Streaming Cryptography

Guang Yang

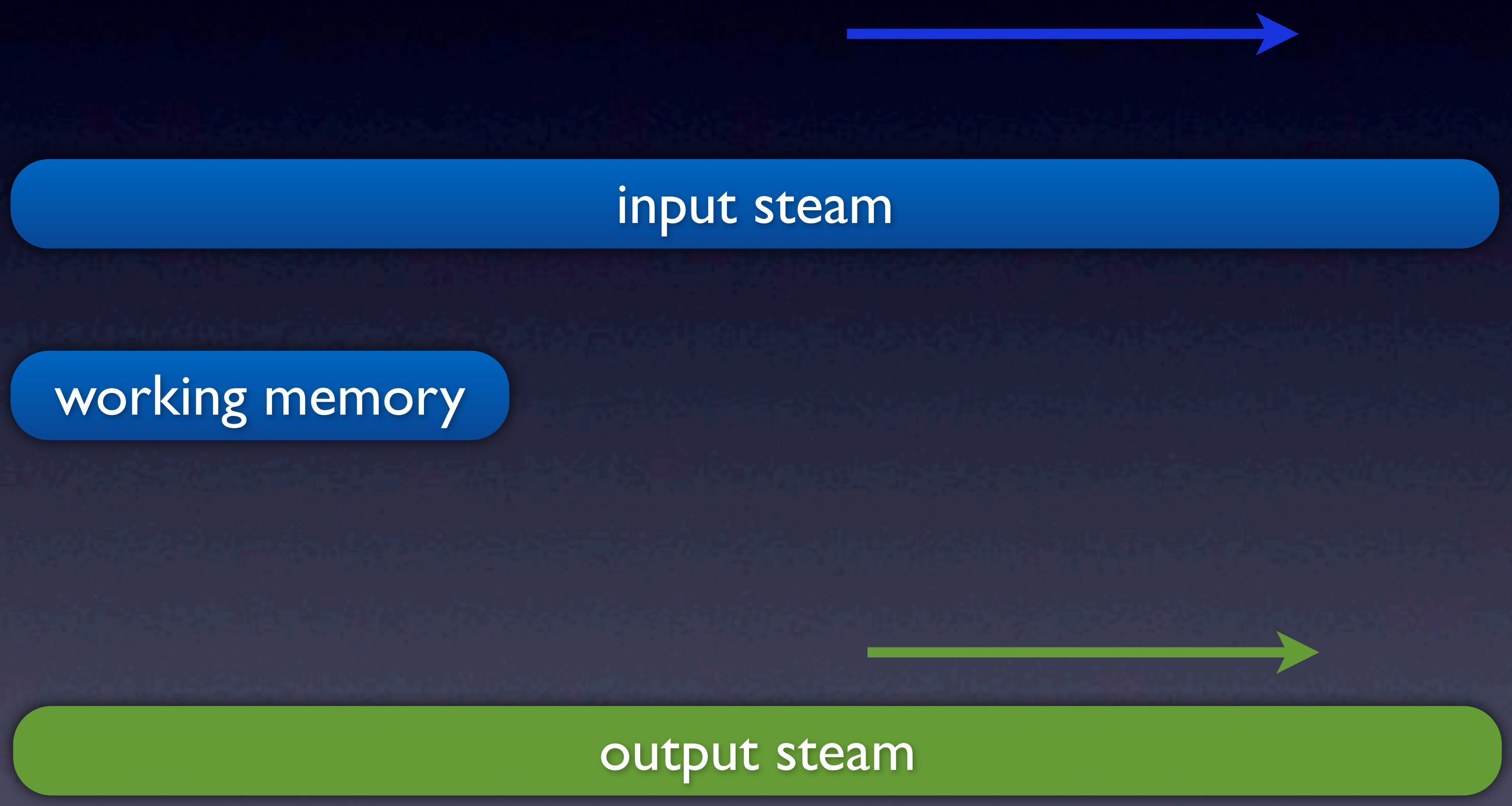
ITCS, Tsinghua University

Aug, 2012

joint work with Periklis Papakonstantinou

Streaming Cryptography

Streaming Model (simplest, aka “online model”)



