Multiparty Generation of an RSA Modulus

Megan Chen Ran Cohen Jack Doerner
Yashvanth Kondi Eysa Lee Schuyler Rosefield
abhi shelat

Northeastern University

http://ia.cr/2020/370

In this work

Multiparty RSA modulus sampling
with malicious security
against a dishonest majority

- First with n-parties from only OT and the Factoring Assumption
- First with cubic communication complexity in modulus length

What are RSA Moduli?

$RSA\ Modulus\ (N)$: a biprime with random, secret factors (p, q)

$Biprime$: product of two primes

Factoring Assumption: no efficient algorithm to factor N

Equivalently: finding $|\mathbb{Z}_N^*|$ is hard

Why RSA Moduli?

Useful For

- Signatures and Encryption
  [RSA77], [Paillier-99]
- Cryptographic Accumulators
  [Benolah-deMare-93], [Camenisch-Lysyanskaya-02],
  [Li-Li-Xue-07], [Boneh-Bünz-Fisch-19]
- VDFs and Timelock Puzzles
  [Rivest-Shamir-Wagner-96], [Boneh-Bonneau-Bünz-Fisch-18],
  [Wesolowski-19], [Pietrzak-19], [Ephraim-Freitag-Komargodski-Pass-19]
- Efficient zk-SNARKs
  [Bünz-Fisch-Szepieniec-19], [Lai-Malavolta-19]
- Many more…

Why RSA Moduli?

Useful For

- Signatures and Encryption

  [RSA77], [Paillier-99]

- Cryptographic Accumulators

  [Benolah-deMare-93], [Camenisch-Lysyanskaya-02], [Li-Li-Xue-07], [Boneh-Bünz-Fisch-19]

- VDFs and Timelock Puzzles

  [Rivest-Shamir-Wagner-96], [Boneh-Bonneau-Bünz-Fisch-18], [Wesolowski-19], [Pietrzak-19], [Ephraim-Freitag-Komargodski-Pass-19]

- Efficient zk-SNARKs

  [Bünz-Fisch-Szepieniec-19], [Lai-Malavolta-19]

- Many more…

History of the Problem

1997

- [Boneh-Franklin-97/01]
- [Frankel-MacKenzie-Yung-98]
- [Poupard-Stern-98
- [Gilboa-99]
- [Malkin-Wu-Boneh-99]
- [Algesheimer-Camenish-Shoup-02]
- [Damgård-Mikkelsen-10]
- [Gavin-12]
- [Hazay-Mikkelsen-Rabin-Toft-12]
- [Frederiksen-Lindell-Osheter-Pinkas-18]
- [Hazay-Mikkelsen-Rabin-Toft-Nicolosi-19]
- [This Work]
- [Chen-Hazay-Ishai-Kashnikov-Micciancio-Riviere-shelat-Muthu-Wang-20]

2020

History of the Problem

1997

- [Boneh-Franklin-97/01]
- [Frankel-MacKenzie-Yung-98]
- [Poupard-Stern-98]
- [Gilboa-99]
- [Malkin-Wu-Boneh-99]
- [Algesheimer-Camenish-Shoup-02]
- [Damgård-Mikkelsen-10]
- [Gavin-12]
- [Hazay-Mikkelsen-Rabin-Toft-12]
- [Frederiksen-Lindell-Osheter-Pinkas-18]
- [Hazay-Mikkelsen-Rabin-Toft-Nicolosi-19]
- [This Work]
- [Chen-Hazay-Ishai-Kashnikov-Micciancio-Riviere-shelat-Muthu-Wang-20]

2020

History of the Problem

1997

- [Boneh-Franklin-97/01]
- [Frankel-MacKenzie-Yung-98]
- [Poupard-Stern-98]
- [Gilboa-99]
- [Malkin-Wu-Boneh-99]
- [Algesheimer-Camenish-Shoup-02]
- [Damgård-Mikkelsen-10]
- [Gavin-12]
- [Hazay-Mikkelsen-Rabin-Toft-12]
- [Frederiksen-Lindell-Osheter-Pinkas-18]
- [Hazay-Mikkelsen-Rabin-Toft-Nicolosi-19]
- [This Work]
- [Chen-Hazay-Ishai-Kashnikov-Micciancio-Riviere-shelat-Muthu-Wang-20]

