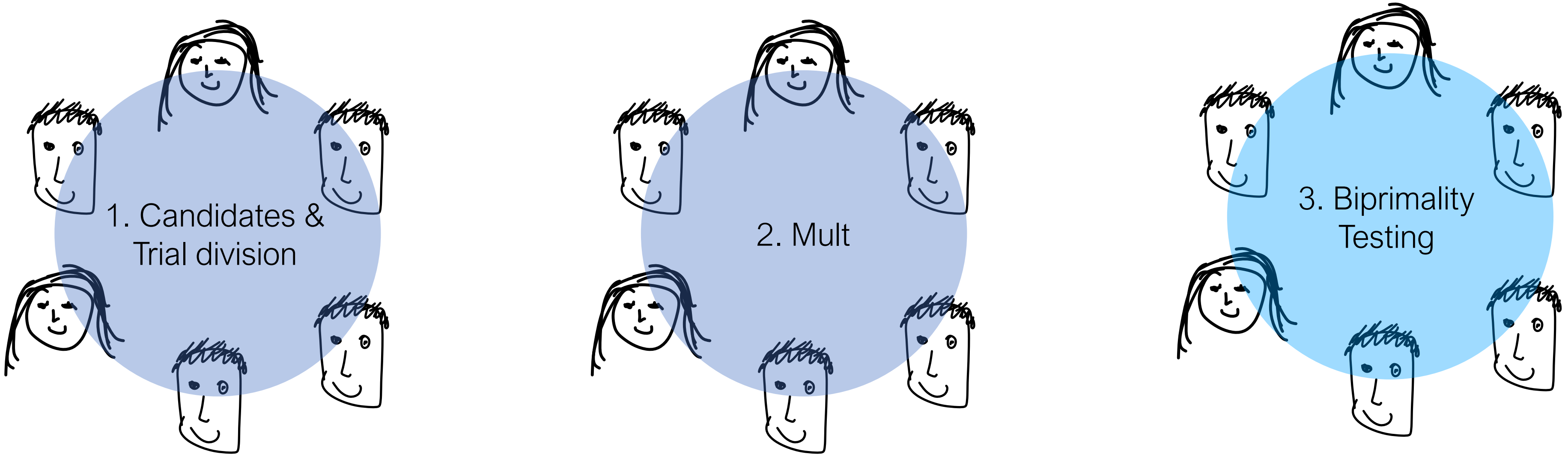


Parties choose p_i, q_i randomly

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

Boneh-Franklin Framework[BF97]