input steam

working memory

output steam

single passes

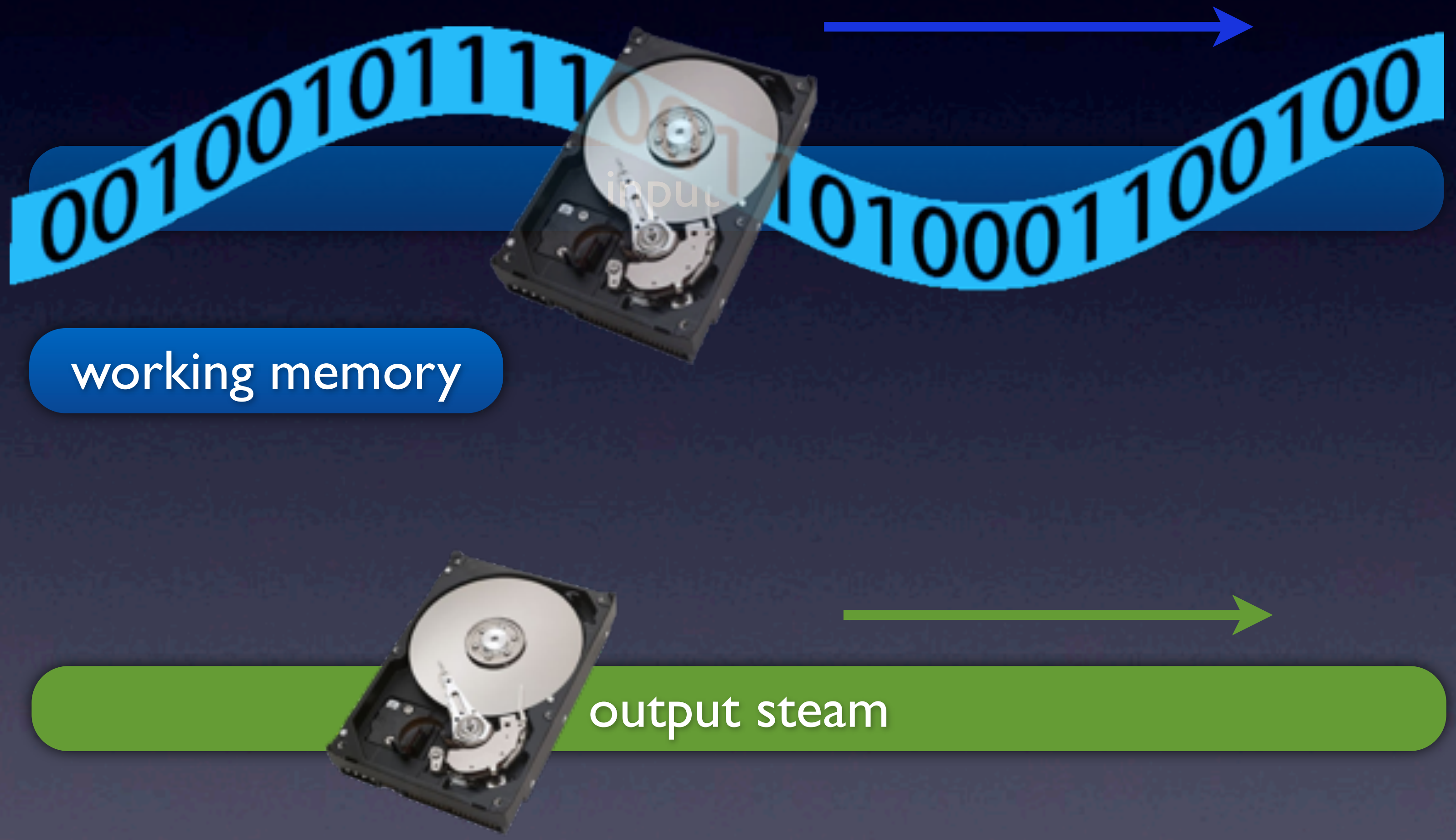
single streams

small working memory

output streams

Streaming Cryptography

Streaming Model (simplest, aka “online model”)



single passes

single streams

small working memory

output streams

Streaming Cryptography

Streaming Model (generalized)

