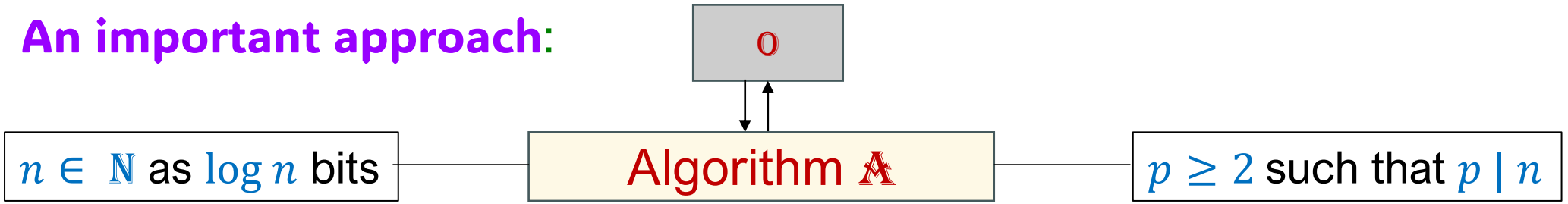
A computational perspective on Carmichael numbers

Joint work with Alan Sikarov and Ilya Volkovich

Nikhil Gupta
Boston College

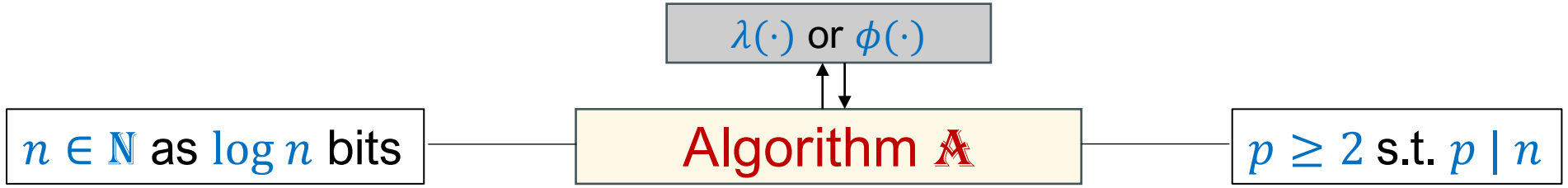
Integer Factoring

- **Gold standard**: The algorithm $\mathbb{A}$ runs in $\text{poly}(\log n)$ time. Looks unachievable!
- **Consequences**: RSA will break and modern cryptography will collapse!
- **An important approach**:



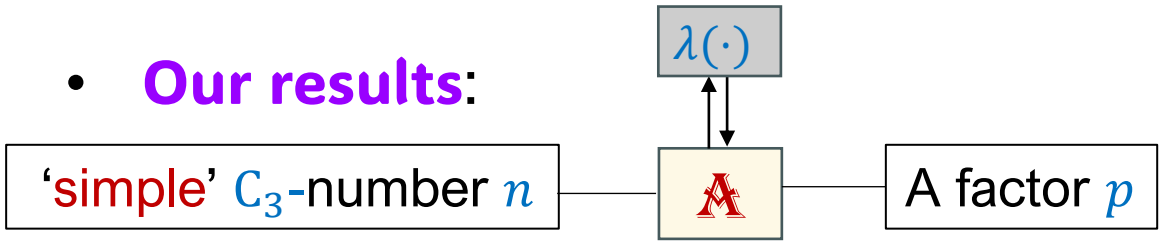
The algorithm $\mathbb{A}$ with oracle access to $\mathbb{A}$ is randomized and runs in $\text{poly}(\log n)$ time.

- $\mathbb{O}$ either computes Euler's Totient function ($\phi(\cdot)$) or Carmichael's Lambda function ($\lambda(\cdot)$)



Towards derandomizing $\mathbb{A}$

- **Status**: Known when $n = pq$ but open if $n = pqr$. Solved if $n = pqr$ is 'size-bounded'.
- **Our work**: n is 'Carmichael' and has 3 prime factors. We call them C_3-numbers.
- **Carmichael number**: A composite n s.t. for every $a < n, \gcd(n, d) = 1, a^n \equiv a \bmod n$.
- **Our results**:



Result 1: $\mathbb{A}$ runs in $\mathrm{poly}(\log n)$ time

Result 2: $\mathbb{A}$ runs in $\mathrm{poly}(\log n)$ time

- **Important tool**: Coppersmith's method to find 'bounded' roots of an $f \in \mathbb{Z}[x, y]$.