Streaming Cryptography

Guang Yang

ITCS, Tsinghua University
Aug, 2012

joint work with Periklis Papakonstantinou
Streaming Cryptography

Streaming Model (simplest, aka “online model”)

- Input stream
- Working memory
- Output stream

- 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

Working memory

Input streams

Output streams
The standard READ-WRITE STREAMING model in the Journal ACM work of Grohe and Sweichartd.
Streaming Cryptography

- Private-key Cryptography
  - OWF
  - PRG
- Public-key Cryptography
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

I 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

I 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

I 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?
Idea

Barrington's Theorem + Randomized Encoding

decompose the result into its computation process
hide extra information

NON-BLACK-BOX!
Previous Result

Fact: $\text{NC}^0 \subsetneq \text{NC}^1 \subsetneq \text{Logspace}$

[AIK04]

OWF in Logspace/NC$^1$

OWF in NC$^0$
Our Result

OWF in NC$^1$ → streaming OWF

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

OWF in NC$^0$ → [AIK04]
**Our Result**

**OWF in NC$^1$**

**streaming OWF**

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

- **types of gates**
  - constant
  - fan-in
  - AND
  - OR
  - NOT

- **depth(C)**

- **size(C) = #gates**

- **C**
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 $\text{NC}^1$

$\text{NC}^1$: poly(n) size and $O(\log n)$ depth.

Hardness assumptions in $\text{NC}^1$:
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!