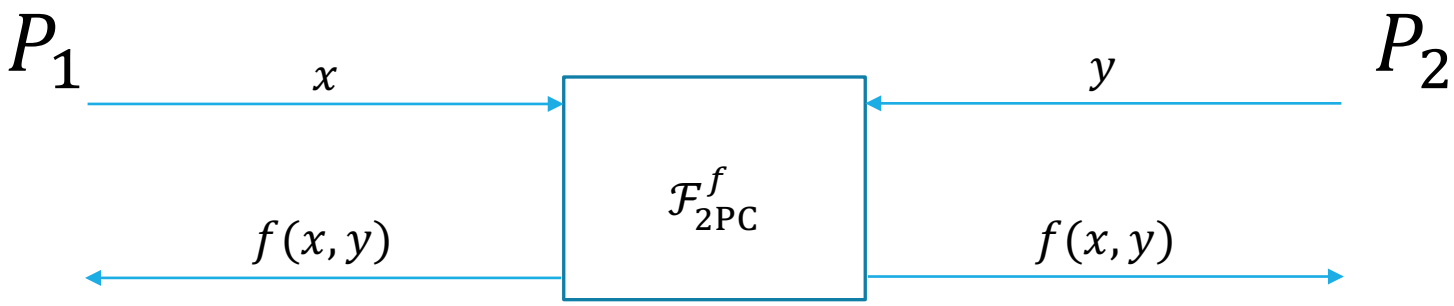
Implementing LEGO

Roberto Trifiletti

Outline

1. Secure Two-party Computation Based on Garbled Circuits.
2. Implementing TinyLEGO using Efficient XOR-Homomorphic Commitments [NST17].
3. DUPLO, a Unified Approach to Cut-and-Choose [Ongoing work].

Secure 2PC



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

Task: realize above scenario using a cryptographic protocol.

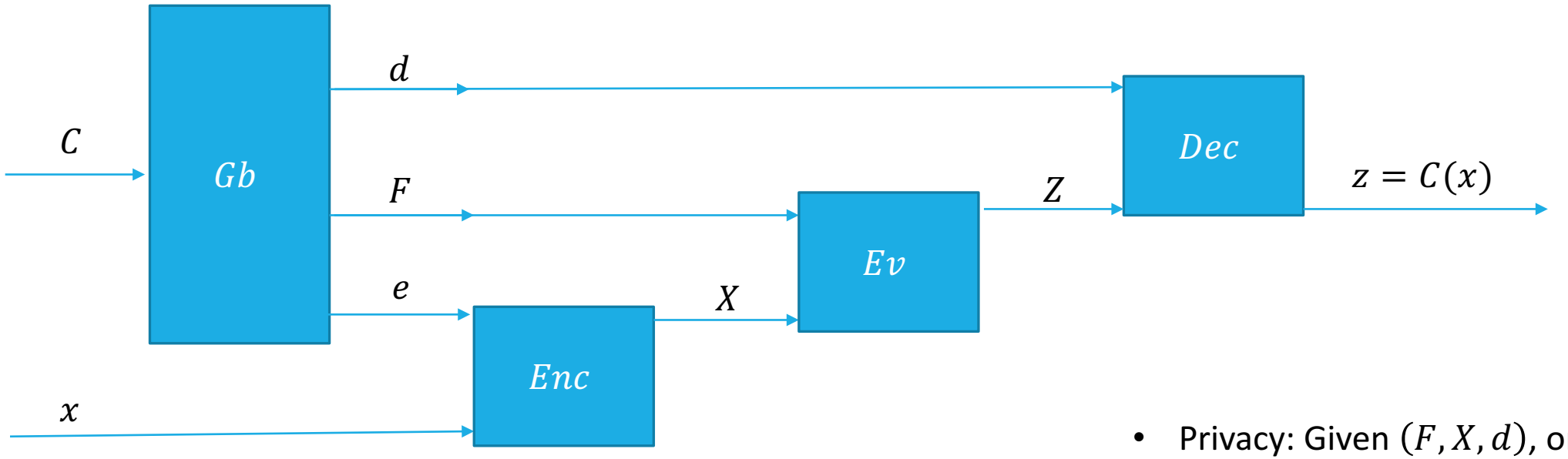
Powerful: can build most other crypto from secure computation.

Applications:

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

Garbling Schemes [BHR12]

$$\mathrm{G} = (Gb, Enc, Ev, Dec)$$



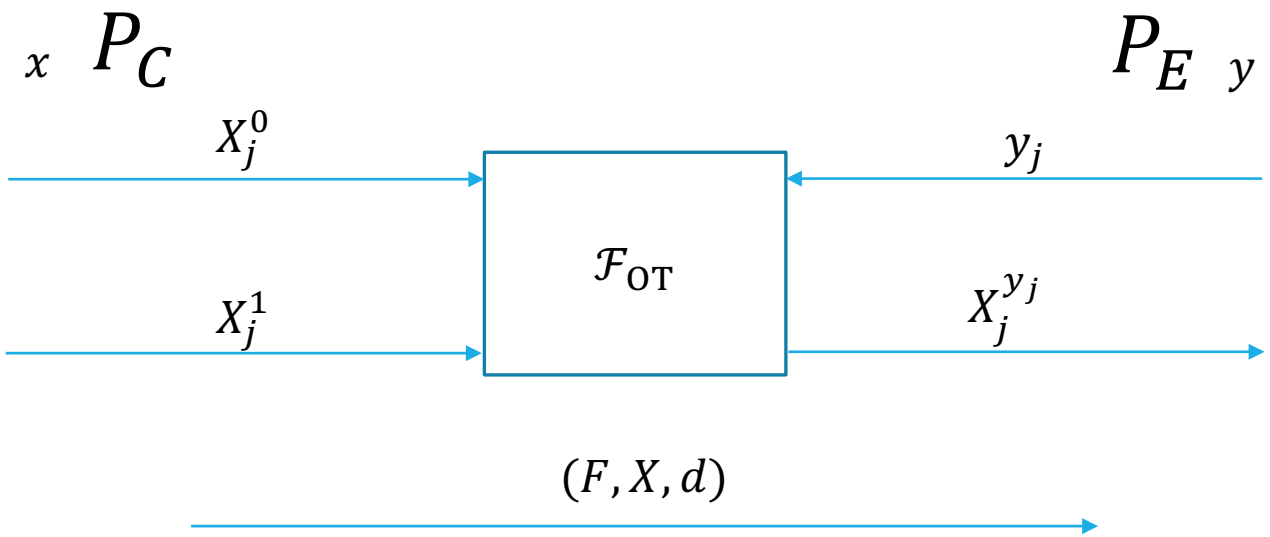
- Privacy: Given (F, X, d), only learn $C(x)$.
- Optimization: Free-XOR [KS08], no data transfer for XOR gates.
- Using Half-Gates [ZRE15], $2k$ bits per AND gate.

Semi-honest: Yao's garbled circuits

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

$(X_1^0, X_1^1, \ldots, X_n^0, X_n^1) \leftarrow e$

$X \leftarrow Enc(x, e)$



$Y \leftarrow (X_1, X_2, X_n)$

$Z \leftarrow Ev(F, Y)$

$z \leftarrow Dec(Z, d)$

$z = C(x, y)$

Malicious adversary

Yao's garbled circuits completely break against *malicious* behavior

- P_C can garble $C' \neq C$ and P_E would never know.
- Selective Failure Attack: Make P_E abort *depending* on his input (thus leaking information about y).

Malicious: “Standard” Cut-and-choose

Main idea

- Send multiple garblings $F_1, F_2, \dots, F_m$, check some, evaluate the rest.
- Not trivial, but possible with care.

Replication cost

- [Bra13, HKE13, Lin13]: s circuits gives 2^{-s} security.
- 40-80x blowup in communication/computation.

Amortization

- [HKKKM14, LR14, LR15, RR16]: $O(s/\log(\#C))$ circuits gives 2^{-s}, i.e. cut-and-choose overhead is amortized over multiple individual computations of C.
- [NO16] Cross&Clean O(1) circuits, mostly theoretical result.

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.
- More efficient bucketing strategy.

How to Solder

Free-XOR: $L_j^1 = L_j^0 \oplus \Delta$, $R_j^1 = R_j^0 \oplus \Delta$, $O_j^1 = O_j^0 \oplus \Delta$ for all AND_j gates.

- A soldering between two wires is then the difference between the 0-labels (truth value):

$$S_{ij}^L = L_i^0 \oplus L_j^0, \qquad S_{ij}^R = R_i^0 \oplus R_j^0, \qquad S_{ij}^O = O_i^0 \oplus O_j^0$$

- Ex: When learning labels X_0^b can now compute

$$X_0^b \oplus S_{01} = (X_0^0 \oplus b \cdot \Delta) \oplus (X_0^0 \oplus X_1^0) = X_1^0 \oplus b \cdot \Delta = X_1^b.$$

Outline

1. ~~Secure Two-party Computation Based on Garbled Circuits.~~
2. Implementing TinyLEGO using Efficient XOR-Homomorphic Commitments [NST17].
3. DUPLO, a Unified Approach to Cut-and-Choose [Ongoing work].

[NST17]

Efficient C++ implementations of [FJNT16] commitment scheme and [FJNT15] LEGO protocol, both heavily utilizing parallelization for maximal performance.