small total #passes

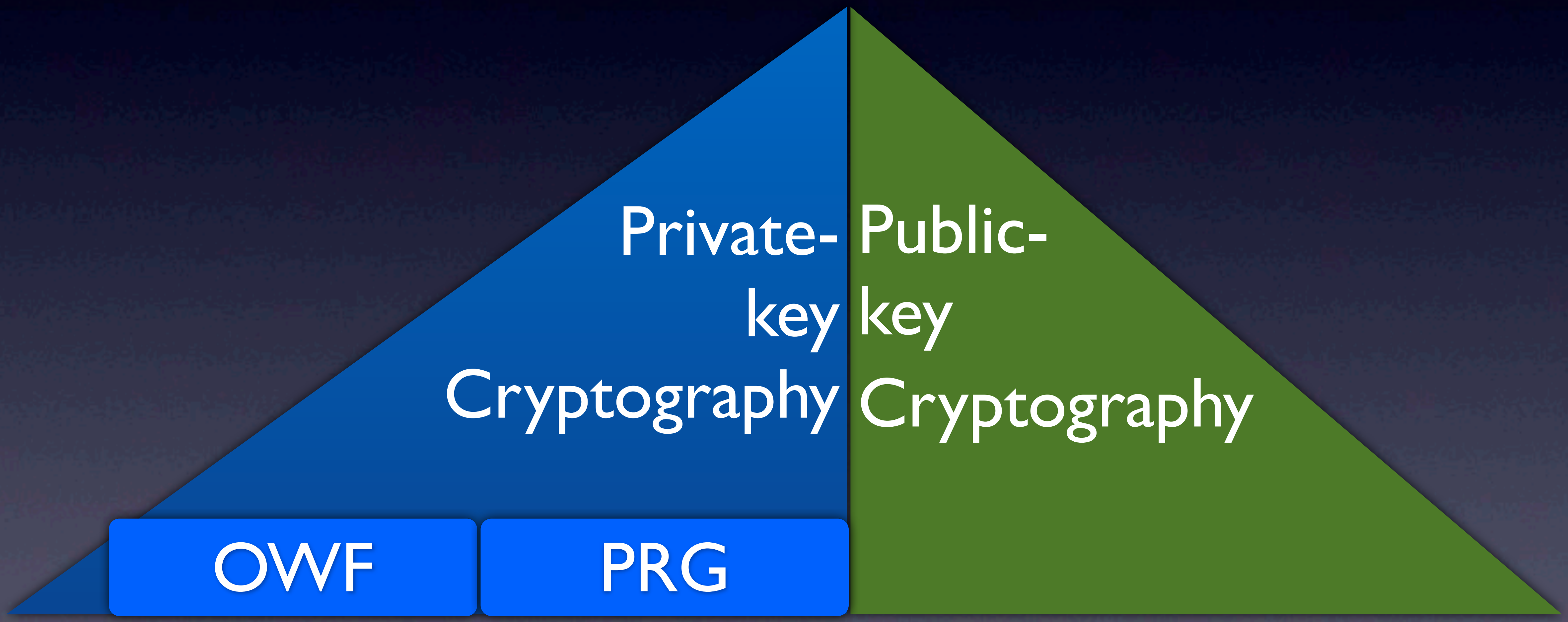
constant #streams

working memory

small working memory

The **standard READ-WRITE STREAMING model** in the Journal ACM work of Grohe and Sweichartd.

Streaming Cryptography


Private-key Cryptography

Public-key Cryptography

OWF

PRG

Streaming Cryptography

- Can we
  do cryptography
  in the streaming model?

Streaming Cryptography

- Can we compute cryptographic primitives (OWF/PRG) in the streaming model?

constant #passes and #streams
O(log n) working memory

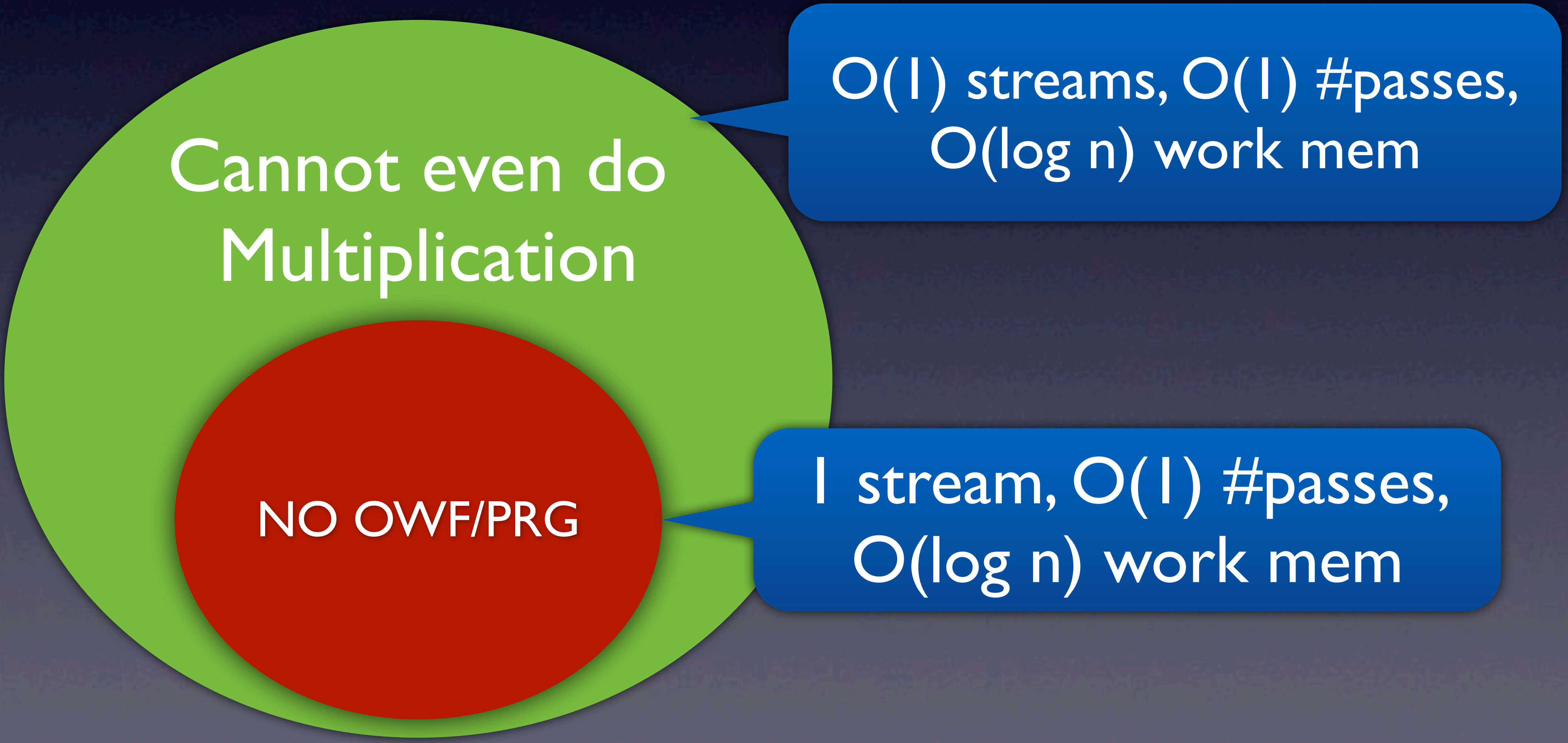
How weak is this setting?

