Diogenes: Lightweight Scalable RSA Modulus Generation with a Dishonest Majority

Megan Chen

Ligero and Northeastern University

LIGERO

Carmit Hazay, 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

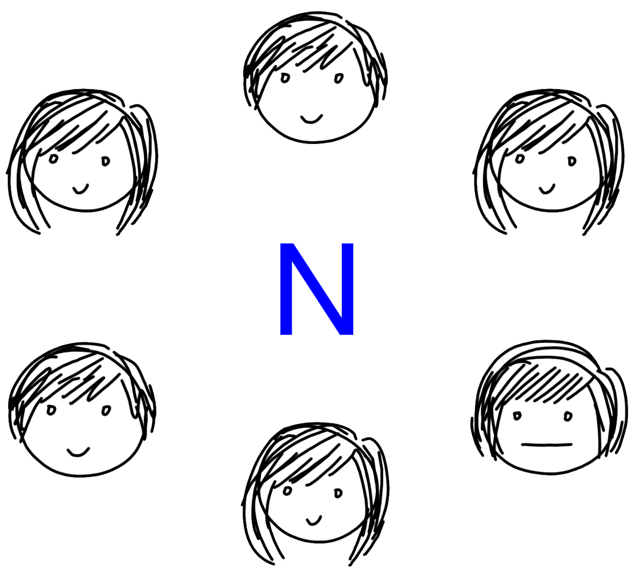
- 1996 - Rivest-Shamir-Wagner timelock puzzle

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

- 2018 - VDF constructions by Pietrzak, Wesolowski

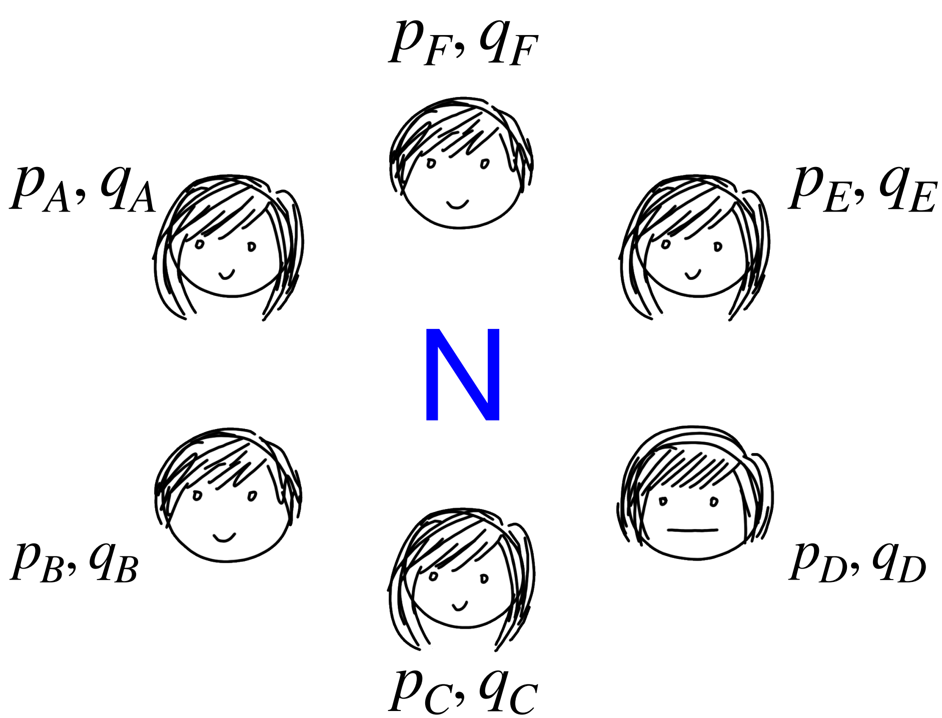
Goal

Parties interact to jointly sample a bi-prime modulus N



Goal

Each party has secret shares of N's factors: p, q



Goal

1024 parties

+

(n-1) active security

Need just 1 honest participant….

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 in Our Setting

Active + n-Party + Dishonest Majority

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: 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	**[HMRT12]**	**Active**	**n**	**t < n**
Passive impl. only	**[FLOP18]**	**Active**	**2**	**t = 1**
	[CCD+20]	Active	n	t < n

Previous Works: State of the Art

	[FLOP18]	
RSA Modulus Size	2048 bits	
Implementation	Passive	
Num Parties	2	
Party Spec	8 GB RAM 8 cores CPU	Let’s do better!
Bandwidth	40 Gbps	
Online Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 thread)	

Previous Works: State of the Art

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

Previous Works: State of the Art

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

Previous Works: State of the Art

	[FLOP18]	**Our Goal**
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	
Online Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 thread)	

Previous Works: State of the Art

	[FLOP18]	**Our Goal**
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	
Online Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 thread)	

Previous Works: State of the Art

	[FLOP18]	**Our Goal**
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	
Online Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 thread)	

Previous Works: State of the Art

	[FLOP18]	**Our Goal**
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**
Online Comm. (Per-Party)	>1.9 GB	
Time	35 sec (8 thread)	

Previous Works: State of the Art

	[FLOP18]	**Our Goal**
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**
Online Comm. (Per-Party)	>1.9 GB	**< 100 MB**
Time	35 sec (8 thread)	

Previous Works: State of the Art

	[FLOP18]	**Our Goal**
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**
Online Comm. (Per-Party)	>1.9 GB	**< 100 MB**
Time	35 sec (8 thread)	**< 20 mins**

Protocol Blueprint

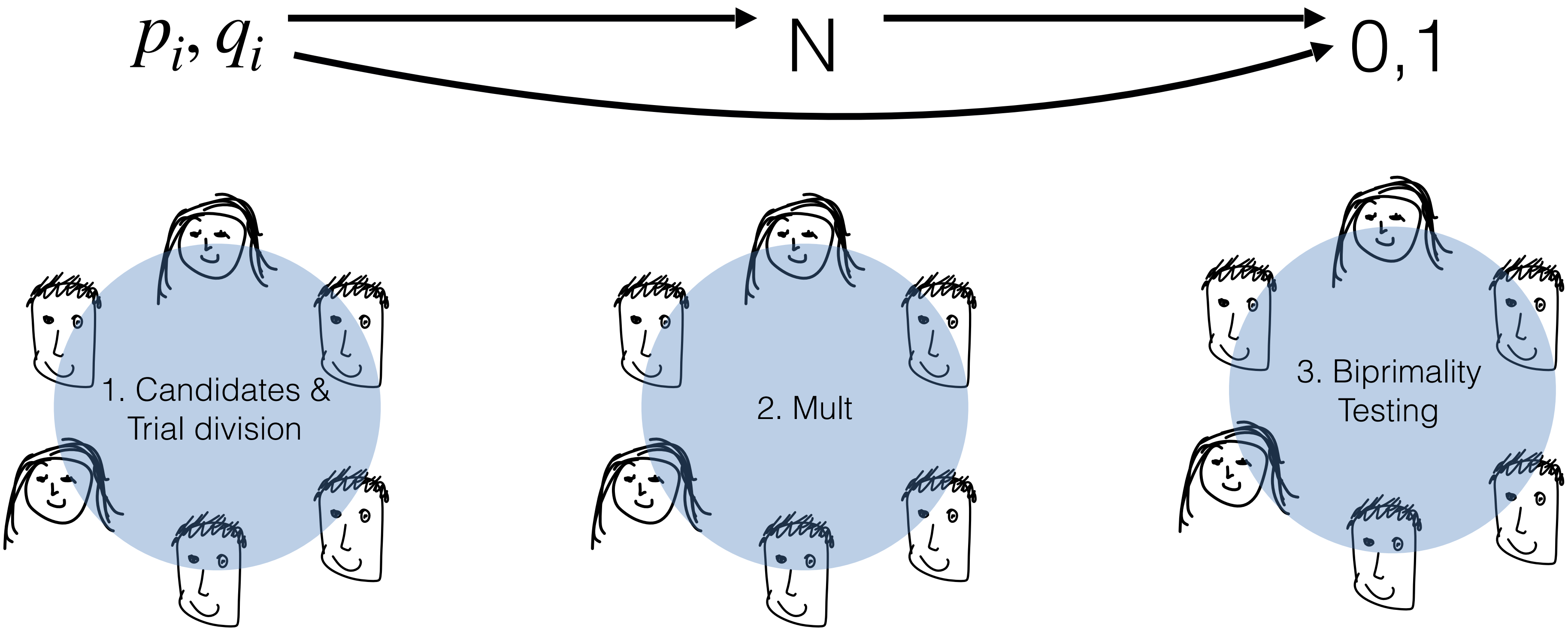
Step 1: Design protocol secure against passive adversary