New approach to dealing with Selective Failure Attack on P_E's input

- No increase of P_E's input as with XOR-tree (s-probe matrix) approach.
- Saves on #OTs, communication and computation.
- XOR-homomorphic commitments combined with *globally correlated* Δ-ROT.

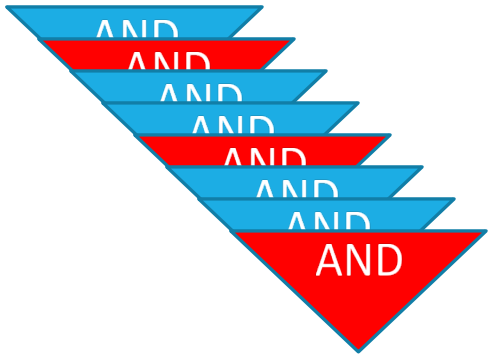
Tightened the analysis of current construction of Δ-ROT (all [IKNP03]-like OT extensions)

- $\sim 5x$ reduction in Δ-ROT communication cost for $k = 128, s = 40$.
- Core part of [NNOB12] "TinyOT". Benefits recent works of [WRK17a, WRK17b, HSV17].

Phase 1: Preprocessing

$n \cdot$ AND → P_C

P_E ← $n \cdot$ AND



Send Gates $\mathrm{G} = \{G_i\}$ and commit to all inp/out wires →

← Send check-set $V \subset G$

Decommit to input and output wires of gates in V →

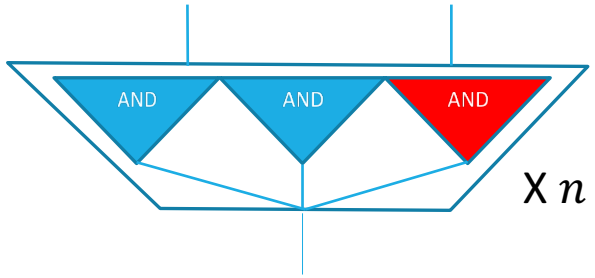
Check that they correspond to AND gates

If OK!

← Permutation B specifying bucketing

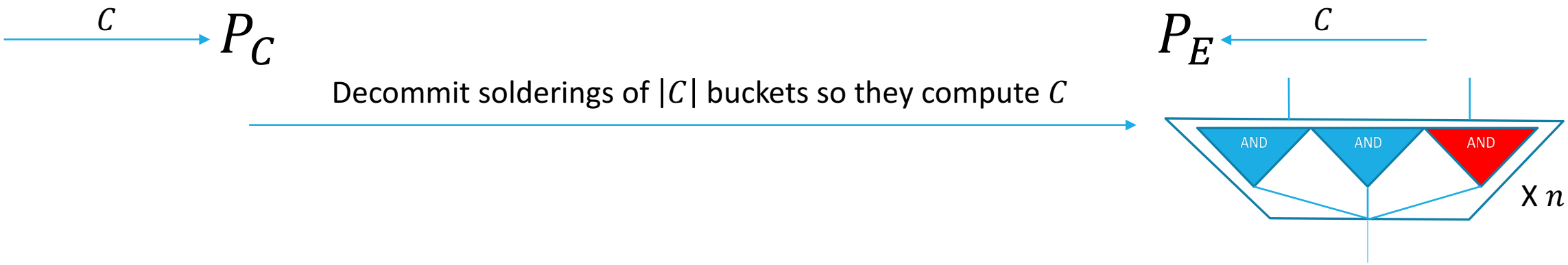
Very parallelizable!

Decommit solderings of remaining gates in according to B →



Everything so far independent of final functionality C

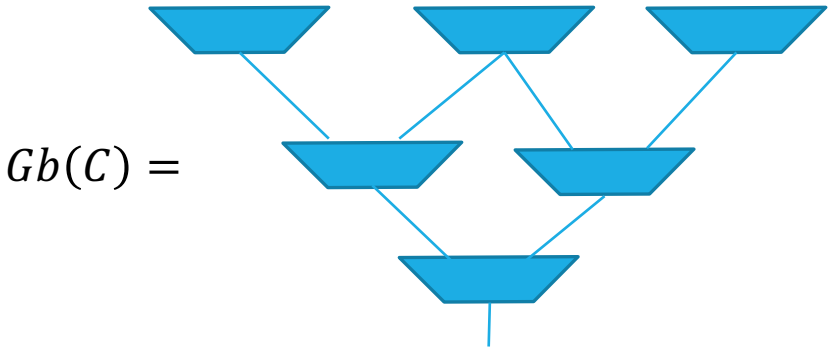
Phase 2: Function soldering



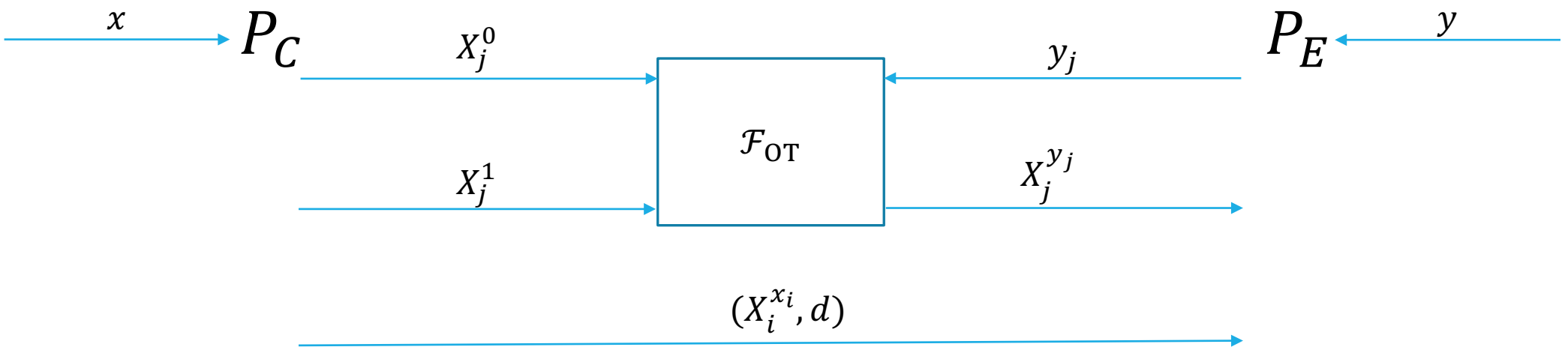
Data transfer cost:

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

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

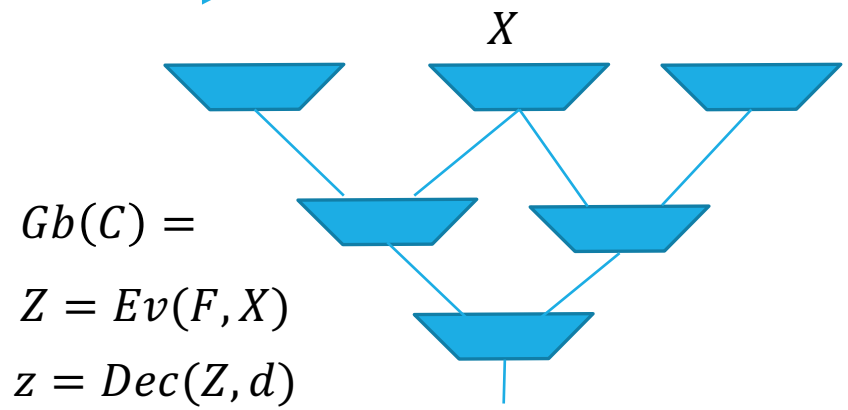


Phase 3: Evaluation

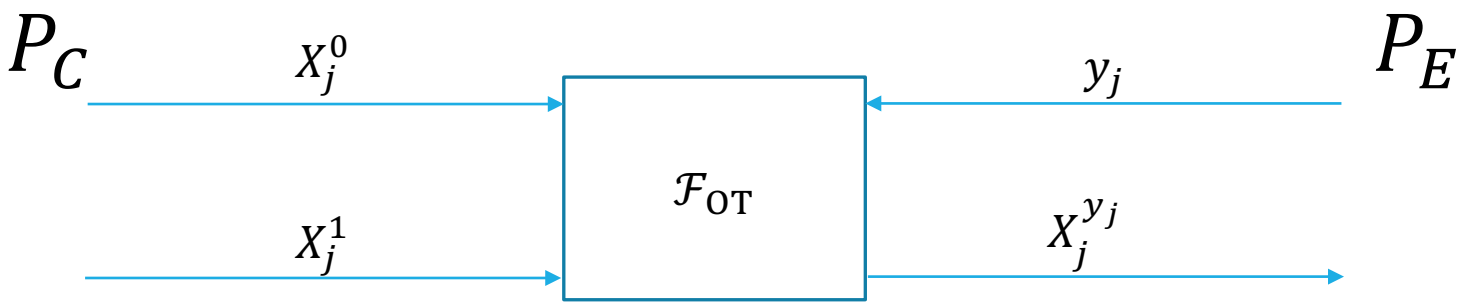


Highlights:

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



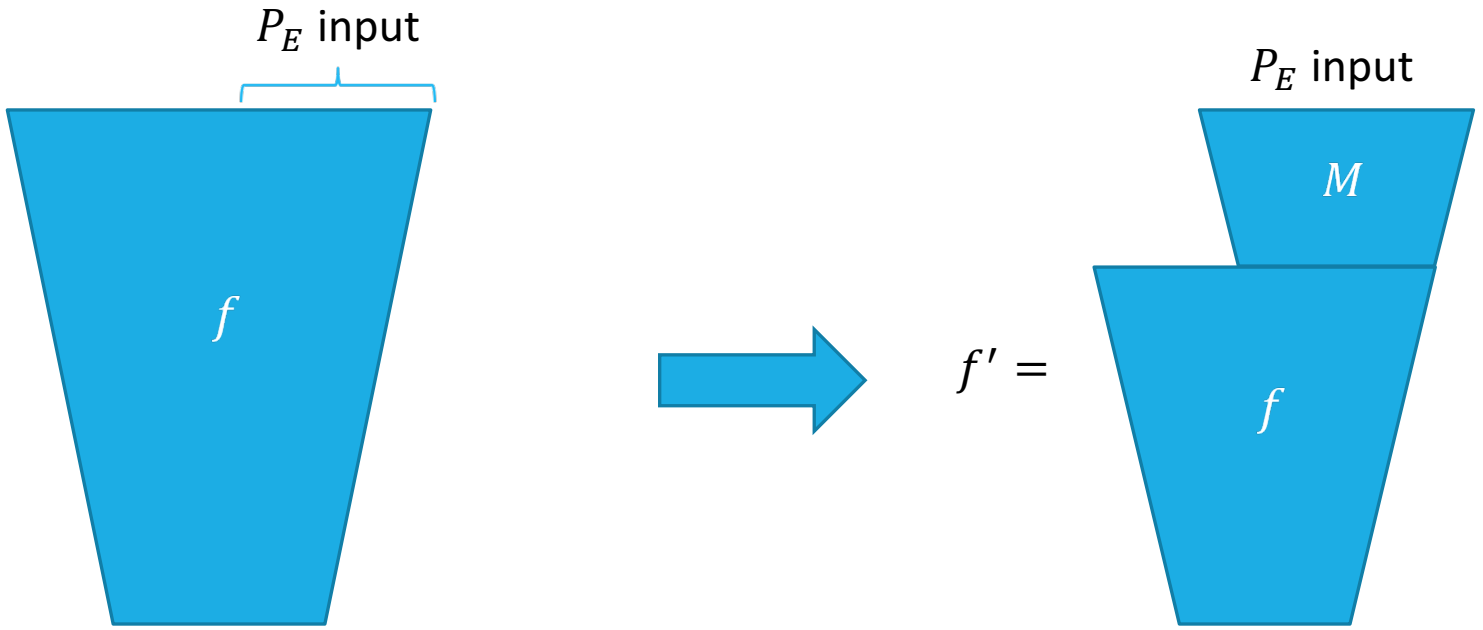
Selective Failure Attack [MF06, KS06]



However, say P_C inputs a value $\widehat{X_j^0} \notin \{X_j^0, X_j^1\}$ for one of the input OTs.

- If P_E does not abort => input is 1.
- If P_E aborts => input is 0.
- In general, P_C can learn l input bits with probability 2^{-l}.
- Hard to deal with due to nature of OT (P_E learns only one value).

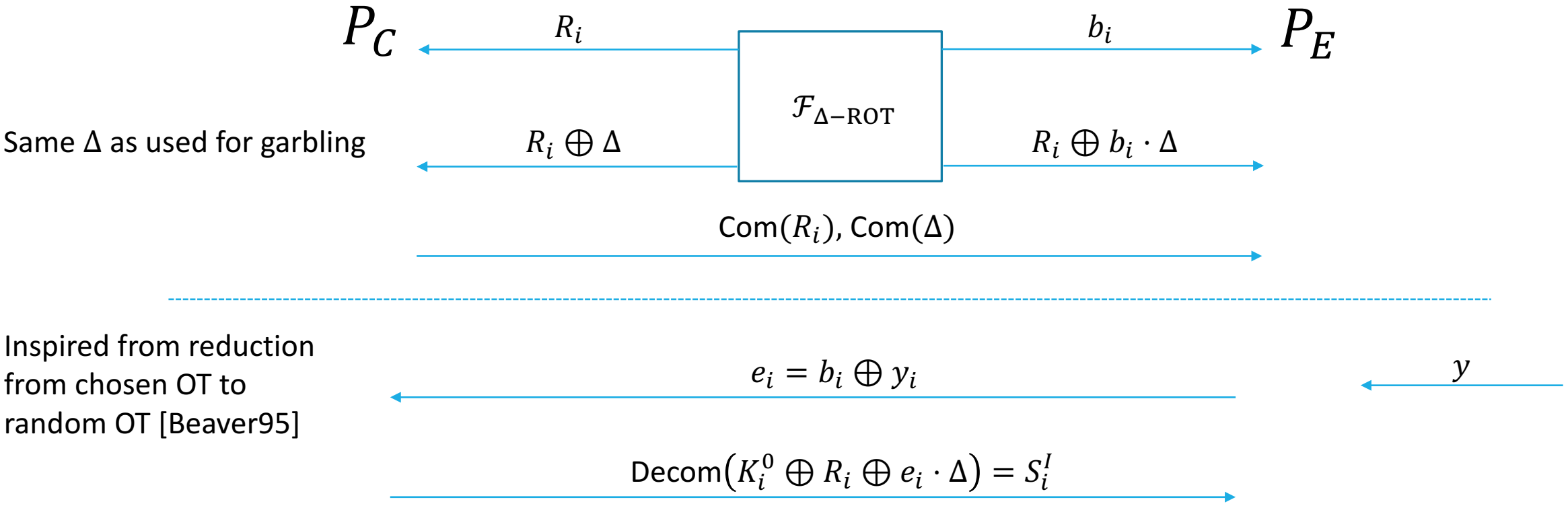
Standard approach: XOR-tree (s-probe matrix) [LP07, sS11]



Guarantee: Learning up to s input bits of P_E for new function $f' = M \circ f$ does not leak anything about the real input y to f.

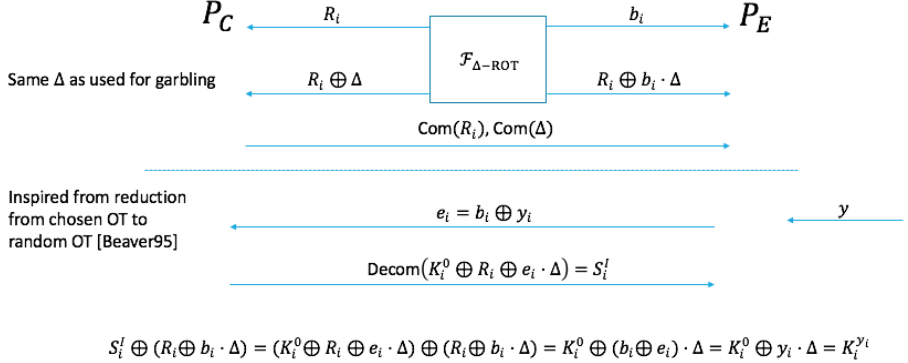
- Increases #OTs, communication and computation time.

New selective failure approach [NST17]



$$S_i^I \oplus (R_i \oplus b_i \cdot \Delta) = (K_i^0 \oplus R_i \oplus e_i \cdot \Delta) \oplus (R_i \oplus b_i \cdot \Delta) = K_i^0 \oplus (b_i \oplus e_i) \cdot \Delta = K_i^0 \oplus y_i \cdot \Delta = K_i^{y_i}$$

Why secure?

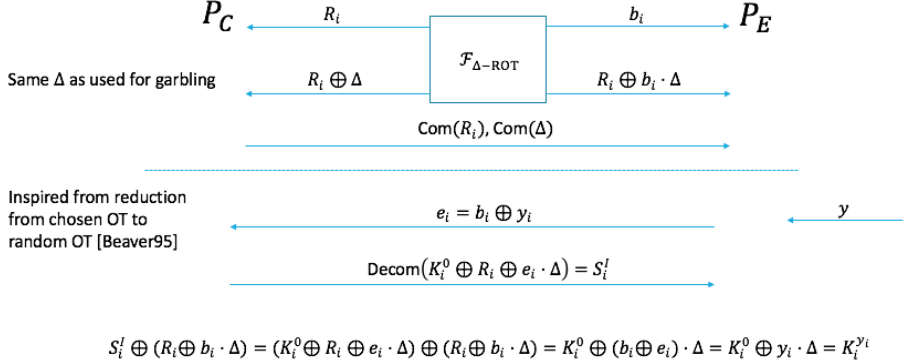


Intuitively, $\mathrm{Decom}\left(K_i^0 \oplus R_i \oplus e_i \cdot \Delta\right)$ is either correct for both, or neither input labels.

Wrong Δ commit:

- $\mathrm{Com}\left(\widehat{\Delta}\right)$ where $\widehat{\Delta} \neq \Delta$, then $S_i^I \oplus (R_i \oplus b_i \cdot \Delta) \notin \{K_i^0, K_i^1\}$ if $b_i = 1$.
- Solution: Cut-and-choose on s OTs. If cheating gets caught with. prob $1 - 2^{-s}$.

Why secure?

