General Purpose Frameworks
for Secure Multi-party Computation

Marcella Hastings
Brett Hemenway
Daniel Noble
Steve Zdancewic

University of Pennsylvania
Beyond Beets: MPC in practice

- Blind auction [BCD+08]
- Satellite collisions [HLO+16]
- Genome edit distance [JKS08]
Beyond Beets: MPC in practice

- Blind auction [BCD+08]
- Satellite collisions [HLO+16]
- Genome edit distance [JKS08]

- Parameter computation [BGM17]

-CASH-
Beyond Beets: MPC in practice

- Blind auction [BCD+08]
- Satellite collisions [HLO+16]
- Genome edit distance [JKS08]

- Parameter computation [BGM17]
- Financial statistics [BLV17]
Beyond Beets: MPC in practice

- Blind auction [BCD+08]
- Satellite collisions [HLO+16]
- Genome edit distance [JKS08]

- Parameter computation [BGM17]
- Financial statistics [BLV17]
- Government applications
Motivating end-to-end frameworks for MPC

▶ Custom one-off solutions are unsustainable
Motivating end-to-end frameworks for MPC

- Custom one-off solutions are unsustainable

- Protocols assumed impractical until Fairplay [MNPS04]
Motivating end-to-end frameworks for MPC

▶ Custom one-off solutions are unsustainable

▶ Protocols assumed impractical until Fairplay [MNPS04]

![Fair Play](image)

▶ Performance improvements rapidly advanced state-of-the-art
  ▶ OT extension [IKNP03]
  ▶ Free XOR gates [KS08]
  ▶ Half-gates [ZRE15]
  ▶ AES-NI
Modern General-Purpose Frameworks

Who are frameworks designed for?
Can the languages express complex, interesting functions?
Are the protocols appropriate for practical settings?
Has software development moved beyond “research code”?

function
description

function
compiler
runtime
output

function input

Framework
Modern General-Purpose Frameworks

Who are frameworks designed for?
Can the languages express complex, interesting functions?
Are the protocols appropriate for practical settings?
Has software development moved beyond “research code”?
Contributions

Survey

- Surveyed 9 frameworks and 2 circuit compilers
- Recorded protocol, feature, implementation details
- Evaluated usability criteria
Contributions

Survey

- Surveyed 9 frameworks and 2 circuit compilers
- Recorded protocol, feature, implementation details
- Evaluated usability criteria

Open-source framework repository

- Three sample programs in every framework
- Docker instances with complete build environments
- Documentation on compilation and execution

github.com/mpc-sok/frameworks
Findings

Most frameworks are in good shape!

- Diverse set of threat models and protocols
- Expressive high-level languages
- Accessible, open-source, and compilable
Findings

Most frameworks are in good shape!

- Diverse set of threat models and protocols
- Expressive high-level languages
- Accessible, open-source, and compilable

Room for improvement

- Engineering limitations
- Barriers to usability
# Inclusion criteria

- **End-to-end**

<table>
<thead>
<tr>
<th>End-to-end</th>
<th>ABY</th>
<th>ABY³</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end</td>
<td>CBMC-GC</td>
<td>dualBatchEx</td>
</tr>
<tr>
<td>End-to-end</td>
<td>Duplo</td>
<td>EMP-toolkit</td>
</tr>
<tr>
<td>End-to-end</td>
<td>EzPC</td>
<td>Fairplay</td>
</tr>
<tr>
<td>End-to-end</td>
<td>FairplayMP</td>
<td>Frigate</td>
</tr>
<tr>
<td>End-to-end</td>
<td>FRESCO</td>
<td>HyCC</td>
</tr>
<tr>
<td>End-to-end</td>
<td>JIFF</td>
<td>JustGarble</td>
</tr>
<tr>
<td>End-to-end</td>
<td>L1</td>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>End-to-end</td>
<td>MPyC</td>
<td>Obliv-C</td>
</tr>
<tr>
<td>End-to-end</td>
<td>ObliVM</td>
<td>PCF</td>
</tr>
<tr>
<td>End-to-end</td>
<td>PICCO</td>
<td>PySyft</td>
</tr>
<tr>
<td>End-to-end</td>
<td>SCALE-MAMBA</td>
<td>SCAPI</td>
</tr>
<tr>
<td>End-to-end</td>
<td>Semi-honest BMR</td>
<td>SEPIA</td>
</tr>
<tr>
<td>End-to-end</td>
<td>Sharemind</td>
<td>SPDZ</td>
</tr>
<tr>
<td>End-to-end</td>
<td>TASTY</td>
<td>TensorSPDZ</td>
</tr>
<tr>
<td>End-to-end</td>
<td>TinyLego</td>
<td>TinyGarble</td>
</tr>
<tr>
<td>End-to-end</td>
<td>VIFF</td>
<td>Wys*</td>
</tr>
<tr>
<td>End-to-end</td>
<td>Wysteria</td>
<td></td>
</tr>
</tbody>
</table>
# Inclusion criteria

