KU LEUVEN

The return of Eratosthenes: Secure Generation of RSA Moduli using Distributed Sieving

Cyprien Delpech de Saint Guilhem, Eleftheria Makri, Dragos Rotaru, Titouan Tanguy

Eleftheria Makri for the Ei/Ψ seminar
TU Einhoven, 27 May, 2021

The return of Eratosthenes:

Secure Generation of RSA Moduli using Distributed Sieving

The return of Eratosthenes:

Secure **Generation of RSA Moduli** using Distributed Sieving

RSA Modulus

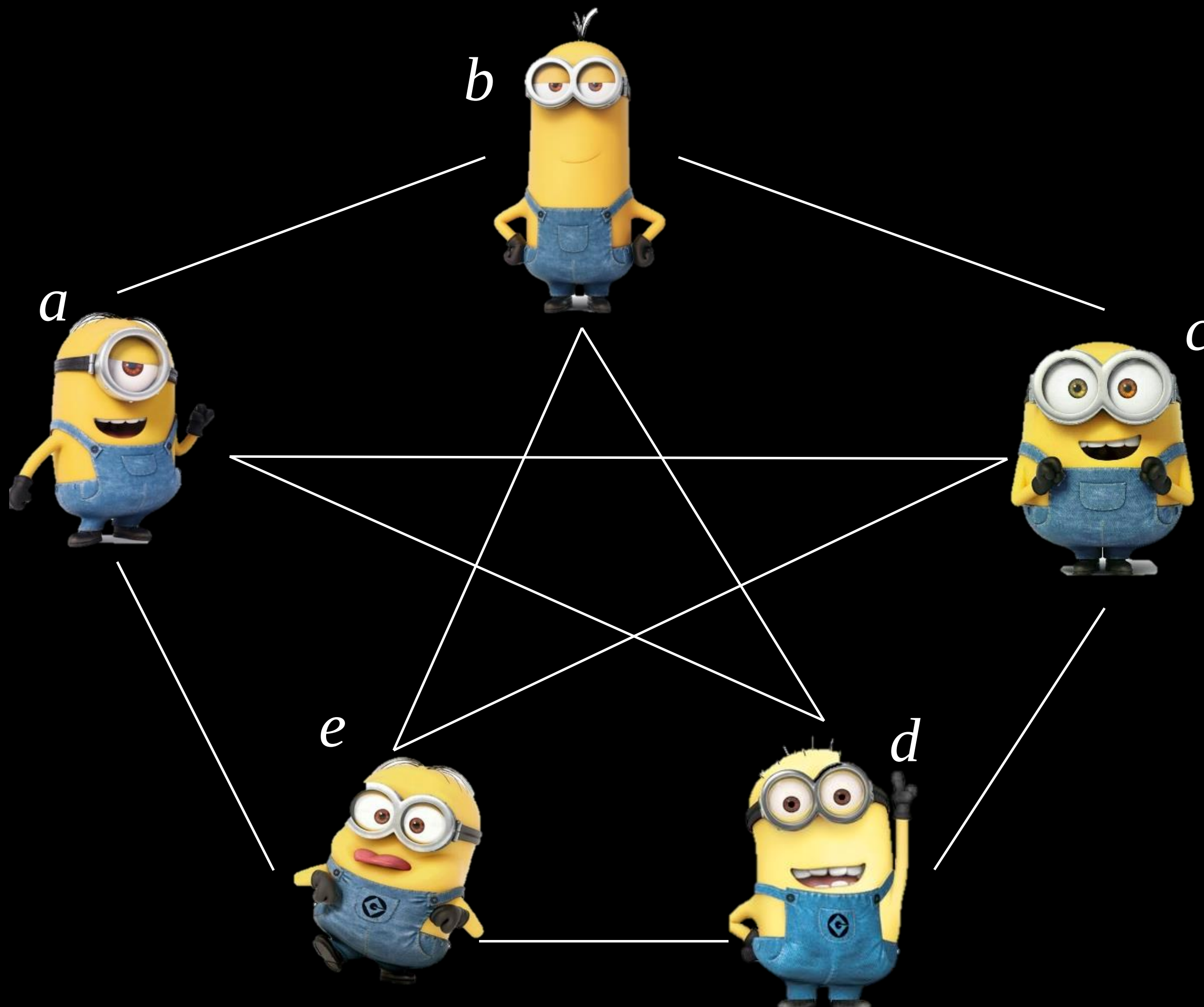
- A biprime (i.e., product of 2 primes), usually denoted by N, with secret prime factors, usually denoted by p and q.

