# SoK: General Purpose Frameworks for Secure Multi-party Computation

**Marcella Hastings** Brett Hemenway Daniel Noble Steve Zdancewic

University of Pennsylvania

## Secure Multi-party Computation (MPC)

Compute an arbitrary function among mutually distrustful parties

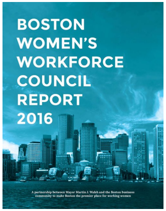
## Secure Multi-party Computation (MPC)

Compute an arbitrary function among mutually distrustful parties

- Set beet prices at auction [BCD+09]
- Input: Beet quantities and prices
- Output: Market clearing price

## Secure Multi-party Computation (MPC)

Compute an arbitrary function among mutually distrustful parties



BOSTON WOMEN'S WORKFORCE COUNCIL REPORT 2016

A partnership between Mayor Martin J. Walsh and the Boston business community to make Boston the premier place for working women

- Set beet prices at auction [BCD+09]
- Input: Beet quantities and prices
- Output: Market clearing price

- Compute statistics on sensitive data [LVB+16,BLV17]
- Input: Salary and payroll data from 150 companies
- Output: Financial statistics and analytics

## 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]

Fair Play
Secure Function Evaluation

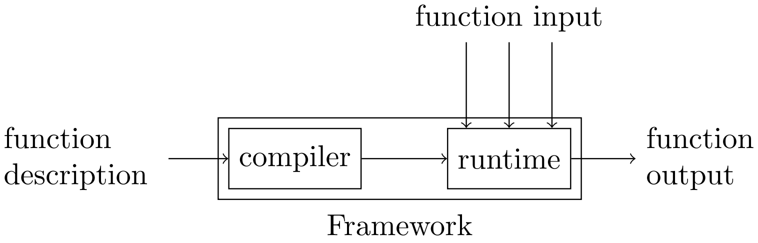
## Motivating end-to-end frameworks for MPC

- Custom one-off solutions are unsustainable
- Protocols assumed impractical until Fairplay [MNPS04]

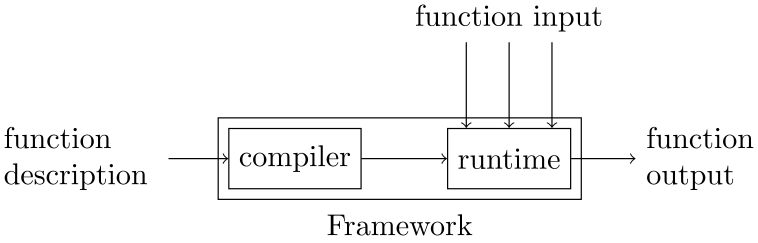
Fair Play
Secure Function Evaluation

- Performance improvements rapidly advanced state-of-the-art
  - OT extension [YKNP03]
  - Free XOR gates [KS08]
  - Half-gates [ZRE15]
  - AES-NI

## Modern General-Purpose Frameworks



## 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

## Frameworks: A brief overview

| | | Protocol family | Parties | Semi-honest | Malicious |
|---|---|---|---|---|---|
| EMP-toolkit | [WMK17] | GC | 2 | ● | ● |
| Obliv-C | [ZH15] | GC | 2 | ● | ○ |
| ObliVM | [LWNHS15] | GC | 2 | ● | ○ |
| TinyGarble | [SHSSK15] | GC | 2 | ● | ○ |
| Wysteria | [RHH14] | MC | 2+ | ● | ○ |
| ABY | [DSZ15] | GC,MC | 2 | ● | ○ |
| SCALE-MAMBA | - | Hybrid | 2+ | ● | ● |
| Sharemind | [BLW08] | Hybrid | 3 | ● | ○ |
| PICCO | [ZSB13] | Hybrid | 3+ | ● | ○ |
| Frigate | [MGCKT16] | - | 2+ | - | - |
| CBMC-GC | [HFKV12] | - | 2+ | - | - |

