Efficient OT Extension and its Impact on Secure Computation

Pushing the Communication Barrier of Passive Secure Two-Party Computation

Michael Zohner (TU Darmstadt)

Joint work with
Ghada Dessouky, Ahmad-Reza Sadeghi, Thomas Schneider,
Shaza Zeitouni (all TU Darmstadt)
Farinaz Koushanfar (UC San Diego)
Applications

Auctions, ...

Private Set Intersection, ...

Machine Learning, ...

Biometric Identification, ...

This work: passive security and security parameter $k = 128$ bit
Secure Two-Party Computation Protocols

Yao's garbled circuits protocol
- Function-dependent setup phase
- Constant round
- $\geq 256$ bit communication per AND (simplex)

Very fast implementations
- Fairplay $\sim 1\,000$ Gates/s
- FastGC $\sim 100\,000$ AND/s
- ObliVM $\sim 3$ million AND/s
- Billion Gate $\sim 100\,000$ AND/s  \hspace{1cm} Passive
- Blazing Fast $\sim 500\,000$ AND/s  \hspace{1cm} Active
- More blazing fast
Secure Two-Party Computation Protocols

GMW
- Function-independent setup phase
- ≥ 256 bit communication per AND (duplex)

Setup Phase: pre-compute multiplication triples (MTs) using OT
- ApricotOT: passive=active ~7 million OTs/s
- ABY: ~3 million AND/s
- TinyOT: ~400 000 AND/s
- SPDZ: 5 000 Mult/s of 128 bit values

Online Phase: simple computation and small messages but multi-round
Status Quo

Good news: extremely fast computation
- JustGarble generates ~2GBit/s traffic per thread
- Passive OT extension generates ~1 Gbit/s traffic per thread

Bad news: communication boundary
- LAN connection provides 1Gbit/s
- Lowerbound on linear garbling schemes [ZRE15]
- Online time of GMW very latency dependent

Computation resources scale better than communication

Bottleneck: Communication and round complexity
Related Work

[KK13] outlines efficient 1ooN OT extension variant
  + Reduces communication per AND in GMW from 256 to 160 bit
  - High computation overhead

[IKMOP13,DZ16,TinyTable] uses multi-input tables for secure computation
  + Reduces communication and rounds in the online phase
  - High setup costs

GESS [KK12] multi-round information-theoretic variant of garbled circuits
  + Reduces communication
  - Unsure how to extend to arbitrary functionalities
What Did We Do?

1) Less communication for GMW
   - Further optimize 1ooN OT extension of [KK13] to compute MTs
   - Reduces communication from 256 to 134 bit per AND

2) Less communication and rounds Lookup Table (LUT) representation
   - Online LUT (O-LUT): efficient online phase
   - Setup LUT (S-LUT): efficient overall evaluation

3) Tool support for generating LUT representations

4) Evaluation on various basic operations
Our Results

Trade more computation for 1) less communication and 2) less rounds

- **2-MT**: GMW from 1-out-of-2 OT extension
- **N-MT**: GMW from 1-out-of-N OT extension [KK13]
- **O-LUT**: LUT protocol with efficient online phase
- **S-LUT**: LUT protocol with better overall communication
Part 1) Less Communication for GMW
Alice holds $m$ pairs of messages $(x_{i,0}, x_{i,1})$

Bob holds $m$-bit string $r$ and wants to obtain $x_{i,r_i}$ in $i$-th OT
Alice and Bob switch roles and perform $k$ base OTs

\[(x_{j,0}, x_{j,1}) \in \{0, 1\}^{2\ell}\]

\[r = (r_1, \ldots, r_m) \in \{0, 1\}^m\]

\[T \in_R \{0, 1\}^{m \times k}\]

for $1 \leq i \leq k$:

\[s \in_R \{0, 1\}^k\]

\[V_i = T_i \oplus s_i \cdot r\]
From 1oo2 OT to 1ooN OT

[IKNP03]

128 bit

1oo2 OT \[\rightarrow\] 1ooN OT

128 \cdot \log N \text{ bit}
From 1oo2 OT to 1ooN OT

1ooN OT can be obtained from $\log N$ invocations of 1oo2 OT extension

Example: 1oo4 OT for $(x_1, \ldots, x_4)$

\[
\begin{align*}
(x_1 \oplus s_1, \ldots, x_4 \oplus s_4) &\rightarrow \text{OT} \quad \text{where } s_i \in \{0, 1\} \\
(x_1, \ldots, x_4) &\rightarrow \text{OT} \quad \text{where } s_i \in \{0, 1\}
\end{align*}
\]
1ooN OT Extension [KK13]