- **End-to-end**
  - consumes a high level language

<table>
<thead>
<tr>
<th>ABY</th>
<th>ABY³</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBMC-GC</td>
<td>dualBatchEx</td>
</tr>
<tr>
<td>Duplo</td>
<td>EMP-toolkit</td>
</tr>
<tr>
<td>EzPC</td>
<td>Fairplay</td>
</tr>
<tr>
<td>FairplayMP</td>
<td>Frigate</td>
</tr>
<tr>
<td>FRESCO</td>
<td>HyCC</td>
</tr>
<tr>
<td>JIFF</td>
<td>JustGarble</td>
</tr>
<tr>
<td>L1</td>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>MPyC</td>
<td>Obliv-C</td>
</tr>
<tr>
<td>ObliVM</td>
<td>PCF</td>
</tr>
<tr>
<td>PICCO</td>
<td>PySyft</td>
</tr>
<tr>
<td>SCALE-MAMBA</td>
<td>SCAPI</td>
</tr>
<tr>
<td>Semi-honest-BMR</td>
<td>SEPIA</td>
</tr>
<tr>
<td>Sharemind</td>
<td>SPDZ</td>
</tr>
<tr>
<td>TASTY</td>
<td>TensorSPDZ</td>
</tr>
<tr>
<td>TinyLego</td>
<td>TinyGarble</td>
</tr>
<tr>
<td>VIFF</td>
<td>Wys*</td>
</tr>
<tr>
<td>Wysteria</td>
<td></td>
</tr>
</tbody>
</table>
### Inclusion criteria

- **End-to-end**
  - consumes a high level language
  - not standard library/API

<table>
<thead>
<tr>
<th>Library/Tool</th>
<th>Library/Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABY</td>
<td>ABY³</td>
</tr>
<tr>
<td>CBMC-GC</td>
<td>dualBatchEx</td>
</tr>
<tr>
<td>Duplo</td>
<td>EMP-toolkit</td>
</tr>
<tr>
<td>EzPC</td>
<td>Fairplay</td>
</tr>
<tr>
<td>FairplayMP</td>
<td>Frigate</td>
</tr>
<tr>
<td>FRESCO</td>
<td>HyCC</td>
</tr>
<tr>
<td>JIFF</td>
<td>JustGarble</td>
</tr>
<tr>
<td>L1</td>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>MPyC</td>
<td>Obliv-C</td>
</tr>
<tr>
<td>ObliVM</td>
<td>PCF</td>
</tr>
<tr>
<td>PICCO</td>
<td>PySyft</td>
</tr>
<tr>
<td>SCALE-MAMBA</td>
<td>SCAPI</td>
</tr>
<tr>
<td>Semi-honest-BMR</td>
<td>SEPIA</td>
</tr>
<tr>
<td>Sharemind</td>
<td>SPDZ</td>
</tr>
<tr>
<td>TASTY</td>
<td>TensorSPDZ</td>
</tr>
<tr>
<td>TinyLego</td>
<td>TinyGarble</td>
</tr>
<tr>
<td>VIFF</td>
<td>Wys*</td>
</tr>
<tr>
<td>Wysteria</td>
<td></td>
</tr>
</tbody>
</table>
## Inclusion criteria

- **End-to-end**
  - consumes a high level language
  - not standard library/API
- **General purpose**

<table>
<thead>
<tr>
<th>ABY</th>
<th>CBMC-GC</th>
<th>EMP-toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplo</td>
<td>EzPC</td>
<td>Fairplay</td>
</tr>
<tr>
<td>FairplayMP</td>
<td>FRESCO</td>
<td>Frigate</td>
</tr>
<tr>
<td>JIFF</td>
<td>L1</td>
<td>HyCC</td>
</tr>
<tr>
<td>MPyC</td>
<td>ObliVM</td>
<td>JustGarble</td>
</tr>
<tr>
<td>PICCO</td>
<td>SCAI</td>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>SCALE-MAMBA</td>
<td>SEPIA</td>
<td>Obliv-C</td>
</tr>
<tr>
<td>Semi-honest-BMR</td>
<td>SPDZ</td>
<td>PCF</td>
</tr>
<tr>
<td>Sharemind</td>
<td>TensorSPDZ</td>
<td>PySyft</td>
</tr>
<tr>
<td>TASTY</td>
<td>TinyGarble</td>
<td>SCAPI</td>
</tr>
<tr>
<td>TinyLego</td>
<td>VIFF</td>
<td>SEPIA</td>
</tr>
<tr>
<td>VIFF</td>
<td>Wysteria</td>
<td>SPDZ</td>
</tr>
<tr>
<td>Wysteria</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Inclusion criteria

- End-to-end
  - consumes a high level language
  - not standard library/API
- General purpose
- Recently updated ($\geq 2014$)

