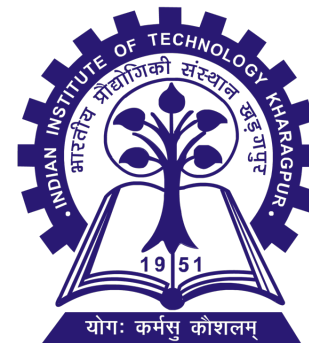


Improved Distributed RSA Key Generation Using the Miller-Rabin Test

Jakob Burkhardt, Ivan Damgård, Tore Kasper Frederiksen, Satrajit Ghosh, Claudio Orlandi



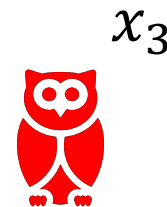
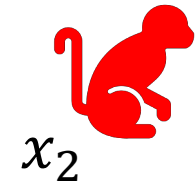
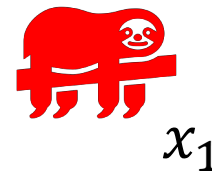
Setting

n parties

$t < n$ passive corruptions

~~Dissemination~~ Dishonest Majority

$$[x]_A = \sum_i x_i \pmod A = x$$

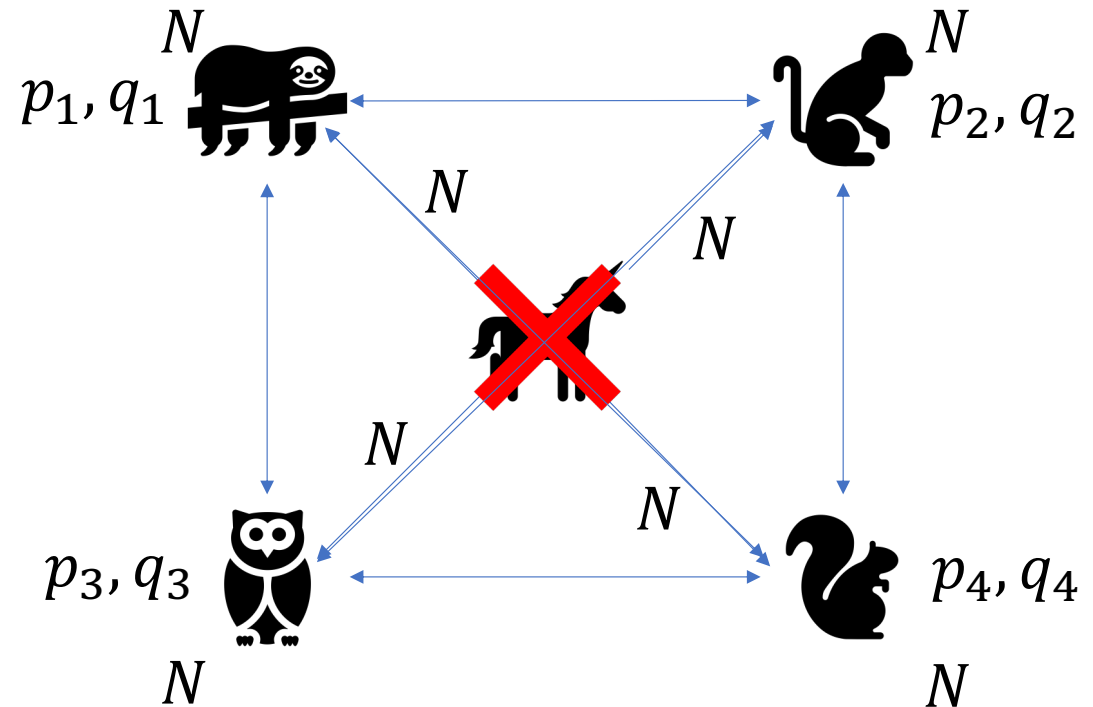


Improved Distributed RSA Key Generation Using the Miller-Rabin Test

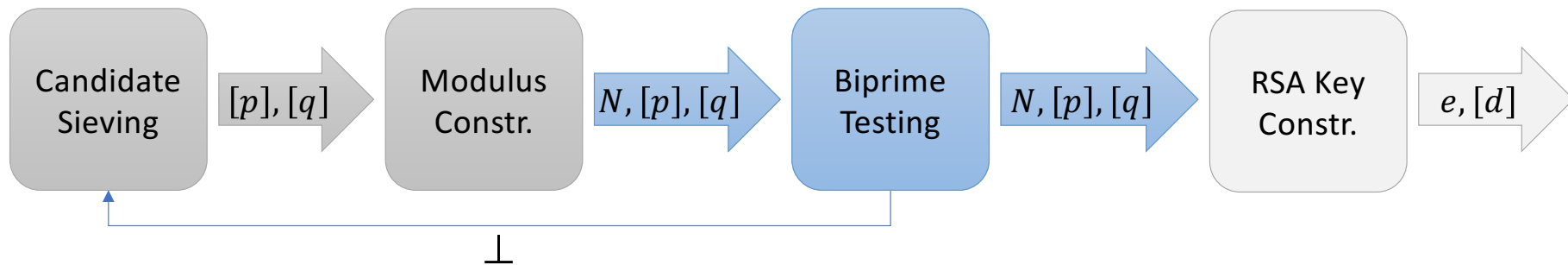
How to securely generate $N = pq$?