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

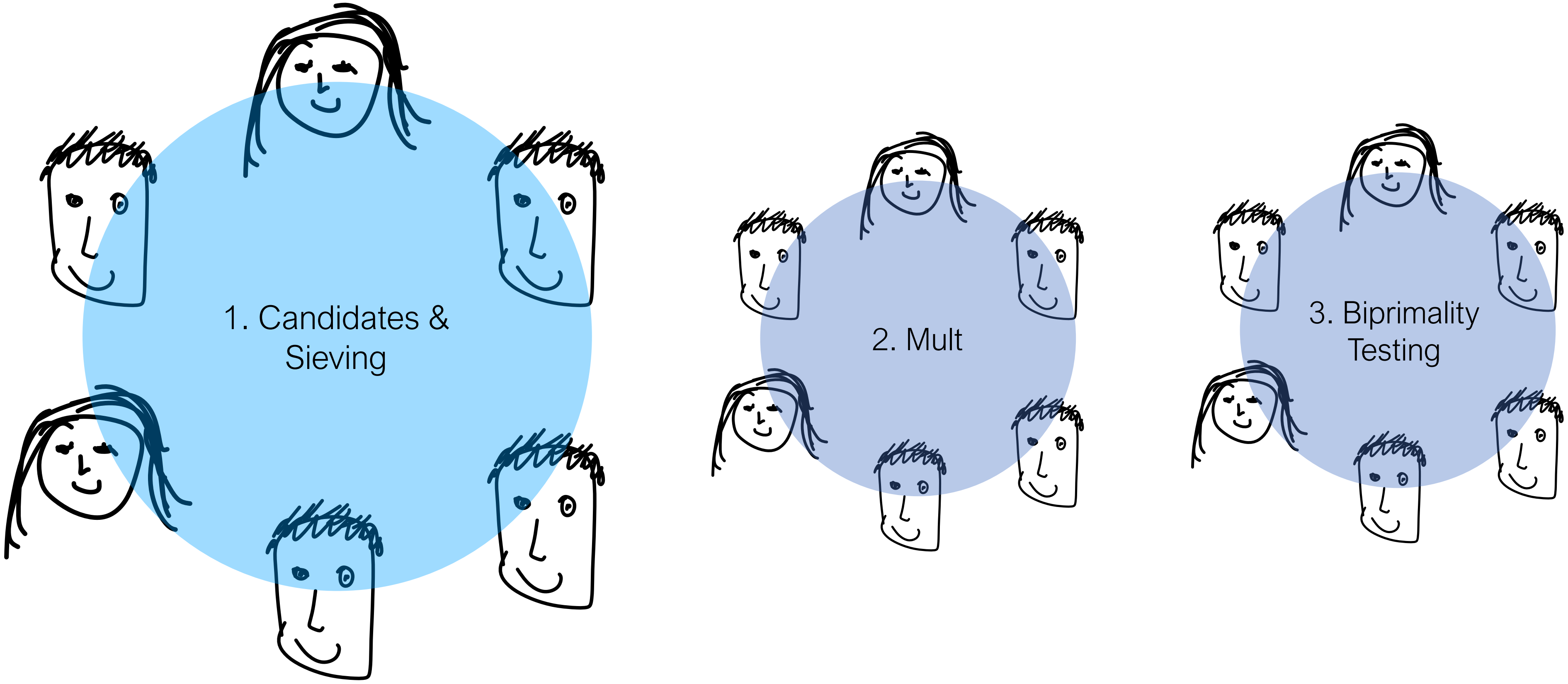


Parties choose p_i, q_i randomly

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

Is N the product of two primes?

Start with Sieving Trick



Candidate Naïve Sampling

A = randomly sampling a 1024-bit prime

B = number is odd

$$\Pr[A|B] \approx \left(\frac{1}{500}\right)$$

$$\Pr[\text{sample biprime}|B] \approx \left(\frac{1}{500}\right)^2$$

Need 250k samples in expectation 👎

Candidate Trial Division [Bru50]

