A Generic Hybrid 2PC Framework with Application to Private Inference of Unmodified Neural Networks (Extended Abstract)

Lennart Braun
Aarhus University, braun@cs.au.dk

Rosario Cammarota
Intel Labs, rosario.cammarota@intel.com

Thomas Schneider
TU Darmstadt, schneider@encrypto.cs.tu-darmstadt.de

Overview

Introduction

Secure Two-Party Computation (2PC)

- compute functionalities $f(x_1, x_2) = (y_1, y_2)$ among distrusting parties
- each party P_i learns only their own input and output.

Example: *Private Inference* for Image Classification

- service provider's trained model and client's image stay private
- only client learns classification result

Our Contributions

Generic 2PC

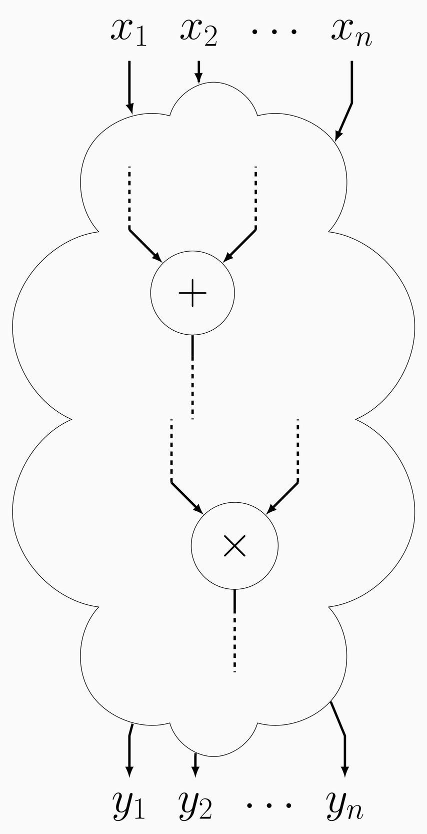
- first 2PC framework with *five different protocols* and *all conversions*
- new protocol *optimization* and conversions
- integration into the recent MOTION [BDST20] framework

Private Inference for Neural Networks

- using *standard MPC techniques* without modifying the networks
- implement common operations as specialized building blocks
- with performance
  - better than using generic 2PC protocols for circuits
  - comparable to recent, highly optimized works
- support for the *Open Neural Network Exchange (ONNX)* file format
  $\Longrightarrow$ *interoperability with deep learning frameworks used in industry*

Five Generic 2PC Protocols

- to evaluate Boolean and arithmetic circuits with semi-honest security
- $\langle x\rangle^S = (\langle x\rangle_1^S, \langle x\rangle_2^S)$ denotes a sharing of value x with protocol S



Yao's Garbled Circuits – Y

- with FreeXOR and Half-Gates optimizations
- $\langle x\rangle^Y = \big((k^0, \Delta), k^0 \oplus x \cdot \Delta\big)$ over $\{0,1\}$

Goldreich-Micali-Wigderson (GMW) – A, B

- $\langle x\rangle^B = (x^1, x^2)$ s. t. $x = x^1 \oplus x^2$ over $\{0,1\}$
- $\langle x\rangle^A = (x^1, x^2)$ s. t. $x = x^1 + x^2$ over $\mathbb{Z}_{2^\ell}$