- The heart of the first public key cryptosystem, where security is based on the factoring assumption.

The return of Eratosthenes:

Secure Generation of RSA Moduli using **Distributed** Sieving

Multiparty Computation



→ Securely compute $f(a, b, c, d, e)$.

The return of Eratosthenes:

Secure Generation of RSA Moduli using Distributed **Sieving**

(Distributed) Sieving*

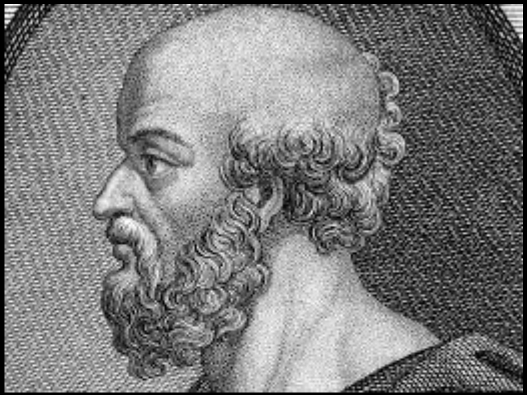
- *p* and *q* are secret, making efficient trial division cumbersome.
- We set M_{Sample} to be the product of all odd (small) primes up to a certain sieving bound.
- Each party selects their share s.t. it is relatively prime to M_{Sample}, meaning that their product is also relatively prime to M_{Sample}.

→Hence, the product of the multiplicative shares has no small prime factors.

*Inspired by Malkin et al. MWB99

The return of Eratosthenes:

Secure Generation of RSA Moduli using Distributed Sieving

The  of 

RSA Modulus: Applications

- Threshold Cryptography
- Permissionless Consensus in Blockchain
- Verifiable Delay Functions
- *Interesting beyond academia: (e.g., Unbound, the VDF Alliance, the Ethereum Foundation, Ligero)*

Related Work

Protocol	Security	Dishonest Majority	#Parties	Test	No Leakage
[BF97]	Passive	×	$n \geq 3$	biprimality	✓
[FMY98]	Active	×	$n \geq 3$	biprimality	✓
[PS98]	Active	✓	$n = 2$	biprimality	×
[Gil99]	Passive	✓	$n = 2$	biprimality	✓
[ACS02]	Passive	×	$n \geq 3$	primality	✓
[DM10]	Active	×	$n = 3$	primality	✓
[HMRT12, HMR+19]	Active	✓	$n \geq 2$	biprimality	✓
[FLOP18]	Active	✓	$n = 2$	biprimality	×
[CCD+20]	Active	✓	$n \geq 2$	biprimality	✓
[CHI+20]	Active*	✓	$n \geq 2$	biprimality	✓
Ours	Active	✓	$n \geq 2$	biprimality	✓

*Diogenes works in the semi-honest coordinator model, and active security is only guaranteed for the non-coordinating parties.

Contribution

- RSA modulus generation protocol working for generic MPC.
- Constructive sampling of candidate primes, by transforming multiplicative sharings to additive, via semi-honest multiplication.
- Jacobi test based biprimality, where the consistency check happens only on shares that pass the test.
- Protocol for converting additive shares over a ring to additive shares over the integers, of independent interest.
- Improved communication cost over the state-of-the-art.