1oo2 OT \(\rightarrow\) 128·log \(N\) bit \(\rightarrow\) 1ooN OT

128 bit

\[ k' \leq 128 \cdot \log N \text{ bit} \]
Generalization to 1ooN OT Extension [KK13]

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

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

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

\[ \text{OT}_m^k \]

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

If \( r_j = 0 \)

If \( r_j = 1 \)

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

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

\[ k \]

\[ k' \]

Hamming distance \( k \)

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

Codewords with HD \( k \)
1ooN OT Extension [KK13] (Efficiency)

The codewords need $k$ bit Hamming distance (HD)

Efficiency of the [KK13] 1ooN OT depends on the underlying code

For $N = 2$: use repetition code
  - Same as the [IKNP03] protocol

For $2 < N \leq 2k$: use a Walsh Hadamard code
  - $h$ codewords with $h$ bit length and HD $h/2$
  - Since we require HD=$k$ we have $2k=256$ bit codewords

For $N > 2k$: use linear codes
  - Achieves $O(k)$ communication instead of $O(k \log N)$
  - Concrete improvements for PSI on 128-bit elements
1ooN OT Extension [KK13]

For 1oo2 bit OT: 80 bit

1oo2 OT \rightarrow 128 \cdot \log N \text{ bit} \rightarrow 1ooN OT

128 \text{ bit} \rightarrow 1oo2 OT

k' \leq 128 \cdot \log N \text{ bit}

[IKNP03] [KK13]
1ooN OT Extension for Short Strings [KK13]

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

Best for \( N=16 \): requires only 320 bits instead of 512 bits
[KK13] Downside: Increased Computation

1oo2 OT extension uses efficient fixed-key AES-128 [BHKR13]

1ooN OT processes values with >128-bit length
  • Too large for AES encryption
  • Replace by AES-256 with key schedule [KSS12] → 30 times slower

Even worse: 1ooN OT needs N evaluations while 1oo2 OT needs \(2\log N\)
  • For \(N=16\): 60x more computation
  • For \(N=256\): 480x more computation
Optimization 1) Improve Computation

Idea: Use pipelining of [GNLP15] for AES-256+KS

AES-256+KS only 9 instead of 30 times slower than fixed-key AES-128
Optimization 2) Short Codes

For $2 < N < 2k$, [KK13] uses Walsh-Hadamard code, which is not size-optimal.

Improve communication using specific codes for specific $N$
- [http://mint.sbg.ac.at/](http://mint.sbg.ac.at/) gives short codes for different parameters

Saves between 25% and 1% communication.
Optimization 3) MTs from 1ooN OT (N-MT)

[KK13] reduces 1ooN OT to 1oo2 OT for computing AND gates

Instead: reduce 1ooN OT to MTs (1oo4 OT)

Best for $N=16$: reducing communication from 256 to 134 bits per AND
Part 2) LUT-based Secure Computation