ABY
CBMC-GC
Duplo
EzPC
FairplayMP
Fresco
JIFF
L1
MPyC
ObliVM
PICCO
SCALE-MAMBA
Semi-honest-BMR
Sharemind
TASTY
TinyLego
VIFF
Wysteria

$ABY^3$
dualBatchEx
EMP-toolkit
Fairplay
Frigate
HyCC
JustGarble
MP-SPDZ
Obliv-C
PCF
PySyft
SCAPI
SEPIA
SPDZ
TensorSPDZ
TinyGarble
Wys*
Inclusion criteria

- **End-to-end**
  - consumes a high level language
  - not standard library/API
- **General purpose**
- **Recently updated (≥ 2014)**
- **Not too recent (< 8/2018)**

<table>
<thead>
<tr>
<th>ABY</th>
<th>CBMC-GC</th>
<th>Duplo</th>
<th>EzPC</th>
<th>FairplayMP</th>
<th>FRESCO</th>
<th>JIFF</th>
<th>L1</th>
<th>MPyC</th>
<th>ObliVM</th>
<th>PICCO</th>
<th>SCALE-MAMBA</th>
<th>Semi-honest-BMR</th>
<th>Sharemind</th>
<th>TASTY</th>
<th>TinyLego</th>
<th>VIFF</th>
<th>Wystleria</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABY³</td>
<td>dualBatchEx</td>
<td>EMP-toolkit</td>
<td>Fairplay</td>
<td>Frigate</td>
<td>HyCC</td>
<td>JustGarble</td>
<td>MP-SPDZ</td>
<td>Obliv-C</td>
<td>PCF</td>
<td>PySyft</td>
<td>SCAPI</td>
<td>SEPIA</td>
<td>SPDZ</td>
<td>TensorSPDZ</td>
<td>TinyGarble</td>
<td>Wys*</td>
<td></td>
</tr>
</tbody>
</table>

---

7 / 30
Inclusion criteria

- End-to-end
  - consumes a high level language
  - not standard library/API
- General purpose
- Recently updated (≥ 2014)
- Not too recent (< 8/2018)
- Not subsumed by new work

<table>
<thead>
<tr>
<th>ABY</th>
<th>CBMC-GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplo</td>
<td>EzPC</td>
</tr>
<tr>
<td>FairplayMP</td>
<td>FRESCO</td>
</tr>
<tr>
<td>JIFF</td>
<td>L1</td>
</tr>
<tr>
<td>MPyC</td>
<td>ObliVM</td>
</tr>
<tr>
<td>PICCO</td>
<td>SCALE-MAMBA</td>
</tr>
<tr>
<td>Semi-honest-BMR</td>
<td>Sharemind</td>
</tr>
<tr>
<td>TASTY</td>
<td>TinyLego</td>
</tr>
<tr>
<td>VIFF</td>
<td>Wysteria</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ABY³</th>
</tr>
</thead>
<tbody>
<tr>
<td>dualBatchEx</td>
</tr>
<tr>
<td>EMP-toolkit</td>
</tr>
<tr>
<td>Fairplay</td>
</tr>
<tr>
<td>Frigate</td>
</tr>
<tr>
<td>HyCC</td>
</tr>
<tr>
<td>JustGarble</td>
</tr>
<tr>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>Obliv-C</td>
</tr>
<tr>
<td>PCF</td>
</tr>
<tr>
<td>PySyft</td>
</tr>
<tr>
<td>SCAPI</td>
</tr>
<tr>
<td>SEPIA</td>
</tr>
<tr>
<td>SPDZ</td>
</tr>
<tr>
<td>TensorSPDZ</td>
</tr>
<tr>
<td>TinyGarble</td>
</tr>
<tr>
<td>Wys*</td>
</tr>
</tbody>
</table>
Inclusion criteria

- **End-to-end**
  - consumes a high level language
  - not standard library/API
- **General purpose**
- **Recently updated** ($\geq 2014$)
- **Not too recent** ($< 8/2018$)
- **Not subsumed by new work**
- **Accessible software**

<table>
<thead>
<tr>
<th>ABY</th>
<th>CBMC-GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplo</td>
<td>EzPC</td>
</tr>
<tr>
<td>FairplayMP</td>
<td>FRESCO</td>
</tr>
<tr>
<td>JIFF</td>
<td>L1</td>
</tr>
<tr>
<td>MPyC</td>
<td>ObliVM</td>
</tr>
<tr>
<td>PICCO</td>
<td>SCALE-MAMBA</td>
</tr>
<tr>
<td>Semi-honest-BMR</td>
<td>Sharemind</td>
</tr>
<tr>
<td>TASTY</td>
<td>TinyLego</td>
</tr>
<tr>
<td>VIFF</td>
<td>Wysteria</td>
</tr>
<tr>
<td>EMP-toolkit</td>
<td>Fairplay</td>
</tr>
<tr>
<td>Frigate</td>
<td>HyCC</td>
</tr>
<tr>
<td>JustGarble</td>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>Obliv-C</td>
<td>PCF</td>
</tr>
<tr>
<td>PySyft</td>
<td>SCAPi</td>
</tr>
<tr>
<td>SEPIA</td>
<td>SPDZ</td>
</tr>
<tr>
<td>TensorSPDZ</td>
<td>TinyGarble</td>
</tr>
<tr>
<td>Wys*</td>
<td></td>
</tr>
</tbody>
</table>
## Inclusion criteria