2020

History of the Problem

1997

- [Boneh-Franklin-97/01]
- [Frankel-MacKenzie-Yung-98]
- [Poupard-Stern-98]
- [Gilboa-99]
- [Malkin-Wu-Boneh-99]
- [Algesheimer-Camenish-Shoup-02]
- [Damgård-Mikkelsen-10]
- [Gavin-12]
- [Hazay-Mikkelsen-Rabin-Toft-12]
- [Frederiksen-Lindell-Osheter-Pinkas-18]
- [Hazay-Mikkelsen-Rabin-Toft-Nicolosi-19]
- [This Work]
- [Chen-Hazay-Ishai-Kashnikov-Micciancio-Riviere-shelat-Muthu-Wang-20]

2020

History of the Problem

1997

- [Boneh-Franklin-97/01]
- [Frankel-MacKenzie-Yung-98]
- [Poupard-Stern-98]
- [Gilboa-99]
- [Malkin-Wu-Boneh-99]
- [Algesheimer-Camenish-Shoup-02]
- [Damgård-Mikkelsen-10]
- [Gavin-12]
- [Hazay-Mikkelsen-Rabin-Toft-12]
- [Frederiksen-Lindell-Osheter-Pinkas-18]
- [Hazay-Mikkelsen-Rabin-Toft-Nicolosi-19]
- [This Work]
- [Chen-Hazay-Ishai-Kashnikov-Micciancio-Riviere-shelat-Muthu-Wang-20]

2020

Overview

1. Problem & Context
2. Our Protocol
3. Efficiency
4. Conclusion

Sampling a Biprime

1. Choose random $p \leftarrow \mathbb{Z}_{2^k}$
2. If p is not prime, repeat step 1
3. Sample q in the same way as p
4. Compute $N = p \cdot q$

Miller-Rabin Test

$$a^{p-1} mod\ p$$

Boneh-Franklin Test

Statistical Param

Subgroup of $\mathbb{Z}_N^*$ with Jacobi Symbol 1

1. Repeat $O(s)$ times:
   a. Choose random $a \leftarrow \mathbb{J}_N$
   b. Test $a^{N-p-q+1} mod\ N = \pm 1$
2. Test $\gcd(N, p+q-1) = 1$

Public!

[BF97, BF01]

Boneh-Franklin Test

Suppose $p = \sum_{i \in [n]} p_i$ and $q = \sum_{i \in [n]} q_i$

Party Count

b. Test $a^{N-p-q+1} mod\ N = \pm 1$

2. Test $\gcd(N, p + q - 1) = 1$

Public!

[BF97, BF01]

Boneh-Franklin Test

Suppose $p = \sum_{i \in [n]} p_i$ and $q = \sum_{i \in [n]} q_i$

Party Count

Boneh and Franklin prove that this leakage is negligible.

$$a^{N-p-q+1} mod \boxed{N} = a^{N+1} \prod_{i \in [n]} \boxed{a^{-p_i - q_i} \bmod N}$$

Public!
Party *i* can compute this

[BF97, BF01]

Trial Division

For candidate prime p:

$p \bmod 3 = 2$ **OK**

$p \bmod 5 = 1$ **OK**

$p \bmod 7 = 5$ **OK**

$p \bmod 11 = 0$ **Reject**

Trial Division

$$\Pr_{p \leftarrow \mathbb{Z}_{2^k}} \left[\begin{array}{l} p \text{ is prime} \mid \\ \quad \text{no divisors} \leq B \end{array} \right] \in \Omega(\log B / k)$$

Trial Division Bound

We need $O(k^2 / \log^2 B)$ tries in expectation

[BF97, BF01]