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63</td>
<td>7c</td>
<td>77</td>
<td>7b</td>
<td>f2</td>
<td>6b</td>
<td>6f</td>
<td>c5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2b</td>
<td>fe</td>
<td>d7</td>
<td>ab</td>
<td>76</td>
</tr>
<tr>
<td>1</td>
<td>ca</td>
<td>82</td>
<td>c9</td>
<td>7d</td>
<td>fa</td>
<td>59</td>
<td>47</td>
<td>f0</td>
<td>ad</td>
<td>d4</td>
<td>a2</td>
<td>af</td>
<td>9c</td>
<td>a4</td>
<td>72</td>
<td>c0</td>
</tr>
<tr>
<td>2</td>
<td>b7</td>
<td>fd</td>
<td>93</td>
<td>26</td>
<td>36</td>
<td>3f</td>
<td>f7</td>
<td>cc</td>
<td>34</td>
<td>a5</td>
<td>e5</td>
<td>f1</td>
<td>71</td>
<td>d8</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>04</td>
<td>c7</td>
<td>23</td>
<td>c3</td>
<td>18</td>
<td>96</td>
<td>05</td>
<td>9a</td>
<td>07</td>
<td>12</td>
<td>80</td>
<td>e2</td>
<td>eb</td>
<td>27</td>
<td>b2</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>09</td>
<td>83</td>
<td>2c</td>
<td>1a</td>
<td>1b</td>
<td>6e</td>
<td>5a</td>
<td>a0</td>
<td>52</td>
<td>3b</td>
<td>d6</td>
<td>b3</td>
<td>29</td>
<td>e3</td>
<td>2f</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>d1</td>
<td>00</td>
<td>ed</td>
<td>20</td>
<td>fc</td>
<td>b1</td>
<td>5b</td>
<td>6a</td>
<td>cb</td>
<td>be</td>
<td>39</td>
<td>4a</td>
<td>4c</td>
<td>58</td>
<td>cf</td>
</tr>
<tr>
<td>6</td>
<td>d0</td>
<td>ef</td>
<td>aa</td>
<td>fb</td>
<td>43</td>
<td>4d</td>
<td>33</td>
<td>85</td>
<td>45</td>
<td>f9</td>
<td>02</td>
<td>7f</td>
<td>50</td>
<td>3c</td>
<td>9f</td>
<td>a8</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>a3</td>
<td>40</td>
<td>8f</td>
<td>92</td>
<td>9d</td>
<td>38</td>
<td>f5</td>
<td>bc</td>
<td>b6</td>
<td>da</td>
<td>21</td>
<td>10</td>
<td>ff</td>
<td>f3</td>
<td>d2</td>
</tr>
<tr>
<td>8</td>
<td>cd</td>
<td>0c</td>
<td>13</td>
<td>ec</td>
<td>5f</td>
<td>97</td>
<td>44</td>
<td>17</td>
<td>c4</td>
<td>a7</td>
<td>7e</td>
<td>3d</td>
<td>64</td>
<td>5d</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>81</td>
<td>4f</td>
<td>dc</td>
<td>22</td>
<td>2a</td>
<td>90</td>
<td>88</td>
<td>46</td>
<td>ee</td>
<td>b8</td>
<td>14</td>
<td>de</td>
<td>5e</td>
<td>0b</td>
<td>db</td>
</tr>
<tr>
<td>a</td>
<td>e0</td>
<td>32</td>
<td>3a</td>
<td>0a</td>
<td>49</td>
<td>06</td>
<td>24</td>
<td>5c</td>
<td>c2</td>
<td>d3</td>
<td>ac</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>e4</td>
<td>79</td>
</tr>
<tr>
<td>b</td>
<td>e7</td>
<td>c8</td>
<td>37</td>
<td>6d</td>
<td>8d</td>
<td>d5</td>
<td>4e</td>
<td>a9</td>
<td>6c</td>
<td>56</td>
<td>f4</td>
<td>ea</td>
<td>65</td>
<td>7a</td>
<td>ae</td>
<td>08</td>
</tr>
<tr>
<td>c</td>
<td>ba</td>
<td>78</td>
<td>25</td>
<td>2e</td>
<td>1c</td>
<td>a6</td>
<td>b4</td>
<td>c6</td>
<td>e8</td>
<td>dd</td>
<td>74</td>
<td>1f</td>
<td>4b</td>
<td>bd</td>
<td>8b</td>
<td>8a</td>
</tr>
<tr>
<td>d</td>
<td>70</td>
<td>3e</td>
<td>b5</td>
<td>66</td>
<td>48</td>
<td>03</td>
<td>f6</td>
<td>0e</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>b9</td>
<td>86</td>
<td>c1</td>
<td>1d</td>
<td>9e</td>
</tr>
<tr>
<td>e</td>
<td>e1</td>
<td>f8</td>
<td>98</td>
<td>11</td>
<td>69</td>
<td>d9</td>
<td>8e</td>
<td>94</td>
<td>9b</td>
<td>1e</td>
<td>87</td>
<td>e9</td>
<td>ce</td>
<td>55</td>
<td>28</td>
<td>df</td>
</tr>
<tr>
<td>f</td>
<td>8c</td>
<td>a1</td>
<td>89</td>
<td>0d</td>
<td>bf</td>
<td>e6</td>
<td>42</td>
<td>68</td>
<td>41</td>
<td>99</td>
<td>2d</td>
<td>0f</td>
<td>b0</td>
<td>54</td>
<td>bb</td>
<td>16</td>
</tr>
</tbody>
</table>
**LUT Representation**