A = randomly sampling a 1024-bit prime

B = sieve up to 863, the 150th prime

$$\Pr[A|B] \approx \left(\frac{1}{60}\right)$$

$$\Pr[\text{sample biprime}|B] \approx \left(\frac{1}{60}\right)^2$$

Need 3600 samples in expectation

Candidate Trial Division: Prior Works

1. Construct p and q

2. Distributed sieving

3. If both pass, multiply

HMRTN12 → El Gamal

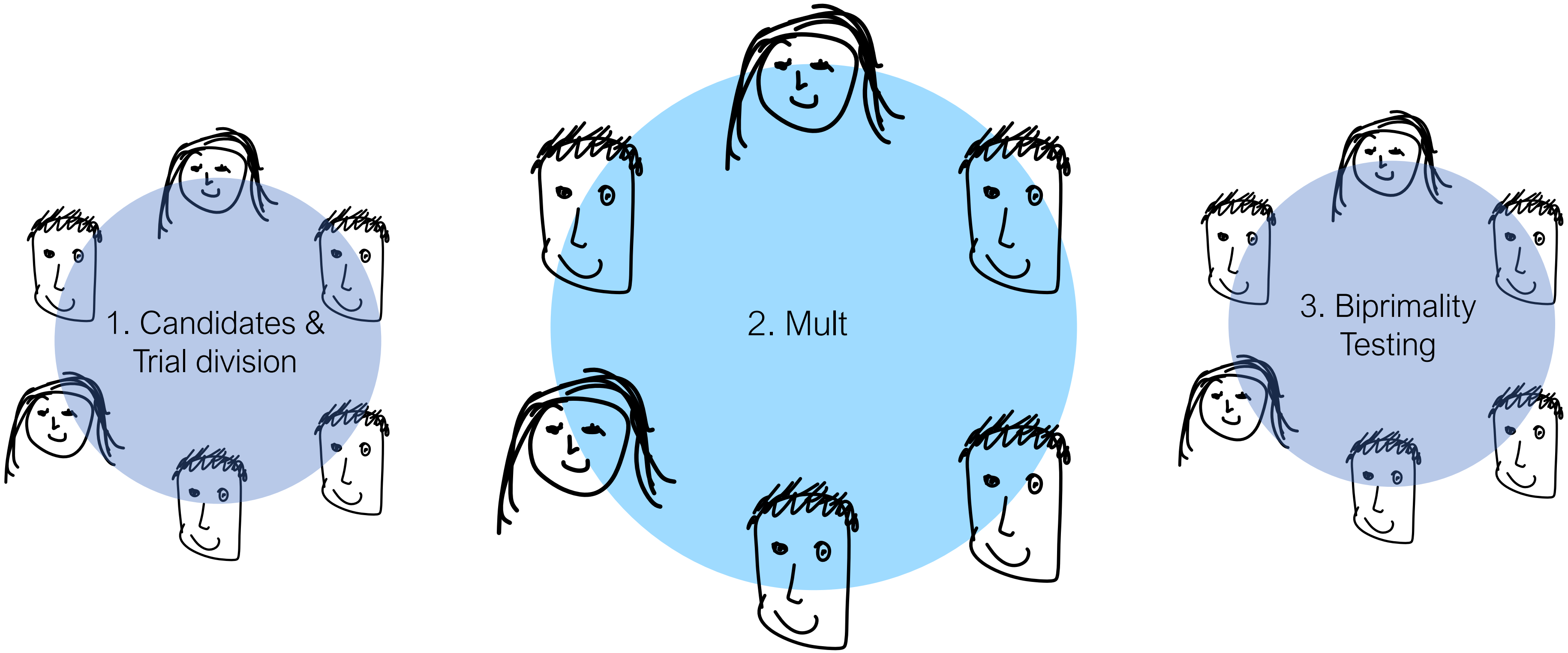
FLOP18 → 1-out-of-k OT

Pairwise communication channels

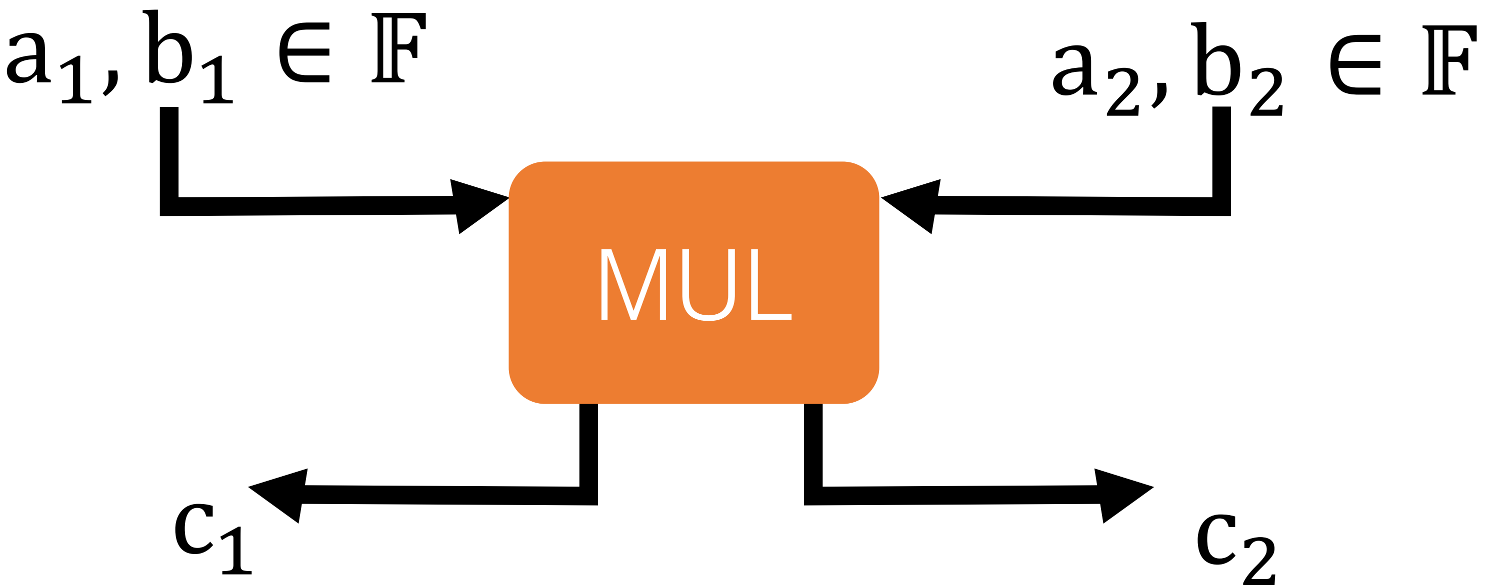
Our Approach

Sieve first,
construct later[CCD+20]

