Constant Round Maliciously Secure 2PC with Function-independent Preprocessing using LEGO

Jesper Buus Nielsen, Thomas Schneider and Roberto Trifiletti

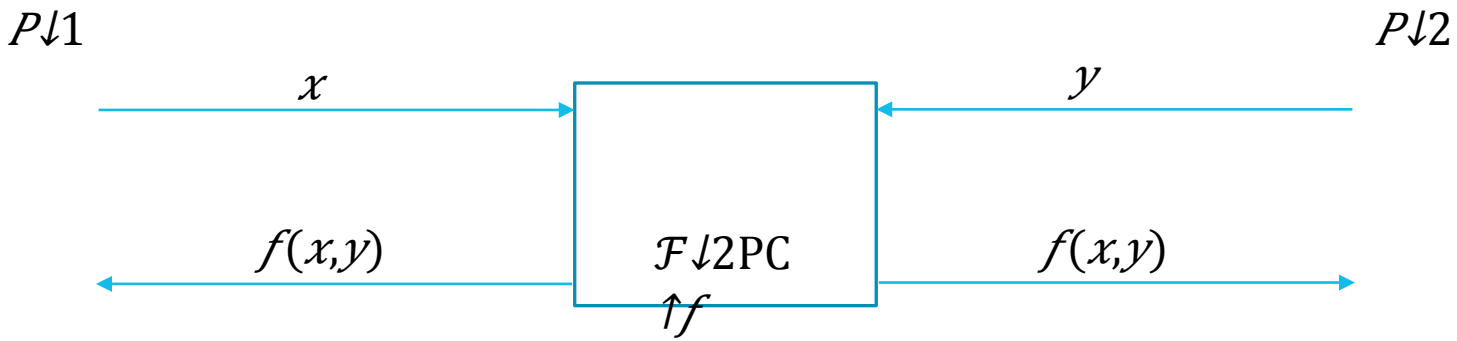
TECHNISCHE UNIVERSITÄT DARMSTADT

AARHUS UNIVERSITY

Outline

1. Intro to Secure Two-party Computation
2. Protocol Overview
3. Experimental Results

Secure 2PC



Nothing but the output $f(x,y)$ is revealed to the parties.