Step 2: Compile to security against active adversary

Step 1: scalable passive protocol

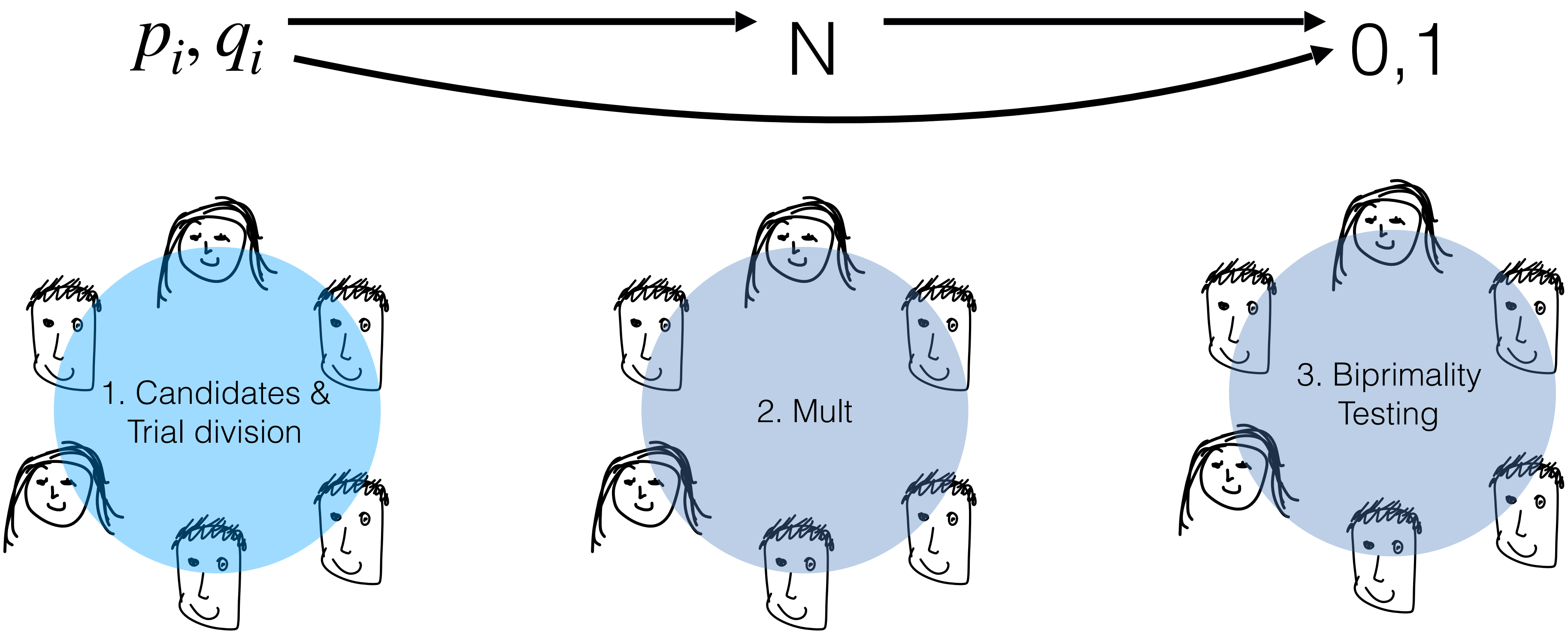
Boneh-Franklin Framework

[BF97]



Boneh-Franklin Framework

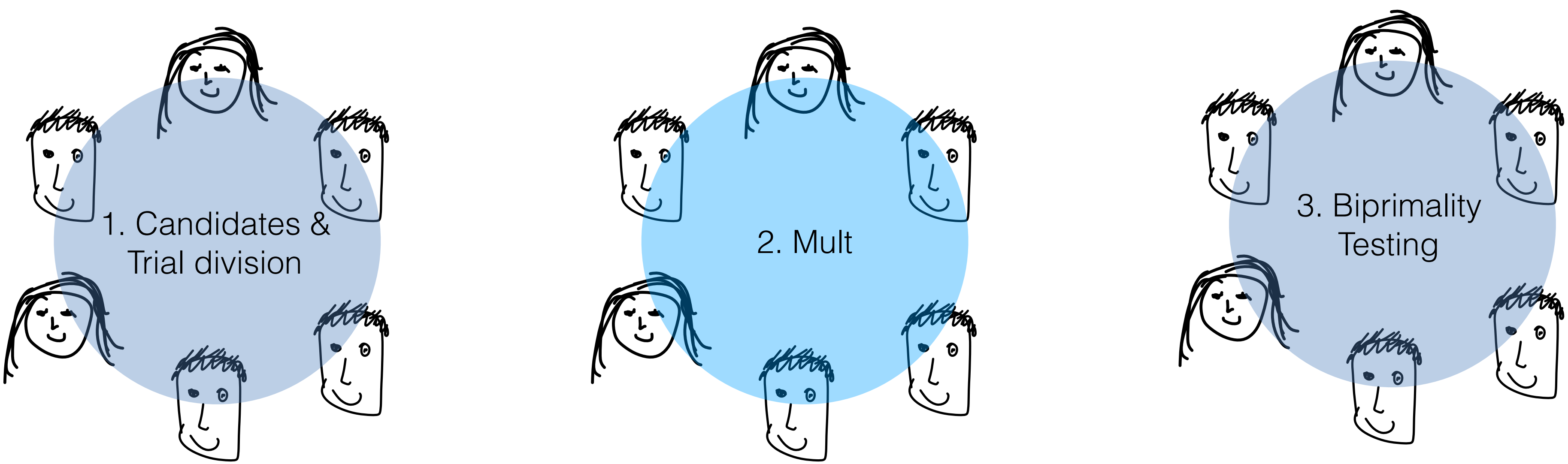
[BF97]



Parties choose p_i, q_i randomly

Boneh-Franklin Framework

[BF97]