Secure Multiplication



Secure Multiplication



$$c_1 + c_2 = \left(\sum a_i\right) \cdot \left(\sum b_i\right)$$

Our Approach: Threshold AHE

- Distributed key generation

  Public key: $\mathbf{PK}$ Secret keys: $\mathbf{sk_1, \ldots, sk_n}$

- Encryption

  $$\mathsf{Enc}_{\mathbf{PK}}(\mathbf{m})$$

- Distributed decryption

  $$\mathbf{m} = \mathsf{Dec}_{\mathbf{sk_1}}(\mathbf{c}) + \cdots + \mathsf{Dec}_{\mathbf{sk_n}}(\mathbf{c})$$

Our Approach: Threshold AHE

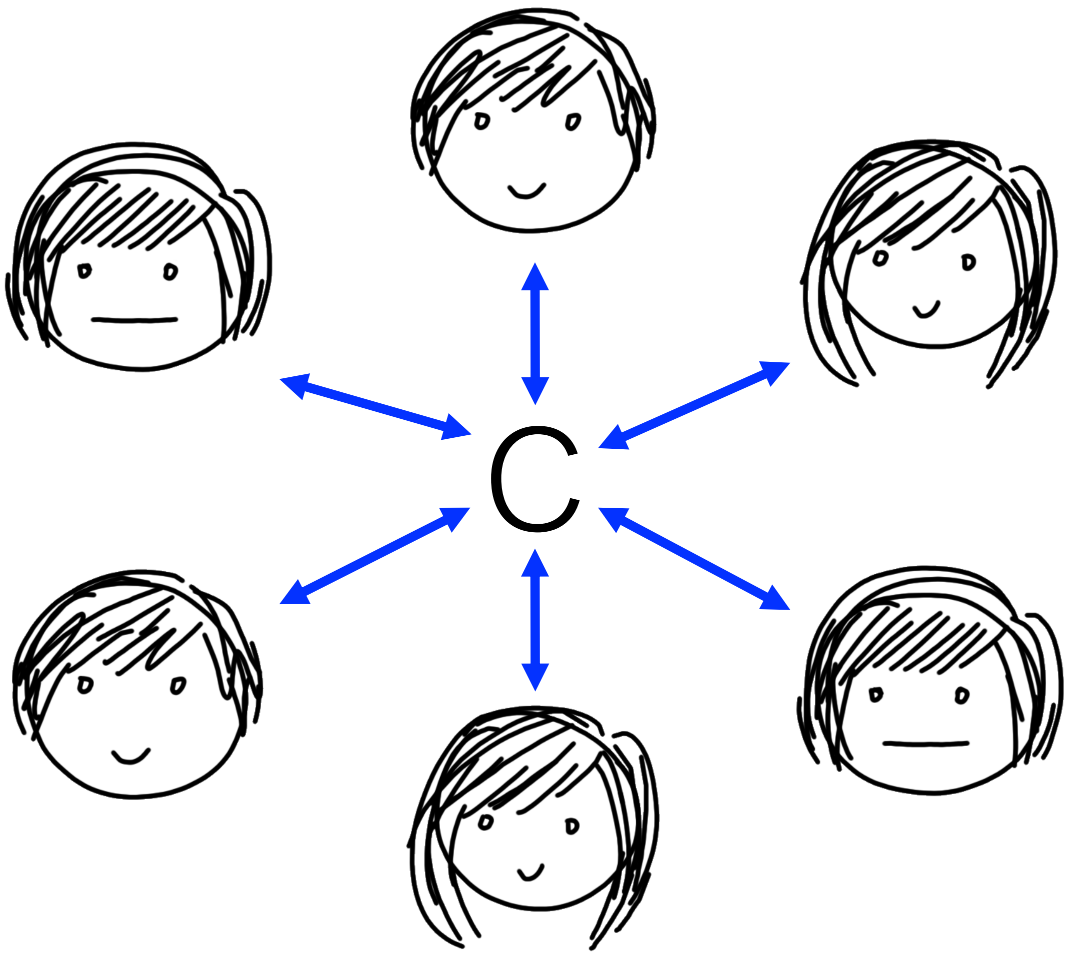
- Addition under encryption

$$\mathrm{Enc}_{\mathrm{PK}}(\mathrm{m}_1) + \mathrm{Enc}_{\mathrm{PK}}(\mathrm{m}_2) = \mathrm{Enc}_{\mathrm{PK}}(\mathrm{m}_1 + \mathrm{m}_2)$$