Impossibility

NO OWF/PRG

1 stream, O(1) #passes, O(log n) work mem

Impossibility



Cannot even do Multiplication

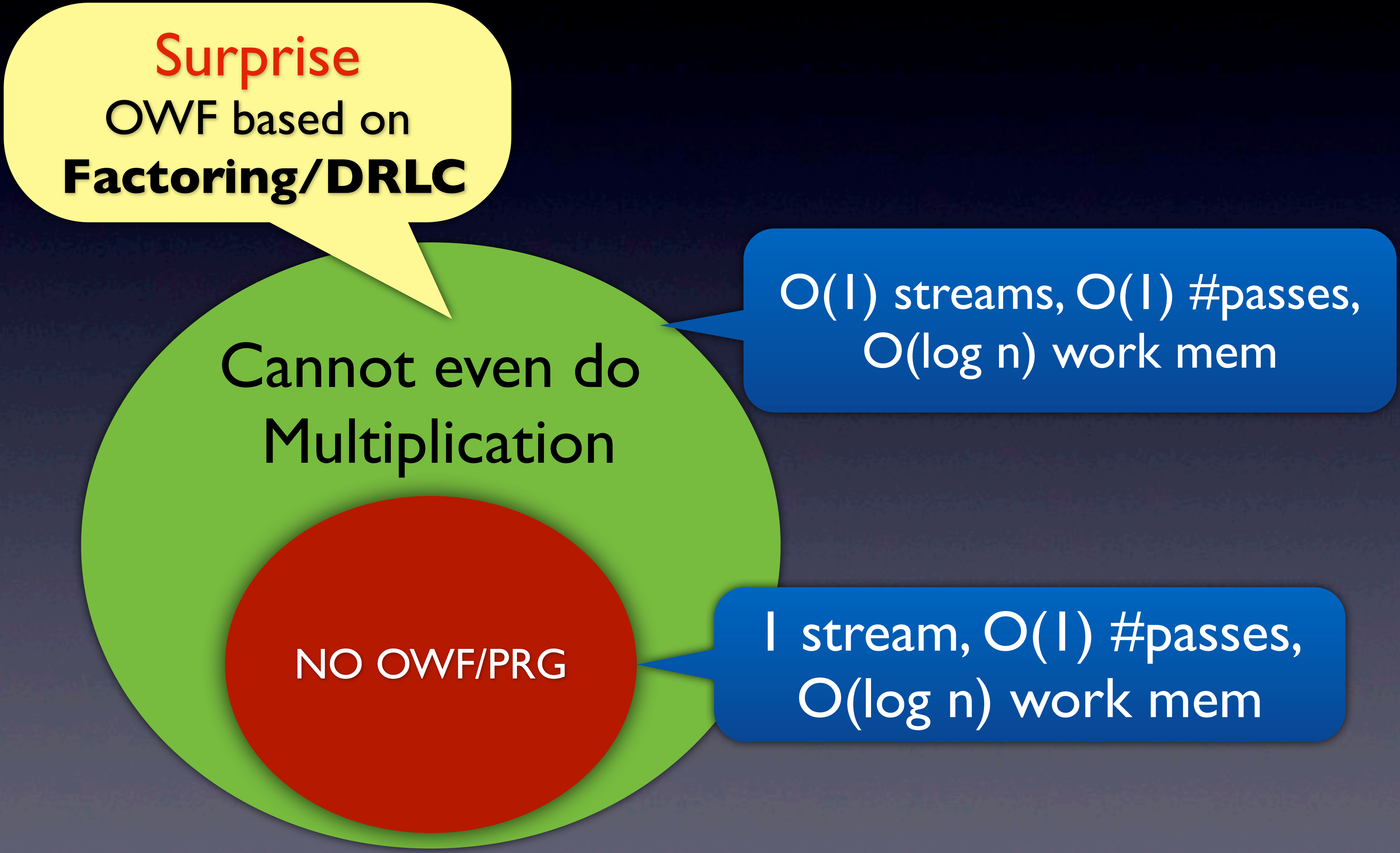
NO OWF/PRG

O(1) streams, O(1) #passes, O(log n) work mem

1 stream, O(1) #passes, O(log n) work mem

The Surprise!