Boolean circuits require one round per layer of AND gates

Process multi-input gates to decrease rounds [IKMOP13,DZ16]

- [IKMOP13] introduced one-time truth table (OTTT) protocol
- [DZ16] showed how to pre-compute OTTTs
1) Represent as table

\[ f(x, y) = x + y \]

\[
\begin{array}{c|cccc}
  y \backslash x & 0 & 1 & 2 & 3 \\
  \hline
  0 & 0 & 1 & 2 & 3 \\
  1 & 1 & 2 & 3 & 4 \\
  2 & 2 & 3 & 4 & 5 \\
  3 & 3 & 4 & 5 & 6 \\
\end{array}
\]

2) Rotate table

\[ r = 2 \]

\[
\begin{array}{c|cccc}
  y \backslash x & 0 & 1 & 2 & 3 \\
  \hline
  0 & 5 & 6 & 3 & 4 \\
  1 & 2 & 3 & 0 & 1 \\
  2 & 3 & 4 & 1 & 2 \\
  3 & 4 & 5 & 2 & 3 \\
\end{array}
\]

3) Secret-Share

\[
\begin{array}{c|cccc}
  y \backslash x & 0 & 1 & 2 & 3 \\
  \hline
  0 & 4 & 4 & 6 & 2 \\
  1 & 5 & 2 & 4 & 5 \\
  2 & 4 & 3 & 5 & 2 \\
  3 & 0 & 3 & 6 & 5 \\
\end{array}
\]

\[ + \]

\[
\begin{array}{c|cccc}
  y \backslash x & 0 & 1 & 2 & 3 \\
  \hline
  0 & 1 & 2 & 5 & 2 \\
  1 & 4 & 1 & 3 & 3 \\
  2 & 6 & 1 & 3 & 0 \\
  3 & 4 & 2 & 3 & 5 \\
\end{array}
\]

4) Distribute

\[ M_0, r \rightarrow P_0 \]

\[ M_1, s \rightarrow P_1 \]
OTTT (Online Phase)

\[ P_0 \left( x=3, \ r=2, \ M_0 \right) \]
\[ P_1 \left( y=2, \ s=1, \ M_1 \right) \]

\[ u=(x+r) \]
\[ v=(y+s) \]

\[
\begin{array}{c|cccc}
 y \backslash x & 0 & 1 & 2 & 3 \\
\hline
 0 & 4 & 4 & 6 & 2 \\
 1 & 5 & 2 & 4 & 5 \\
 2 & 4 & 3 & 5 & 2 \\
 3 & 0 & 3 & 6 & 5 \\
\end{array}
\]

\[
M_0[u,v]=3
\]
\[
M_1[u,v]=2
\]

Output \( f(x,y)=5 \)

Compute \( f(x,y) \)

\[
\begin{array}{c|cccc}
 y \backslash x & 0 & 1 & 2 & 3 \\
\hline
 0 & 1 & 2 & 5 & 2 \\
 1 & 4 & 1 & 3 & 3 \\
 2 & 6 & 1 & 3 & 0 \\
 3 & 4 & 2 & 3 & 5 \\
\end{array}
\]

Output \( f(x,y)=5 \)
Pre-Computing OTTTs via Circuits [DZ16]

Use circuit-based protocols to pre-compute all table entries

1) Represent the function as circuit C

2) \( P_0 \) chooses random \( r \), \( P_1 \) chooses random \( s \)

3) For all \( 0 \leq i, j \leq 3 \), \( P_0 \) and \( P_1 \) evaluate \( C(r+i, s+j) = (M_0[i,j], M_1[i,j]) \)

For circuits with \( \delta \) input bits requires \( 2^\delta \) evaluations with \( \leq \delta-1 \) ANDs
LUT Protocols based on 1ooN OT

Circuit-based pre-computation of [DZ16] adds great overhead
  • For $\delta$-input LUTs $2^\delta$ overhead compared to Boolean circuit

Idea: Use 1ooN OT to obliviously transfer OTTTTs
  • LUT communication becomes independent of the circuit cost

We outline two LUT protocols:
  • Online LUT with low online communication
  • Setup LUT with low overall but higher online communication
Online LUT (O-LUT)

