Pushing the Communication Barrier in 2PC using Lookup Tables

Ghada Dessouky*, Farinaz Koushanfar†, Ahmad-Reza Sadeghi*, Thomas Schneider*, Shaza Zeitouni*, and Michael Zohner*

*Technische Universität Darmstadt
†University of California, San Diego
Secure 2PC

\[ f(x, y) \]

\[ f \]

\[ f(x, y) \]
Secure 2PC

Alice < Alice
Secure 2PC

This work: semi-honest (passive) security
Applications of Secure 2PC

Sugar Beet Auction [BCD+09]
Applications of Secure 2PC

Sugar Beet Auction [BCD+09]  

Face Recognition [EFG+09]
Applications of Secure 2PC

Sugar Beet Auction [BCD+09]

Face Recognition [EFG+09]

Blind En/Decryption [Dyadic]

Michael Zohner - Pushing the Communication Barrier in 2PC
Applications of Secure 2PC

Sugar Beet Auction [BCD+09]  
Face Recognition [EFG+09]

Blind En/Decryption [Dyadic]  
Stable Matching [DES16]
Generic Secure 2PC

Two prominent techniques: Yao's protocol and GMW

Both evaluate Boolean circuits securely
- XOR gates are „free“
- AND gates cost sym. crypto / comm.

Difference: round complexity
- Yao is constant round
- GMW requires interaction per AND gate
Currently: 3 million ANDs/s per thread, however:

- We have hit a comm. lower-bound per AND for Yao [ZRE15]
- Run-time for GMW often is mostly network latency
Lookup Tables

\[ a \quad b \quad c \quad \wedge \quad \oplus \quad \wedge \quad d \]

\[ a \quad b \quad c \quad \rightarrow \quad \text{LUT} \quad \rightarrow \quad d \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Our Contributions

Develop lookup table (LUT)-based protocols

Tool support for generating LUT circuits

Evaluation and comparison

(Paper: improve building blocks & comm. for GMW)
Lookup Table Protocols

LUT

\[
\begin{array}{ccc|c}
 a & b & c & d \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 \\
 0 & 1 & 0 & 0 \\
 0 & 1 & 1 & 0 \\
 1 & 0 & 0 & 0 \\
 1 & 0 & 1 & 0 \\
 1 & 1 & 0 & 1 \\
 1 & 1 & 1 & 0 \\
\end{array}
\]
1ooN Oblivious Transfer

Bob obliviously obtains one of N messages s.t.
- Alice does not learn Bob's choice $c$
- Bob does not learn Alice's other messages

Most efficient protocol 1ooN OT: [KK13]
Intuition of the Protocols

Use \([KK13]\) 1ooN OT to perform table lookups

\[
\begin{array}{cccc}
\text{LUT:} & & & \\
\hline
a & b & c & d \\
\hline
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

\[(a \| b \| c) \in [1...8]\)

\[\text{LUT}[(a \| b \| c)]\]
LUT Protocols

We develop two LUT protocols based on [KK13] OT

• Online Phase LUT (OP-LUT)
• Setup Phase LUT (SP-LUT)
Generating LUT Circuits

\[ z = x + y \]
Tool Support for LUTs

Generating LUT circuits is difficult and error-prone
  • Automation is required

Idea: FPGAs internally operate on single output LUTs
  • Use ABC logic synthesis to generate single output LUTs

Add post-processing to improve efficiency
Combining LUTs

FPGAs only support single output LUTs

We combine LUTs with similar inputs to improve efficiency

SP-LUT Communication: 512 bits

SP-LUT Communication: 380 bits
Extracting XORs

Since XORs are free, we can extract them

Example $z = (x \oplus y)$
Comparison
• Mostly: \textit{SP-LUT} \textless \textit{GMW} \textless \textit{OP-LUT} \textless \textit{Yao}

• Boolean circuits perform better for sequential structures

• LUT circuits perform best for tree based structures
Communication

- Mostly: SP-LUT < GMW < OP-LUT < Yao
- Boolean circuits perform better for sequential structures
- LUT circuits perform best for tree based structures
Communication

- Mostly: SP-LUT < GMW < OP-LUT < Yao
- Boolean circuits perform better for sequential structures
- LUT circuits perform best for tree based structures
Interaction Rounds

• **Yao** is constant round
• Mostly: **SP-LUT < OP-LUT < GMW**
• Exception: Multiplication with Ripple-carry addition
Interaction Rounds

- **Yao** is constant round
- Mostly: **SP-LUT < OP-LUT < GMW**
- Exception: Multiplication with Ripple-carry addition
Empirical Evaluation

AES encryption of 1000 blocks using 4 threads
- LAN (1 GBit network, 0.2 ms latency)
- WAN (28 MBit network, 122ms latency)
Conclusion

Communication is bottleneck in 2PC

Developed LUT protocols based on 100N OT

Tool chain for compiling LUT circuits

Showed that LUT protocols can improve communication
Thank you for your attention
From 1oo2 OT to 1ooN OT

[IKNP03]

128 bit

1oo2 OT

128 log N bit

1ooN OT

[KK13]

\[ k' \leq 128 \log N \text{ bit} \]
Our Results

![Graph showing communication (MBytes) vs. time symmetric crypto (ms) for various protocols. Notable points include Yao, GMW, PnP [BMR90], GRR-3 [NPS99], Unopt. [LP04], FK-AES [BHKR13], Half-Gates [ZRE15], Free-XOR [KS08], Unopt. [CHK+12], and 2x Lower Bound [ZRE15].]
1ooN OT Extension [KK13]

\[ T \in \mathbb{R} \{0, 1\}^{m \times k} \]

\[ s \in \mathbb{R} \{0, 1\}^k \]

\[ V_i = T_i \oplus s_i \cdot r \]

for \( 1 \leq i \leq k \):

\[ (T_i, T_i \oplus r) \]

if \( r_j = 0 \) if \( r_j = 1 \)

\[ \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \]

Hamming distance \( k \)

\[ k' \leq 128 \log N \]

Codewords with HD \( k \)

Michael Zohner - Pushing the Communication Barrier in 2PC
From 1oo2 OT to 1ooN OT

- 1ooN OT can be obtained from logN 1oo2 OTs

- Example 1oo4:

\[
\begin{align*}
(m_{0,0}, m_{0,1}) & \quad \xrightarrow{\text{OT}} \quad (m_{0,0}, s_0) \\
(m_{1,0}, m_{1,1}) & \quad \xrightarrow{\text{OT}} \quad (m_{1,0}, s_1) \\
\end{align*}
\]

\[
\begin{align*}
s_0 & \in \{0, 1\} \\
\end{align*}
\]

\[
\begin{align*}
x_0 \oplus m_{0,0} & \oplus m_{1,0} \\
x_1 \oplus m_{0,0} & \oplus m_{1,1} \\
x_2 \oplus m_{0,1} & \oplus m_{1,0} \\
x_3 \oplus m_{0,1} & \oplus m_{1,1} \\
\end{align*}
\]
From 1ooN OT to 1oo2 OT

• Surprising insight: reducing 1ooN OT to single bit 1oo2 OT saves communication

• Best for N=16: Requires 320 bits instead of 512 bits