Possibility



Surprise

OWF based on **Factoring/DRLC**

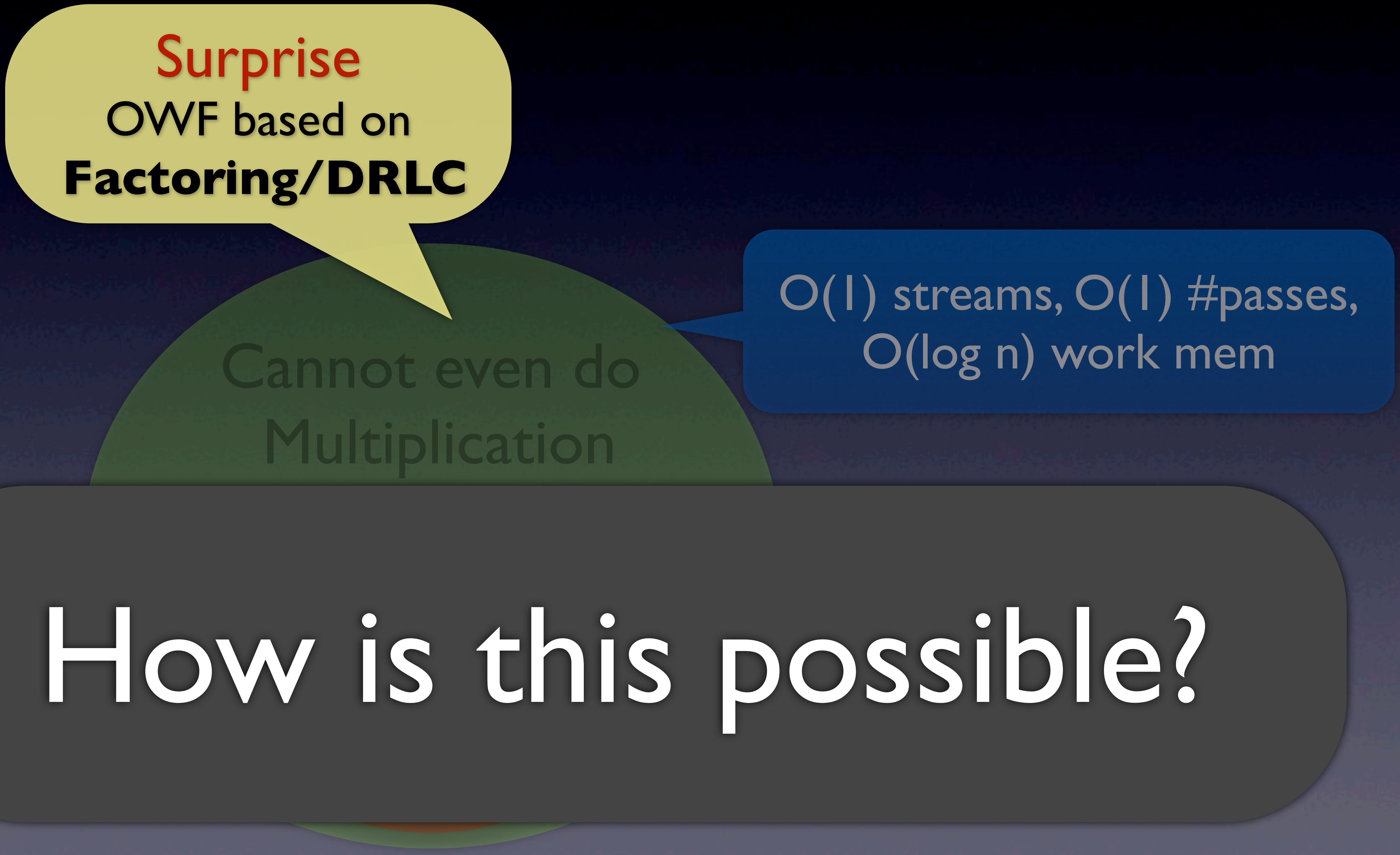
Cannot even do Multiplication

NO OWF/PRG

O(1) streams, O(1) #passes, O(log n) work mem

1 stream, O(1) #passes, O(log n) work mem

Possibility



Surprise
OWF based on
Factoring/DRLC

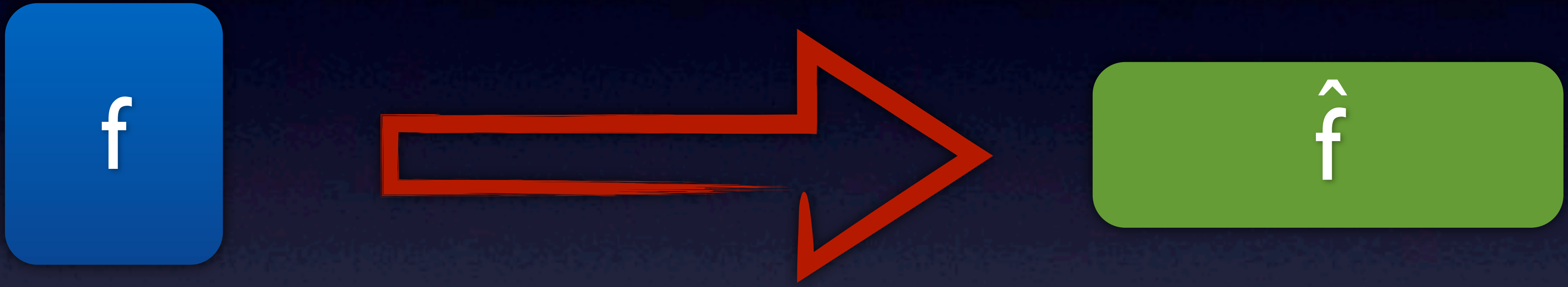
Cannot even do
Multiplication

O(1) streams, O(1) #passes,
O(log n) work mem

How is this possible?

