Efficient Private File Retrieval by Combining ORAM and PIR

Travis Mayberry
Erik-Oliver Blass
Agnes Chan
Hiding Access Patterns

Oblivious RAM
• Communication: High
• Rounds: Multiple
• Client computation: None
• Server computation: None

Private Information Retrieval
• Communication: Low
• Rounds: One
• Client computation: Low
• Server computation: High
Contributions

• We introduce a PIR bucket construction which allows recent ORAM protocols to be merged with PIR
• Consider the notion of an ORAM’s data latency or online data
  – We define latency to be the amount of communication required before the client has full access to the requested data
• Using our bucket construction with the tree-based scheme of Shi et. al., we obtain an ORAM protocol with:
  – The lowest communication overhead of any constant-client-memory Oblivious RAM
  – Optimal data latency
• We evaluate our scheme on Amazon AWS and show that it has very low overall query time and monetary cost per query
Notation

- $n$ : Number of blocks in the ORAM
- $\ell$ : Size of each block in bits
- $k$ : Size of one ciphertext in bits

Helpful sample values:

\[
\begin{align*}
n &= 2^{25} & \text{Database} &= 4 \text{ TB} \\
\ell &= 1 \text{ MB} \\
k &= 2048 \text{ bits}
\end{align*}
\]
Shi et al

- First poly-logarithmic worst-case oblivious RAM
- New tree based construction
- Achieves $O(\ell \cdot \log^3 n)$ communication, with relatively good constants
- Consists of two phases: data access, and eviction
$\mathcal{O}(\log n)$
Private Information Retrieval

• Traditionally very computationally expensive, conjectured that it might never be feasible [SC07]
• Recently advances in homomorphic encryption have lead to practical schemes [MBC13][MG08], especially when $\ell$ is large compared to $n$
$$O(nk + \ell)$$
To change $X_i$ to $X'$, encrypt "delta":

$Y_j = X'_j - X_{i,j}$

<table>
<thead>
<tr>
<th>Query</th>
<th>Server Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ E(0) \times E(Y_1) ]</td>
<td>[ E(0) \times E(Y_1) ]</td>
</tr>
<tr>
<td>[ E(0) \times E(Y_2) ]</td>
<td>[ E(0) \times E(Y_2) ]</td>
</tr>
<tr>
<td>[ E(0) \times E(Y_3) ]</td>
<td>[ E(0) \times E(Y_3) ]</td>
</tr>
<tr>
<td>[ E(0) \times E(Y_4) ]</td>
<td>[ E(0) \times E(Y_4) ]</td>
</tr>
</tbody>
</table>

$$nk \ell$$

$O(nk + \ell)$
### Encrypted Delta

<table>
<thead>
<tr>
<th>E(0)</th>
<th>E(0)</th>
<th>E(0)</th>
<th>E(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E(Y_1)</td>
<td>E(Y_2)</td>
<td>E(Y_3)</td>
<td>E(Y_4)</td>
</tr>
<tr>
<td>E(0)</td>
<td>E(0)</td>
<td>E(0)</td>
<td>E(0)</td>
</tr>
<tr>
<td>E(0)</td>
<td>E(0)</td>
<td>E(0)</td>
<td>E(0)</td>
</tr>
</tbody>
</table>

### Encrypted Database

<table>
<thead>
<tr>
<th>E(X_{1,1})</th>
<th>E(X_{1,2})</th>
<th>E(X_{1,3})</th>
<th>E(X_{1,4})</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>E(X_{2,1})</td>
<td>E(X_{2,2})</td>
<td>E(X_{2,3})</td>
</tr>
<tr>
<td>+</td>
<td>E(X_{3,1})</td>
<td>E(X_{3,2})</td>
<td>E(X_{3,3})</td>
</tr>
<tr>
<td>+</td>
<td>E(X_{4,1})</td>
<td>E(X_{4,2})</td>
<td>E(X_{4,3})</td>
</tr>
</tbody>
</table>

\[ E(0) \quad E(0) \quad E(0) \quad E(0) \]
Bucket ORAM

$O(\log n)$
\[ O(\ell \cdot \log n) \quad \Rightarrow \quad O(\log n \cdot k + \ell) \]
PIR Bucket

• Read blocks using linear PIR
• Write blocks using linear PIR-Writing
• Requires only additively homomorphic encryption!
What does this give us?