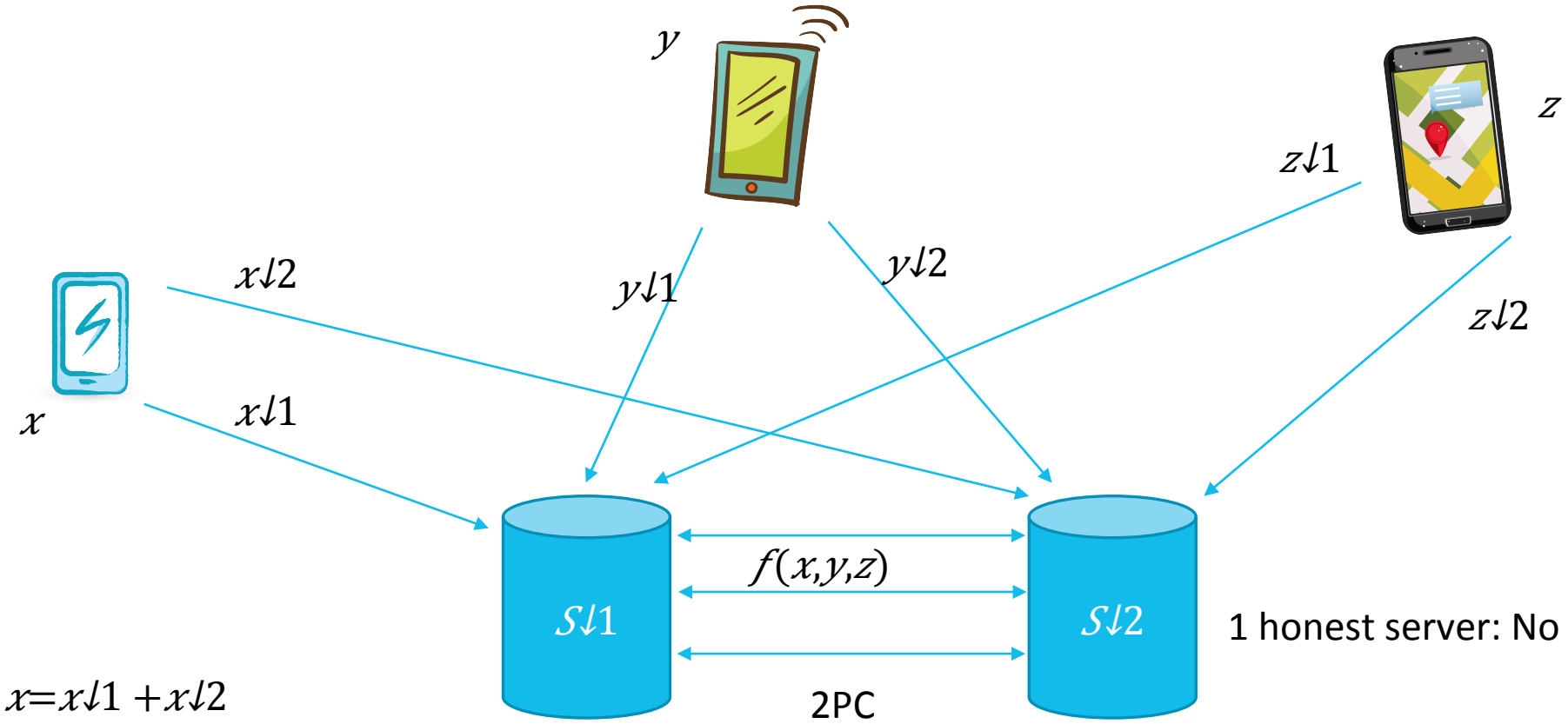
Task: realize above scenario using a cryptographic protocol.

Powerful: can build most other crypto from secure computation.

Applications:

- Privacy preserving data analysis
- Secure outsourcing
- Company benchmarking
- Satellite collision detection

Example Application: Secure outsourcing



1 honest server: No client info can leak

Secret sharing: $x=x\downarrow 1 + x\downarrow 2$

Security models

Two main types

- Semi-honest: The servers run the protocol/code as prescribed. Guaranteed that data cannot leak if servers do not collude.
  - Protects against breaches “after-the-fact”, but not if a server is taken over during computation.
- Malicious: No assumptions on server behavior. As long as one server is honest, data cannot leak.
  - Protects against online attacks, robustness.

Security at a price

- Malicious security much harder/expensive than semi-honest. Often 10-100x in computation/ communication.

In this Work

First implementation of *constant round* malicious 2PC with *function-independent* preprocessing

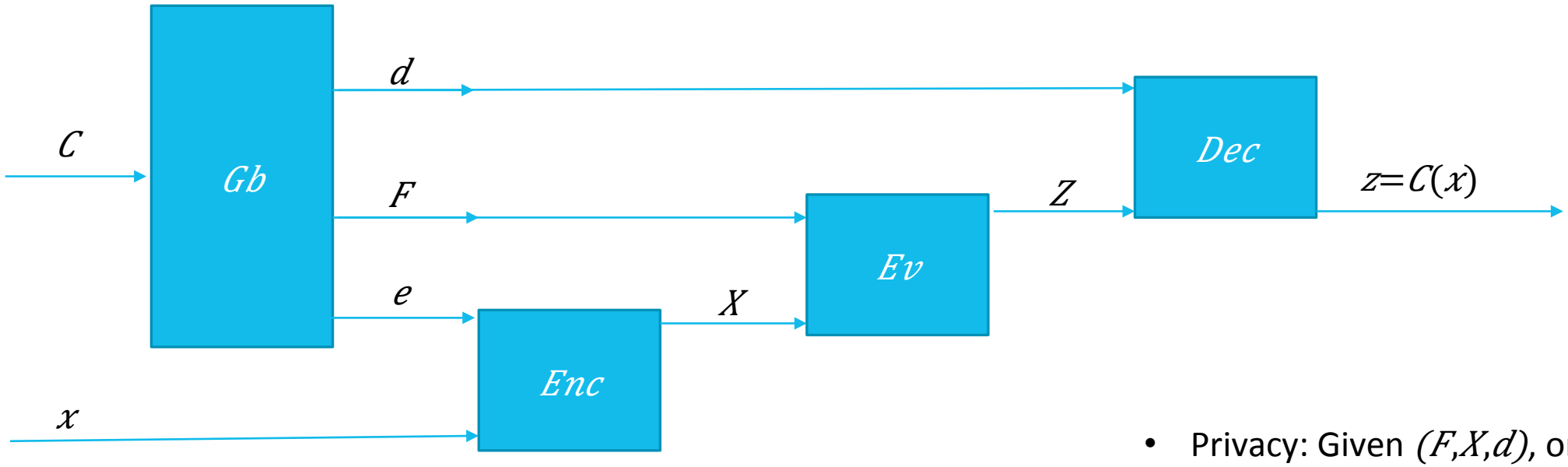
- Allows the servers to run up to 90% of the total computation independent of clients and function(s).
- Function-dependent computation matches the semi-honest setting.
- Improves clients' experience as latency is significantly reduced.