VISA
4000 0012 3456 7899
4000
VALID FROM▸ 00/00
EXPIRES END ▸ 00/00 V
CARDHOLDER NAME

Idea



f

$\hat{f}$

Barrington’s Theorem

+

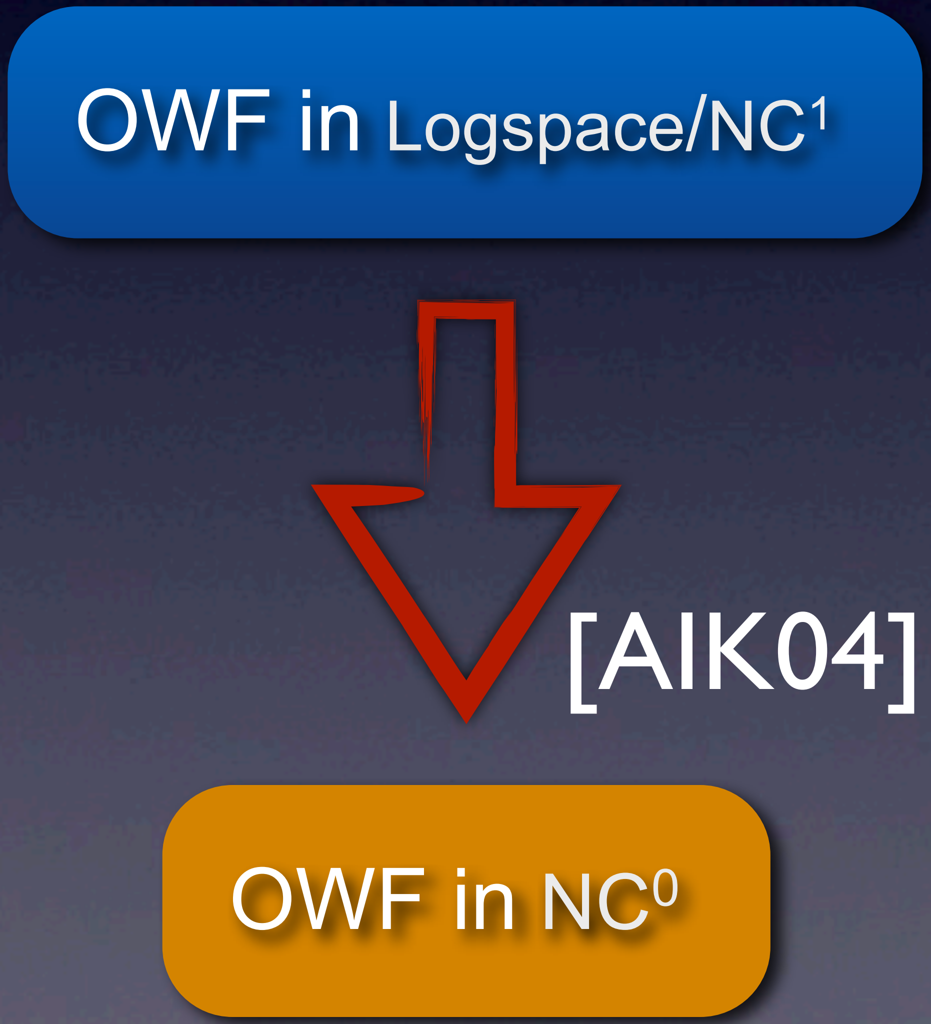
Randomized Encoding

decompose the result into its computation process

hide extra information

NON-BLACK-BOX !

Previous Result



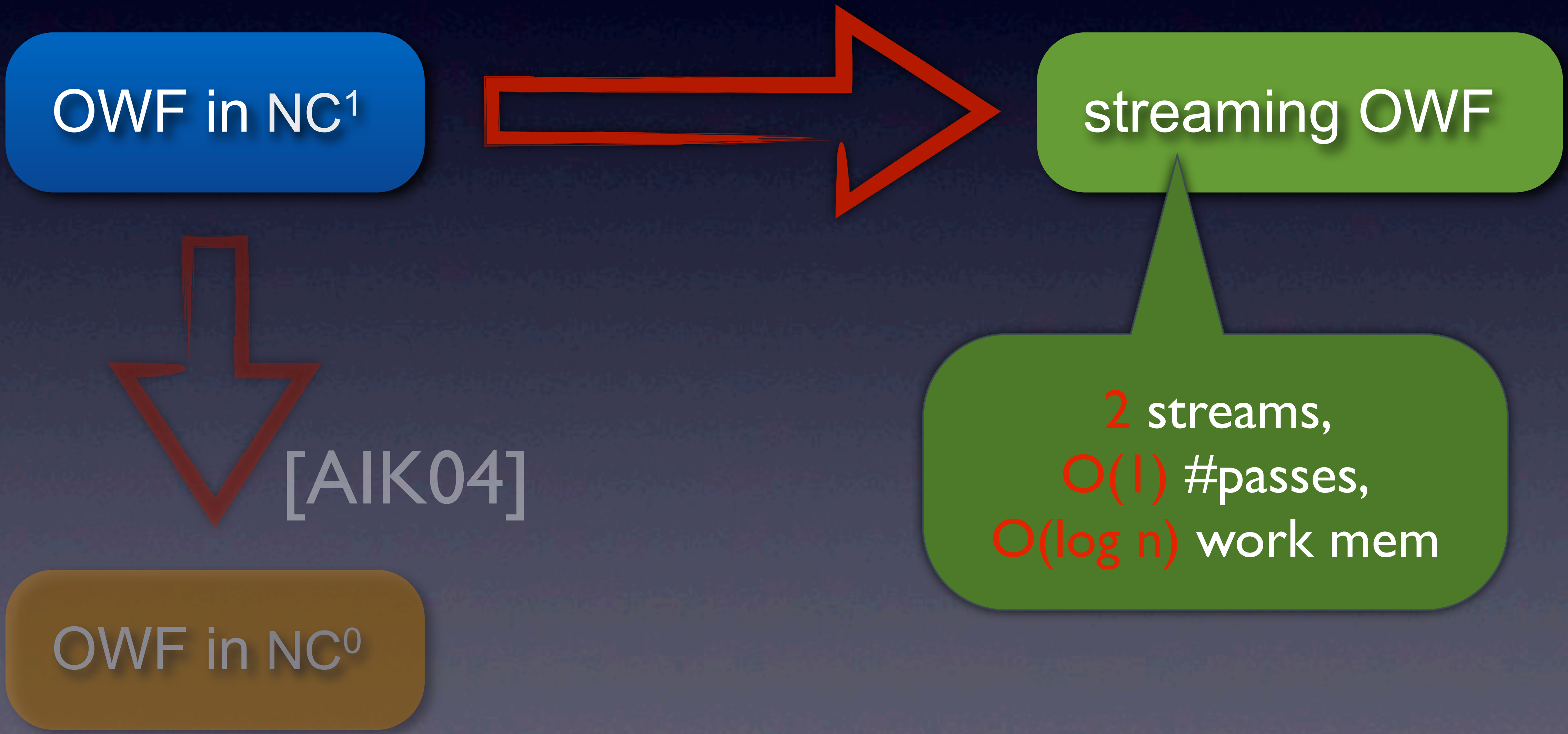
OWF in Logspace/NC^1

[AIK04]