ABY2.0 [PSSY21] Secret Sharing – α, β

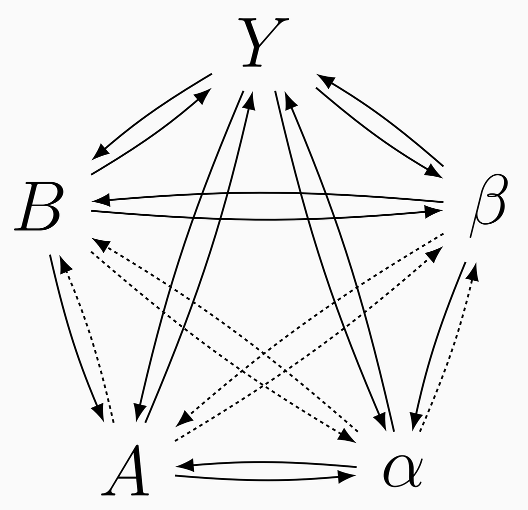
- $\langle x\rangle^\beta = (\Delta_x; \langle \delta_x\rangle^B)$ s. t. $\Delta_x = x \oplus \delta_x$ over $\{0,1\}$
- $\langle x\rangle^\alpha = (\Delta_x; \langle \delta_x\rangle^A)$ s. t. $\Delta_x = x + \delta_x$ over $\mathbb{Z}_{2^\ell}$

Auxiliary Protocols and Preprocessing

- based on Oblivious Transfer

Conversion Protocols

Conversions Between All Five Protocols



- change representation of shared values
- evaluation of hybrid circuits combining Boolean and arithmetic components
- 14 direct (⟶), 6 as composition (⋯►)
- based on protocols from ABY [DSZ15], ABY2.0 [PSSY21], and MOTION [BDST20], with improvements

Improvements to ABY [DSZ15] and ABY2.0 [PSSY21]

- security parameter κ, bit length ℓ

$B \to A$: using shared bits

	setup	online	
	comm.	comm.	rounds
ABY	$\ell\kappa$	$\ell^2/2+\ell$	2
this	$\ell\kappa + \ell^2/2$	2ℓ	1

$Y \to A$: *without* online comm.

	setup	online	
	comm.	comm.	rounds
ABY	$\ell\kappa$	$\ell^2/2+\ell$	2
this	$(2\ell-1)\kappa+\ell$	0	0

$\beta \to \alpha$: optimized setup phase

	setup comm.
ABY2.0	$\ell\kappa + \ell^2$
this	$(\ell-1)\kappa + \ell^2/2 - \ell/2$

Optimized setup for mixed products

- $\langle b\cdot n\rangle^A \leftarrow \langle b\rangle^B \cdot \langle n\rangle^A$
- $\langle b\cdot n\rangle^\alpha \leftarrow \langle b\rangle^\beta \cdot \langle n\rangle^\alpha$
- $\langle b_1\cdot b_2\rangle^\alpha \leftarrow \langle b_1\rangle^\beta \cdot \langle b_2\rangle^\beta$

ABY [DSZ15] vs. ABY2.0 [PSSY21]

Multiplications / ANDs / Matrix Products / Convolutions

Online Communication

- GMW: linear in *input* size
- ABY2.0: linear in *output* size

Setup Phase

- GMW: batched generation of Beaver triples $(\langle a\rangle^A, \langle b\rangle^A, \langle a\cdot b\rangle^A)$
- ABY2.0: function-dependent setup $\Longrightarrow$ SIMD even more important

Conversions

- newer conversions improve on original ABY [DSZ15] protocols
- ABY: $A/B \to Y$: 2 rounds, $Y \to A/B$: no communication
- ABY2.0: $\alpha/\beta \leftrightarrow Y$: both 1 round

$\Longrightarrow$ same overall round complexity when alternating A/α and Y

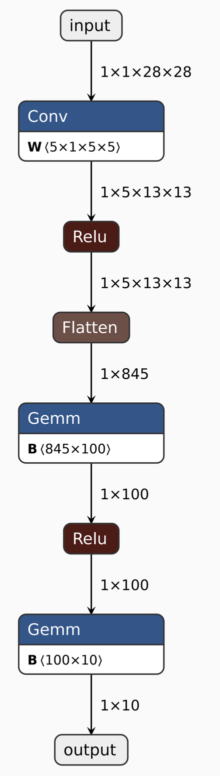
Neural Networks

- many batched operations $\Longrightarrow$ reduces disadvantage of ABY2.0 setup
- ABY2.0 clearly better for ReLU

Neural Network Building Blocks

Idea

- exploit high-level structure of networks, and do not compile to circuits
- use generic protocols, but implement them in a optimized way
- do not change the networks' architectures