- Scalar multiplication under encryption

$$\mathrm{a} \cdot \mathrm{Enc}_{\mathrm{PK}}(\mathrm{m}) = \mathrm{Enc}_{\mathrm{PK}}(\mathrm{a} \cdot \mathrm{m})$$

Our Approach: Untrusted Coordinator



Performs only public operations

Our Approach: Threshold AHE

	P_i	C
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

Our Approach: Threshold AHE

	P_i	C
	PK	
Key Generation	sk_i	
Parties' secret shares	p_i, q_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	$p \cdot q$	

State-of-the-Art TAHE

Paillier?

- Circular choice

El Gamal?

- Inefficient decryption (discrete log)

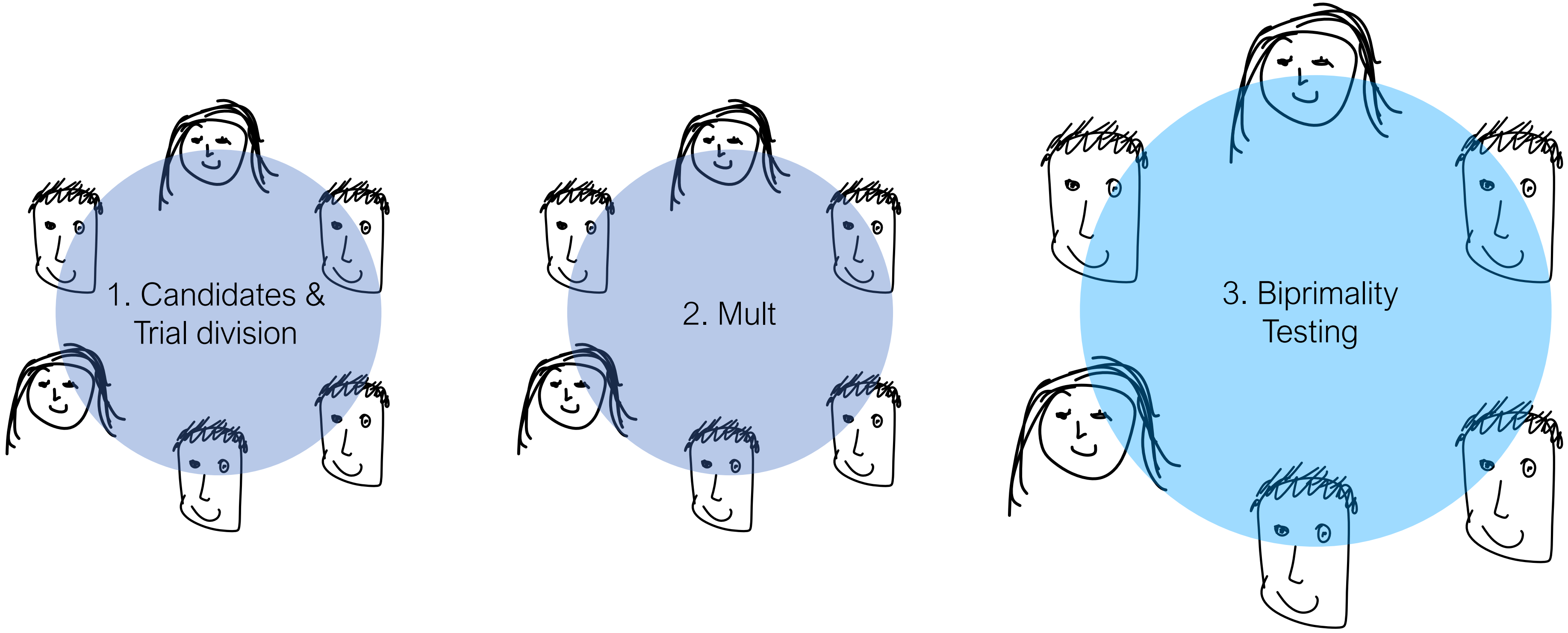
LWE?