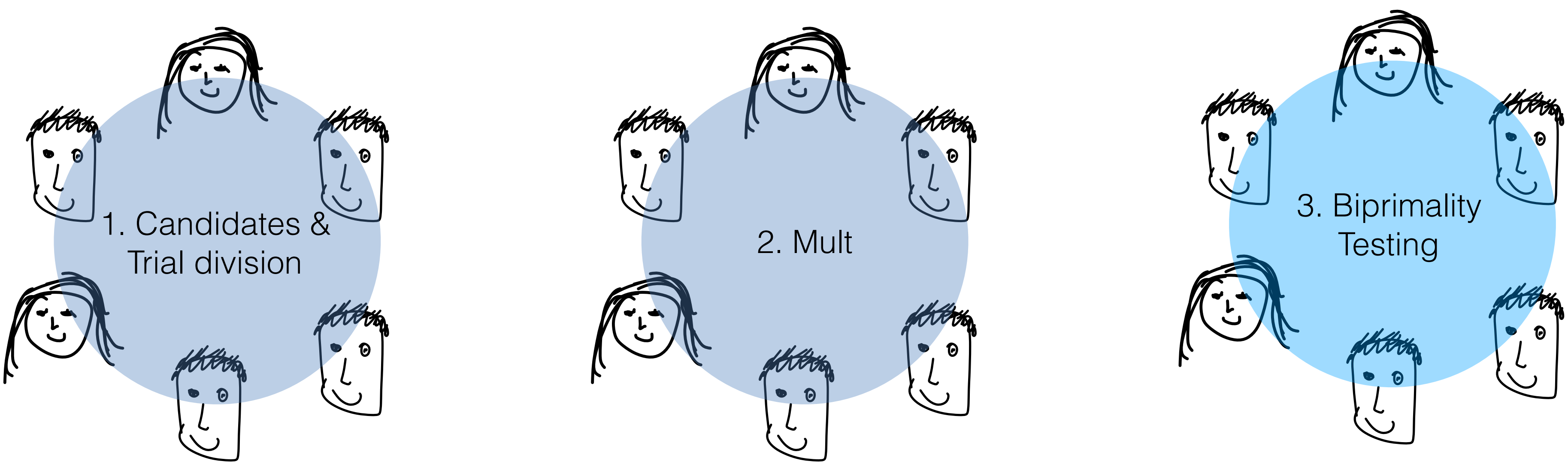


Parties choose
p_i, q_i randomly

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

Boneh-Franklin Framework

[BF97]

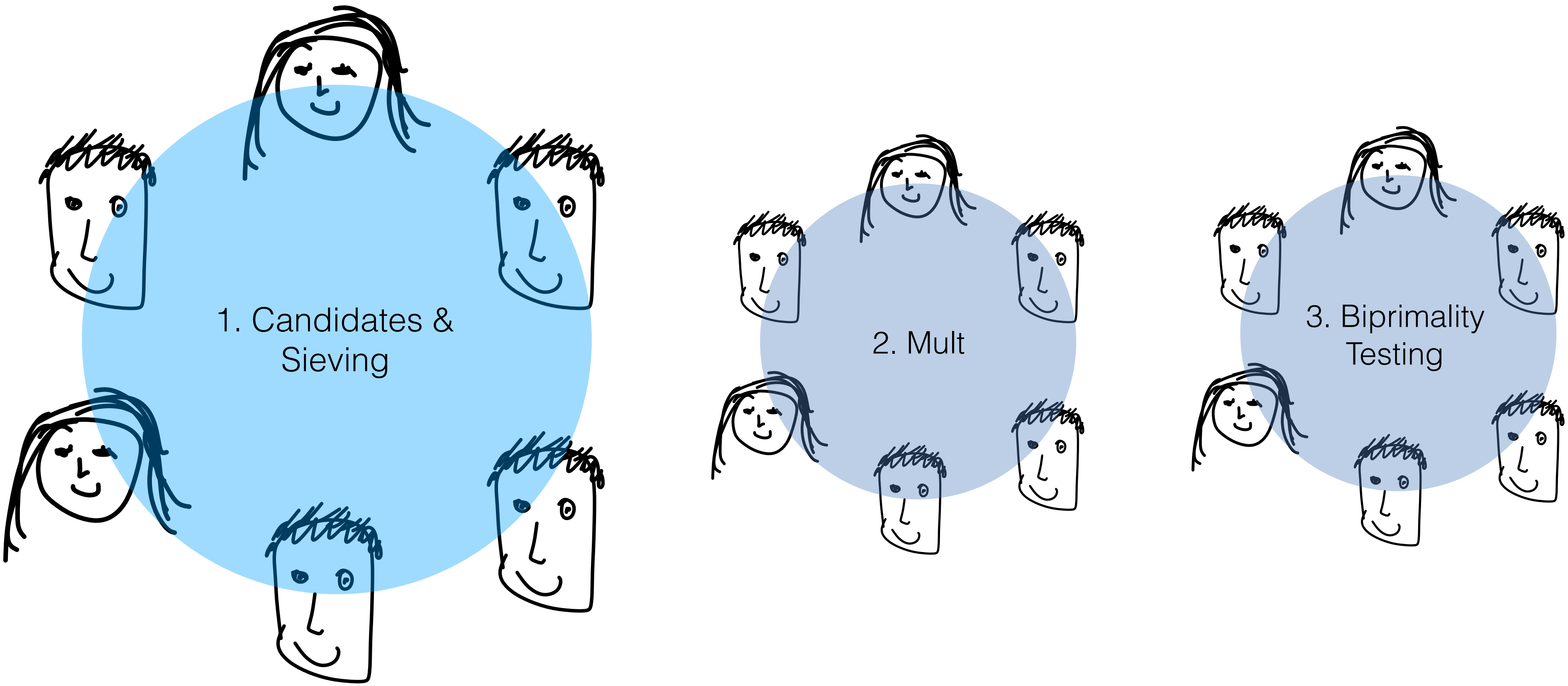


Parties choose p_i, q_i randomly

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

Is N the product of two primes?

Start with sieving trick



Candidate Trial Division: Prior Works

1. Pick p and q shares.
2. Joint Trial division.
3. If both pass, multiply.

HMRTN12	Uses El Gamal
FLOP18	Uses 1-out-of-k OT

Candidate Trial Division [Bru50]

A = randomly sampling a 1024-bit prime

B = prime is odd

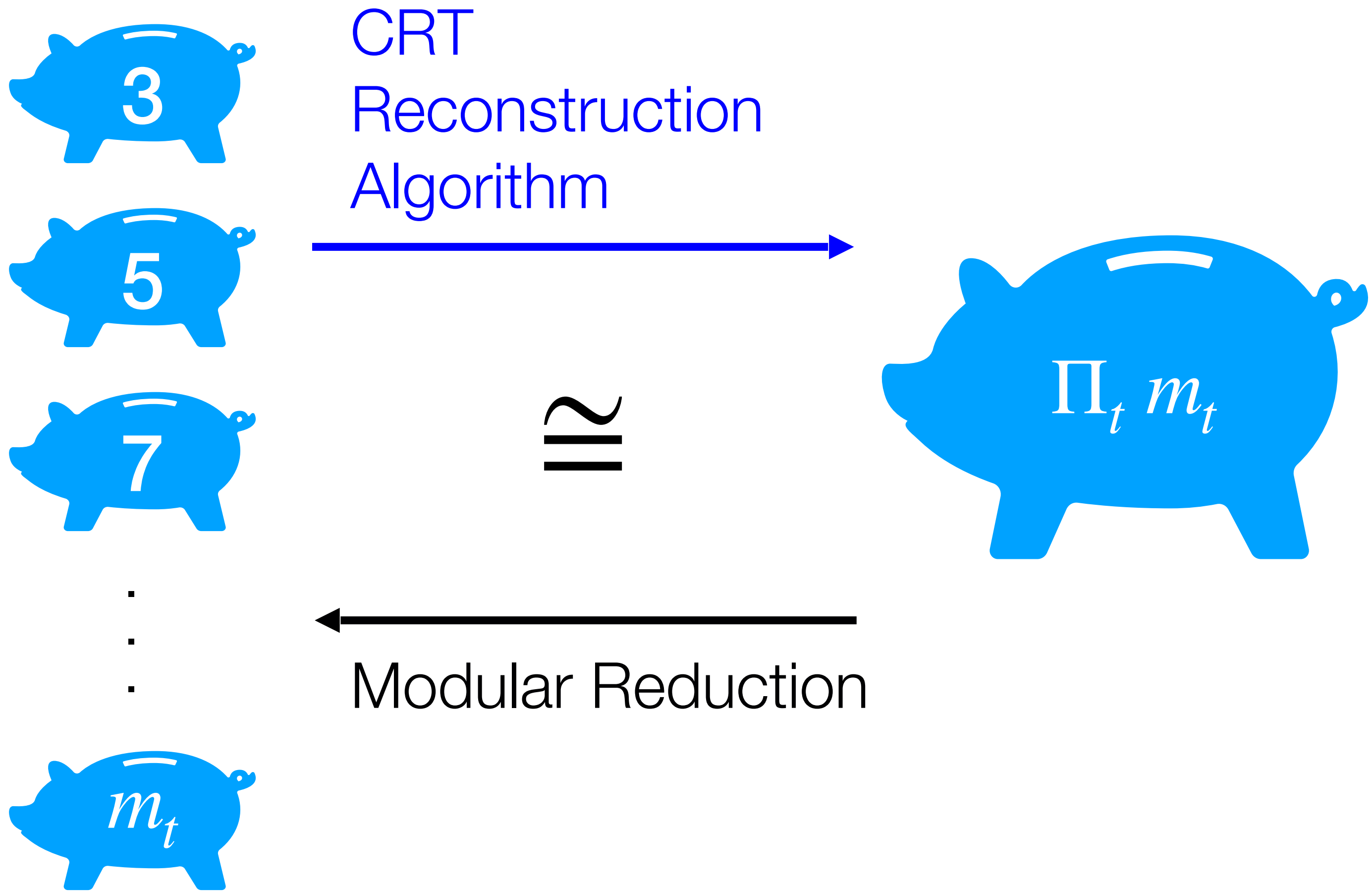
$$Pr[A \mid B] \approx \left(\frac{1}{500}\right)$$