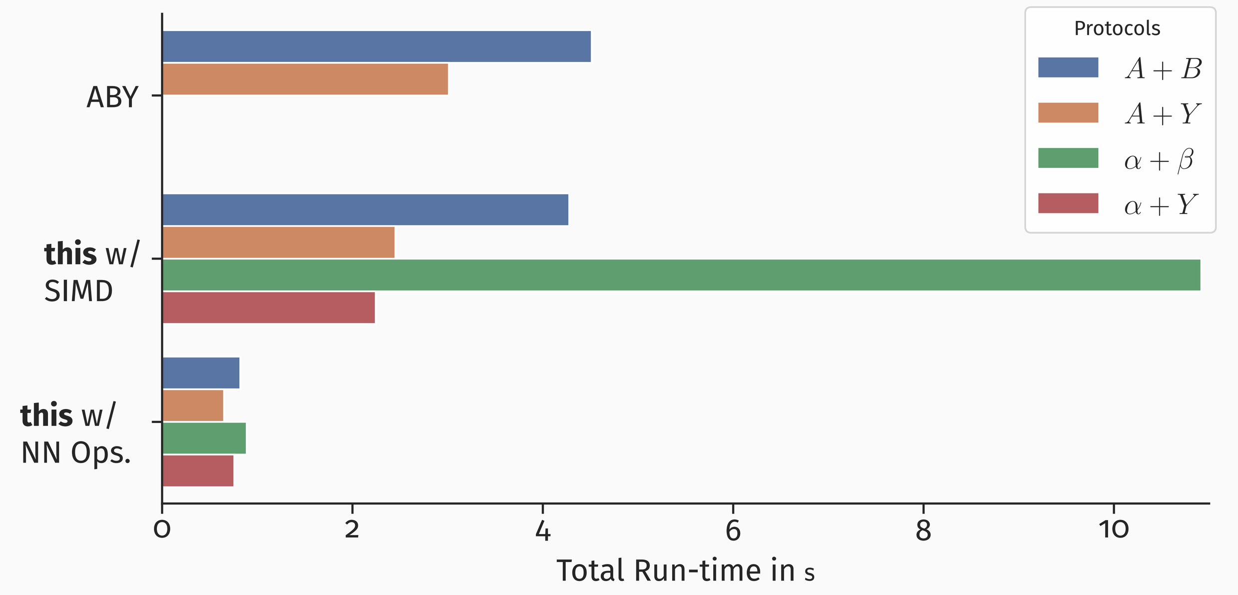
Currently Supported Tensor Operations

- fixed-point arithmetic with truncation by [MZ17]
- fully-connected and convolutional layers (A/α)
- ReLU (multiple variants $Y/B/\beta/A+B/\alpha+\beta$)
- MaxPool (using optimized circuits $Y/B/\beta$)
- AveragePool (A/α)