Show for the first time that LEGO technique for malicious 2PC is highly practical.

- Up to 50x faster than previous protocols if ignoring cost of independent preprocessing.
- Within factor 3x if comparing total costs.

Garbling Schemes [BHR12]

$G=(Gb,Enc,Ev,Dec)$



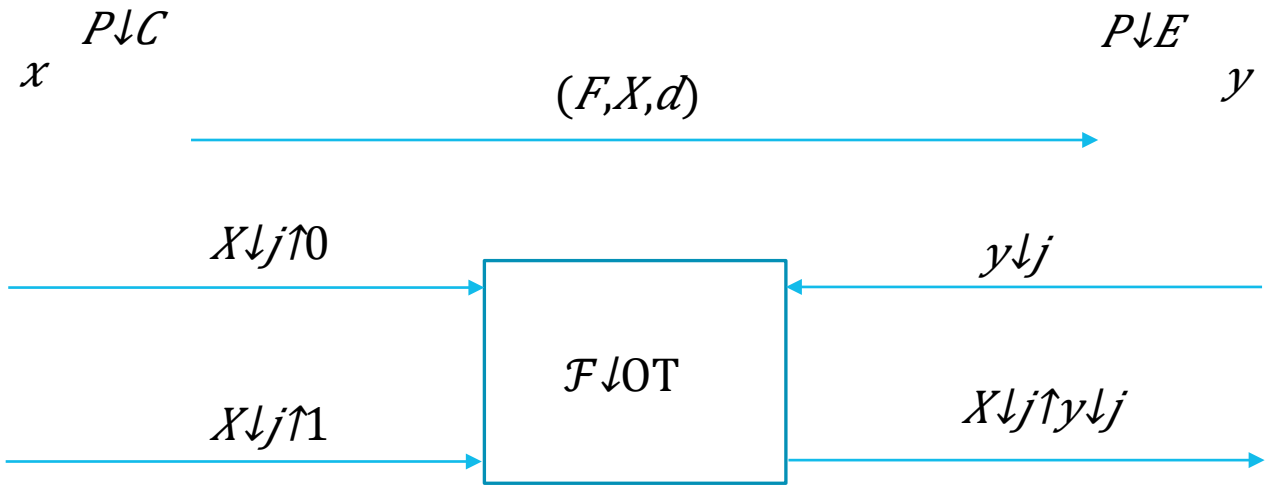
- Privacy: Given (F,X,d), only learn $C(x)$.
- Optimization: Free-XOR [KS08], no data transfer for XOR gates.