- Does not support all AHE operations

Ring-LWE → more efficient, flexible

- Supports AHE, better parameters, packing

[BF97]'s Distributed Biprimality Test



- Test whether N is the product of two primes
- Don't leak p or q
- Extension of Miller-Rabin primality test [Rabin80]
- Probabilistic - need to repeat s times

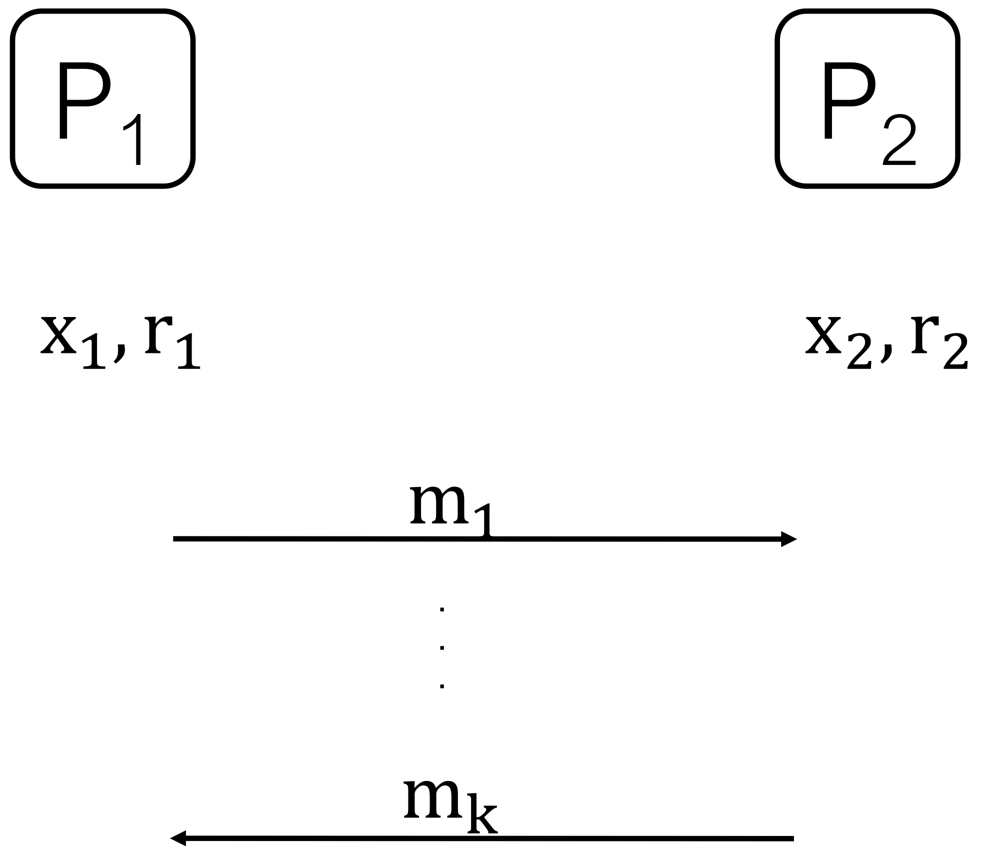
Step 2: Security against Active Adversaries