Goal:

- Interactive protocol
- Generate $N = pq$ (biprime)
- Keep $[p]$ and $[q]$ secret shared



RSA Key Generation



Improved **Distributed RSA Key Generation** Using the Miller-Rabin Test

Distributed Biprimality Testing

→ [ACS02]

Miller-Rabin [M76],[R80]
Exp. in **secret domain**

[DM10] ($n = 3$ & $t = 1$)

Exponentiation **modulo N**
Good **average case** error bounds
Set-up assumption

→ [BF97]

$O(s)$ repetitions

[FMY98],[PS98],
[HMRT12]

[G99],[FLOP18],
[CHI+21],[CDK+22],
[GMRT21]

↓
Our work ($t < n$)

New distributed divisibility test
Statistically secure
1 iteration
No Set-up assumption

Miller-Rabin Test variant

Goal: Given $p|N$ with $p \equiv 3 \pmod{4}$, test if p is a prime

1. $v \in_R \mathbb{Z}_N$

2. $p \mid (v^{(p-1)/2} \bmod N) - 1$ OR $p \mid (v^{(p-1)/2} \bmod N) + 1$

Divisibility Test \Rightarrow Biprimality Test

Miller-Rabin Test variant in MPC

Goal: Given $p|N$ with $p \equiv 3 \pmod{4}$, test if p is a prime

1. $v \in_R \mathbb{Z}_N$

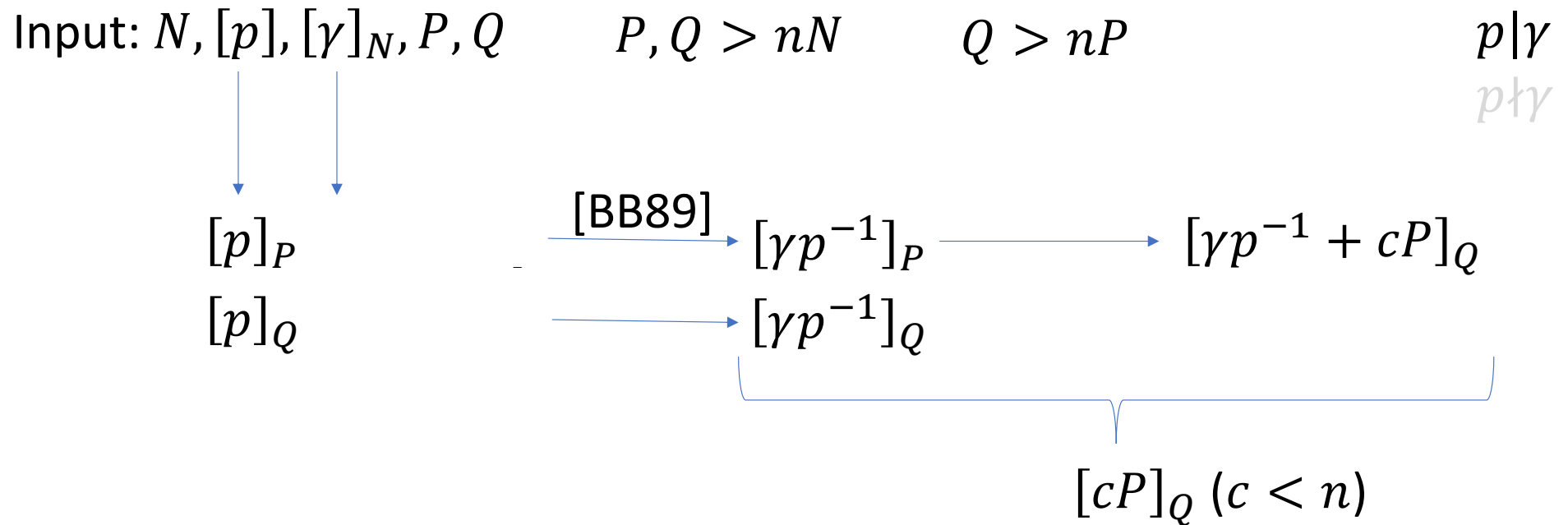
2. $p \mid (v^{(p-1)/2} \bmod N) - 1$

$$[p] \mid (v^{([p]-1)/2} \bmod N) - 1$$

$$\langle v^{(p-1)/2} \rangle_N - 1$$

$$[v^{(p-1)/2} - 1]_N = [\gamma]_N \quad ?$$

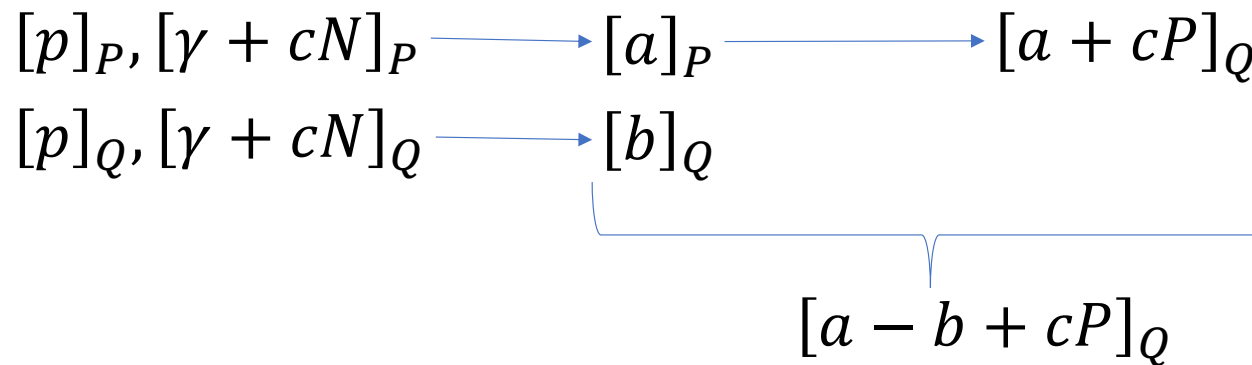
Distributed Divisibility Test



Distributed Divisibility Test

Input: $N, [p], [\gamma]_N, P, Q$ $P, Q > nN$ $Q > nP$

$p|\gamma$
 $p \nmid \gamma$



Distributed Divisibility Test

Input: $N, [p], [\gamma]_N, P, Q$

$$\prod_{i < n} ([z]_Q - iP) \stackrel{?}{=} 0 \quad (\text{[BB89] to achieve } O(1) \text{ rounds})$$