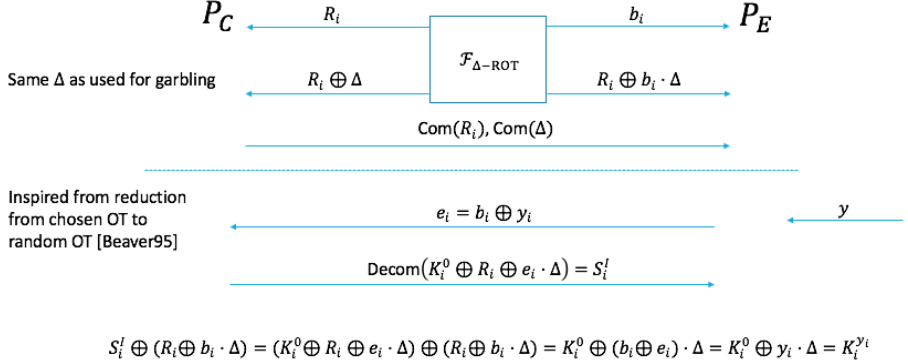


Intuitively, $\mathrm{Decom}\left(K_i^0 \oplus R_i \oplus e_i \cdot \Delta\right)$ is either correct for both, or neither input labels.

Wrong R_i commit:

- $\mathrm{Com}\left(\widehat{R_i}\right)$ where $\widehat{R_i} \neq R_i$, then $S_i^I \oplus (R_i \oplus b_i \cdot \Delta) \notin \{K_i^0, K_i^1\}$.
- Solution: Majority sized Input Authenticators, if $S_i^I \oplus (R_i \oplus b_i \cdot \Delta) \notin \{K_i^0, K_i^1\}$ will reject and P_E can abort.

Why secure?

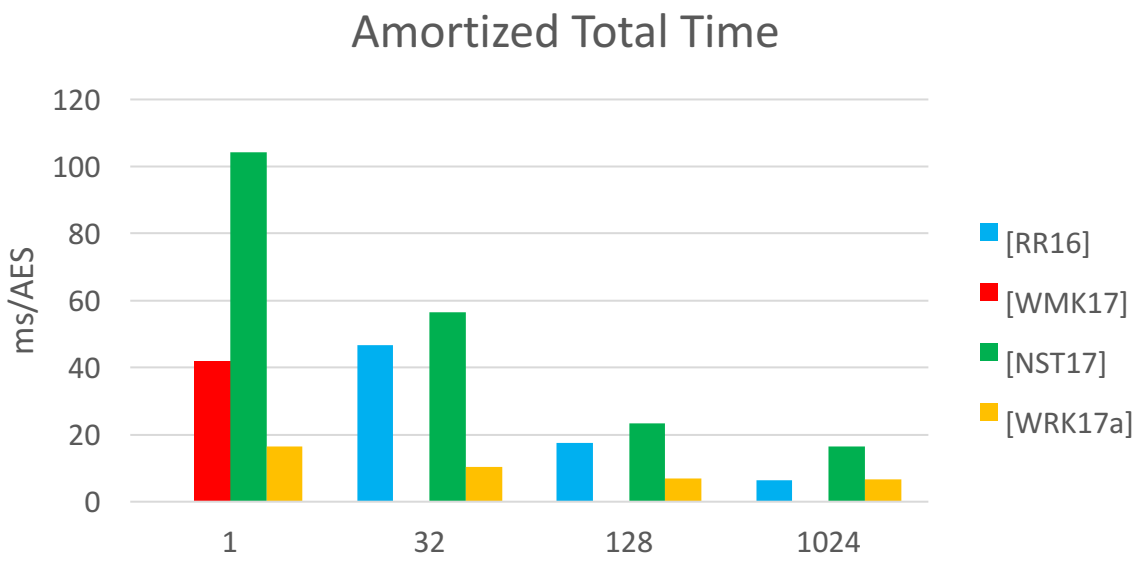


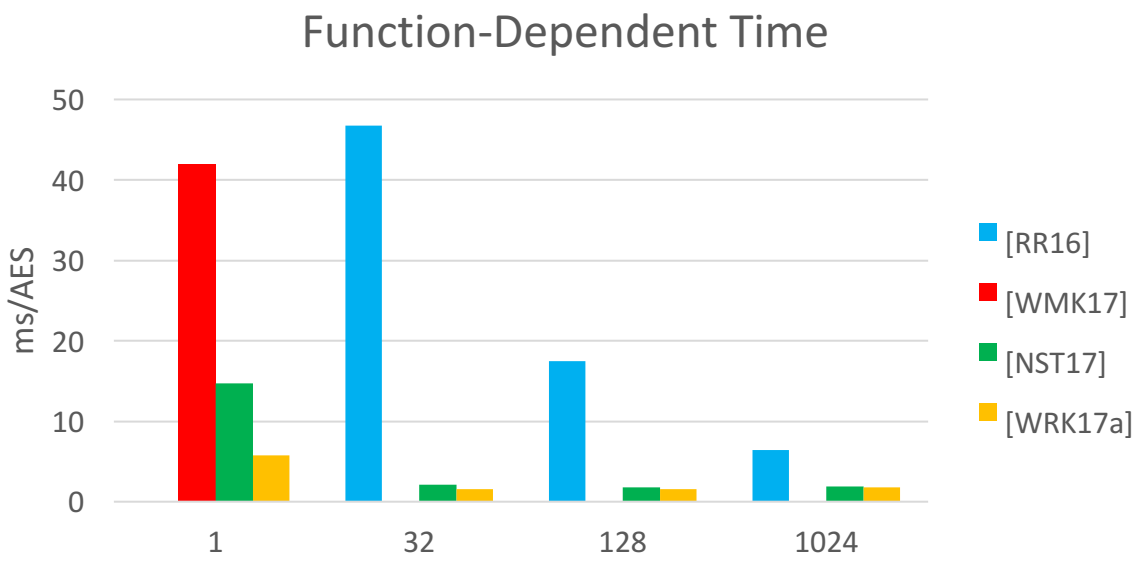
Intuitively, $\mathrm{Decom}\left(K_i^0 \oplus R_i \oplus e_i \cdot \Delta\right)$ is either correct for both, or neither input labels.

Flipped R_i commit:

- $\mathrm{Com}\left(\widehat{R_i}\right)$ where $\widehat{R_i} = R_i \oplus \Delta$, then $S_i^I \oplus (R_i \oplus b_i \cdot \Delta) = K_i^{1-y_i}$.
- Solution: In preprocessing, P_C sends $\mathrm{Decom}\left(lsb\left(K_i^0\right)\right)$. Note: $lsb(\Delta) = 1$.

Performance Comparison (AES-128)

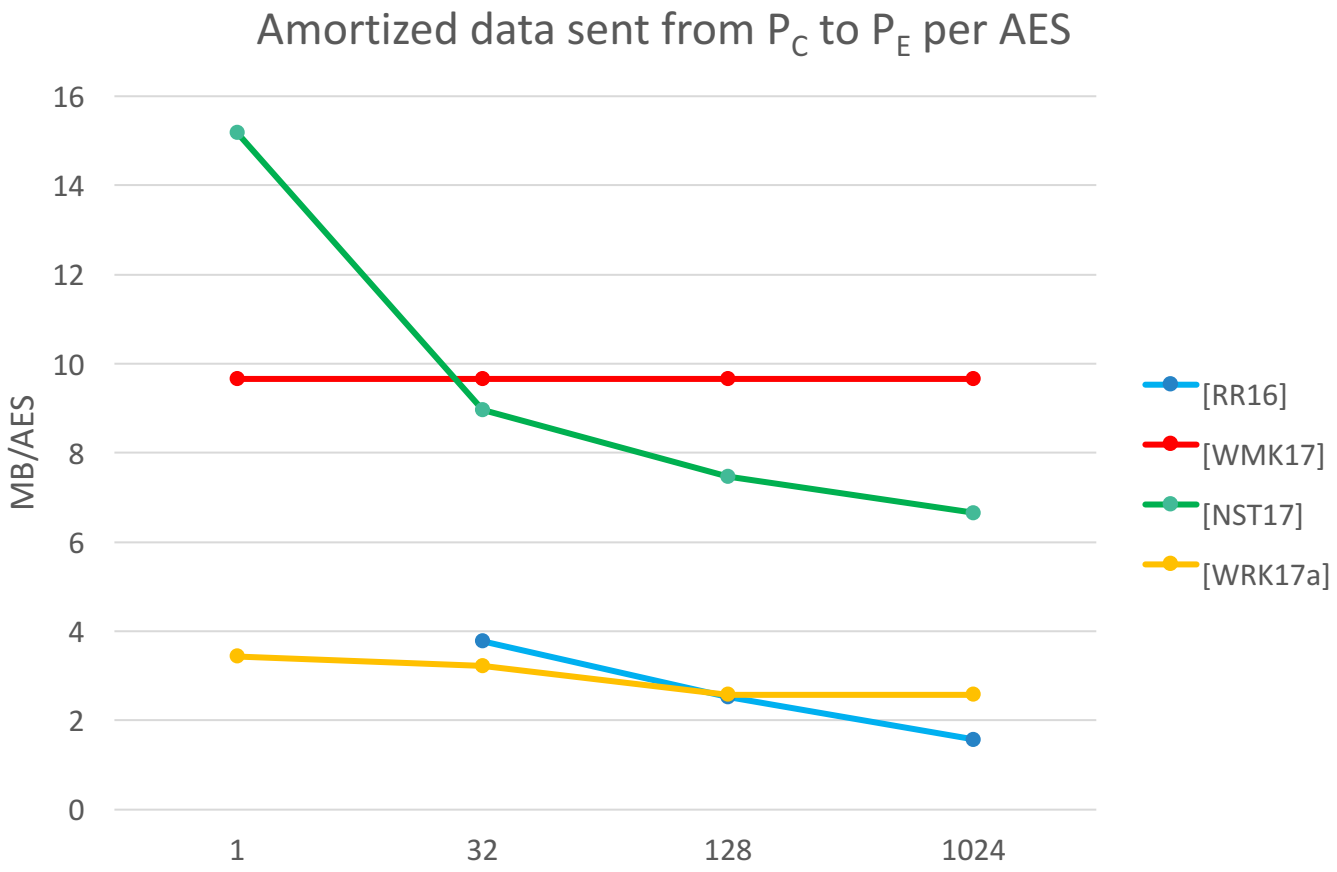




AWS c4.8x instances, 10Gbit LAN. Excl Base OT cost.