OWF in NC^0

Fact: $NC^0 \subsetneq NC^1 \subsetneq$ Logspace

Our Result



OWF in NC^1

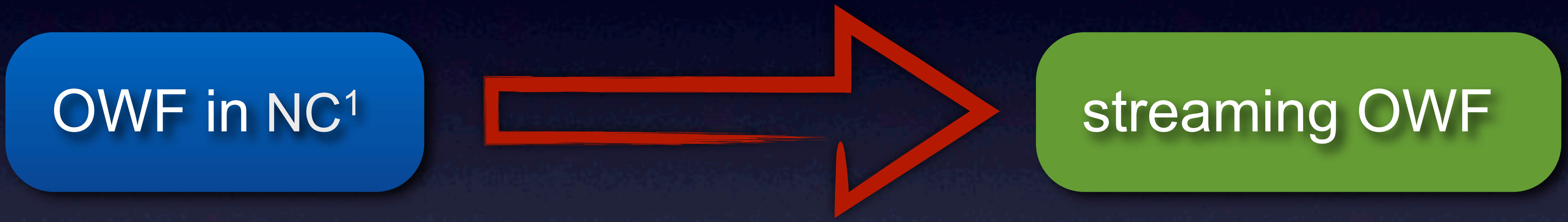
streaming OWF

2 streams,
O(1) #passes,
O(log n) work mem

[AIK04]