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

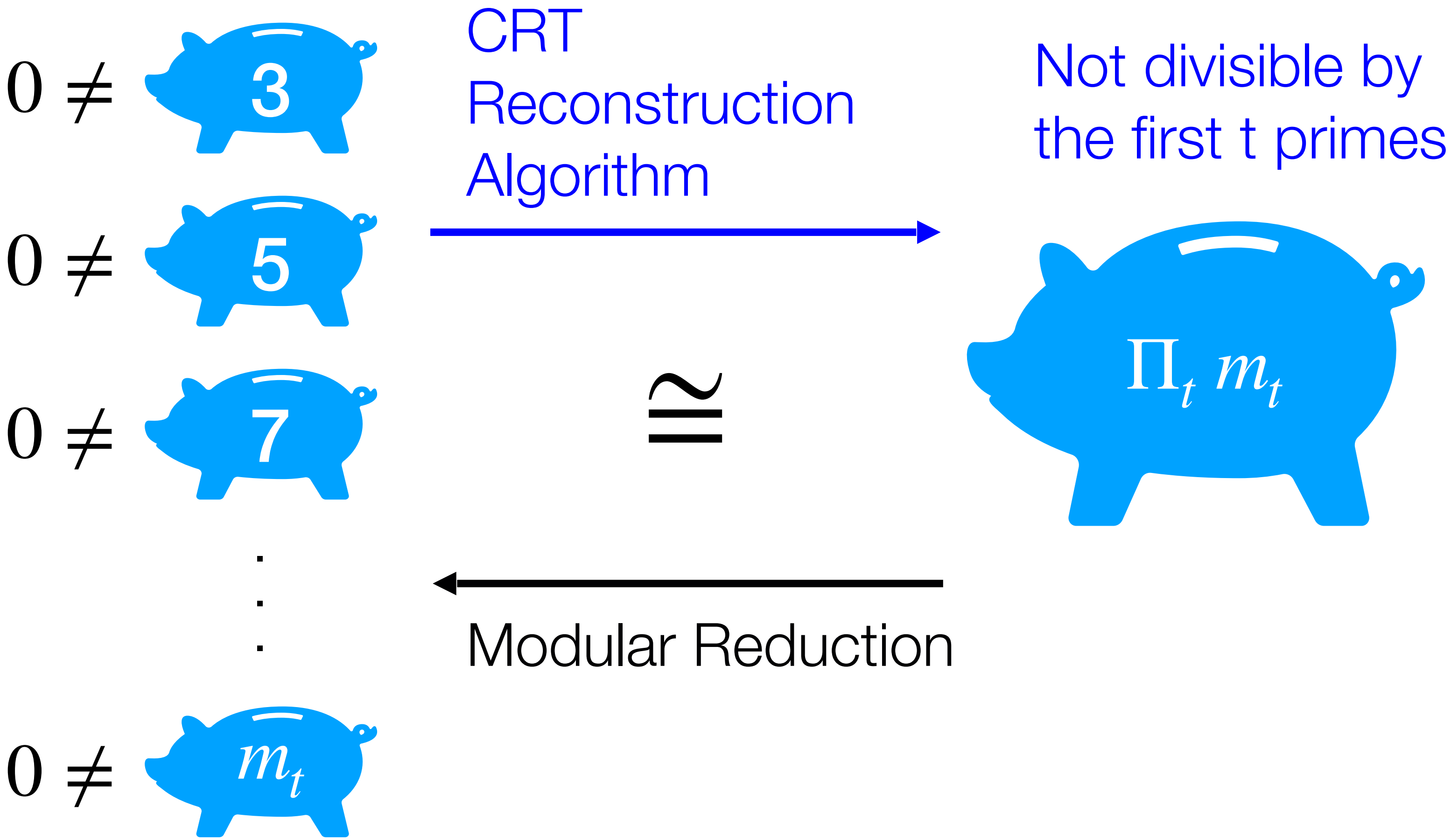
Need **250k** samples in expectation,
Large multiplication for N

Candidate Construction: Chinese Remainder Theorem (CRT)

Moduli are relatively prime!



Candidate Construction: Sieving Trick [CCD+20]



Candidate Trial Division [Bru50]