The Boneh-Franklin Blueprint

1. Pick prime candidates (via trial division)

2. Securely multiply candidates

3. Biprimality testing

Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\ q$ and reveal N
3. Jacobi biprimality test
4. Consistency check
5. GCD test

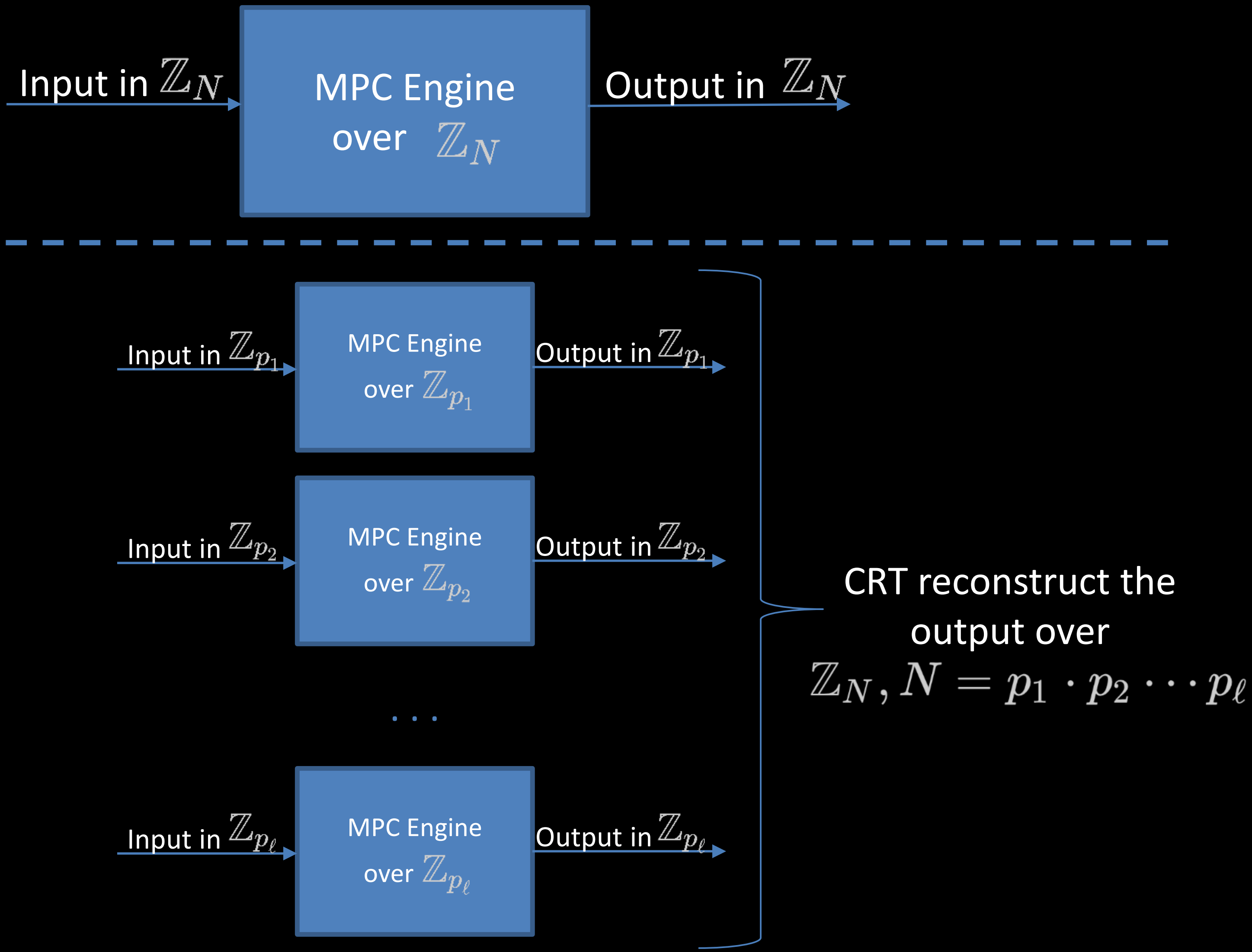
Our Protocol

1. **Sample candidate primes *p* and *q***
2. Securely compute $N = p\ q$ and reveal N
3. Jacobi biprimality test
4. Consistency check
5. GCD test