Recent Progress

	[HMRT12] [HMRTN19]	[FLOP18]
Security	Malicious	Malicious
Parties	n	2
Corruptions	$n - 1$	1
Uses HE	Yes	No
Assumptions	DCR, DDH	OT
ZK over Crypto	Yes	Almost No
Extra Leakage	No	Yes
Covert DOS	No	Yes

Incurs $O(s)$ overhead

Open Question

	[HMRT12] [HMRTN19]	[FLOP18]	Open
Security	Malicious	Malicious	Malicious
Parties	n	2	n
Corruptions	$n - 1$	1	$n - 1$
Uses HE	Yes	No	No
Assumptions	DCR, DDH	OT	OT
ZK over Crypto	Yes	Almost No	No
Extra Leakage	No	Yes	No
Covert DOS	No	Yes	No

Our Contribution

	[HMRT12] [HMRTN19]	[FLOP18]	[This Work]
Security	Malicious	Malicious	Malicious
Parties	n	2	n
Corruptions	$n-1$	1	$n-1$
Uses HE	Yes	No	No
Assumptions	DCR, DDH	OT	OT, Factoring
ZK over Crypto	Yes	Almost No	No
Extra Leakage	No	Yes	No
Covert DOS	No	Yes	No
Analysis	Monolithic	Monolithic	Modular
Comm. Cost	?	$\tilde{O}(sk^4)$	$\tilde{O}(nsk^3)$

Overview

1. Problem & Context
2. Our Protocol
3. Efficiency
4. Conclusion

Protocol Overview

1. Sample p, q as integer shares
2. Compute $N = p \cdot q$
3. Biprimality test on N
4. Retroactive Consistency check

Shared by most works

Analogous to [FLOP18] “Proof of Honesty”

Protocol Overview

1. Sample p, q as integer shares
2. Compute $N = p \cdot q$
3. Biprimality test on N
4. Retroactive Consistency check

Chinese Remainder Thm

Thm: If $m_1, m_2, \ldots, m_\ell$ are unique primes

then $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell} \simeq \mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

Chinese Remainder Thm

Thm: If $m_1, m_2, \ldots, m_\ell$ are unique primes

then $\underbrace{\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell}}_{\text{Product of Fields}} \simeq \underbrace{\mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}}_{\text{Ring Modulo Product}}$

"CRT Form" **"Standard Form"**

$$\vec{x} \mapsto p \Leftrightarrow x_j \equiv p (mod\ m_j)$$

Constructive Sampling

1. Sample $\vec{x}$
2. Check $x_j \neq 0 \; \forall j$
3. Construct p from $\vec{x}$
4. Interpret p as integer

Legend: $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell}$ $\mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

Distributed Sampling

Alice	Bob
1. Sample $\vec{x}$	1. Sample $\vec{y}$
2. Check $x_j \neq 0 \; \forall j$	2. Check $y_j \neq 0 \; \forall j$
3. Construct p_A from $\vec{x}$	3. Construct p_B from $\vec{y}$
4. Interpret p_A as integer	4. Interpret p_B as integer

Legend: $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell}$ $\mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

Distributed Sampling

Alice	Bob
1. Sample $\vec{x}$	1. Sample $\vec{y}$
2. Jointly check $x_j + y_j \not\equiv 0 (\mathrm{mod}\, m_j) \quad \forall j$	
3. Construct p_A from $\vec{x}$	3. Construct p_B from $\vec{y}$
4. Interpret p_A as integer	4. Interpret p_B as integer

$$p = p_\mathrm{A} + p_\mathrm{B} \not\equiv 0 (\mathrm{mod}\, m_j)$$

Legend: $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell} \quad \mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

Distributed Sampling

Alice	Bob
1. Sample $\vec{u}$	1. Sample $\vec{v}$
2. Jointly check $u_j + v_j \not\equiv 0 (\mathrm{mod}\, m_j) \quad \forall j$	
3. Construct q_A from $\vec{u}$	3. Construct q_B from $\vec{v}$
4. Interpret q_A as integer	4. Interpret q_B as integer

$$q = q_\mathrm{A} + q_\mathrm{B} \not\equiv 0 (\mathrm{mod}\, m_j)$$

Legend: $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell} \quad \mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

Protocol Overview

1. Sample p, q as integer shares

2. Compute $N = p \cdot q$

3. Biprimality test on N

4. Retroactive Consistency check

Multiparty Mul

Protocols for Securely Computing

$$N = p \cdot q$$

Slow!