[RR16]: "Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution", USENIX 16.
[WMK17]: "Faster Two-Party Computation Secure Against Malicious Adversaries in the Single-Execution Setting", Eurocrypt 17.
[WRK17a]: "Authenticated Garbling and Efficient Maliciously Secure Two-Party Computation", ePrint 17.

Observations



N	[RR16]	[WMK17]	[NST17]	[WRK17a]
1	-	9,66	15,18	3,43
32	3,78	9,66	8,97	3,21
128	2,52	9,66	7,46	2,57
1024	1,58	9,66	6,66	2,57

LEGO has too high concrete communication complexity compared to other approaches.

Commitment cost dominate the preprocessing phase, ~70% of total time.

Outline

1. ~~Secure Two-party Computation Based on Garbled Circuits.~~
2. ~~Implementing TinyLEGO using Efficient XOR-Homomorphic Commitments [NST17].~~
3. DUPLO, a Unified Approach to Cut-and-Choose [Ongoing work].

Cost of Malicious 2PC

Two main costs of malicious 2PC based on Cut-and-Choose (C&C) of garbled circuits

- Replication factor, data communication *per gate*.
- Soldering cost, data communication *per input/output wire*.

Best implementations of two main approaches

- Whole-circuit C&C, e.g. [RR16, WMK17]
  - High replication factor $O(s)$, low soldering cost.
  - [RR16] amortizes over many executions of *same* function, $O(s/\log(\#C))$.
- Gate-level C&C, e.g. [NST17, ZH17, (WRK17a)]
  - Low replication factor, LEGO-factor $O(s/\log(|C|))$. [WRK17a] best constants.
  - High soldering cost, $O(k|C|)$, [NST17] best constants.

Balls and Buckets [LR14]

Experiment:

- Let N be number of buckets, B the bucket size and $p = {}^1/_2$ is the checking probability.
- Adv $\mathcal{A}$ prepares $2NB$ balls and sends to Chal $\mathcal{C}$.
- $\mathcal{C}$ inspects NB balls at *random* and aborts if any are filthy.
- $\mathcal{C}$ packs the remaining NB balls at *random* into N buckets each of size B.
- $\mathcal{A}$ wins the game if any of the N buckets contain only filthy balls.

Theorem

$$\Pr[\mathcal{A} \text{ wins}] \leq \frac{\binom{2NB-t}{NB-t}}{\binom{2NB}{NB}} \cdot N \cdot \binom{t}{B} \cdot \binom{NB}{B}^{-1} \leq 2^{-t} \cdot N \cdot \binom{t}{B} \cdot \binom{NB}{B}^{-1}$$

where t is number of filthy balls.

As N increases, the bucket size B can be lowered accordingly.

Survive C&C

#Filthy subsets of size B

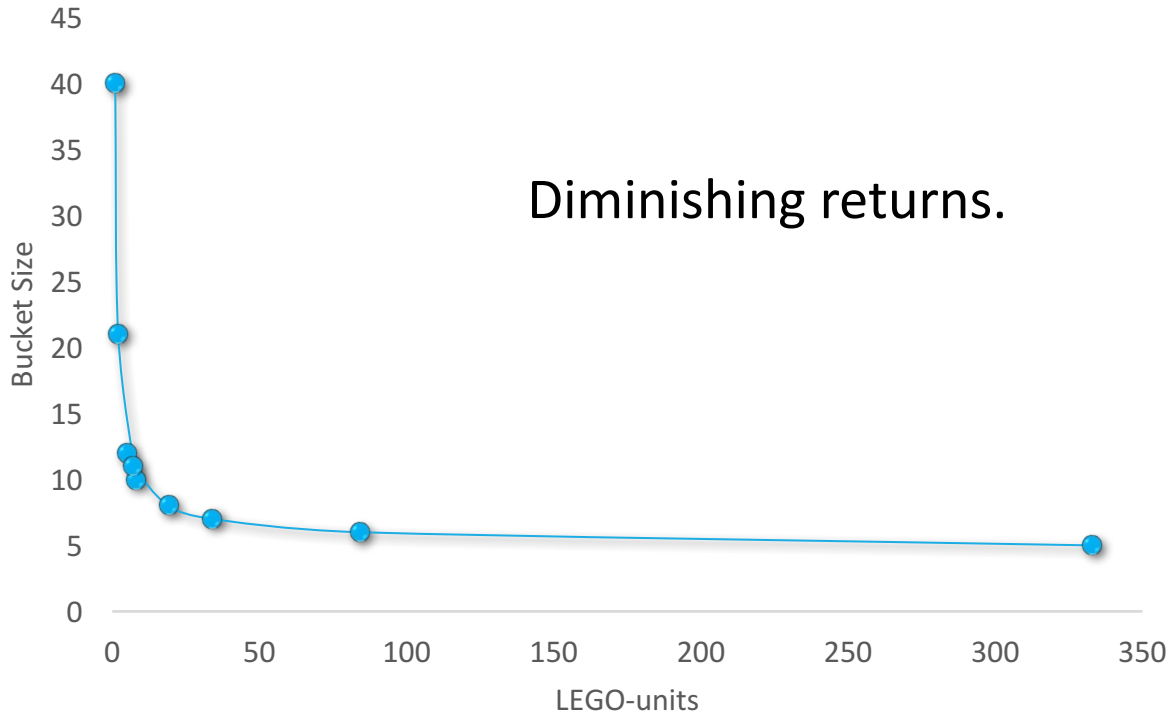
Union Bound

Prob. a bucket with B filthy balls.

LEGO-factor

All mentioned protocols rely on the same “bucketing” analysis for replication factor $O(s/\log(|C|))$

#LEGO-units (N)	Bucket Size
1	40
2	21
8	10
333	5
3,151	4
280,132	3
> 200,000,000,000	2



Diminishing returns.

Check component prob. $p = 1/2$. Stat sec. $s = 40$.

Introducing DUPLO

Joint work with Vladimir Kolesnikov, Jesper Buus Nielsen, Mike Rosulek and Ni Trieu.

Propose LEGO style 2PC protocol with arbitrary component size

- Whole-circuit C&C has component size |C|.
- Gate C&C has component size 1.
- DUPLO supports *any size in this range*, also distinct component types.
- Similar idea considered in [GLMY16] for semi-honest setting to get function-independent preprocessing.

Fewer components than LEGO, but larger

- Low replication factor due to LEGO-factor diminishing returns.
- Lower commitment and soldering cost due to fewer overall input/output wires.
- Results in more efficient protocol, especially for programs comprised of many identical subcomponents (e.g. loops, arithmetic operations, subroutine calls).
- Implementation is up to 7x faster than [WMK17] and 5x faster than [RR16] for certain circuits.

Scaling LEGO to larger components

LEGO uses global free-XOR difference Δ

- To not leak during C&C, a gate is checked on a single input only. Only caught with prob. $^1/_4$ ([ZH17] catches cheating with prob. $^1/_2$).
- Does not scale to large input sizes, worst-case catch bad circuit with prob. $^1/_{2^n}$ for n-input components.

DUPLO uses distinct Δ_i free-XOR differences for each garbled component

- Allows to catch a checked component with prob. 1. Better concrete LEGO-factor.
- Can use variant of the C&C seed checking technique of [GMS08]. Check circuits much cheaper.

Soldering with distinct differences

Recall, with LEGO solderings are XOR of 0-label

- And P_C always commits to 0-label K^0, and the 1-label is defined as $K^0 \oplus \Delta$.

Consider garbled components G_i and G_j with distinct differences Δ_i and Δ_j.