Sampling

- Distributed sieving, sampling multiplicative shares without small prime factors
- Semi-honest multiplication on the shares, allowing additive errors
- Transform to additive shares, while ensuring they fall within bounds that determine the bit-length of the primes
- Input into MPC-CRT engines

MPC – CRT


Input in $\mathbb{Z}_N$
MPC Engine over $\mathbb{Z}_N$
Output in $\mathbb{Z}_N$
Input in $\mathbb{Z}_{p_1}$
MPC Engine over $\mathbb{Z}_{p_1}$
Output in $\mathbb{Z}_{p_1}$
Input in $\mathbb{Z}_{p_2}$
MPC Engine over $\mathbb{Z}_{p_2}$
Output in $\mathbb{Z}_{p_2}$
. . .
Input in $\mathbb{Z}_{p_\ell}$
MPC Engine over $\mathbb{Z}_{p_\ell}$
Output in $\mathbb{Z}_{p_\ell}$
CRT reconstruct the output over $\mathbb{Z}_N, N = p_1 \cdot p_2 \cdots p_\ell$

Our Protocol

1. Sample candidate primes *p* and *q*
2. **Securely compute $N = p\ q$ and reveal N**
3. Jacobi biprimality test
4. Consistency check
5. GCD test

Combine

- Extend the CRT representation, so that the product is taken over the integers (i.e., prevent overflow)
- Perform "standard" secure multiplication over the MPC-CRT engines
- Reveal and CRT-Reconstruct the product N
- Check that N falls within the predetermined bounds, and it is coprime to M_{Sample}

Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\, q$ and reveal N
3. **Jacobi biprimality test**
4. Consistency check
5. GCD test

Jacobi Test

- Sample public $\gamma \in \mathbb{Z}_N$ s.t. the Jacobi symbol $(\frac{\gamma}{N}) = 1$
- Securely compute $\phi(N)/4$ in the exponent of γ
- Abort if $\gamma^{\phi(N)/4} \neq \pm 1$

- *This test accepts false positives with probability ½. We repeat the test sec times to increase the probability of N being a biprime to* $2^{-\text{sec}}$