Semi-honest: Yao's garbled circuits

$(F,e,d) \leftarrow Gb(C)$

$(X\downarrow 1\uparrow 0, X\downarrow 1\uparrow 1, \ldots, X\downarrow n\uparrow 0, X\downarrow n\uparrow 1) \leftarrow e$

$X \leftarrow Enc(x,e)$



$Y \leftarrow (X\downarrow 1, X\downarrow 2, X\downarrow n)$

$Z \leftarrow Ev(F,Y)$

$z \leftarrow Dec(Z,d)$

$z = C(x,y)$

Malicious adversary

Yao's garbled circuits completely break against malicious behavior.

- $P{\downarrow}C$ can garble $C' \neq C$ and $P{\downarrow}E$ would never know.
- Selective Failure Attack: Make $P{\downarrow}E$ abort depending on his input (thus leaking information about y).

Malicious: “Standard” Cut-and-choose

Main idea

- Send multiple garblings $F\downarrow 1$,$F\downarrow 2$,…,$F\downarrow m$, check some, evaluate the rest.
- Not trivial to ensure nothing can go wrong.

Replication cost

- [Bra13,HKE13,Lin13]: s circuits gives $2\uparrow -s$ security.
- 40-80x blowup in communication/computation.

Amortization

- [LR15,RR16]: $\mathcal{O}(s/\log(\#C))$ circuits gives $2\uparrow -s$, i.e. cut-and-choose overhead is amortized over multiple individual computations of C.

LEGO

[NO09] introduced LEGO technique for maliciously secure 2PC based on cut-and-choose of Garbled Circuits.

Considers gates instead of circuits for cut-and-choose.

- Asymptotic improvement, $\mathcal{O}(s/\log(|C|))$ vs $\mathcal{O}(s)$.
- Allows preprocessing that is independent of C.
- Requires "soldering" individual gates to form a circuit using homomorphic commitments.

[NO09] downsides

- Expensive public-key operations for each gate of the circuit.
- Incompatible with optimizations of Yao's garbled circuits.

[FJNNO13, FJNT15, FJNT16] Improvements

- Eliminate public-key operations for each gate.
- Compatible with all known optimizations.
- Efficient XOR-homomorphic commitment scheme based on ECC and OT.

Folklore: LEGO is asymptotically efficient, but not practical due to the commitment overhead.

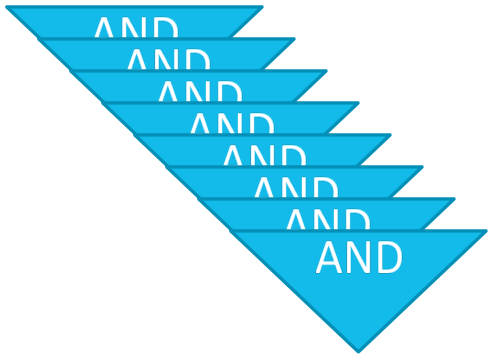
Outline

1. ~~Intro to Secure Two-party Computation~~
2. Protocol Overview
3. Experimental Results

Phase 1: Preprocessing

$n\cdot$ AND $P\downarrow C$

$P\downarrow E$ $n\cdot$ AND



Send Gates G=$\{G\downarrow i\}$ and commit to all inp/out wires

Send check-set $V\subset G$

Decommit to input and output wires of gates in V

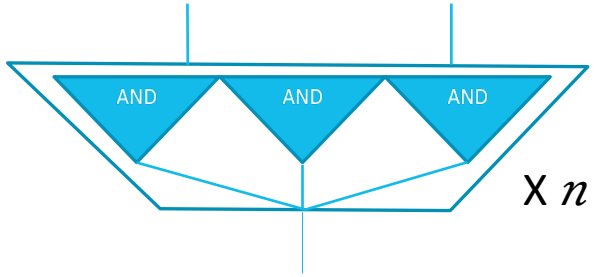
Check that they correspond to AND gates

If OK!

Permutation B specifying bucketing

Very parallelizable!

