Constant-weight PIR: Single-round Keyword PIR via Constant-weight Equality Operators

Rasoul Akhavan Mahdavi, Florian Kerschbaum

Usenix 2022, Boston, US
Homomorphic Encryption

- Method for computation over encrypted data
- Circuits from basic operations: + and *
Homomorphic Encryption

- Method for computation over encrypted data
- Circuits from basic operations: + and *
- Multiplication is expensive
- Mult. Depth determines parameters

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (μs)</th>
<th>Noise Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N = 2048$</td>
<td>$N = 4096$</td>
</tr>
<tr>
<td>Addition</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>Plain Mult.*</td>
<td>32.235</td>
<td>39.520</td>
</tr>
<tr>
<td>Multiplication</td>
<td>-</td>
<td>3823</td>
</tr>
<tr>
<td>Substitution</td>
<td>-</td>
<td>768</td>
</tr>
</tbody>
</table>

Larger Parameters
Homomorphic Encryption

- Method for computation over encrypted data
- Circuits from basic operations: $+$ and $\times$
- Multiplication is expensive
- Mult. Depth determines parameters

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (\mu s)</th>
<th>Noise Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N = 2048$</td>
<td>$N = 4096$</td>
</tr>
<tr>
<td>Addition</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>Plain Mult.*</td>
<td>127135</td>
<td>30520</td>
</tr>
<tr>
<td>Multiplication</td>
<td>-</td>
<td>3823</td>
</tr>
<tr>
<td>Substitution</td>
<td>-</td>
<td>768</td>
</tr>
</tbody>
</table>

1. Use Fewer Multiplications
2. Reduce the Multiplicative Depth

Larger Parameters
Equality Operators using HE

- Definition

\[ x \rightarrow \text{if} \rightarrow \text{output 1} \]
\[ y \rightarrow \text{else if} \rightarrow \text{output 0} \]
Equality Operators using HE

- Definition
- Important building block
  - Private Information Retrieval, Private Set Intersection, ...

\[
\begin{align*}
x & \quad \text{if} \\
y & \quad \text{else if} \\
\end{align*}
\]
Equality Operators using HE

- Definition:
- Important building block
  - Private Information Retrieval, Private Set Intersection, ...
- Existing equality operators

\[ D = \{0, 1\}^\ell \]

Plain

\[ f_{FP}(x, y) = \prod_{y[i]=0} (1 - x[i]) \prod_{y[i]=1} x[i] \]

Arithmetic

\[ f_{AF}(x, y) = \prod_{i=0}^{\ell - 1} \left(1 + (x[i] - y[i])^2\right) \]
Equality Operators using HE

- Multiplicative depth depends on the element size

\[ f_{AP}(x, y) = \prod_{i=0}^{\ell-1} \left( 1 + (x[i] - y[i])^2 \right) \]

- Limits scalability of equality using HE

\# of multiplications: \( 2^\ell \)

Multiplicative depth: \( \log_2 \ell + 1 \)
Equality Operators using HE

- Multiplicative depth depends on the element size

\[ f_{\text{AP}}(x, y) = \prod_{i=0}^{l-1} \left( 1 + (x[i] - y[i])^2 \right) \]

- Limits scalability of equality using HE

\# of multiplications: \(2^l\)

Multiplicative depth: \(\log_2 l + 1\)

We need a better approach
Constant-weight Code

Constant-weight codewords

Exactly \textbf{k} bits set to one

\begin{align*}
\text{m=}6, \text{ k=}2 \\
0000 & 11 \\
0001 & 01 \\
0001 & 10 \\
0010 & 01 \\
0010 & 10 \\
0011 & 00 \\
0011 & 00 \\
\vdots & \\
\vdots & \\
\end{align*}
Equality Operator for Constant-weight Codewords

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x_{m-1}$</th>
<th>$x_{m-2}$</th>
<th>...</th>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$x_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Plain
Equality Operator for Constant-weight Codewords

\[ \mathbf{x} = x_{m-1} x_{m-2} \ldots x_2 x_1 x_0 \]

\[ \mathbf{y} = 0 \quad 1 \quad \ldots \quad 0 \quad 0 \quad 1 \]

\[ x_{m-2} \times \cdots \times x_0 = e \]
Equality Operator for Constant-weight Codewords