Our Protocol

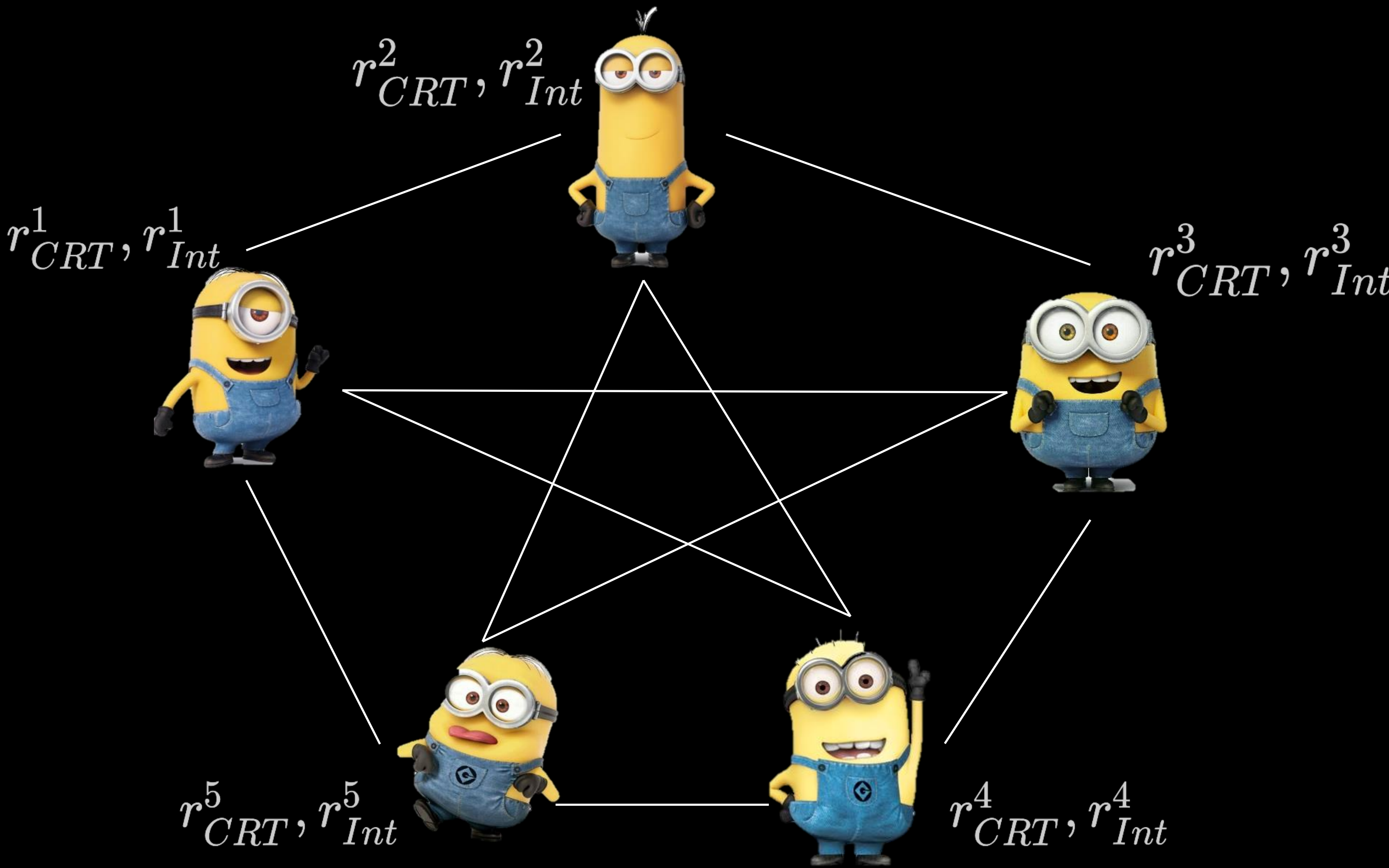
1. Sample candidate primes p and q
2. Securely compute $N = p\,q$ and reveal N
3. Jacobi biprimality test
4. **Consistency check**
5. GCD test