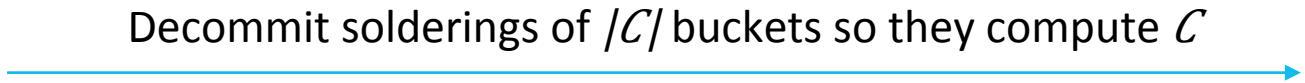
Decommit solderings of remaining gates in according to B

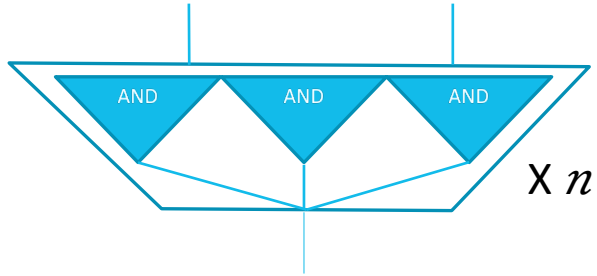


Everything so far independent of final functionality C

Phase 2: Function soldering

Decommit solderings of $|C|$ buckets so they compute C

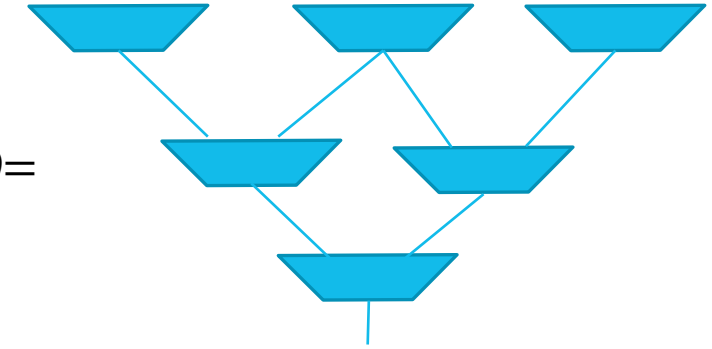




Data transfer cost:

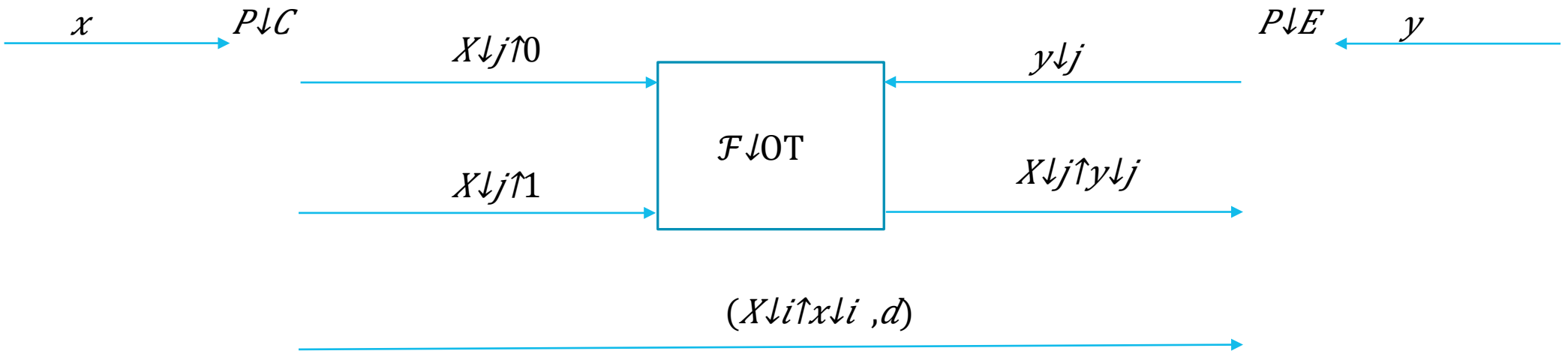
- $2 \cdot |C|$ decommits
- With [FJNT16] commit scheme: $2 \cdot |C| \cdot k + c$ (~1 garbled circuit).
- Non-LEGO: $\mathcal{O}(s \cdot |C| \cdot k)$