GMW Paradigm

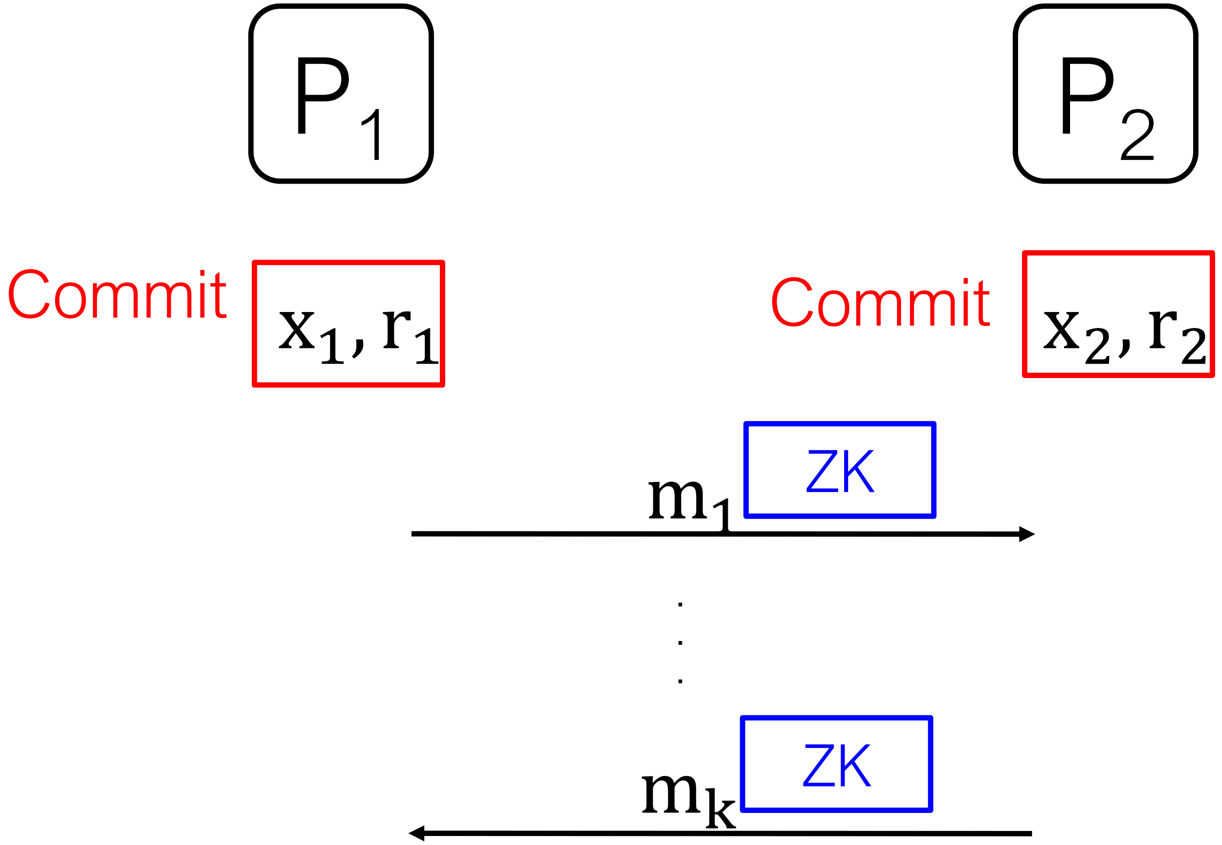
aka Zero-Knowledge Proofs

aka ”I will prove I did everything honestly!”

GMW Paradigm: Passive Protocol



GMW Paradigm: Active Protocol



GMW Paradigm: Our Compiler

P_1 P_2

Commit $\boxed{\cancel{x_1}, r_1}$ Commit $\boxed{\cancel{x_2}, r_2}$

$\xrightarrow{m_1}$

$\vdots$

$\xleftarrow{m_k}$

$\boxed{\text{ZK}}$

What ZK Protocol to Use?

Need:

- Fast prover
- Prover runs on a 1 CPU 2 GB RAM machine
- Prove operations over

  Lattice Operations over Ring $Z_Q[x]/x^n+1$ where

  $$Z_Q = Z_{p1} \times \ldots \times Z_{p21}$$

  Modulus generation - operations in

  $$F_2, F_3, F_5, \ldots, F_{823}$$

  Jacobi test – Exponentiation operations over

  $$Z^*_N \text{ (2048-bit number)}$$

What ZK Protocol to Use?

Need:

- Fast prover
- Prover runs on a 1 CPU 2 GB RAM machine
- Prove operations over

Lattice Operations over Ring $Z_Q [x]/x^n+1$ where

$Z_Q = Z_{p1} \times \ldots \times Z_{p21}$

Modulus generation - operations in

$F_2, F_3, F_5, \ldots, F_{823}$

LIGERO

Jacobi test – Exponentiation operations over

Z^*_N (2048-bit number)

Tailor-made
Sigma Protocol

What ZK Protocol to Use?