- **End-to-end**
  - consumes a high level language
  - not standard library/API
- **General purpose**
- **Recently updated** (*≥ 2014*)
- **Not too recent** (*< 8/2018*)
- **Not subsumed by new work**
- **Accessible software**

<table>
<thead>
<tr>
<th>ABY</th>
<th>CBMC-GC</th>
<th>ABY$^3$ dualBatchEx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplo</td>
<td>EzPC</td>
<td>EMP-toolkit</td>
</tr>
<tr>
<td>FairplayMP</td>
<td>FRESCO</td>
<td>Fairplay</td>
</tr>
<tr>
<td>JIFF</td>
<td>L1</td>
<td>Frigate</td>
</tr>
<tr>
<td>MPyC</td>
<td></td>
<td>HyCC</td>
</tr>
<tr>
<td>ObliVM</td>
<td>PICCO</td>
<td>JustGarble</td>
</tr>
<tr>
<td>SCALE-MAMBA</td>
<td>Semi-honest-BMR</td>
<td>MP-SPDZ</td>
</tr>
<tr>
<td>Sharemind</td>
<td>TASTY</td>
<td>Obliv-C</td>
</tr>
<tr>
<td>TinyLego</td>
<td>VIFF</td>
<td>PCF</td>
</tr>
<tr>
<td>VIFF</td>
<td></td>
<td>PySyft</td>
</tr>
<tr>
<td>Wysteria</td>
<td></td>
<td>SCAPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEPIA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPDZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TensorSPDZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TinyGarble</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wys*</td>
</tr>
</tbody>
</table>
## Frameworks: A brief overview

<table>
<thead>
<tr>
<th>Protocol family</th>
<th>Parties</th>
<th>Semi-honest</th>
<th>Malicious</th>
<th>DH majority</th>
<th>IT secure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EMP-toolkit [WMK17]</strong></td>
<td>GC 2</td>
<td>● ●</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Obliv-C [ZH15]</strong></td>
<td>GC 2</td>
<td>● ○</td>
<td>-</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td><strong>ObliVM [LWNHS15]</strong></td>
<td>GC 2</td>
<td>● ○</td>
<td>-</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td><strong>TinyGarble [SHSSK15]</strong></td>
<td>GC 2</td>
<td>● ○</td>
<td>-</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td><strong>Wysteria [RHH14]</strong></td>
<td>MC 2+</td>
<td>● ○ ○ ○</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ABY [DSZ15]</strong></td>
<td>GC,MC 2</td>
<td>● ○</td>
<td>-</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td><strong>SCALE-MAMBA [-]</strong></td>
<td>Hybrid 2+</td>
<td>● ● ●</td>
<td>○</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sharemind [BLW08]</strong></td>
<td>Hybrid 3</td>
<td>● ○ ○ ●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PICCO [ZSB13]</strong></td>
<td>Hybrid 3+</td>
<td>● ○ ○ ○</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Frigate [MGCKT16]</strong></td>
<td>- 2+</td>
<td>- - - -</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CBMC-GC [HFKV12]</strong></td>
<td>- 2+</td>
<td>- - - -</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GC** = Garbled Circuit  
**MC** = Multi-party circuit-based
Frameworks: A brief overview

<table>
<thead>
<tr>
<th>Protocol family</th>
<th>Parties</th>
<th>Semi-honest</th>
<th>Malicious</th>
<th>DH majority</th>
<th>IT secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP-toolkit</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>●</td>
<td>-</td>
</tr>
<tr>
<td>Obliv-C</td>
<td>GC</td>
<td>2+</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>ObliVM</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TinyGarble</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wysteria</td>
<td>MC</td>
<td>2+</td>
<td>●</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ABY</td>
<td>GC,MC</td>
<td>2</td>
<td>●</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SCALE-MAMBA</td>
<td>Hybrid</td>
<td>2+</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Sharemind</td>
<td>Hybrid</td>
<td>3</td>
<td>●</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PICCO</td>
<td>Hybrid</td>
<td>3+</td>
<td>●</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frigate</td>
<td>-</td>
<td>2+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBMC-GC</td>
<td>-</td>
<td>2+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

GC = Garbled Circuit  MC = Multi-party circuit-based
Garbled circuit protocols

Introduced by [Yao82, Yao86]

- Functions represented as Boolean circuits
- Typically semi-honest, 2-party
- Constant-round communication, volume $\propto$ circuit size