Neural Network Benchmarks

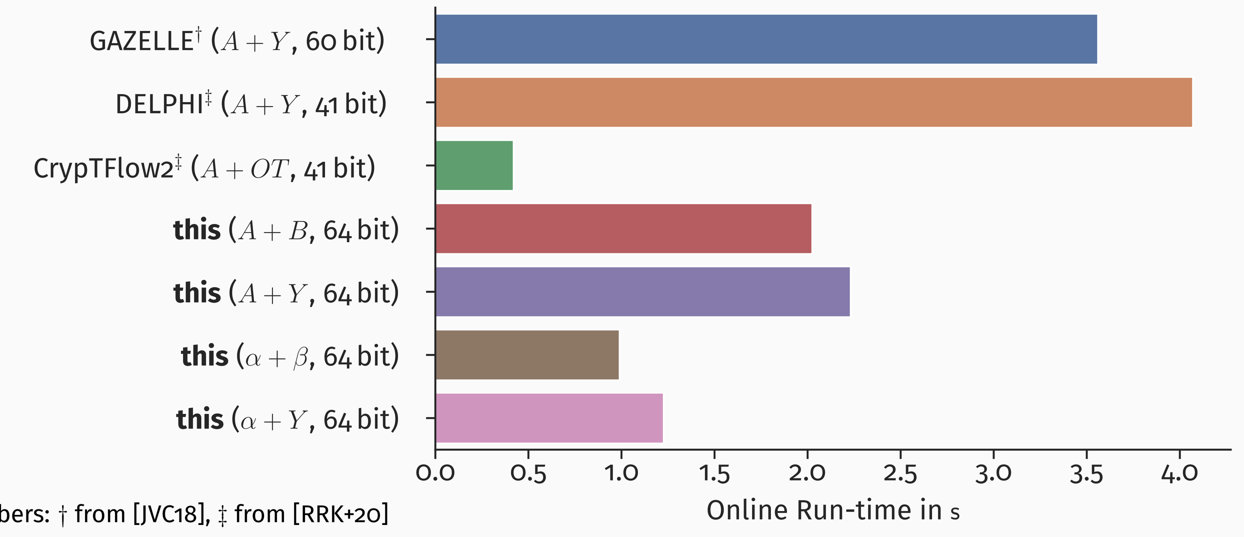
Small Network: MiniONN MNIST [LJLA17]

Neural Network Building Blocks vs. Generic 2PC for Hybrid Circuits



Larger Network: MiniONN CIFAR-10 [LJLA17]

Comparison with Prior and Concurrent Work [JVC18; MLS+20; RRK+20]



Numbers: † from [JVC18], ‡ from [RRK+20]

Extending the MOTION Framework ⇒ MOTION2NX

Extending and Improving the Framework

- implementation of the five generic 2PC protocols and conversions
- architectural improvements to increase flexibility and performance
  - cleaner interfaces, decoupled components
  - new system for asynchronous communication
  - executors allow for different execution strategies
- single instruction multiple data (SIMD) operations
- automatic collection of run-time statistics and metadata
- support for HyCC-generated [BDK+18] hybrid circuits

Support for Neural Networks

- secure tensor data types
- neural network building blocks
- parallelized tensor operations
- ONNX support for interoperability with PyTorch, TensorFlow, etc.

Open Source under an MIT License

- available on GitHub: `https://encrypto.de/code/MOTION2NX`

More Information

Extended Abstract

`https://encrypto.de/papers/BCS21PriMLNeurIPS.pdf`

Acknowledgments

This work is supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreements No. 803096 (SPEC) and No. 850990 (PSOTI), and the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) — SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

References

[BDK+18] N. Büscher, D. Demmler, S. Katzenbeisser, D. Kretzmer, and T. Schneider. *HyCC: Compilation of Hybrid Protocols for Practical Secure Computation*. In: *CCS 2018*. 2018.

[BDST20] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko. *MOTION – A Framework for Mixed-Protocol Multi-Party Computation*. To appear in ACM Transactions on Privacy and Security (TOPS). 2020. URL: `https://ia.cr/2020/1137`.

[DSZ15] D. Demmler, T. Schneider, and M. Zohner. *ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation*. In: *NDSS 2015*. 2015.

[JVC18] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. *GAZELLE: A Low Latency Framework for Secure Neural Network Inference*. In: *USENIX Security 2018*. 2018.

[LJLA17] J. Liu, M. Juuti, Y. Lu, and N. Asokan. *Oblivious Neural Network Predictions via MiniONN Transformations*. In: *CCS 2017*. 2017.

[MLS+20] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa. *Delphi: A Cryptographic Inference Service for Neural Networks*. In: *USENIX Security 2020*. 2020.

[MZ17] P. Mohassel and Y. Zhang. *SecureML: A System for Scalable Privacy-Preserving Machine Learning*. In: *S&P*. 2017.

[PSSY21] A. Patra, T. Schneider, A. Suresh, and H. Yalame. *ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation*. In: *USENIX Security 2021*. 2021.

[RRK+20] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma. *CrypTFlow2: Practical 2-Party Secure Inference*. In: *CCS 2020*. 2020.