A = randomly sampling a 1024-bit prime

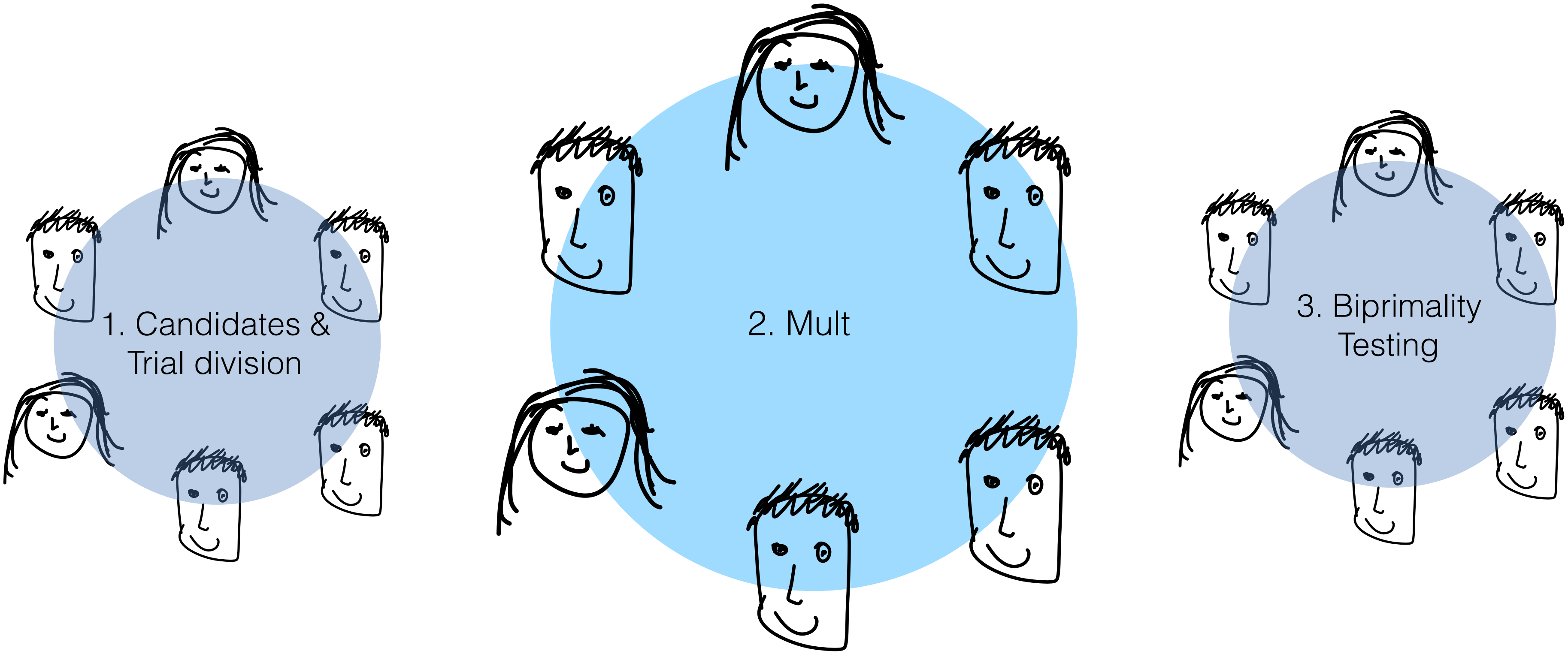
B = sieve up to 863, the 150th prime

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

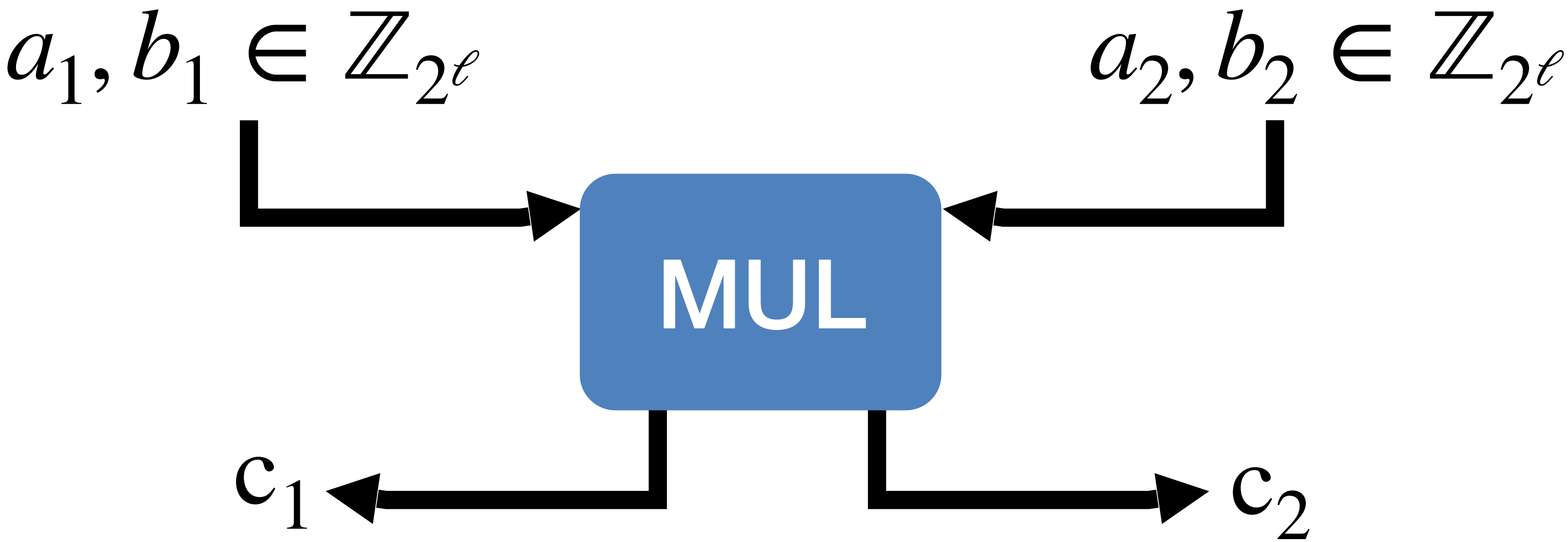
$$Pr[\text{sample biprime} \mid B] \approx \left(\frac{1}{60}\right)^2$$

Need **3600** samples in expectation,
Construct N using a **series of small** mults

Add Multiplier



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: PK Secret keys: $sk_1, \ldots, sk_n$