Diagram:

- Garble
- Evaluate
- Function output
- Runtime

\[\text{garble} \rightarrow \text{evaluate} \rightarrow \text{function output} \]

\[\text{runtime} \]
## Frameworks: A brief overview

<table>
<thead>
<tr>
<th>Protocol family</th>
<th>Parties</th>
<th>Semi-honest</th>
<th>Malicious</th>
<th>DH majority</th>
<th>IT secure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EMP-toolkit</strong></td>
<td>GC 2</td>
<td>● ●</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td><strong>Obliv-C</strong></td>
<td>GC 2+</td>
<td>● ● ● ●</td>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td><strong>ObliVM</strong></td>
<td>GC 2</td>
<td>● ● ○</td>
<td></td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td><strong>TinyGarble</strong></td>
<td>GC 2</td>
<td>● ○ ○</td>
<td></td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td><strong>Wysteria</strong></td>
<td>MC 2+</td>
<td>● ○ ○ ○</td>
<td></td>
<td>○</td>
<td>o</td>
</tr>
<tr>
<td><strong>ABY</strong></td>
<td>GC,M C 2</td>
<td>● ○ ○</td>
<td></td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td><strong>SCALE-MAMBA</strong></td>
<td>Hybrid 2+</td>
<td>● ● ● ●</td>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td><strong>Sharemind</strong></td>
<td>Hybrid 3</td>
<td>● ○ ○ ●</td>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td><strong>PICCO</strong></td>
<td>Hybrid 3+</td>
<td>● ○ ○ ○</td>
<td></td>
<td>○</td>
<td>o</td>
</tr>
<tr>
<td><strong>Frigate</strong></td>
<td>- 2+</td>
<td>- - - -</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td><strong>CBMC-GC</strong></td>
<td>- 2+</td>
<td>- - - -</td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

**GC** = Garbled Circuit  
**MC** = Multi-party circuit-based
Multi-party circuit-based protocols

Introduced by [GMW87, BGW88, CCD88]

- Functions represented as Boolean or arithmetic circuits
- Data represented as linear secret shares
- Various threat models and protocol types (information-theoretic or cryptographic)
- Rounds, volume of communication $\propto$ multiplication gates
## Frameworks: A brief overview

<table>
<thead>
<tr>
<th>Framework</th>
<th>Protocol family</th>
<th>Parties</th>
<th>Semi-honest</th>
<th>Malicious</th>
<th>DH majority</th>
<th>IT secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP-toolkit</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>●</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>GC</td>
<td>2+</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Obliv-C [ZH15]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>ObliVM [LWNHS15]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>TinyGarble [SHSSK15]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>Wysteria [RHH14]</td>
<td>MC</td>
<td>2+</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td><strong>ABY [DSZ15]</strong></td>
<td>GC,MC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>SCALE-MAMBA</td>
<td>Hybrid</td>
<td>2+</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Sharemind [BLW08]</td>
<td>Hybrid</td>
<td>3</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>PICCO [ZSB13]</td>
<td>Hybrid</td>
<td>3+</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Frigate [MGCKT16]</td>
<td>-</td>
<td>2+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBMC-GC [HFKV12]</td>
<td>-</td>
<td>2+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**GC = Garbled Circuit**

**MC = Multi-party circuit-based**
## Frameworks: A brief overview

<table>
<thead>
<tr>
<th>Protocol family</th>
<th>Parties</th>
<th>Semi-honest</th>
<th>Malicious</th>
<th>DH majority</th>
<th>IT secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP-toolkit [WMK17]</td>
<td>GC 2</td>
<td>● ● -</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Obliv-C [ZH15]</td>
<td>GC 2+</td>
<td>● ● ● ●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>ObliVM [LWNHS15]</td>
<td>GC 2</td>
<td>● ○ -</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>TinyGarble [SHSSK15]</td>
<td>GC 2</td>
<td>● ○ -</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Wysteria [RHH14]</td>
<td>MC 2+</td>
<td>● ○ ○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>ABY [DSZ15]</td>
<td>GC,MC 2</td>
<td>● ○ -</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>SCALE-MAMBA [-]</td>
<td>Hybrid 2+</td>
<td>● ● ● ●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Sharemind [BLW08]</td>
<td>Hybrid 3</td>
<td>● ○ ○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>PICCO [ZSB13]</td>
<td>Hybrid 3+</td>
<td>● ○ ○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Frigate [MGCKT16]</td>
<td>- 2+</td>
<td>- - - -</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBMC-GC [HFKV12]</td>
<td>- 2+</td>
<td>- - - -</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

GC = Garbled Circuit  
MC = Multi-party circuit-based
Hybrid protocols

- Integrates optimized subprotocols for common functions
  - Bitwise operators in arithmetic settings
  - Matrix operations
- Seamless front-end experience (no explicit protocol selection)
- Currently: One-to-one mapping from operations to protocols
Hybrid protocols