OWF in NC^0

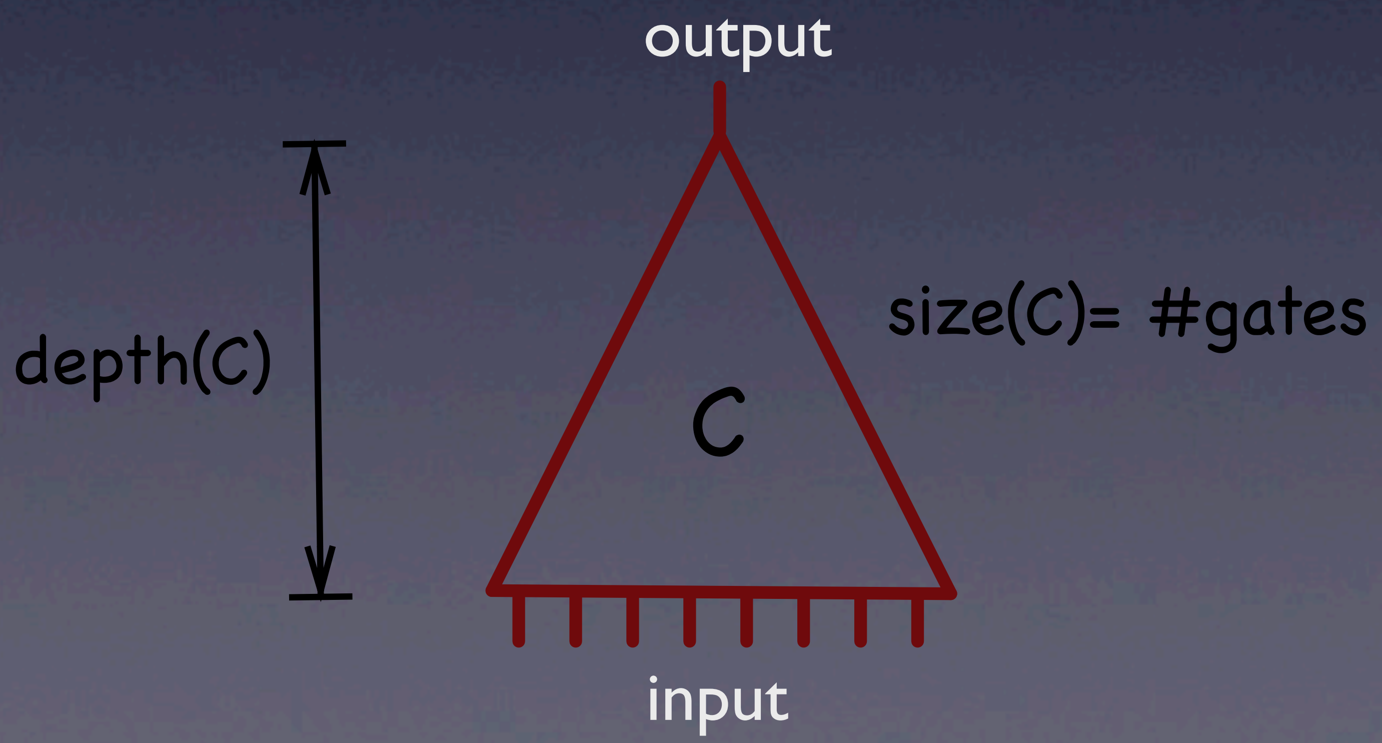
Our Result

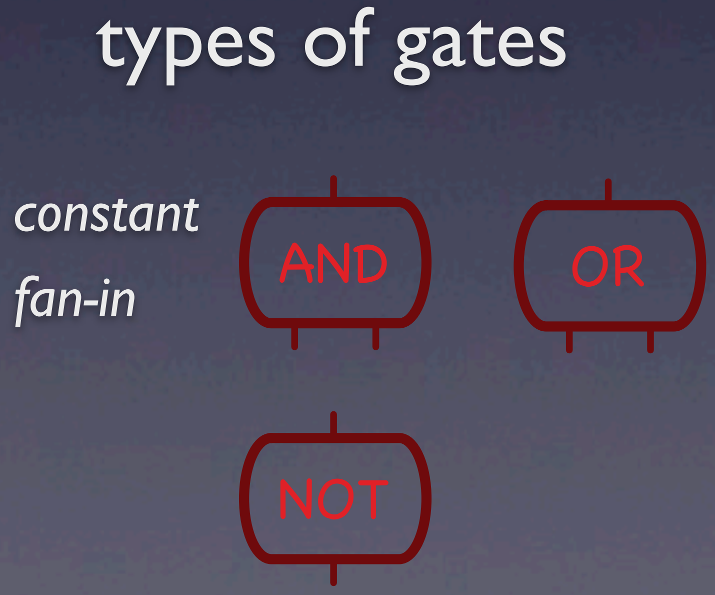


OWF in NC^1

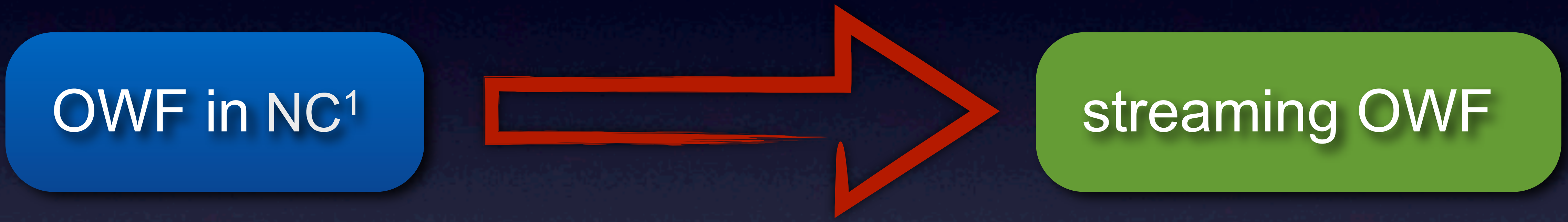
streaming OWF

NC^1: poly(n) size and O(log n) depth.





Our Result



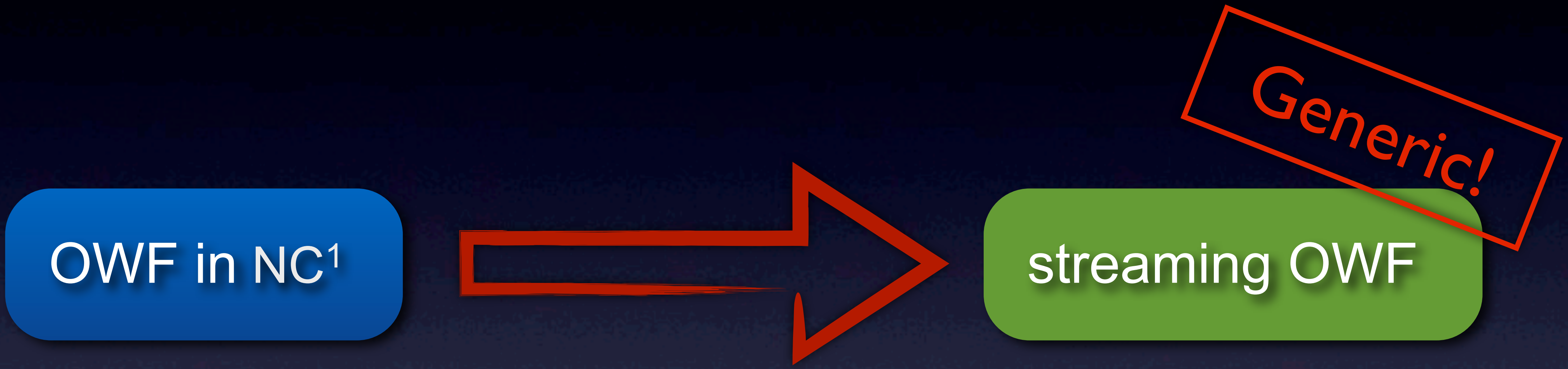
OWF in NC^1

streaming OWF

NC^1: poly(n) size and O(log n) depth.

Hardness assumptions in NC^1:
Factoring, Decoding Random Linear Code, Discrete Logarithm, Lattice assumptions, etc.

Our Result



OWF in NC^1

Generic!

streaming OWF

NC^1: poly(n) size and O(log n) depth.

Hardness assumptions in NC^1:
Factoring, Decoding Random Linear Code, Discrete Logarithm, Lattice assumptions, etc.

Other Results

- Streaming OWF => Streaming PRG
- More efficiently streaming OWF from DRLC
- Linear Stretch PRG if DRLC is exp. hard

Ongoing Work

- Apply this technique to all, known, basic Private-Key and Public-Key Crypto systems

Thanks!