- Encryption

$$\text{Enc}_{PK}(m)$$

- Distributed decryption

$$m = \text{Dec}_{sk_1}(c) + \ldots + \text{Dec}_{sk_n}(c)$$

Our Approach: Threshold AHE

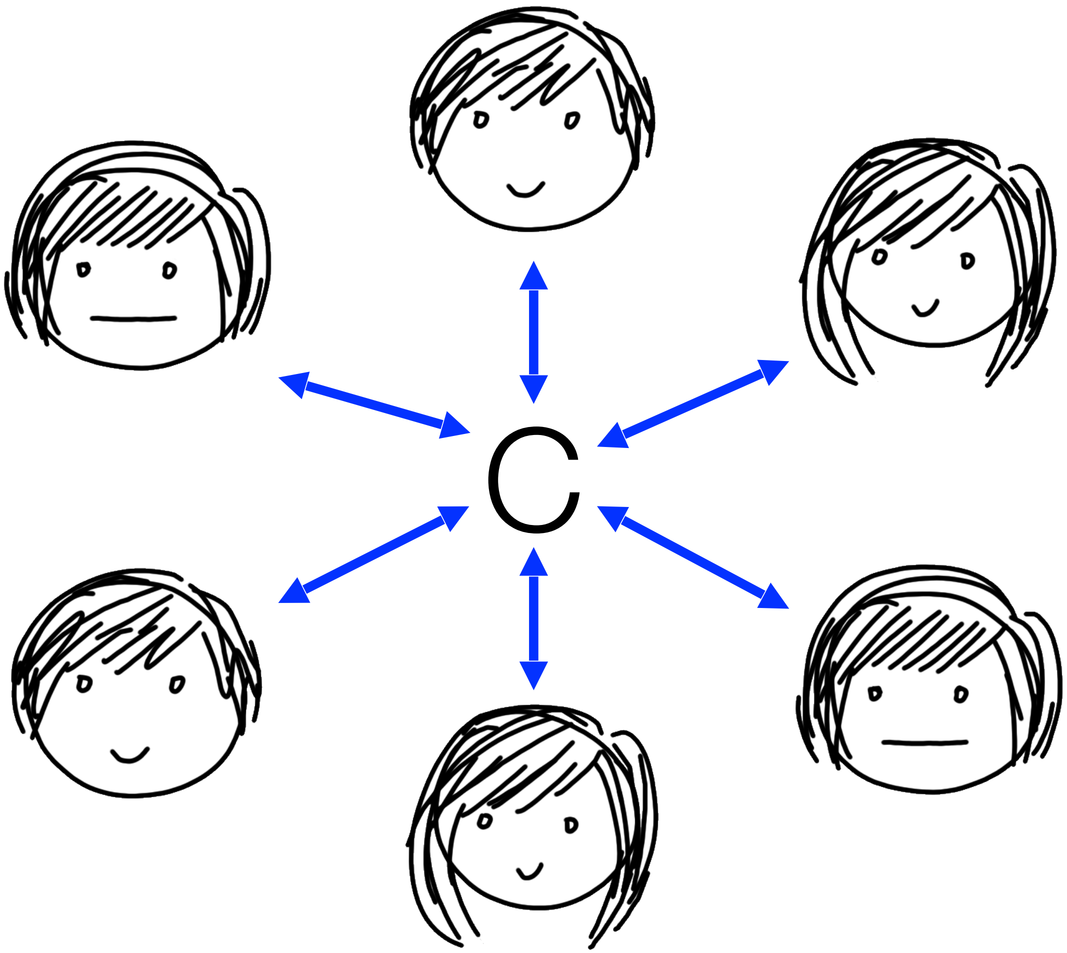
- Addition under encryption

$$\text{Enc}_{PK}(m_1) + \text{Enc}_{PK}(m_2) = \text{Enc}_{PK}(m_1 + m_2)$$

- Scalar multiplication under encryption

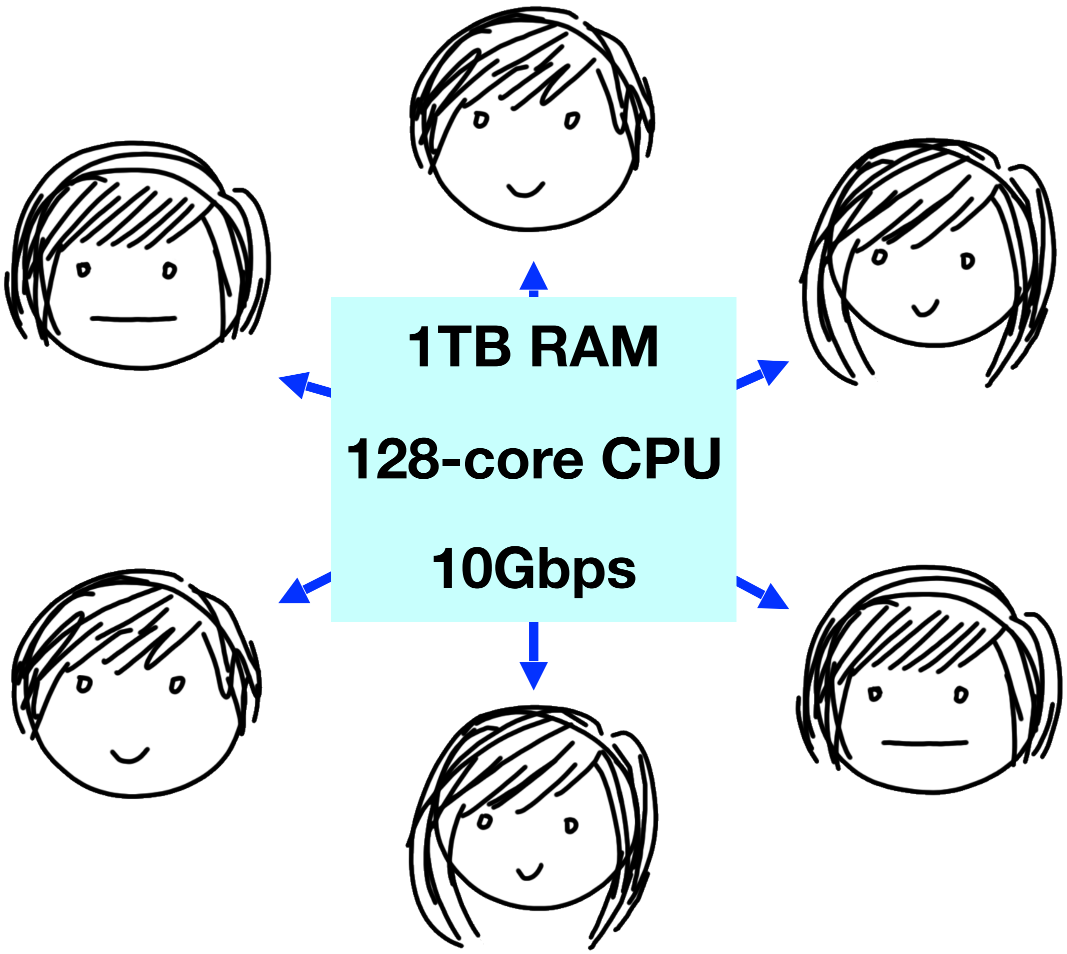
$$a \cdot \text{Enc}_{PK}(m) = \text{Enc}_{PK}(a \cdot m)$$

Our Approach: Coordinator



- Untrusted
- Does public operations (AHE Aggregations)
- Not in party count

Our Approach: Coordinator



- Untrusted
- Does public operations (AHE Aggregations)
- Not in party count

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

Our Approach: Threshold AHE

	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	$p \cdot q$	

State-of-the-Art TAHE

Paillier?

- Circular choice

El Gamal?

- Inefficient decryption (discrete log)

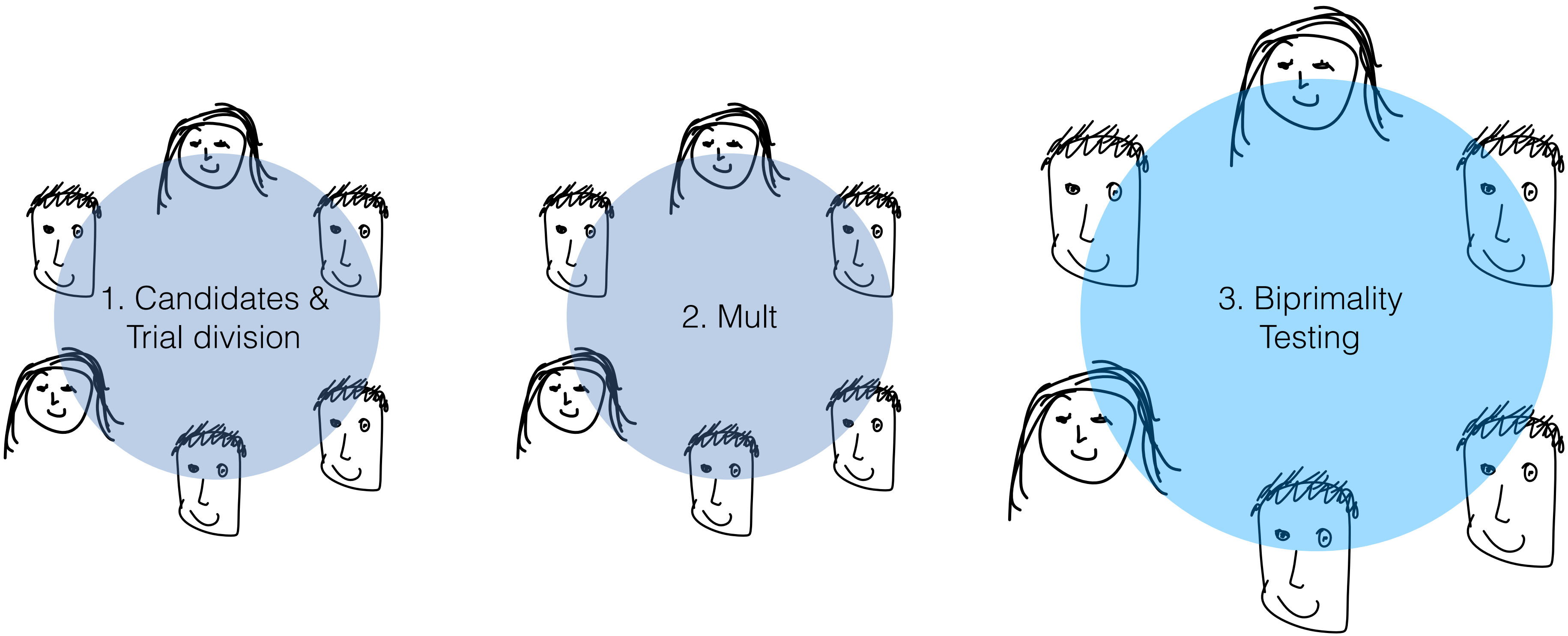
From LWE?