- Integrates optimized subprotocols for common functions
  - Bitwise operators in arithmetic settings
  - Matrix operations
- Seamless front-end experience (no explicit protocol selection)
- Currently: One-to-one mapping from operations to protocols
<table>
<thead>
<tr>
<th>Framework</th>
<th>Protocol family</th>
<th>Parties</th>
<th>Semi-honest</th>
<th>Malicious</th>
<th>DH majority</th>
<th>IT secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP-toolkit [WMK17]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>●</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>&quot;</td>
<td>GC</td>
<td>2+</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Obliv-C [ZH15]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>ObliVM [LWNHS15]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>TinyGarble [SHSSK15]</td>
<td>GC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>Wysteria [RHH14]</td>
<td>MC</td>
<td>2+</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>ABY [DSZ15]</td>
<td>GC, MC</td>
<td>2</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>SCALE-MAMBA [-]</td>
<td>Hybrid</td>
<td>2+</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Sharemind [BLW08]</td>
<td>Hybrid</td>
<td>3</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>PICCO [ZSB13]</td>
<td>Hybrid</td>
<td>3+</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Frigate [MGCKT16]</td>
<td>-</td>
<td>2+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBMC-GC [HFKV12]</td>
<td>-</td>
<td>2+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

GC = Garbled Circuit \hspace{1cm} \hspace{1cm} MC = Multi-party circuit-based
Sample programs
Implementations for every framework are on Github!

- **mult3**: Multiply 3 integers
- **innerprod**: Sum of pairwise product of vectors
- **xtabs**: Sums by category (type)

\[
\begin{array}{ccc}
\square & \ast & \square \\
\square & \ast & \square \\
\end{array} = \square
\]
Sample programs
Implementations for every framework are on Github!

▶ \textbf{mult3}: Multiply 3 integers
▶ \textbf{innerprod}: Sum of pairwise product of vectors
▶ \textbf{xtabs}: Sums by category (type)
Sample programs

Implementations for every framework are on Github!

- **mult3**: Multiply 3 integers
- **innerprod**: Sum of pairwise product of vectors
- **xtabs**: Sums by category (type)

\[
\begin{pmatrix}
\text{id} & \text{value} \\
\hline
\end{pmatrix}
\times
\begin{pmatrix}
\text{id} & \text{type} \\
\hline
\end{pmatrix}
\rightarrow
\begin{pmatrix}
\hline
\end{pmatrix}
\]
Sample programs
Implementations for every framework are on Github!

- **mult3**: Multiply 3 integers
- **innerprod**: Sum of pairwise product of vectors
- **xtabs**: Sums by category (type)

Items we don’t encompass
- Stress testing on large data
- Floating point behavior (division, type conversion)
- Overall performance comparison
Design decisions

- **Architecture**: system structure and data representation
- **Circuit model**: representing data-independent paradigm
- **Language accessibility**: cryptographic abstraction level
Design decisions: Architecture

How much should the framework rely on existing languages and compilers?
Design decisions: Architecture
How much should the framework rely on existing languages and compilers?

▶ Independent: Novel language specification and toolchain.

Examples:
ObliVM, Wysteria, SCALE-MAMBA, Frigate
Design decisions: Architecture
How much should the framework rely on existing languages and compilers?

- **Independent**: Novel language specification and toolchain.
- **Extension**: Modifies existing language.

Examples:
Obliv-C, TinyGarble, PICCO, CBMC-GC
Design decisions: Architecture
How much should the framework rely on existing languages and compilers?

- **Independent**: Novel language specification and toolchain.
- **Extension**: Modifies existing language.

Examples:
Obliv-C, TinyGarble, PICCO, CBMC-GC
Design decisions: Architecture

How much should the framework rely on existing languages and compilers?

- **Independent:** Novel language specification and toolchain.
- **Extension:** Modifies existing language.
- **Library:** Fully implemented within existing existing language.

Examples:
EMP-toolkit, ABY
Design decisions: Architecture

No clear winner

Independent languages allow formal security guarantees, control over front-end experience.

Libraries and extensions take advantage of features in the host language
- C’s speed
- Javascript’s parallelization
Design decisions: Execution flow

Intermediate representation (bytecodes, circuits)

Explicit optimization
Better modularity

Continuous execution (interpreter, source-to-source compiler)

Better dynamic behavior
Design decisions: Data representation

Bitwise representation (Boolean circuit)

+ Easy to use with garbled circuit protocols
+ Extends to arbitrary-sized numbers
− Not well suited to large arithmetic
Design decisions: Data representation

Bitwise representation (Boolean circuit)
  + Easy to use with garbled circuit protocols
  + Extends to arbitrary-sized numbers
  - Not well suited to large arithmetic

Linear secret shares (arithmetic circuit)
  + Well suited to large arithmetic
  - No natural interpretation of Boolean operations
Design decisions: Data representation

Bitwise representation (Boolean circuit)
- Easy to use with garbled circuit protocols
- Extends to arbitrary-sized numbers
  - Not well suited to large arithmetic

Linear secret shares (arithmetic circuit)
- Well suited to large arithmetic
  - No natural interpretation of Boolean operations

Recommendation: Hybrid model
Hidden conversion between representations
Design decisions: Data-independent construction

Should designers reveal “non-traditional” performance characteristics?