$Gb(C) =$



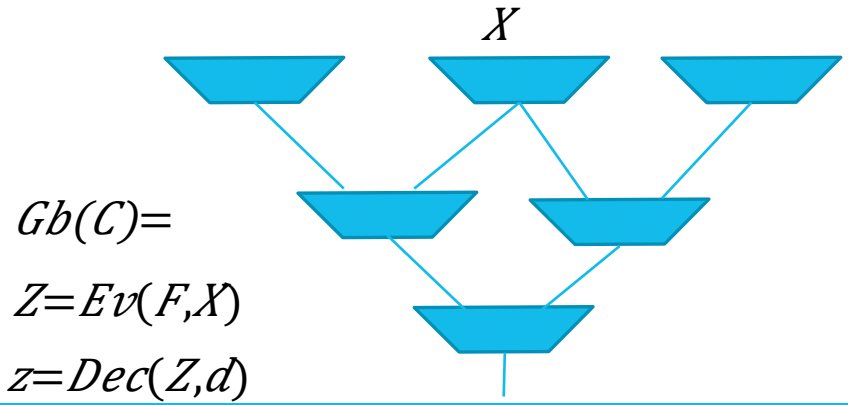
k comp. security param, s stat. security param.

Phase 3: Evaluation



Highlights:

- LEGO: Single set of input keys vs. non-LEGO: one per eval circuit.
- Optimal 2 rounds (3 if $P\downarrow C$ gets output)
- Computation: Evaluating $\mathcal{O}(s/\log(|C|)\ )$ garbled circuits.



Outline

1. ~~Intro to Secure Two-party Computation~~
2. ~~Protocol Overview~~
3. Experimental Results