- Does not support all AHE operations

From Ring-LWE.

- Supports AHE, better parameters, packing

[BF97]'s Biprimality Test



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

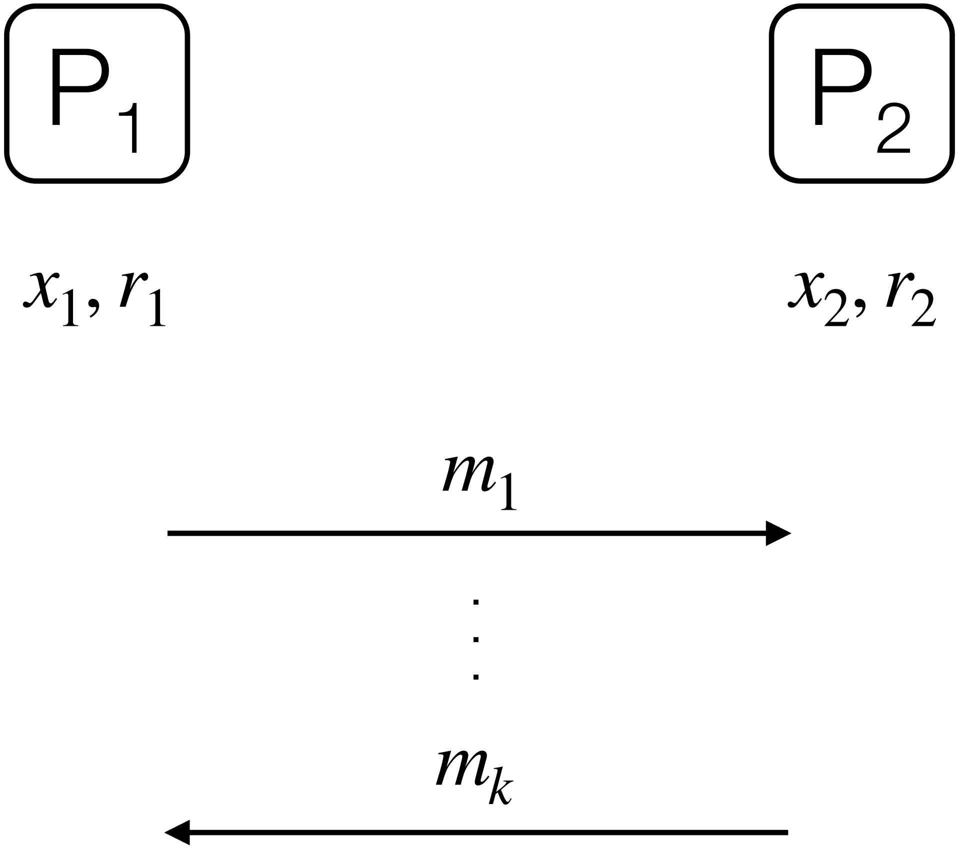
Step 2: Security against active adversaries

GMW paradigm

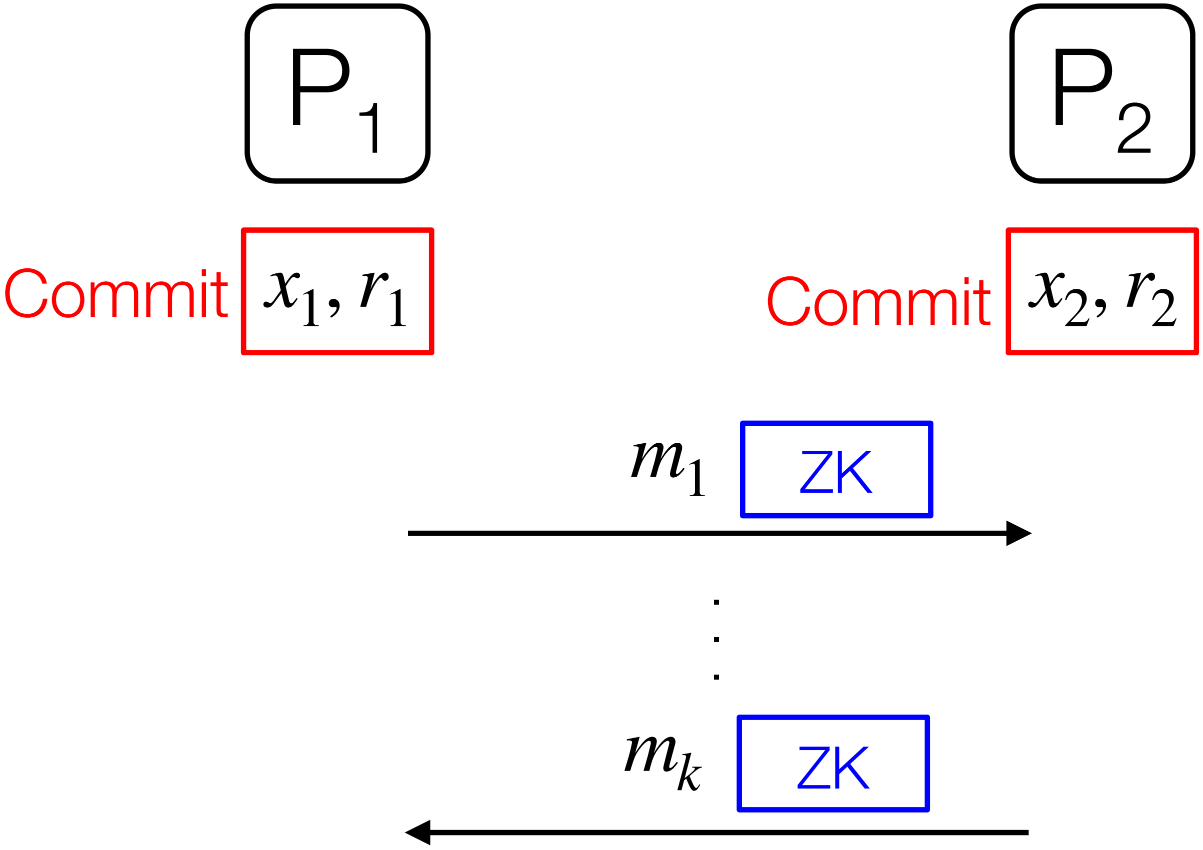
aka Zero-Knowledge Proofs

aka ”I will prove I did everything honestly!”