Circuits are a data-independent representation.

Branching programs are flattened in this model.

Non-expert users might not recognize this performance disparity.
Data independence: Private conditionals

Should branching programs reveal atypical performance?

Obliv-C: traditional paradigm

```c
obliv int result;
obliv if (a >= b) {
    result = a * a;
} else {
    result = b;
}
```
Data independence: Private conditionals
Should branching programs reveal atypical performance?

Obliv-C: traditional paradigm

```c
obliv int result;
obliv if (a >= b) {
    result = a * a;
} else {
    result = b;
}
```

EMP-toolkit: explicit branch selection

```c
Bit a_bigger = a.geq(b);
Integer result = b.select(a_bigger, a * a);
```
Data independence: Private conditionals

Should branching programs reveal atypical performance?

Obliv-C: traditional paradigm

```c
obliv int result;
obliv if (a >= b) {
    result = a * a;
} else {
    result = b;
}
```

EMP-toolkit: explicit branch selection

```c
Bit a_bigger = a.geq(b);
Integer result = b.select(a_bigger, a * a);
```

Recommendation

Depends on your users, but data independence is a good paradigm
Design decisions: Cryptographic abstraction level
Should the user have control over the underlying cryptographic representation?

Frigate: standard (C-style) abstraction

```c
int result = 0;
for (int i=0; i<LEN; i++) {
    result = result + (A.data[i] * B.data[i]);
}
```
Design decisions: Cryptographic abstraction level
Should the user have control over the underlying cryptographic representation?

Frigate: standard (C-style) abstraction

```c
int result = 0;
for (int i=0; i<LEN; i++) {
    result = result + (A.data[i] * B.data[i]);
}
```

PICCO: custom primitive, high level abstraction

```c
int result = A @ B;
```
Design decisions: Cryptographic abstraction level
Should the user have control over the underlying cryptographic representation?

**ABY: Low-level access**

```c
share *A, *B;
A = circ->PutMULGate(A, B);
A = circ->PutSplitterGate(A);
for (uint32_t i = 1; i < LEN; i++) {
    A->set_wire_id(0, circ->PutADDGate(A->get_wire_id(0),
                                   A->get_wire_id(i)));
}
A->set_bitlength(1);
share *result = circ->PutOUTGate(A, ALL);
```
Software engineering

Complicated, non-trivial build systems

- Set up certificate authority or PKI
- Compile specific OpenSSL version from source
- No dependency lists, manual search for compile errors
- Estimated time: 1-2 weeks per framework

Significant software projects

- Cryptographic protocols
- Distributed communication
- Interfacing with other systems

ObliVM: We couldn't return more than 32 bits
Software engineering

Complicated, non-trivial build systems

- Set up certificate authority or PKI
- Compile specific OpenSSL version from source
- No dependency lists, manual search for compile errors
- Estimated time: 1-2 weeks per framework

Significant software projects

- Cryptographic protocols
- Distributed communication
- Interfacing with other systems
Software engineering

Complicated, non-trivial build systems
- Set up certificate authority or PKI
- Compile specific OpenSSL version from source
- No dependency lists, manual search for compile errors
- Estimated time: 1-2 weeks per framework

Significant software projects
- Cryptographic protocols
- Distributed communication
- Interfacing with other systems
  - ObliVM: We couldn’t return more than 32 bits
Documentation

- **Language documentation**: How do I write secure code?
- **Code samples**: What does a working example look like?
- **Code documentation**: How does this example work?
- **Online support**: Where can I ask questions?
- **Open-source**: Can I run this without complex licensing?

Half the frameworks have no more than 3 of these 😐
Limited language documentation is frustrating

► CBMC-GC:

```c
int mpc_main(int alice, int bob) {
    return alice * bob;
}
```

```
$ make
[...]
Uncaught exception: Unknown literal: 33. Did you forget to return a value or assign a value to a OUTPUT variable?
```
Limited language documentation is frustrating

► CBMC-GC: Arguments must be called INPUT_<var>

```c
int mpc_main(int INPUT_alice, int INPUT_bob) {
    return INPUT_alice * INPUT_bob;
}
```

$ make
[...]
Gates: 5648 with 1986 Non-XOR and 0 LUTs
Depth: 151 with 32 Non-XOR
Limited language documentation is frustrating

- **CBMC-GC:** Arguments must be called `INPUT_<var>`
- **ObliVM:**

```c
int main(int alice, int bob){
    secure int result = alice * bob;
    return result;
}
```

