Scalable and private media consumption with Popcorn

Trinabh Gupta*¶, Natacha Crooks*δ, Whitney Mulhern¶, Srinath Setty†, Lorenzo Alvisi*, and Michael Walfish¶

*The University of Texas at Austin
¶New York University
δMPI-SWS
†Microsoft Research
Why? Because media diet can reveal sensitive information.

Attacks: [NDSS14, ESORICS13, ESORICS12, ...]
Private Information Retrieval (PIR) provably hides requests but ...

- Each request must **touch the entire library**.

- There is a tension between overhead and content protection.

- PIR **assumes fixed-size objects**, but media sizes vary.
Popcorn tailors PIR for media to meet our three requirements.

Its per-request dollar cost is 3.87x times that of a non-private baseline.
Rest of this talk

• Background on PIR.

• Design (tailoring of PIR) and evaluation of Popcorn.
Background on Information-theoretic PIR (ITPIR)

Server 1

(M1 ⊕ M3 ⊕ M5)
⊕ (M1 ⊕ M5)

M3 = (M1 ⊕ M3 ⊕ M5)

Reply = M1 ⊕ M3 ⊕ M5

1, 3, 5

(M1 ⊕ M3 ⊕ M5)

Client

1, 5

No collusion

Server 2

(M1 ⊕ M3 ⊕ M5)

Reply = M1 ⊕ M5

(M1 ⊕ M3 ⊕ M5)

Table:

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table:

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>M4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Given these, how can we build a system that is low cost and compatible with commercial streaming?
Popcorn composes ITPIR and CPIR to get desirable properties from both

![Diagram showing the relationship between Client, ITPIR, Key Library (holding Enc(M1) to Enc(M5)), and Akamai with corresponding Enc(M1) to Enc(M5)].
<table>
<thead>
<tr>
<th>ITPIR</th>
<th>CPIR</th>
<th>Popcorn</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheap operations (XORs)</td>
<td>expensive operations (5-10x ITPIR)</td>
<td>mostly cheap operations</td>
</tr>
<tr>
<td>does not respect controls on content dissemination</td>
<td>respects controls on content dissemination</td>
<td>respects controls on content dissemination</td>
</tr>
<tr>
<td>process entire library per request</td>
<td>process entire library per request</td>
<td>?</td>
</tr>
</tbody>
</table>
Popcorn batches requests to amortize the overhead of ITPIR

Observation: Same I/O work for each request!

Benefits of batching:

- I/O transfers are amortized.
- CPU cycles are reduced as matrix multiplication algorithms exploit cache locality.
Strawman: Group requests that arrive during an epoch

- Client A, Client B, Client C

Time diagram:
- Epoch
- Start handling A, B, C
- Playback position
- Client A’s playback buffer
- Wait for server to form batch
- Client perceived delay = epoch + epsilon
Strawman: Group requests that arrive during an epoch

**Issue**: Hard to get both a small delay and a large batch
Popcorn exploits streaming to form large batches with small startup delay

client

perceived delay (d)
d + t

d + 2t
d + 3t

chunks of a movie

server

• Inspired by pyramid broadcasting [MMCN95]
Other design considerations

• Popcorn must handle variable-sized media objects.
  Response: Change bitrates to make movies of the same size.
Outline

✓ Background on PIR.

✓ Design (tailoring of PIR) of Popcorn.

• Evaluation of Popcorn.
Experiment method

• Baselines:
  - Non-private system (Apache server)
  - State-of-the-art CPIR [XPIR PETS16]
  - State-of-the-art ITPIR [Percy++] modified to support streaming
  - CPIR and ITPIR extended with the strawman batching scheme

• Netflix-like library: 8000 movies, 90 minutes, 4Mbps

• Workload: 10K clients arrive within 90 minutes according to a Poisson process

• Estimate per-request dollar cost using Amazon’s pricing model
  - CPU: $0.0076/hour
  - I/O bandwidth: $0.042/Gbps-hour
  - Network: $0.006/GB
<table>
<thead>
<tr>
<th>System</th>
<th>CPUs</th>
<th>I/O (Gbps)</th>
<th>Network (relative to non-private)</th>
<th>$ relative to non-private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-private</td>
<td>0</td>
<td>0</td>
<td>1x</td>
<td>1x</td>
</tr>
<tr>
<td>CPIR</td>
<td>11.6</td>
<td>64</td>
<td>5x</td>
<td>265x</td>
</tr>
<tr>
<td>ITPIR</td>
<td>3.1</td>
<td>64</td>
<td>2x</td>
<td>256x</td>
</tr>
<tr>
<td>ITPIR++ (delay 15s)</td>
<td>0.65</td>
<td>3</td>
<td>2x</td>
<td>14x</td>
</tr>
<tr>
<td>ITPIR++ (delay 10min)</td>
<td>0.41</td>
<td>0.058</td>
<td>2x</td>
<td>2.5x</td>
</tr>
<tr>
<td>Popcorn (delay 15s)</td>
<td>0.74</td>
<td>0.23</td>
<td>2x</td>
<td>3.87x</td>
</tr>
</tbody>
</table>
Related work

• Improving performance of PIR.
  • Distributing work [FC13, TDSC12], cheaper crypto [PETS16, ESORICS14, ISC10, TKDE13, WEWoRC07], bucketing [DBSec10, PETS10], batching [FC15, JoC04], secure co-processors [PET03, FAST13, NDSS08, IBM Systems Journal01]

• Protecting library content in ITPIR [RANDOM98, S&P07, WPES13]

• Handling variable-sized objects [CCSW14, NDSS13]

• Prior PIR implementations [Percy++, PETS16, CCSW14]

• Video-on-demand [MMCN95]
Take-away points

• It is possible to build a private, backwards compatible, and low-cost media delivery system ...

• ... by tailoring PIR to media delivery.

• The per-request cost in Popcorn is 3.87x that of a non-private baseline.