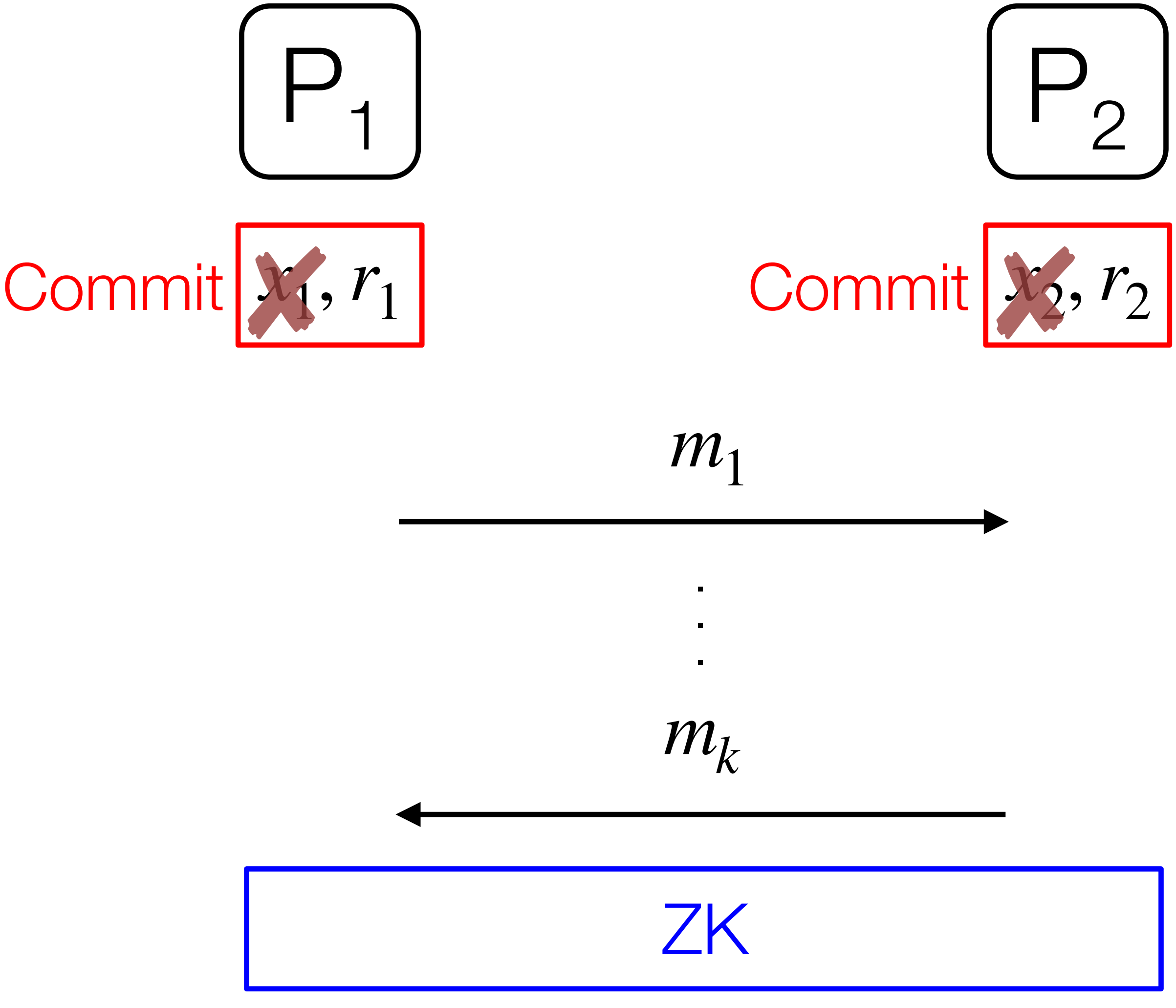
GMW Paradigm: Passive Protocol



GMW Paradigm: Active Protocol



GMW Paradigm: Our compiler



ZK Considerations

- Lattices - Operations in Ring

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

- Modulus generation - Operations in

$$Z_2, Z_3, Z_5, \ldots, Z_{823}$$

- Jacobi test - Operations in

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

ZK Schema

Party i — Coordinator

Commit($rand_{TAHE}$, $rand_{shares}$) →

← Passive Protocol →

Commit($rand_{sigma}$) →

Sigma-protocol proof →

ZK Proof that all actions are correct →

What ZK protocol to use?

Needs:

- Memory efficient
- Supports commit-and-prove
- Versatile: composable!

Ligero [AHIV17] + Sigma [Sho00]

The proofs

Ligero

- Range proofs on noise for Ring-LWE
- Other proofs - Correctness of everything else

Sigma

- Correctness of Jacobi test (for biprimality testing)

Coordinator security

- only AGGREGATES
- has no inputs or randomness
- publishes transcript, thus publicly verifiable

Summary: Our Protocol

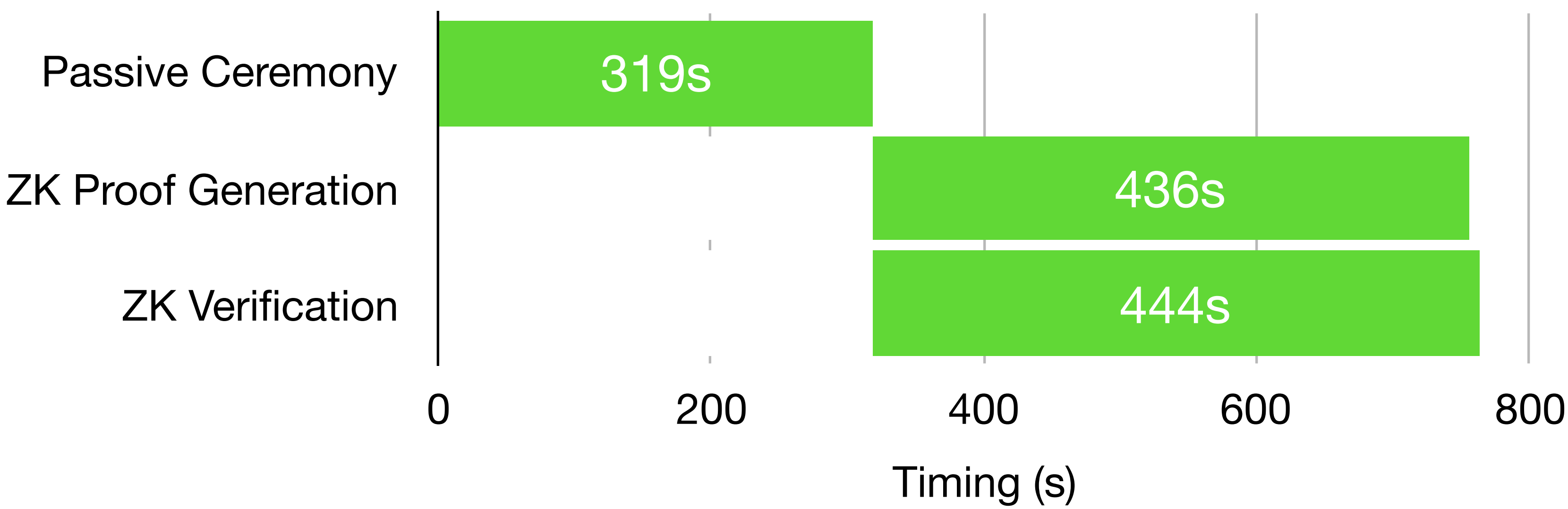
Key Setup	Generate threshold keys
Generate Candidates	Sample pre-approved primes
Compute Products	Use TAHE to compute candidates
Biprimality test	BF biprimality test
Certification	Ligero ZK + Sigma

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 19s	5m 19s
ZK Proof Generation	7m 16s	12m 35s
ZK Verification	7m 24s	12m 43s



VDF Day Trial Run

Spec

- ~25 parties (VDF day attendees!)
- Coordinator on AWS
- 2 runs. Passive succeeded, but active didn't complete.

Takeaways

- We previously tested on AWS + (few real life parties)
- Identifiable abort requires rigorous testing
- Thanks to VDF day, we learned a lot about real world conditions
- Stay tuned, for next demo!

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**
Online Comm. (Per-Party)	>1.9 GB	✗ **< ~~100 MB~~ 200 MB**
Time	35 sec (8 thread)	✓ **< 20 mins**

Thank You