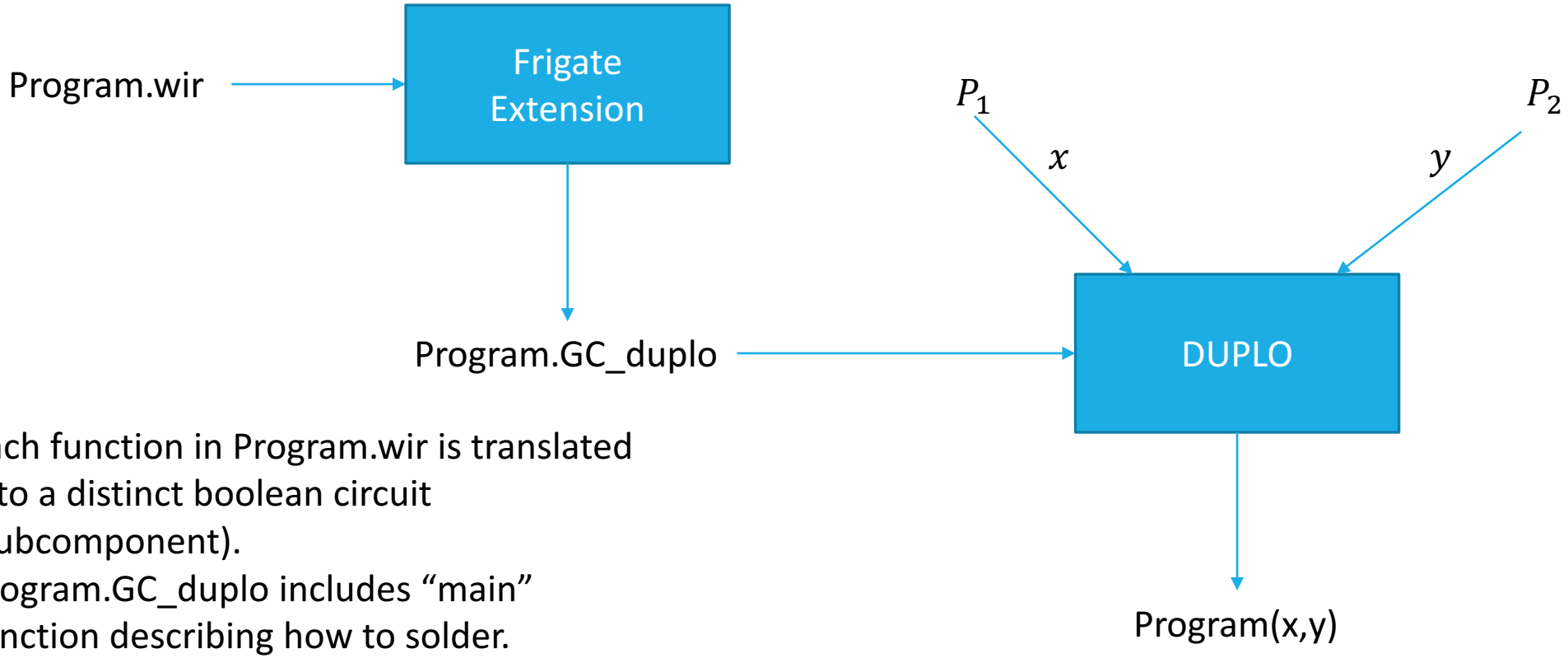
- No single value S^K can turn a label $K_i^b = K_i^0 \oplus b \cdot \Delta_i$ into label $K_j^b = K_j^0 \oplus b \cdot \Delta_j$.

Use approach of [AHMR15] to solder circuits together

- XOR-homomorphic commitments to indicator bits.
- Essentially same cost as LEGO using various tricks and optimizations.

A Tool for Program Decomposition

We extend the recent Frigate compiler [MGCBT16] to output circuits in a format suitable for DUPLO. Same input language as Frigate (C-like syntax).

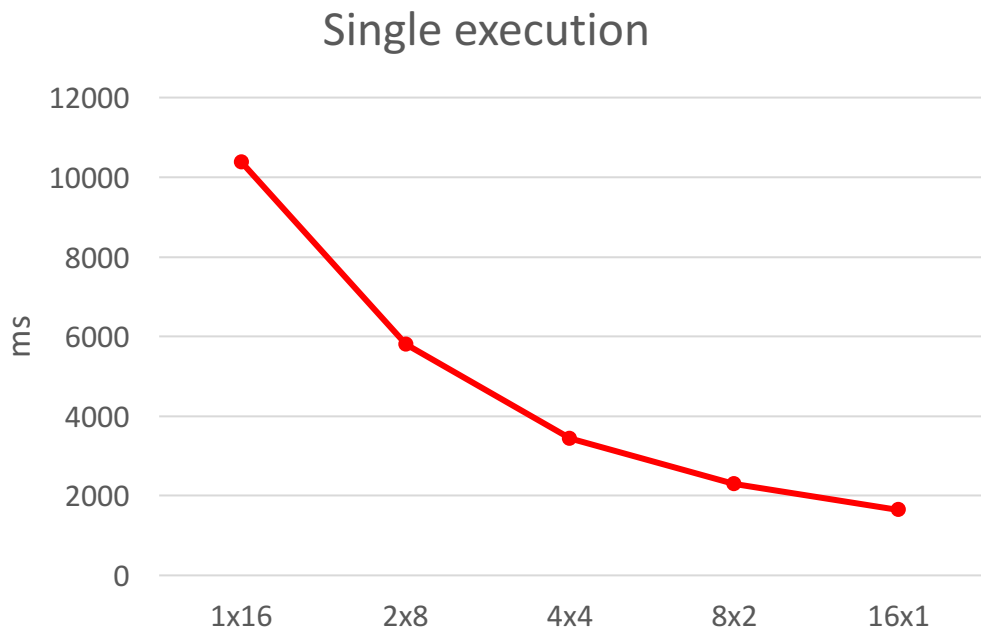


Each function in Program.wir is translated into a distinct boolean circuit (subcomponent).
Program.GC_duplo includes “main” function describing how to solder.

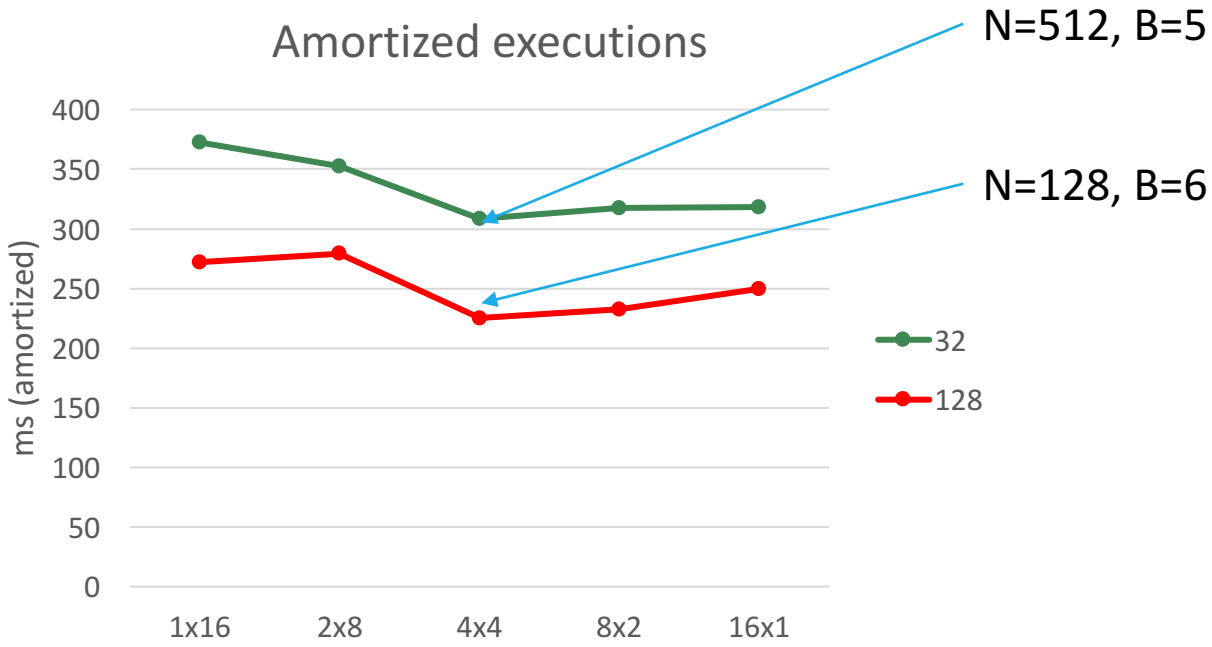
AES-CBC-MAC-16 Decomposition

Contains 102,400 AND gates

- Can naturally be split into 1x16, 2x8, 4x4, 8x2, 16x1 AES subcomponents.