**Plain**

\[
\begin{array}{cccccc}
\mathbf{x} & x_{m-1} & x_{m-2} & \cdots & x_2 & x_1 & x_0 \\
\mathbf{y} & 0 & 1 & \cdots & 0 & 0 & 1
\end{array}
\]

\[
x_{m-2} \times \cdots \times x_0 = e
\]

**Arithmetic**

\[
\begin{array}{cccccc}
\mathbf{x} & x_{m-1} & x_{m-2} & \cdots & x_2 & x_1 & x_0 \\
\mathbf{y} & y_{m-1} & y_{m-2} & \cdots & y_2 & y_1 & y_0
\end{array}
\]

\[
\begin{array}{cccccc}
\mathbf{x} & x_{m-1} & x_{m-2} & \cdots & x_2 & x_1 & x_0 \\
\mathbf{y} & y_{m-1} & y_{m-2} & \cdots & y_2 & y_1 & y_0
\end{array}
\]

13
Equality Operator for Constant-weight Codewords

\[ x \equiv y \iff x_0 \cdot x_1 \cdots x_{m-2} = e \]

Plain

\[
\begin{array}{cccc}
  x_{m-1} & x_{m-2} & \cdots & x_2 & x_1 & x_0 \\
  y_{m-1} & y_{m-2} & \cdots & y_2 & y_1 & y_0 \\
\end{array}
\]

Arithmetic

\[
\begin{array}{cccc}
  x_{m-1} & x_{m-2} & \cdots & x_2 & x_1 & x_0 \\
  y_{m-1} & y_{m-2} & \cdots & y_2 & y_1 & y_0 \\
\end{array}
\]

Inner product

\[
\frac{1}{k!} \prod_{i=0}^{k-1} (c - i) = e
\]
Equality Operator for Constant-weight Codewords

Plain

\[
\begin{array}{cccccc}
\mathbf{x} & x_{m-1} & x_{m-2} & \ldots & x_2 & x_1 & x_0 \\
\mathbf{y} & 0 & 1 & \ldots & 0 & 0 & 1 \\
\end{array}
\]

\[
x_{m-2} \times \cdots \times x_0 = e
\]

# of Mult = \(k\)  
Mult. Depth = \(\log_2 k\)

Arithmetic

\[
\begin{array}{cccccc}
\mathbf{x} & x_{m-1} & x_{m-2} & \ldots & x_2 & x_1 & x_0 \\
\mathbf{y} & y_{m-1} & y_{m-2} & \ldots & y_2 & y_1 & y_0 \\
\end{array}
\]

Inner product

\[
\frac{1}{k!} \prod_{i=0}^{k-1} (c - i) = e
\]

# of Mult = \(m + k\)  
Mult. Depth = \(\log_2 k + 1\)
Evaluation of Equality Operators

Constant-weight Plain Operator

- 10x faster than folklore
- Attributed to
  - Less operations
  - Smaller Parameters
- Larger representation size
  - Higher memory usage
  - Slower runtime for less memory
Evaluation of Equality Operators

Arithmetic Operators

- Comparable runtime
- More operations for constant-weight
- Better when smaller parameters are used

<table>
<thead>
<tr>
<th></th>
<th>$n$</th>
<th>$2^8$</th>
<th>$2^{16}$</th>
<th>$2^{32}$</th>
<th>$2^{64}$</th>
<th>$2^{128}$</th>
<th>$2^{256}$</th>
<th>$2^{512}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Folklore</td>
<td>$\ell$</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td></td>
<td>Mult Depth</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>$N = 8192$</td>
<td>0.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N = 16384$</td>
<td>2.2</td>
<td>4.6</td>
<td>9.2</td>
<td>19</td>
<td>37</td>
<td>74</td>
<td>149</td>
<td></td>
</tr>
</tbody>
</table>

| Arithmetic Constant-weight | $k$ | 4     | 8       | 16       | 32       | 64       | 128       | 256       |
|                            | Mult Depth | 3     | 4        | 5        | 6        | 7         | 8         | 9         |
| $m$ | 11   | 19    | 36      | 68       | 132      | 261      | 517       |           |
| $N = 8192$   | 0.53  | -     | -       | -        | -        | -        | -         | -         |
| $N = 16384$  | 2.2   | 4.3   | 8.2     | 16       | 31       | 63       | 123       |           |

| Arithmetic Constant-weight | $k$ | 2     | 4       | 8       | 16       | 32       | 64       | 128       |
|                            | Mult Depth | 2     | 3        | 4        | 5        | 6         | 7         | 8         |
| $m$ | 24   | 37    | 64      | 117      | 221      | 427      | 838       |           |
| $N = 8192$   | 0.85  | 1.3   | -       | -        | -        | -        | -         | -         |
| $N = 16384$  | 4.3   | 6.4   | 11      | 21       | 40       | 78       | 154       |           |
Evaluation of Equality Operators