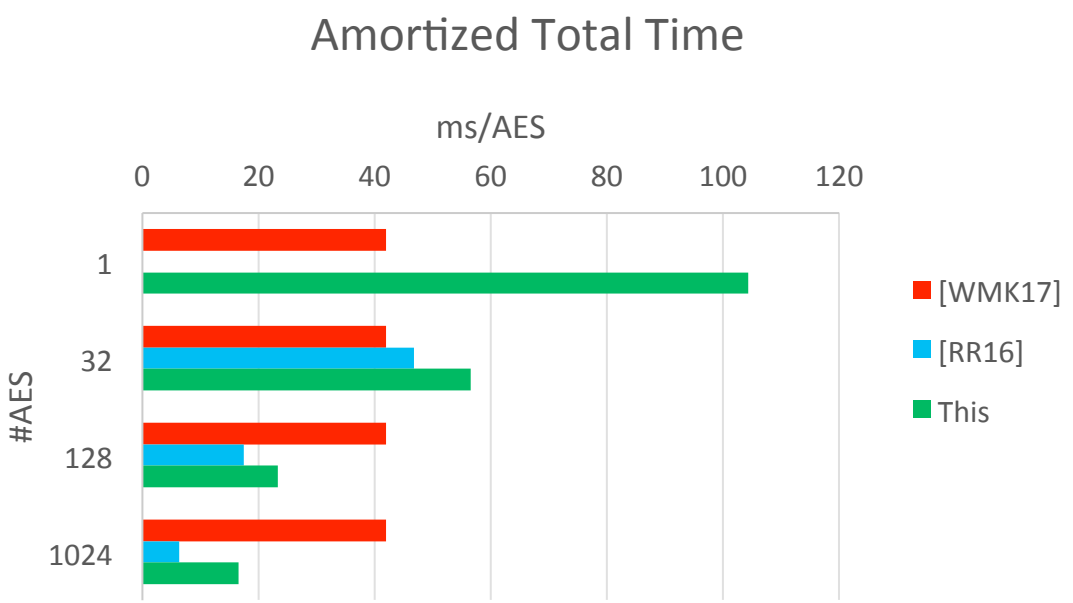
Observations

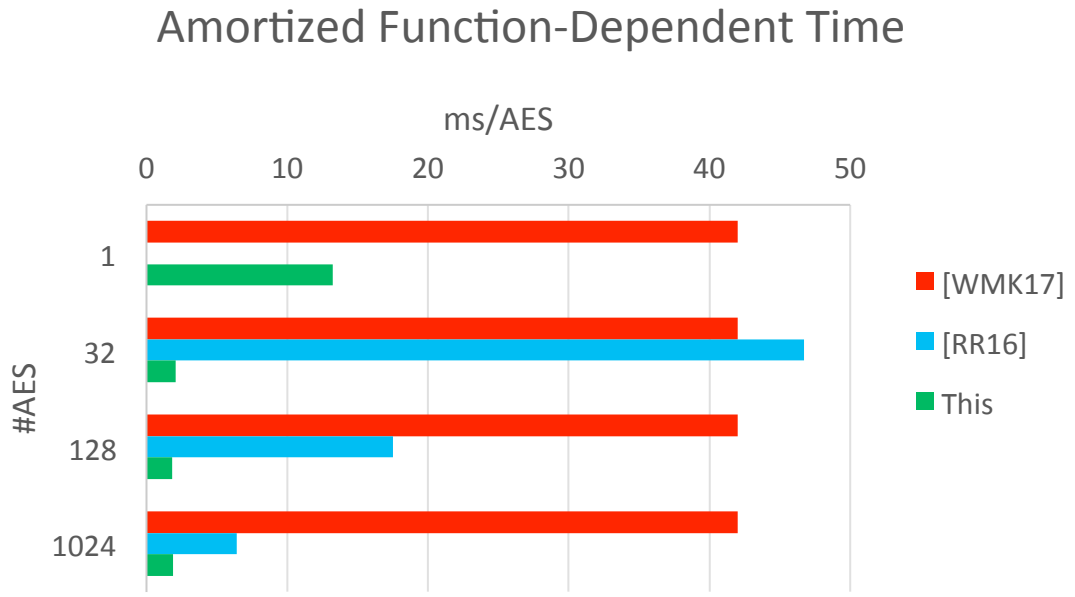
The overhead of the commitments dominate the preprocessing phase, $\sim 70\%$ of total time.

- Spent great care optimizing the commitment scheme implementation.
- Includes utilizing efficient BitMatrix transposition and Intel AVX instructions for computing several linear combinations in parallel over hundreds of millions of values.

Clear that network bandwidth is the major bottleneck.

Performance Comparison (AES-128)





AWS c4.8x instances, LAN

[WMK17]: “Faster Two-Party Computation Secure Against Malicious Adversaries in the Single-Execution Setting”, Eurocrypt 17
[RR16]: “Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution”, USENIX 16

Source: **https://github.com/AarhusCrypto/TinyLEGO**

In Conclusion

LEGO is competitive with state-of-the-art 2PC,

and even surpasses previous best results if utilizing function-independent preprocessing.

Thank you

References

[BHR12] Mihir Bellare, Viet Tung Hoang, Phillip Rogaway: **Foundations of Garbled Circuits, CCS 2012.**

[KS08] Vladimir Kolesnikov, Thomas Schnieder: **Improved Garbled Circuit: Free XOR Gates and Applications, ICALP 2008**

[NO09] Jesper Buus Nielsen, Claudio Orlandi: **LEGO for Two Party Secure Computation, TCC 2009.**

[FJNNO13] Tore Kasper Frederiksen, Thomas P. Jakobsen, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi: **MiniLEGO: Efficient Secure Two-Party Computation From General Assumptions, Eurocrypt 2013.**

[FJNT15] Tore Kasper Frederiksen, Thomas P. Jakobsen, Jesper Buus Nielsen, Roberto Trifiletti: **TinyLEGO: An Interactive Garbling Scheme for Maliciously Secure Two-party Computation, ePrint 2015.**

[FJNT16] Tore Kasper Frederiksen, Thomas P. Jakobsen, Jesper Buus Nielsen, Roberto Trifiletti: **On the Complexity of Additively Homomorphic UC Commitments, TCC-A 2016.**

[RR16] Peter Rindal, Mike Rosulek: **Faster Malicious 2-Party Secure Computation with Online/Offline Dual Execution, USENIX 2016.**

[WMK17] Xiao Wang, Alex J. Malozemoff, Jonathan Katz: **Faster Two-Party Computation Secure Against Malicious Adversaries in the Single-Execution Setting, Eurocrypt 2017.**