Consistency Check

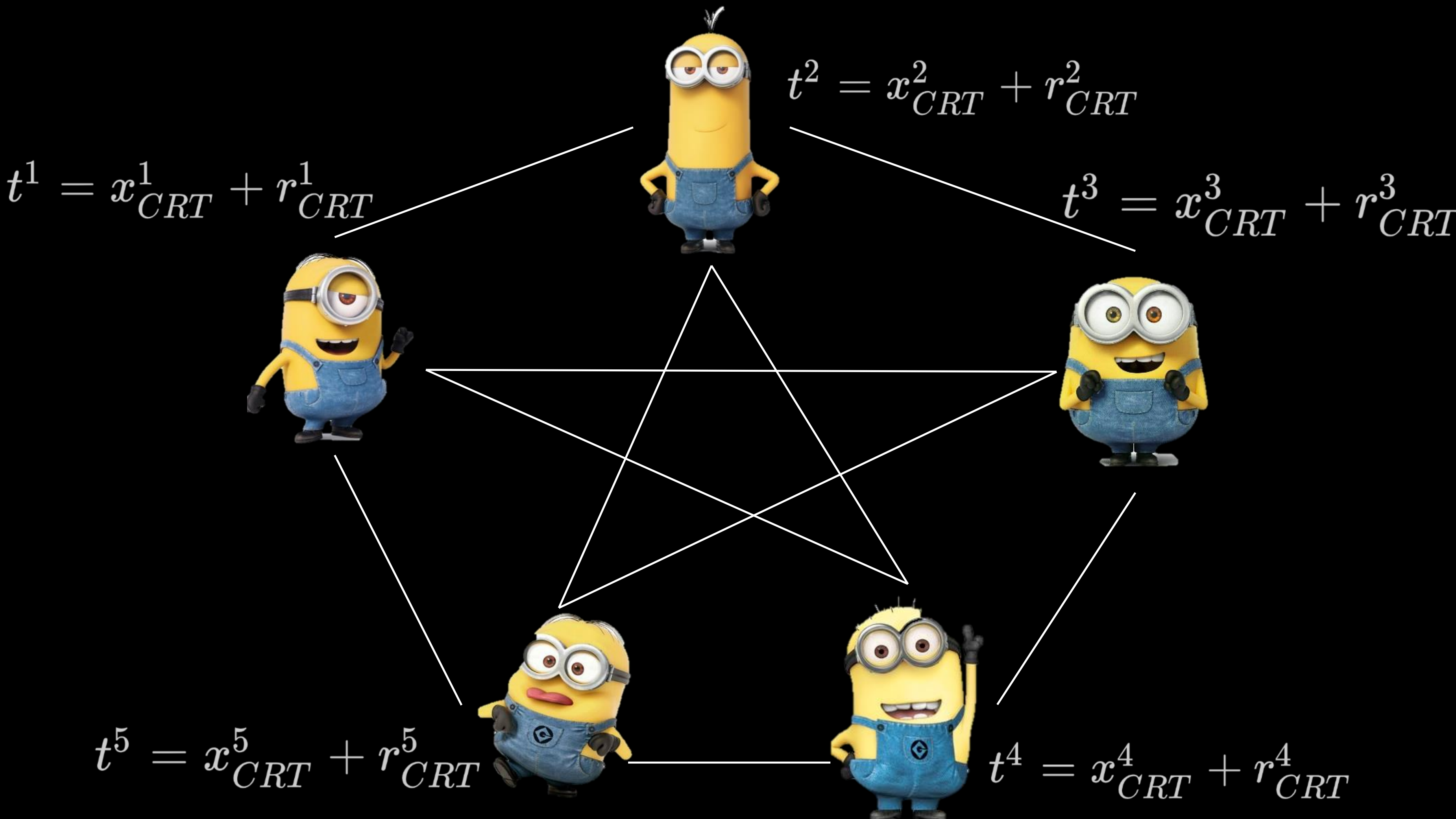
- *This check ensures security against malicious parties, who contributed inconsistent shares to the Jacobi test.*

1. LevelUp s.t. the CRT representation allows the consistency check computations to be performed without overflow.
2. Sample bounded randomness and multiplicatively mask the secret exponent
3. Convert the CRT represented masked sharing to a sharing over the integers