Need:

- Fast prover
- Prover runs on a 1 CPU 2 GB RAM machine
- Prove operations over

Lattice Operations over Ring $Z_Q[x]/x^n+1$ where

$Z_Q = Z_{p1} \times \ldots \times Z_{p21}$

Modulus generation - operations in

$F_2, F_3, F_5, \ldots, F_{823}$

LIGERO

Jacobi test – Exponentiation operations over

Z^*_N (2048-bit number)

Tailor-made Sigma Protocol

Summary: Our Protocol

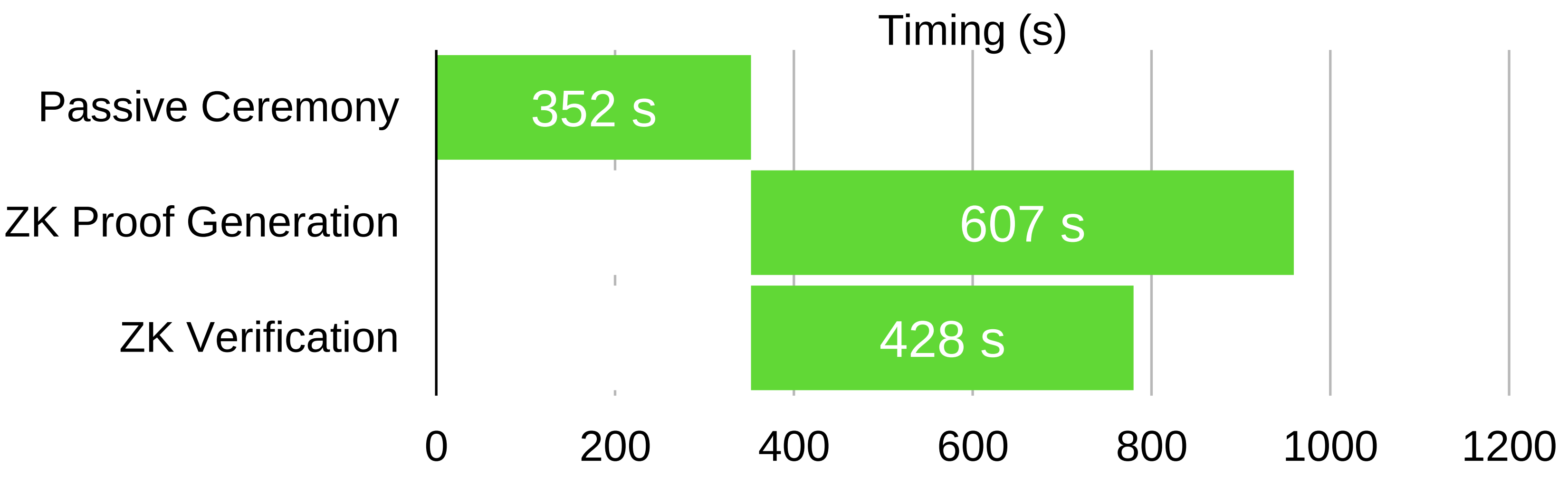
Key Setup	Generate threshold keys
Generate Candidates	Sample pre-approved primes
Compute Products	Use TAHE to compute candidates
Biprimality test	BF biprimality test
Certification	Ligero ZK + Sigma

Performance Metrics: 10,000 Parties (Passive)

Parties	Coordinator	Total time (s)
64	`m5.metal`	61.8
128	”	74.3
256	”	104.8
512	”	137.6
1024	”	205.8
1500	`r5.24xlarge`	266.8
2000	”	416.5
4500	”	1282.6
10000	”	2111.8

Performance Metrics: 1024 Parties (Active)

Stage	Timing Per Step	Cumulative Time
Passive Protocol	5m 52s	
ZK Proof Generation	11m 07s	
ZK Verification	7m 08s	17m 06s



Conclusion

	[FLOP18]	Our Goal	
Modulus size	2048 bits	**2048 bits**	
Implementation	Passive	**Active**	✓
Num Parties	2	**1024**	✓
Party Spec	8 GB RAM 8 cores CPU	**2 GB RAM single-core CPU**	✓
Network speed	40 Gbps	**1 Mbps 100 ms latency**	✓
Comm. (Per-Party)	>1.9 GB	**< ~~100 MB~~** 200 MB	✗
Time	35 sec (8 thread)	**< 20 mins**	✓

https://github.com/ligeroinc/LigeroRSA

Thank You