Arithmetic Operators

- Comparable runtime
- More operations for constant-weight
- Better when smaller parameters are used
- Parallelization is better for constant-weight
Let’s Build a PIR Protocol
Private Information Retrieval

- Retrieve an element from a database without the server learning which element is retrieved
Private Information Retrieval

- Retrieve an element from a database without the server learning which element is retrieved
- Single server and multi-server
  - We focus on single-server PIR
Private Information Retrieval

- Retrieve an element from a database without the server learning which element is retrieved
- Single server and multi-server
  - We focus on single-server PIR
- Variations:
Private Information Retrieval

- Retrieve an element from a database without the server learning which element is retrieved
- Single server and multi-server
  - We focus on single-server PIR
- Variations:

<table>
<thead>
<tr>
<th>Index PIR</th>
<th>Keyword PIR</th>
</tr>
</thead>
</table>

  Example
  - 256-bit identifiers
  - filenames
Private Information Retrieval

- Retrieve an element from a database without the server learning which element is retrieved
- Single server and multi-server
  - We focus on single-server PIR
- Variations:
  - Index PIR (easier)
  - Keyword PIR (harder)
  - SealPIR
  - MulPIR
Private Information Retrieval

- Retrieve an element from a database without the server learning which element is retrieved
- Single server and multi-server
  - We focus on single-server PIR
- Variations:
  - Index PIR (easier)
  - Keyword PIR (harder)
  - SealPIR
  - MulPIR

Extra round Probabilistic Failure
Constant-weight PIR
Constant-weight PIR
Constant-weight PIR

User

+ \left( c = d \right) \times \text{file}

\vdots

+ \left( c = d \right) \times \text{file}

\text{file}
PIR for Sparse Databases

- Slower in a packed database
- Advantageous as domain grows
PIR for Sparse Databases

- Slower in a packed database
- Advantageous as domain grows
- # of operations doesn’t depend on domain size

<table>
<thead>
<tr>
<th>Method</th>
<th>Mult Depth</th>
<th>Query Bit-length</th>
<th># of Operations (Excluding Expansion)</th>
<th>Download Cost (in cts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SealPIR</td>
<td>$d - 1$</td>
<td>$d \left\lceil \frac{d}{\sqrt{</td>
<td>S</td>
<td>}} \right\rceil$</td>
</tr>
<tr>
<td>MulPIR</td>
<td>$d - 1$</td>
<td>$d \left\lceil \frac{d}{\sqrt{</td>
<td>S</td>
<td>}} \right\rceil$</td>
</tr>
<tr>
<td>CwPIR</td>
<td>$[\log k]$</td>
<td>$O\left( \frac{1}{k!}</td>
<td>S</td>
<td>+ k \right)$</td>
</tr>
</tbody>
</table>
PIR for Sparse Databases

Smaller representation size for fixed multiplicative depth

Encoding size as a function of multiplicative depth
PIR with Large Response Size

- The response must fit into a ciphertext
- SealPIR/MulPIR are repeated when the response is large
Takeaways

- Constant-weight Equality Operators
  - suitable for homomorphic encryption
  - faster than existing operators
- Constant-weight Keyword PIR
  - using equality operators
  - single-round
  - faster for sparse databases (i.e. keyword PIR)
  - faster for larger response sizes
Takeaways

- **Constant-weight Equality Operators**
  - suitable for homomorphic encryption
  - faster than existing operators

- **Constant-weight Keyword PIR**
  - using equality operators
  - single-round
  - faster for sparse databases (i.e. keyword PIR)
  - faster for larger response sizes

GitHub: RasoulAM/constant-weight-pir
Thank you!

Questions?