1. OT Based:
   $\Omega(k^2)$ Comm. $\Omega(k^2)$ Comp.
2. AHE (Paillier) Based:
   $\Omega(k)$ Comm. $\Omega(k^2 \log k)$ Comp.

with recent $O(n \log n)$ multiplication techniques

$\Omega(k^3)$ realistically

where $|p| = |q| = k$

CRT Multiplication

If $\vec{x} \mapsto p$ and $\vec{u} \mapsto q$

then $\vec{x} \odot \vec{u} \mapsto p \cdot q$

Elementwise

Modular Product

$$x_j, u_j \not\equiv 0 \Leftrightarrow p \cdot q \not\equiv 0 (mod\ m_j)$$

Reuse work for “free” zero-testing, merged sample-mult operation

---

Legend: $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell}$ $\mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

CRT Multiplication

If $\vec{x} \mapsto p$ and $\vec{u} \mapsto q$

then $\vec{x} \odot \vec{u} \mapsto p \cdot q$

Elementwise Modular Product

Modular Product

Complexity in $O(k \log k) \subset o(k^2)$

If $k = |m_1 \cdot \ldots \cdot m_\ell|$

then $|\max \vec{m}| \in O(\log k)$

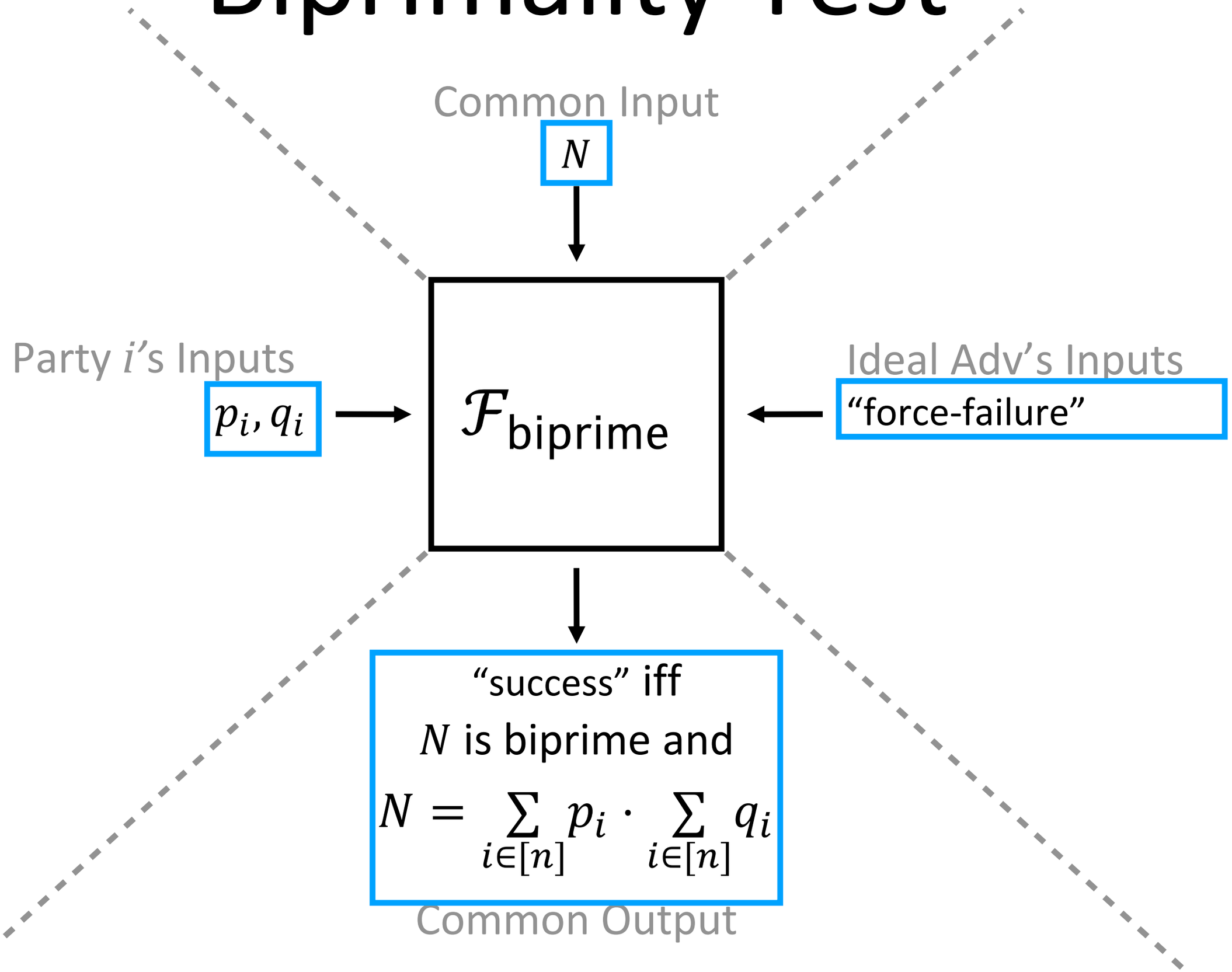
and $\ell \in O(k / \log k)$

Legend: $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_\ell}$ $\mathbb{Z}_{m_1 \cdot \ldots \cdot m_\ell}$

Protocol Overview

1. Sample p, q as integer shares
2. Compute $N = p \cdot q$
3. Biprimality test on N
4. Retroactive Consistency check

Biprimality Test



Protocol Overview

1. Sample p, q as integer shares
2. Compute $N = p \cdot q$
3. Biprimality test on N