• Better asymptotic communication
  – Old: $O(\ell \cdot \log^3 n)$
  – New: $O(k \cdot \log^3 n + \ell \cdot \log^2 n)$

<table>
<thead>
<tr>
<th></th>
<th>Worst-Case</th>
<th>Practical Worst-Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shi et al</td>
<td>$O(l \cdot \log^3(N))$</td>
<td>$O(l \cdot \log^2(N))$</td>
</tr>
<tr>
<td>Kushilevitz</td>
<td>$O\left(\frac{l \cdot \log^2(N)}{\log \log(N)}\right)$</td>
<td>$O(l \cdot \log^3(N))$</td>
</tr>
<tr>
<td>Path-PIR Additive</td>
<td>$O(k \cdot \log^3(N) + l \cdot \log^2(N))$</td>
<td>$O(l \cdot \log(N))$</td>
</tr>
<tr>
<td>Path-PIR FHE</td>
<td>$O(k \cdot \log(N) + l \cdot \log(N))$</td>
<td>$O(k + l)$</td>
</tr>
<tr>
<td>Optimal</td>
<td>$O(\log(N) + l)$</td>
<td>$O(\log(N) + l)$</td>
</tr>
</tbody>
</table>

Also interesting: good latency!
1) Client requests to read block 5
1) Client requests to read block 5
2) Naïve way: use PIR to retrieve 1\textsuperscript{st} element of each bucket

\[ O((\ell + k) \cdot \log n) \]
1) Client requests to read block 5
2) Naïve way: use PIR to retrieve 1\textsuperscript{st} element of each bucket
3) Use PIR again to retrieve 3\textsuperscript{rd} element of previous results

\[ O(k \cdot \log n + \ell) \]

This is optimal!
What good is that?

• Latency represents how responsive the ORAM is to client interactions
  – If most of the communication happens in the background, after the client receives their data, it is much more acceptable in real world scenarios
• Also allows the client to take advantage of interesting network asymmetries…
Cell network data is expensive 😞

WiFi Data is cheap 😊

Defer eviction while you are out

Complete “bookkeeping” when you get home
<table>
<thead>
<tr>
<th></th>
<th>Latency</th>
<th>Worst-Case</th>
<th>Practical Worst-Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shi et al</td>
<td>$O(l \cdot \log^2(N))$</td>
<td>$O(l \cdot \log^3(N))$</td>
<td>$O(l \cdot \log^2(N))$</td>
</tr>
<tr>
<td>Kushilevitz</td>
<td>$O(\frac{l \cdot \log^2(N)}{\log \log(N)})$</td>
<td>$O(\frac{l \cdot \log^2(N)}{\log \log(N)})$</td>
<td>$O(l \cdot \log^3(N))$</td>
</tr>
<tr>
<td>Path-PIR Additive</td>
<td>$O(k \cdot \log(N) + l)$</td>
<td>$O(k \cdot \log^3(N) + l \cdot \log^2(N))$</td>
<td>$O(l \cdot \log(N))$</td>
</tr>
<tr>
<td>Path-PIR FHE</td>
<td>$O(k + l)$</td>
<td>$O(k \cdot \log(N) + l \cdot \log(N))$</td>
<td>$O(k + l)$</td>
</tr>
<tr>
<td>Optimal</td>
<td>$O(\log(N) + l)$</td>
<td>$O(\log(N) + l)$</td>
<td>$O(\log(N) + l)$</td>
</tr>
</tbody>
</table>
Communication Comparison

Overall (per query)

Latency (per query)
But what about expensive computation?

Total Time (per query) vs. Database Size (TB)

Dollar Cost on AWS (per query) vs. Database Size (TB)
Conclusion

• We have introduced a technique for applying PIR to ORAM protocols which results in significantly decreased communication.

• Combining our technique with an existing scheme leads to an efficient ORAM protocol with very low (optimal) latency.

• Our protocol was tested on Amazon AWS and shown to be cheaper and faster than related work.