Use $100N$ OT to transfer OTTTTs for all possible choices of $s$

1) $P_0$ chooses random $r$ and $M_0$ and prepares $M_{1,s'} = f(i+r,j+s') \oplus M_0$ for all $0 \leq i,j,s' \leq 3$

2) $P_0$ and $P_1$ perform a $1004$ OT, where $P_1$ chooses a random table $s$

For $\delta$ inputs the parties have to transfer $2^\delta$ tables of $2^\delta$ bits each
Setup LUT (S-LUT)

High setup communication for OTTTs with $\delta$ inputs

- Using circuits: between $138 \cdot 2^\delta$ and $(\delta-1) \cdot 138 \cdot 2^\delta$ bits
- Using 1ooN OT: $2^{2^\delta}$ bits

Problem: OTTTs are heavy since they require outputs for all possible inputs

Idea: pre-compute 1ooN OT in the setup phase and only update results in the online phase
Improving **S-LUT** Round Complexity

- **OT Pre-computation**
  - \( P_0 \) to \( P_1 \)
  - Choices:
    - Choice 1
    - Choice 2
    - Choice \( d \)
  - Updates:
    - Update 1
    - Update 2
    - Update \( d \)
  - Total: \( 2d \) rounds

- **Role Switching [Huang12]**
  - \( P_0 \) to \( P_1 \)
  - Choices:
    - Choice 1
    - Choice 2
    - Choice 3
  - Updates:
    - Update 1
    - Update 2
    - Update \( d \)
  - Total: \( d + 1 \) rounds
LUT Efficiency (Setup Phase)

Setup Communication LUTs

- O-LUT
- S-LUT
- [DZ16]

Number of Inputs vs. Communication [Bytes]
LUT Efficiency (Online Phase)

Online Communication LUTs

Communication [Bytes]

Number of Inputs

- O-LUT, DZ[16]
- S-LUT
LUT Efficiency (Total)

![Graph showing Total Communication LUTs vs Number of Inputs for different types of LUTs: [DZ16], O-LUT, and S-LUT. The graph illustrates the linear increase in communication LUTs with the number of inputs.](image)
Communication vs Boolean Circuits

Total Communication LUTs

Optimum for O-LUT: 4 inputs (105% of N-MT)
Optimum for S-LUT: 7 inputs (45% of N-MT)
Part 3) Generating LUT Representations

![Diagram showing two LUT (Look-Up Table) blocks connected with inputs X1 to X6 and outputs Z1 and Z2.](image-url)
Yet Another Compiler?

Re-doing the work for LUTs is a time-consuming and error-prone task

=> Automate the generation of LUT representations

Idea: FPGAs internally operate on single output LUTs

We use the ABC Logic synthesis tool to generate single output LUTs
Grouping Multi-Output LUTs

Problem: FPGAs only support LUTs with one output bit

We post-process and group LUTs with the same or similar inputs

S-LUT Communication: 512 bits  S-LUT Communication: 380 bits
Extracting XORs

Values are bitwise XOR secret-shared
  • Allows free XOR and evaluation of AND gates using MTs

Example: $x = y$
Part 4) Empirical Comparison
Setting

Evaluate communication and rounds for basic operations
- Addition
- Multiplication
- Comparison
- AES S-Box
- Floating-Point Operations

Evaluate different approaches
- **2-MT**: GMW using 1oo2 OT extension (260 bits)
- **N-MT**: GMW using 1ooN OT extension (138 bits)
- **O-LUT**: for LUTs with up to 4 inputs
- **S-LUT**: for LUTs with up to 8 inputs
Mostly: $S$-LUT < $N$-MT < $O$-LUT < $2$-MT

$2$-MT and $N$-MT perform better for Ripple-carry based circuits

LUT approaches perform best for tree based structures
Rounds Basic Operations

- Mostly: S-LUT < O-LUT < MT
- Exception: Multiplication with Ripple-carry addition
Evaluation on Applications: AES

AES encryption of 1 block using 4 threads

- LAN (1 GBit, 0.2 ms latency)
- WAN (120 Mbit, 100 ms latency)
AES encryption of 1,000 blocks using 4 threads

- LAN (1 GBit, 0.2 ms latency)
- WAN (120 Mbit, 100ms latency)
Take-Away Message

Traded more computation for less communication and rounds

GMW costs ~one ciphertext per AND

LUT protocols can reduce communication and rounds even further
Questions?

Contact:  http://encrypto.de
References