4. Retroactive Consistency check

Consistency Check

$$f(N, \vec{x}, \vec{y}, \vec{u}, \vec{v})$$

Predicate

Primarily checking $\vec{x} + \vec{u} \mapsto p$ (resp. q)

Evaluation Strategies:

1. Secure if biprimality test succeeded
2. Privacy-free if biprimality test failed

   Necessary to prevent DoS

Overview

1. Problem & Context
2. Our Protocol
3. Efficiency
4. Conclusion

Instantiation

- Multiplication: [Doerner-Kondi-Lee-shelat-18] using OT extension with Silent OT [Boyle-Couteau-Gilboa-Ishai-Kohl-Rindal-Scholl-19]
- Consistency check: Authenticated garbling [Wang-Ranellucci-Katz-17]

Efficiency

- Failed instance: $O(nsk)$
- Successful instance: $\tilde{O}(n^2sk^2)$
- Expected iterations: $O(k^2/\log^2 k)$
- Total cost: $O(nsk^3/\log^2 k)$
- Concretely, mult is no longer the bottleneck

Overview

1. Problem & Context
2. Our Protocol
3. Efficiency
4. Conclusion

Summary

1. CRT representation gives us “free” sampling and faster mult
2. Consistency check only requires an algebraic predicate
3. Mult no longer the bottleneck, room to optimize consistency check

Additional Results

1. Constant Rounds
2. Concrete Cost Analysis
3. Perfect SH Sec in Hybrid Model
4. Modular Security Analysis

Thanks!

Multiparty Generation of an RSA Modulus

Megan Chen Ran Cohen Jack Doerner
Yashvanth Kondi Eysa Lee Schuyler Rosefield
abhi shelat

Northeastern University

http://ia.cr/2020/370