$ ./run-compiler 12345 multiply.lcc

Was expecting one of: ⟨ IDENTIFIER ⟩ ... "[" ... "@" ... "i" ...
Limited language documentation is frustrating

- **CBMC-GC**: Arguments must be called `INPUT_<var>`
- **ObliVM**: `alice` and `bob` are reserved keywords

```c
int main(int aaaaa, int bbb)
{
    secure int result = aaaaa * bbb;
    return result;
}
```

```
$ ./run-compiler 12345 multiply.lcc
[INFO] The program type checks
[INFO] Compiling mult3.lcc succeeds
[INFO] Compilation finishes successfully.
```
Limited language documentation is frustrating

▶ CBMC-GC: Arguments must be called INPUT_<var>
▶ ObliVM: alice and bob are reserved keywords
▶ Wysteria:

```wysteria
let richer = \x:ps . \w:W x nat .

let b @ sec(x) =
  wfold x (w, 0, \accum:nat . \p:ps . \n:nat .
    if accum > n then accum
    else n )

in b

let all = { !Alice, !Bob } in
let w = (wire !Alice:10) ++ (wire !Bob:100) in
richer all w
```

$ wysteria –i-am Alice –gmw-port 9000 examples/tutorial.wy
File examples/fakemill.wy, line 1, character 16: syntax error at ‘:'
Limited language documentation is frustrating

- **CBMC-GC:** Arguments must be called \texttt{INPUT\_<var>}
- **ObliVM:** \texttt{alice} and \texttt{bob} are reserved keywords
- **Wysteria:** Language docs don’t account for parser limitations

```ocaml
let richer = \(x:ps\{\text{true}\}) . \(w:W \times \text{nat}\) .
let tmp \@ par(x) =
  let b \@ sec(x) =
  let result = \(wfold\) \(\times\) \[w ; 0;
    \(\text{accum:nat}\) . \(p:ps\{\text{true}\}\) . \(n:\text{nat}\) .
    if accum > n then accum
    else n \]
  in result
  in b
  in \text{wire} x:tmp
in let all = \{ !\text{Alice}, !\text{Bob} \} in
  let w = (\text{wire} !\text{Alice}:10) \text{++} (\text{wire} !\text{Bob}:100) in
  richer all w
```

\$ \text{wysteria \text{--i-am Alice \text{--gmw-port 9000 examples/tutorial.wy}}}
done with type checking the program
Limited language documentation is frustrating

- **CBMC-GC**: Arguments must be called `INPUT_<var>`
- **ObliVM**: `alice` and `bob` are reserved keywords
- **Wysteria**: Language docs don’t account for parser limitations
- **EMP-toolkit**: $\approx 1$ comment per 600 lines of code
Documentation appreciation and recommendations

Frameworks with excellent documentation

- **ABY**: 35-page language guide; only slightly out-of-date
- **SCALE-MAMBA**: 100+ pages of documentation
- **Sharemind**: Auto-generated language guide online
Documentation appreciation and recommendations

Frameworks with excellent documentation

▶ **ABY**: 35-page language guide; only slightly out-of-date
▶ **SCALE-MAMBA**: 100+ pages of documentation
▶ **Sharemind**: Auto-generated language guide online

Two recommendations for maintainers

▶ Multiple types of documentation drastically increase usability
▶ Online resources are sustainable and reduce workload
  ▶ Produces a living FAQ
  ▶ Allows users to interact
Documentation appreciation and recommendations

Frameworks with excellent documentation

▶ **ABY**: 35-page language guide; only slightly out-of-date
▶ **SCALE-MAMBA**: 100+ pages of documentation
▶ **Sharemind**: Auto-generated language guide online

Two recommendations for maintainers

▶ Multiple types of documentation drastically increase usability
▶ Online resources are sustainable and reduce workload
  ▶ Produces a living FAQ
  ▶ Allows users to interact

Documentation issues aren’t fundamental
IARPA HECTOR includes usability criteria
Future directions in MPC frameworks

Seamless support for multiple settings
▶ Most frameworks implement a single threat model and protocol

Take advantage of work in other disciplines
▶ Heavy-duty circuit compilers (TinyGarble)
▶ Formal guarantees about program operation (Wysteria, ObliVM)

Maintaining the repository
▶ Recently added: MP-SPDZ, JIFF
▶ Please contribute your framework!
  ABY$^3$, EzPC, FRESCO, HyCC, MPyC
General Purpose Frameworks for Secure Multi-party Computation

Marcella Hastings
Brett Hemenway
Daniel Noble
Steve Zdancewic

University of Pennsylvania

github.com/mpc-sok/frameworks
FAQ: How fast are they? Which one is fastest?

A: Kind of fast
  - O(30 seconds) for `mult3`
  - O(3 minutes) for `xtabs`

A: It’s hard to compare them
  - Different architectures
  - Different intermediate representations
  - Different threat models