ConvInt Protocol

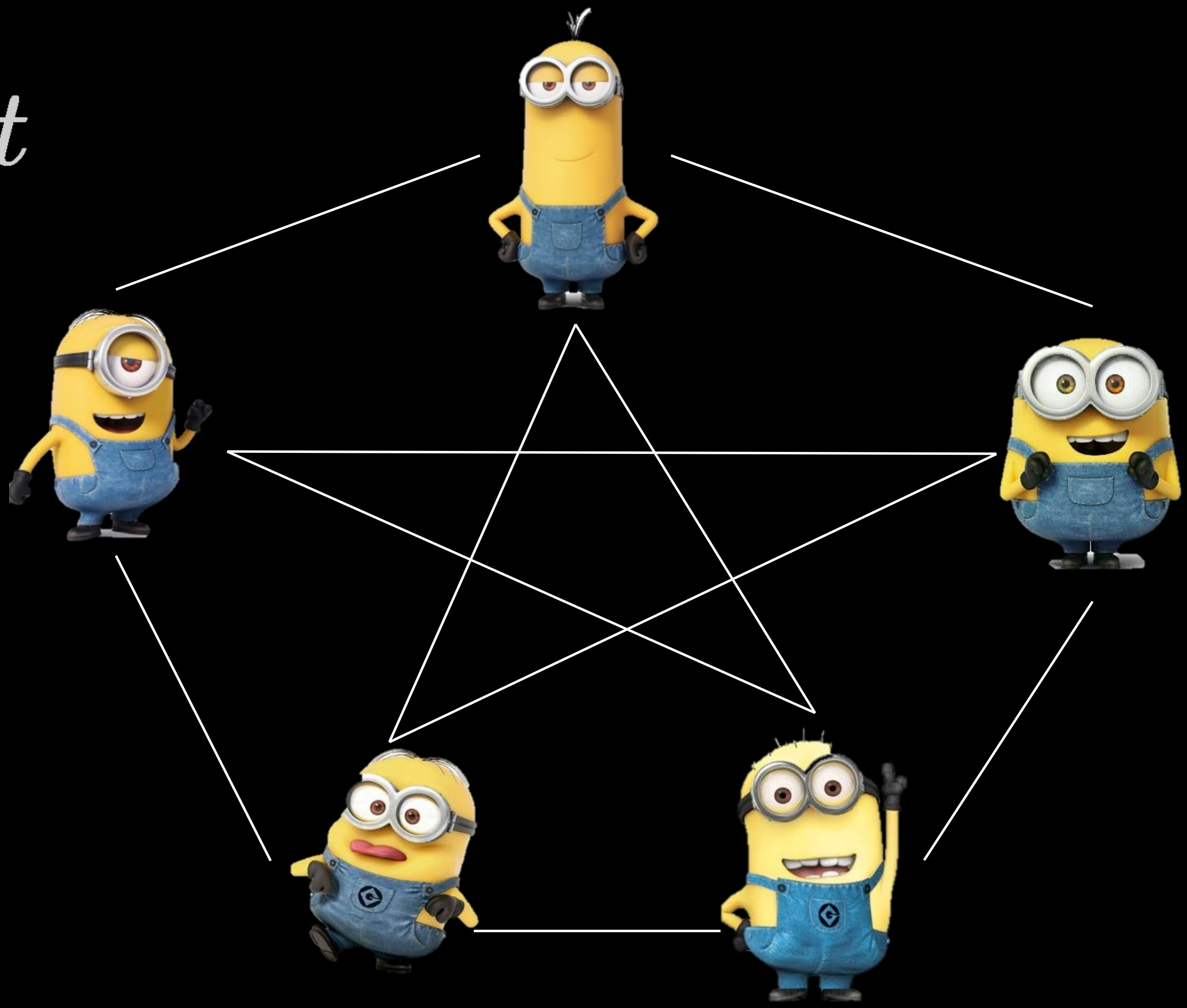


ConvInt Protocol

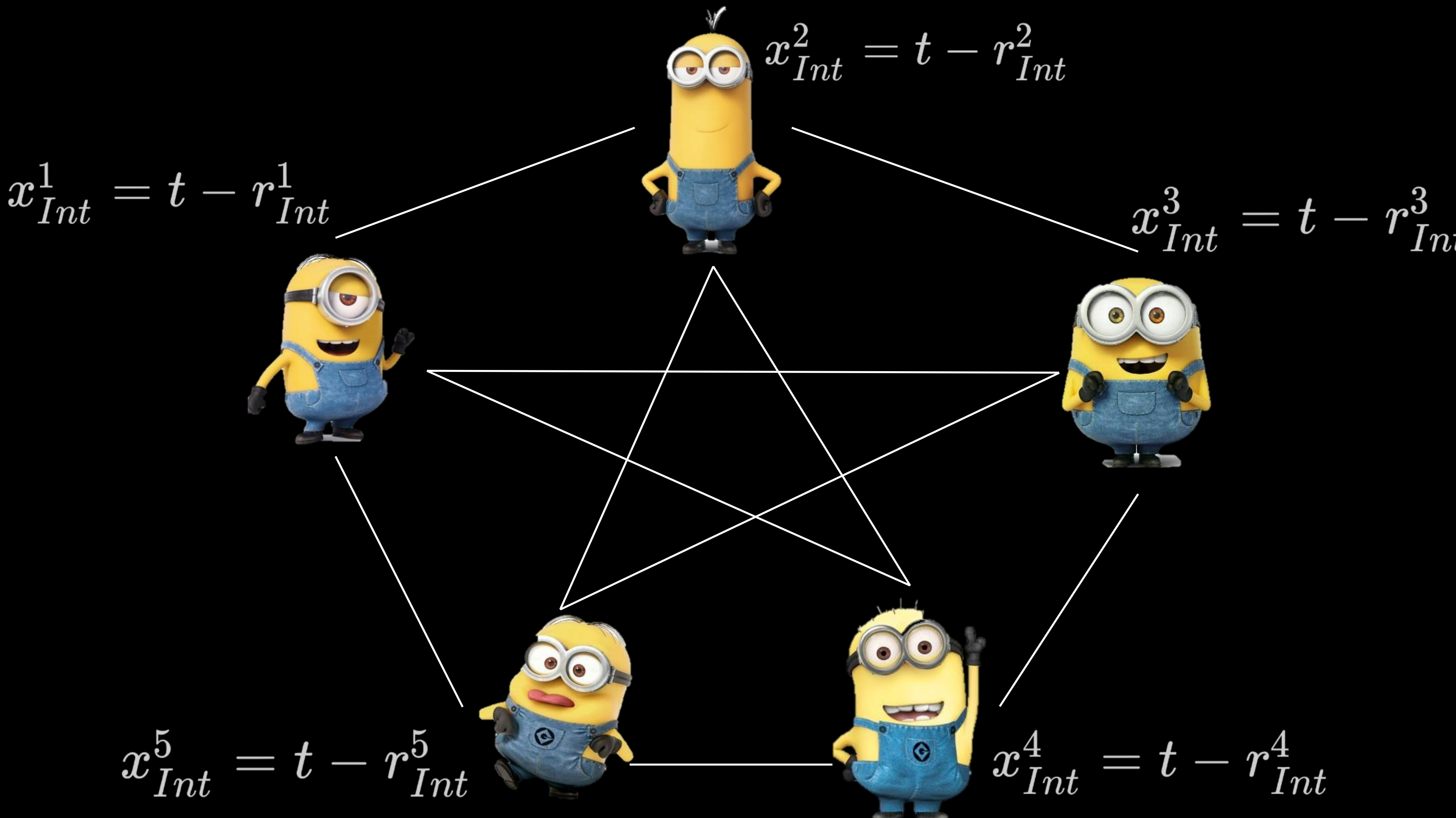


ConvInt Protocol

Open t



ConvInt Protocol



Our Protocol

1. Sample candidate primes p and q
2. Securely compute $N = p\, q$ and reveal N
3. Jacobi biprimality test
4. Consistency check
5. **GCD test**

Efficiency Analysis (1/2)

Scheme	CCD+20	Ours	CCD+20	Ours	CCD+20	Ours
κ	**1024**	**1024**	**1536**	**1536**	**2048**	**2048**
semi-honest (MB)	139	41.68	416	116.55	910	243.3
malicious (GB)	20.81	0.64	43.42	1.188	74.52	1.99

Communication cost per party, for 2-party protocol.

Efficiency Analysis (2/2)

Scheme	CCD+20	Ours	CCD+20	Ours	CCD+20	Ours
κ	**1024**	**1024**	**1536**	**1536**	**2048**	**2048**
semi-honest (MB)	2.09	4.34	6.24	12.17	13.65	25.23
malicious (GB)	1020	68.8	4734	153.2	8100	281.91

Communication cost per party, for 16-party protocol.

Summary of Contributions

- RSA modulus generation protocol working with generic MPC.
- Fully exploit Distributed Sieving techniques, and public knowledge to perform it semi-honestly without degrading the overall security.
- Convert to Integer protocol, of independent interest.
- Up to 37x better communication cost compared to CCD+20.