$$[z]_Q = \begin{matrix} [a - b + cP]_Q \\ [cP]_Q \quad (c < n) \end{matrix} \quad \text{How to distinguish?}$$

Miller-Rabin Test variant

Goal: Given $p|N$ with $p \equiv 3 \pmod{4}$, test if p is a prime

1. $v \in_R \mathbb{Z}_N$

2. $p \mid \underbrace{\left(v^{(p-1)/2} \bmod N \right) - 1}_{\gamma} \text{ OR } p \mid \left(v^{(p-1)/2} \bmod N \right) + 1$

Divisibility Test \Rightarrow Biprimality Test

Distributed Divisibility Test

$$\prod_{i < n} ([z^+]_q - iP) \prod_{i < n} ([z^-]_q - iP) \stackrel{?}{=} 0$$

Security

Our protocol is passively and statistically secure, given secure multiplication as a black box. Different implementations of multiplication lead to different concrete instantiations:

Approach	Corruptions
Beaver Triples	$t < n$
Shamir secret sharing (+ conversion)	$t < n/2$
Replicated secret sharing (+ conversion)	$t < n/2$
OT-based [G99]	$t < n$

Active Security: from [DOS18] compiler, details later.

Improved Distributed RSA Key Generation Using the Miller-Rabin Test

Experimental Evaluation (Throughput)

Unit: Biprimality tests per second

Honest Majority

$ N $	Ours	[BF97]
2048	124	2.69
3072	52.0	0.915
4096	28.7	0.405

Dishonest Majority

$ N $	Ours	[BF97]
2048	0.312	2.04
3072	0.187	0.775
4096	0.122	0.361

Setup:

- Java (openJDK 19)
- Exponentiations in C through JNI
- Micro-benchmarking (JMH)
- AWS t3.micro
(Intel Xeon 8000, 1 GB RAM)
- Ubuntu
- One thread
- 1 Gb connection, 10 ms latency

Active Security

Compiler from [DOS18]: each party in passive protocol emulated by several parties, checking each others' work. No zero-knowledge, hence overhead factor 3 or 4.