1 AES best subcomponent

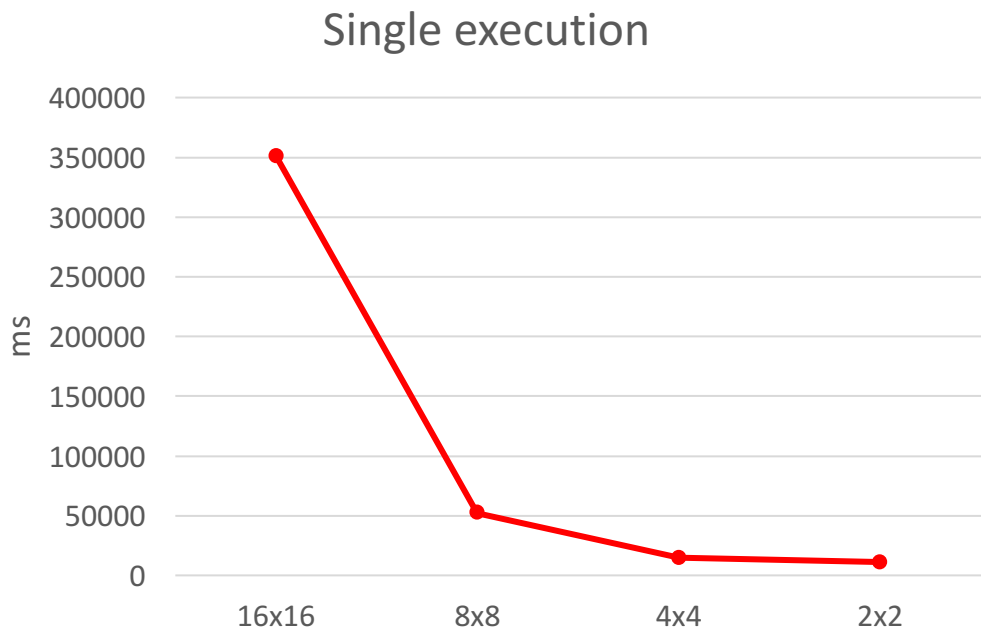


4 AES best subcomponent

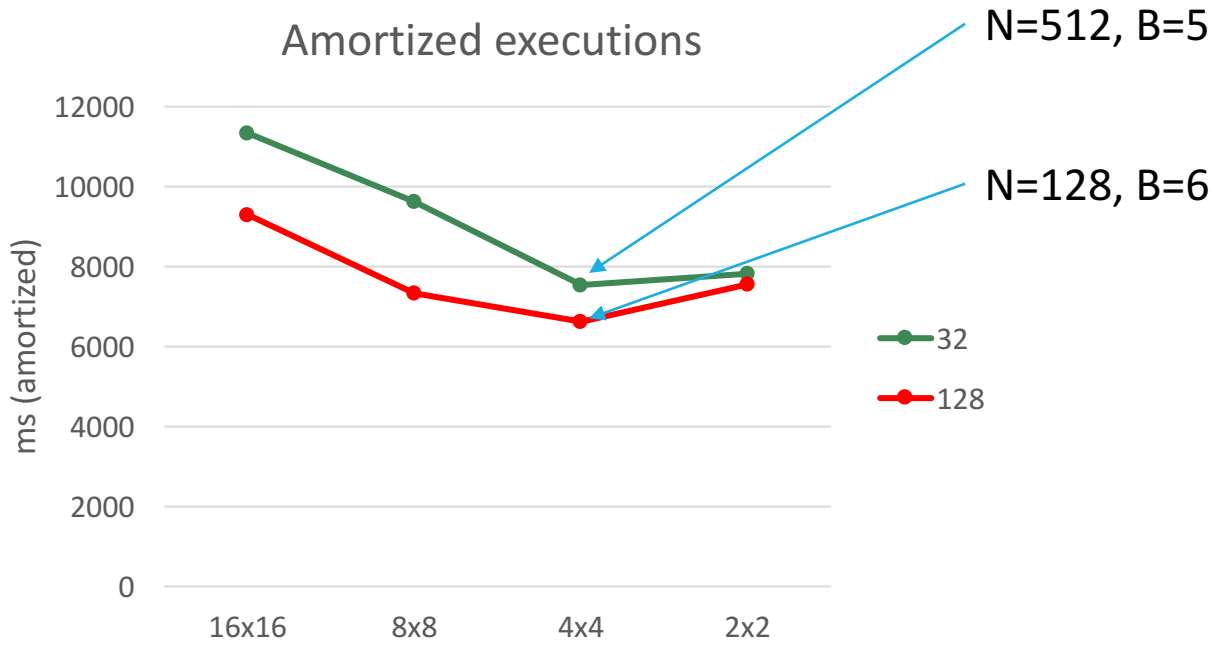
32-bit 16x16 Matrix Multiplication

Contains 4,194,304 AND gates

- Can split computation into $m \; x \; m$ 32-bit matrix product and matrix addition blocks, for $m = 2,4,8,16$.

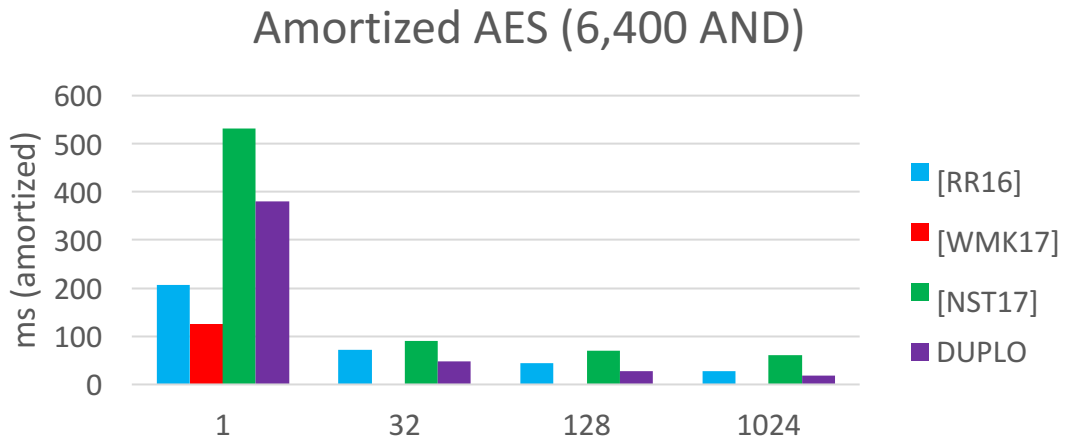


2x2 block matrix best subcomponent

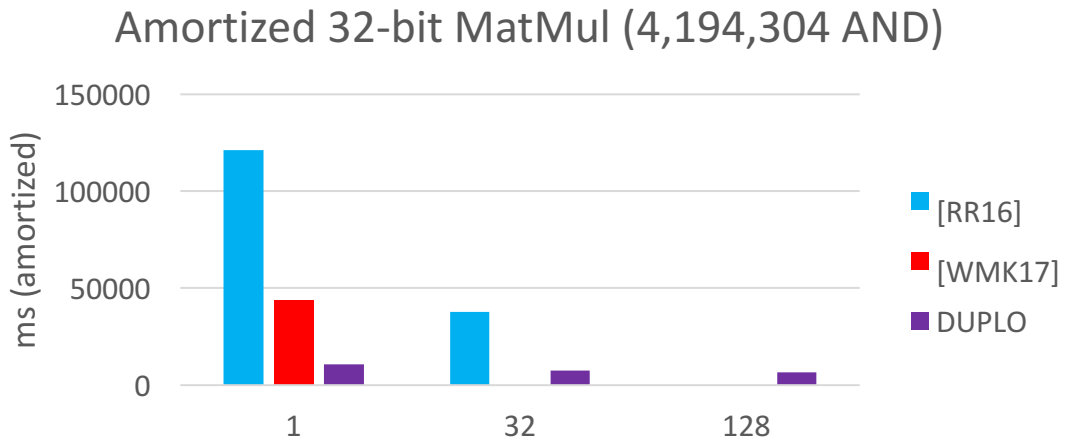


4x4 block matrix best subcomponent

Comparison (Same hardware, 1Gbit LAN)

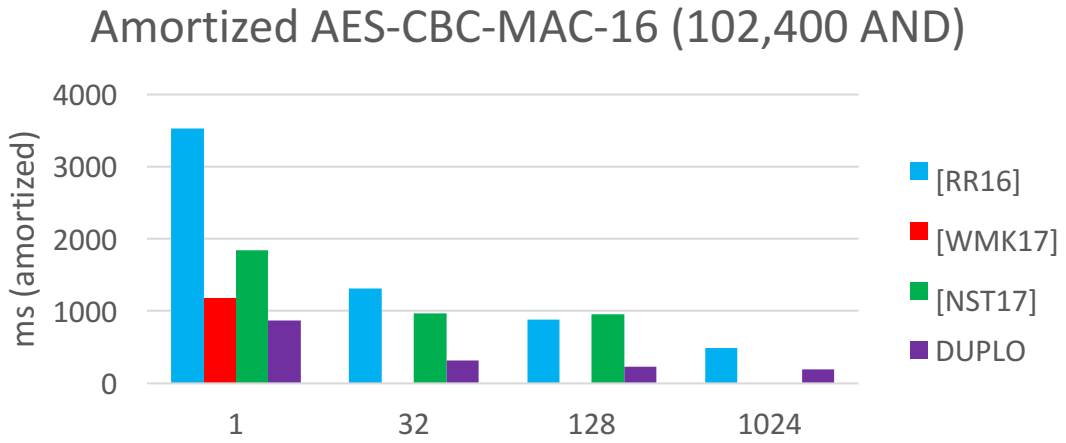


N	[RR16]	[WMK17]	[NST17]	DUPLO
1	207	125	531	380
32	71	-	90	48
128	44	-	70	28
1024	28	-	61	18

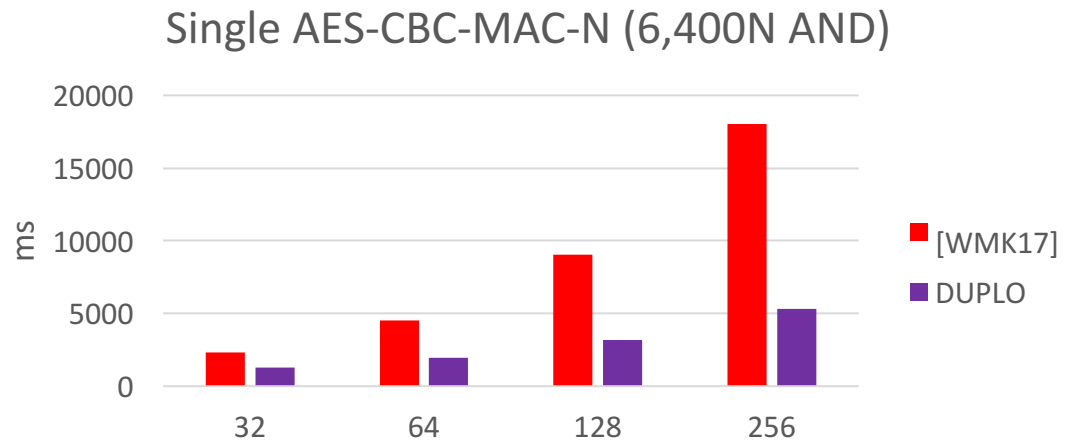


N	[RR16]	[WMK17]	DUPLO
1	121,113	43,930	10,955
32	37,684	-	7,539
128	Out of memory	-	6,622