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

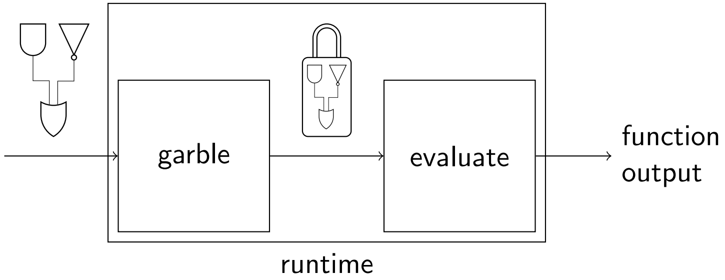
## Frameworks: A brief overview

| | | Protocol family | Parties | Semi-honest | Malicious |
|---|---|---|---|---|---|
| EMP-toolkit | [WMK17] | GC | 2 | ● | ● |
| Obliv-C | [ZH15] | GC | 2 | ● | ○ |
| ObliVM | [LWNHS15] | GC | 2 | ● | ○ |
| TinyGarble | [SHSSK15] | GC | 2 | ● | ○ |
| Wysteria | [RHH14] | MC | 2+ | ● | ○ |
| ABY | [DSZ15] | GC,MC | 2 | ● | ○ |
| SCALE-MAMBA | - | Hybrid | 2+ | ● | ● |
| Sharemind | [BLW08] | Hybrid | 3 | ● | ○ |
| PICCO | [ZSB13] | Hybrid | 3+ | ● | ○ |
| Frigate | [MGCKT16] | - | 2+ | - | - |
| CBMC-GC | [HFKV12] | - | 2+ | - | - |

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

## Garbled circuit protcols

Introduced by [Yao82, Yao86]



- Function represented as Boolean circuits
- Typically semi-honest, 2-party

## Frameworks: A brief overview

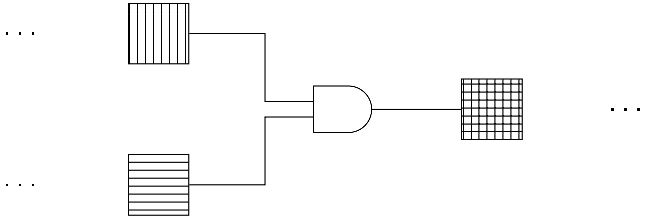
| | | Protocol family | Parties | Semi-honest | Malicious |
|---|---|---|---|---|---|
| EMP-toolkit | [WMK17] | GC | 2 | ● | ● |
| Obliv-C | [ZH15] | GC | 2 | ● | ○ |
| ObliVM | [LWNHS15] | GC | 2 | ● | ○ |
| TinyGarble | [SHSSK15] | GC | 2 | ● | ○ |
| Wysteria | [RHH14] | MC | 2+ | ● | ○ |
| ABY | [DSZ15] | GC,MC | 2 | ● | ○ |
| SCALE-MAMBA | - | Hybrid | 2+ | ● | ● |
| Sharemind | [BLW08] | Hybrid | 3 | ● | ○ |
| PICCO | [ZSB13] | Hybrid | 3+ | ● | ○ |
| Frigate | [MGCKT16] | - | 2+ | - | - |
| CBMC-GC | [HFKV12] | - | 2+ | - | - |

GC = Garbled Circuit

MC = Multi-party circuit-based

## Multi-party circuit-based protcols

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)

## Frameworks: A brief overview

| | | Protocol family | Parties | Semi-honest | Malicious |
|---|---|---|---|---|---|
| EMP-toolkit | [WMK17] | GC | 2 | ● | ● |
| Obliv-C | [ZH15] | GC | 2 | ● | ○ |
| ObliVM | [LWNHS15] | GC | 2 | ● | ○ |
| TinyGarble | [SHSSK15] | GC | 2 | ● | ○ |
| Wysteria | [RHH14] | MC | 2+ | ● | ○ |
| ABY | [DSZ15] | GC,MC | 2 | ● | ○ |
| SCALE-MAMBA | - | Hybrid | 2+ | ● | ● |
| Sharemind | [BLW08] | Hybrid | 3 | ● | ○ |
| PICCO | [ZSB13] | Hybrid | 3+ | ● | ○ |
| Frigate | [MGCKT16] | - | 2+ | - | - |
| CBMC-GC | [HFKV12] | - | 2+ | - | - |

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

## Inner product: Illustrating language abstractions

### Frigate: standard (C-style) abstraction

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

## Inner product: Illustrating language abstractions

Frigate: standard (C-style) abstraction

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

PICCO: custom primitive, high level abstraction

```
int result = A @ B;
```

## Inner product: Illustrating language abstractions

### ABY: Low-level access

```
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

## 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 buying something?

Half the frameworks have no more than 3 of these ☹

## Limited language documentation is frustrating

- CBMC-GC:

```
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>`

```
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:

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

```
$ ./run-compiler 12345 multiply.lcc
[ERROR] Error: Parsing Error Encountered " "alice" "alice "" at
line 3, column 21.
Was expecting one of: ⟨ IDENTIFIER ⟩ ... "[" ... "@" ... "¡" ...
```

## Limited language documentation is frustrating

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

```
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: Language docs don't account for parser limitations

## 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: ≈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

## What's next for MPC?

Engineering and usability challenges aside, MPC is in good shape!

Usability challenges have been acknowledged (IARPA HECTOR)

Consider working with programming languages researchers

Our repository is actively maintained!

# SoK: General Purpose Frameworks for Secure Multi-party Computation

**Marcella Hastings** Brett Hemenway Daniel Noble Steve Zdancewic

University of Pennsylvania

`github.com/mpc-sok/frameworks`