Passive	Active	Comment
$n, t^2 + t$	n, t	
$n = 3, t = 2$	$n = 3, t = 1$	Compiler+preprocessing
$n = 5, t = 2$	$n = 5, t = 1$	No preprocessing, hence faster
$n = 4, t = 3$	$n = 4, t = 1$	Guaranteed output delivery

Summary

Distributed divisibility test

→ Distributed (bi)primality test

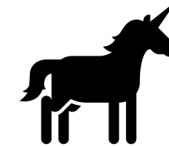
→ Distributed RSA key generation

Statistically secure

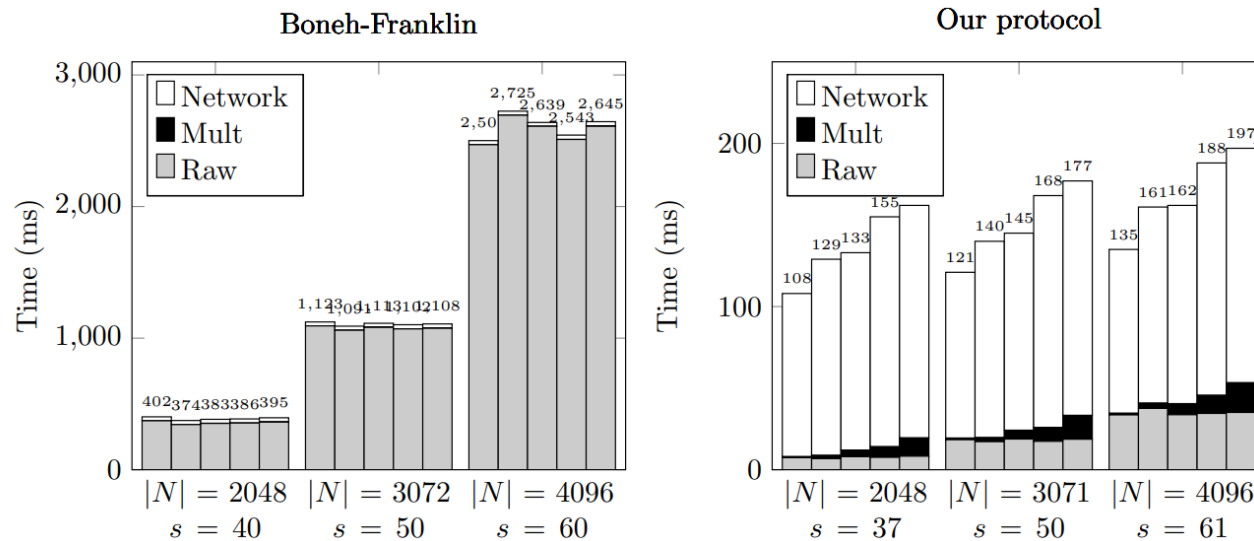
→ Assuming secure multiplication

Take-home message: if you have a small number of parties and can assume honest majority, then read our paper!

Thank you!



Experimental Evaluation (Total time)



Setup:

- Java (openJDK 19)
- Exponentiations in C through JNI
- Micro-benchmarking (JMH)
- AWS t3.micro
(Intel Xeon 8000, 1 GB RAM)
- Ubuntu
- One thread
- 1 Gb connection, 10 ms latency

Fig. 3: Total computation time of our protocol and the Boneh-Franklin protocol for 3, 5, 7, 9 and 11 parties in the honest majority model assuming 1 Gigabit network with 10 ms latency. The left bar in a pair represents 3 parties then 5, 7, and 9 parties (using Shamir secret sharing).

Experimental Evaluation (Computation)

$ N $	Latency		Throughput		Price	
	Ours	[BF97]	Ours	[BF97]	Ours	[BF97]
Shamir						
2048	108	402	124	2.69	58.2	174
3072	121	1,124	52.0	0.915	89.3	439
4096	135	2,501	28.7	0.405	122	917
Replicated secret sharing						
2048	111	402	97.6	2.69	104	180
3072	123	1,124	43.2	0.915	158	444
4096	142	2,501	24.4	0.405	213	911
Gilboa						
2048	3,440	523	0.312	2.04	15,100	727
3072	5,550	1,326	0.187	0.775	33,200	1,661
4096	8,550	2,814	0.122	0.361	58,400	3,062

Table 7: Computation time for 3 parties on a 1 gigabit network with 10 ms latency for different multiplication protocols. s is 37, 50, 61 for $|N| = 2048, 3072, 4096$ respective for our protocol and 40, 50, 60 for Boneh-Franklin.

Setup:

- Java (openJDK 19)
- Exponentiations in C through JNI
- Micro-benchmarking (JMH)
- AWS t3.micro
(Intel Xeon 8000, 1 GB RAM)
- Ubuntu
- One thread
- 1 Gb connection, 10 ms latency