Comparison (Same hardware, 1Gbit LAN)

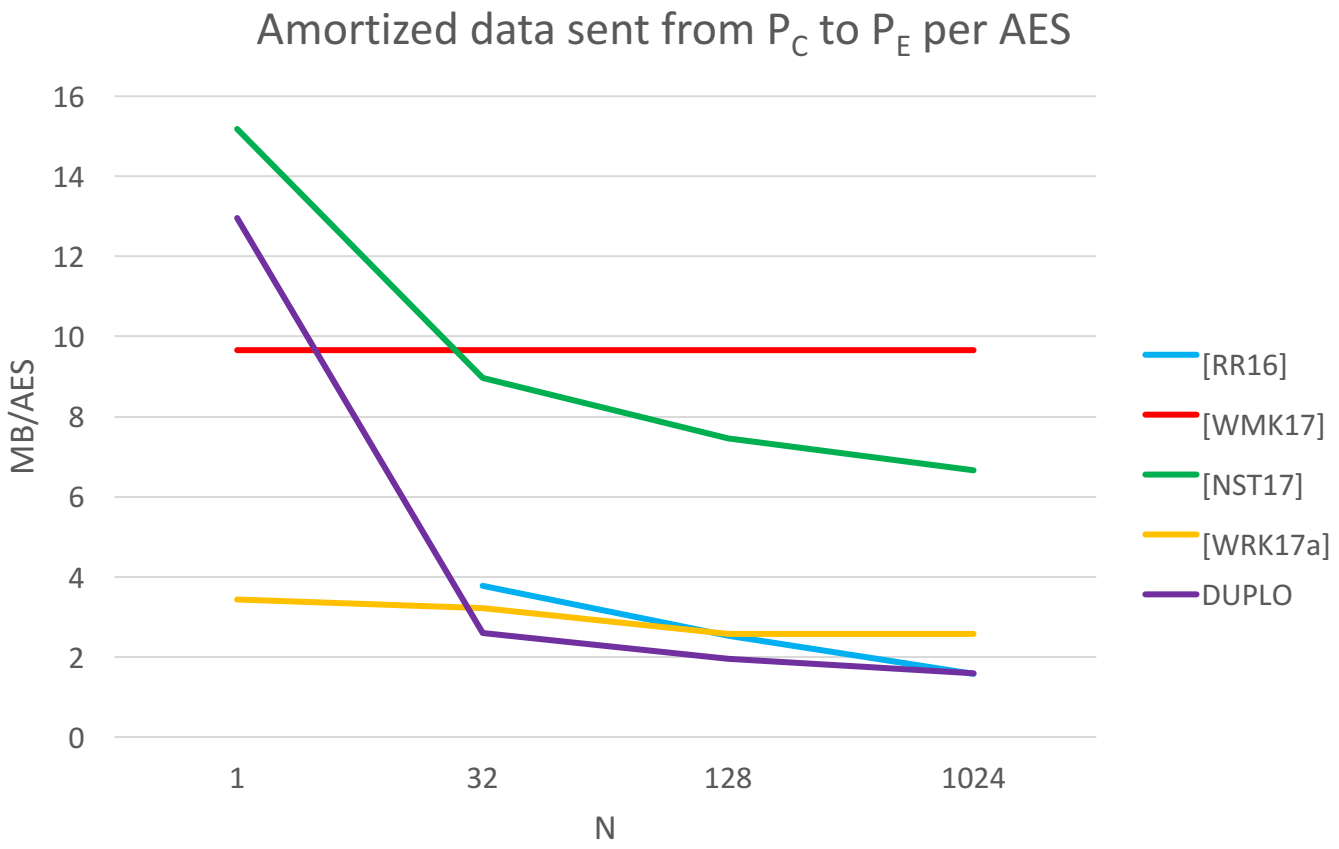


N	[RR16]	[WMK17]	[NST17]	DUPLO
1	3,531	1,177	1,842	871
32	1,317	-	974	317
128	881	-	953	227
1024	488	-	Out of memory	190



N	[WMK17]	DUPLO
32	2,298	1,279
64	4,539	1,981
128	9,029	3,187
256	18,003	5,346

Comparing Communication



N	[RR16]	[WMK17]	[NST17]	[WRK17a]	DUPLO
1	-	9,66	15,18	3,43	12,96
32	3,78	9,66	8,97	3,21	2,60
128	2,52	9,66	7,46	2,57	1,96
1024	1,58	9,66	6,66	2,57	1,59

Thank you

Questions?

Paper coming to ePrint “soon”

Source Code: https://github.com/AarhusCrypto/DUPLO

References

[NST17] Jesper Buus Nielsen, Thomas Schneider, Roberto Trifiletti: **Constant Round Maliciously Secure 2PC with Function-independent Preprocessing using LEGO, NDSS 2017.**

[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.**

[ZRE15] Samee Zahur, Mike Rosulek, David Evans: **Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using Half Gates, Eurocrypt 2015.**

[Bra13] Luís T. A. N. Brandão: **Secure Two-Party Computation with Reusable Bit-Commitments, via a Cut-and-Choose with Forge-and-Lose Technique, Asiacrypt 2013.**

[HKE13] Yan Huang, Jonathan Katz, David Evans: **Efficient Secure Two-Party Computation Using Symmetric Cut-and-Choose, CRYPTO 2013.**

[Lin13] Yehuda Lindell: **Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries, CRYPTO 2013.**

[HKKKM14] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, Alex J. Malozemoff : **Amortizing Garbled Circuits, CRYPTO 2014.**

[LR14] Yehuda Lindell, Ben Riva: **Cut-and-Choose Based Two-Party Computation in the Online/Offline and Batch Settings, CRYPTO 2014.**

[LR15] Yehuda Lindell, Ben Riva: **Blazing Fast 2PC in the Offline/Online Setting with Security for Malicious Adversaries, CCS 2015.**

References

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

[NO16] Jesper Buus Nielsen, Claudio Orlandi: **Cross and Clean: Amortized Garbled Circuits with Constant Overhead, TCC 2016-B.**

[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.**

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, Erez Petrank: **Extending Oblivious Transfers Efficiently, CRYPTO 2003.**

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Sai Sheshank Burra: **A New Approach to Practical Active-Secure Two-Party Computation, CRYPTO 2012.**

[WRK17a] Xiao Wang, Samuel Ranellicci, Jonathan Katz: **Authenticated Garbling and Efficient Maliciously Secure Two-Party Computation, ePrint 2017.**

[WRK17b] Xiao Wang, Samuel Ranellicci, Jonathan Katz: **Authenticated Garbling and Efficient Maliciously Secure Multi-Party Computation, ePrint 2017.**

[HSV17] Carmit Hazay, Peter Scholl, Eduardo Soria-Vazquez: **Low Cost Constant Round MPC Combining BMR and Oblivious Transfer, ePrint 2017.**

References

[MF06] Payman Mohassel, Matthew Franklin: **Efficiency Tradeoffs for Malicious Two-party Computation, PKC 2006.**

[KS06] Mehmet S. Kiraz and Berry Schoenmakers: **A Protocol Issue for the Malicious Case of Yao's Garbled Circuit Construction, SITB 2006.**

[LP07] Yehuda Lindell, Benny Pinkas: **An efficient protocol for secure two-party computation in the presence of malicious adversaries, Eurocrypt 2007.**

[sS11] abhi shelat, Chih-Hao Shen: **Fast two-party secure computation with minimal assumptions**, CCS 13

[Beaver95] Donald Beaver: **Precomputing Oblivious Transfer, CRYPTO 1995.**

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

[ZH17] Ruiyu Zhu, Yan Huang: **Faster LEGO-based Secure Computation without Homomorphic Commitments, ePrint 2017.**

[GLMY16] Adam Groce, Alex Ledger, Alex J. Malozemoff, Arkady Yerukhimovich: **CompGC: Efficient Offline/Online Semi-honest Two-party Computation, ePrint 2016.**

[GMS08] Vipul Goyal, Payman Mohassel, Adam Smith: **Efficient Two Party and Multi Party Computation Against Covert Adversaries, Eurocrypt 2008.**

[AHMR15] Arash Afshar, Zhangxiang Hu, Payman Mohassel, Mike Rosulek: **How to Efficiently Evaluate RAM Programs with Malicious Security, Eurocrypt 2015.**

[MGCBT16] Benjamin Mood, Debayan Gupta, Henry Carter, Kevin R. B. Butler, Patrick Traynor: **Frigate: A Validated, Extensible, and Efficient Compiler and Interpreter for Secure Computation, EuroS&P 2016.**