MPCLEAGUE: ROBUST 4-PARTY COMPUTATION FOR PRIVACY-PRESERVING MACHINE LEARNING

Nishat Koti, Arpita Patra & Ajith Suresh *
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India
{kotis,arpita,ajith}@iisc.ac.in

ABSTRACT

Secure computation has demonstrated its potential in several practical use-cases, particularly in privacy-preserving machine learning (PPML). Robustness, the property that guarantees output delivery irrespective of adversarial behaviour, and efficiency, are the two first-order asks of a successfully deployable PPML framework. Towards this, we propose the first *robust*, highly-efficient mixed-protocol framework, **MPCLeague** that works with four parties, offers malicious security, and supports ring. **MPCLeague** has a multifold improvement over ABY3 (Mohassel et al. CCS'18), a 3-party framework achieving security with abort, and improves upon Trident (Chaudhari et al. NDSS'20), a 4-party framework achieving security with fairness. **MPCLeague**'s competence is tested with extensive benchmarking for deep neural networks such as LeNet and VGG16, and support vector machines.

Recent advancements in machine learning (ML) have shown progress in various applications such as autonomous driving (Schwarting et al., 2018), voice assistance (Xiong et al., 2018), and even surpassed humans in healthcare applications such as classifying echocardiograms (Madani et al., 2017). To address the issue of privacy of individuals who contribute to the training data, the field of privacy-preserving machine learning (PPML) has been emerging, which allows to carry out ML tasks in a privacy-preserving way. Further, to aid the compute-intensive PPML tasks, the paradigm of secure outsourced computation (SOC) has gained popularity, where clients can outsource the training/prediction computations to computationally powerful servers that can be availed on a 'pay-per-use' basis from Cloud service providers. Lately, secure multiparty computation (MPC) based techniques (Mohassel & Zhang, 2017; Mohassel & Rindal, 2018; Riazi et al., 2018; Makri et al., 2019; Wagh et al., 2019; Chaudhari et al., 2019; 2020; Patra & Suresh, 2020; Byali et al., 2020) for PPML in SOC setting have gained interest, where a server enacts the role of a party in the MPC protocol. MPC (Yao, 1982; Goldreich et al., 1987; Chaum et al., 1988) allows mutually distrusting parties to compute a function in a distributed fashion while guaranteeing *privacy* of the parties' inputs and *correctness* of their outputs against any coalition of t parties.

Efficiency is the first ask of a PPML framework. We work towards this goal in the 4-party (4PC) setting (honest majority) (Gordon et al., 2018; Chaudhari et al., 2020; Byali et al., 2020; Koti et al., 2020). Robustness is yet another essential of a successfully deployable PPML protocol via MPC. *Robustness* or *guaranteed output delivery* (GOD) ensures that all honest protocol participants receive the output of the computation irrespective of the adversarial behaviour. In the SOC setting, carrying out the computation via a robust MPC protocol among the servers prevents the adversary from aborting the computation and guarantees correct output delivery to the clients, regardless of the adversary's misbehaviour. Thus, a robust protocol prevents the adversary from repeatedly causing the computations to rerun, thereby saving a client's valuable time and resources. Preventing repeated failures, robustness upholds the trust and interest of clients in the system. Several works (Mohassel & Rindal, 2018; Wagh et al., 2019; Patra & Suresh, 2020; Chaudhari et al., 2020) realizing PPML

*https://www.csa.iisc.ac.in/%7Ecris/

via MPC settle for weaker guarantees such as abort[1] and fairness[2]. Achieving the strongest notion of GOD without degrading performance is an interesting goal which is the focus of this work.

To enhance practical efficiency, many recent works (Baum et al., 2016; Damgård et al., 2018; Keller et al., 2018; Chaudhari et al., 2019; 2020; Patra & Suresh, 2020; Byali et al., 2020) resort to the preprocessing paradigm, which casts protocols into two phases; a preprocessing phase where input-independent (but function-dependent), computationally heavy tasks can be computed, followed by a fast online phase. This paradigm is well suited for PPML in SOC where ML algorithms are evaluated several times, and the algorithm is known beforehand. Further, recent works (Demmler et al., 2015; Damgård et al., 2018; 2019) propose MPC protocols over 32 or 64 bit rings to leverage CPU optimizations. We follow the preprocessing paradigm and work over ring.

MPC protocols can be categorized as low-latency and high-throughput, where the former, based on garbled circuits (GC), require higher communication than the latter. High-throughput protocols work over secret shared data over the boolean ring $\mathbb{Z}_2$ or arithmetic ring $\mathbb{Z}_{2^\ell}$ and aim at minimizing the communication overhead (bandwidth) at the expense of non-constant rounds. While high-throughput protocols enable efficient computation of functions such as addition, multiplication and dot-product, other functions such as division are best performed using garbled circuits. Further, activation functions such as ReLU used in Neural Networks (NN) training, alternate between multiplication and comparison, where multiplication is better suited in the arithmetic world and comparison is more efficient when performed in the boolean world. Hence, MPC protocols working over different representations (arithmetic/boolean/garbled circuit based) can be mixed to achieve better efficiency. This motivated the need for mixed protocols (Demmler et al., 2015; Mohassel & Zhang, 2017; Mohassel & Rindal, 2018; Aly et al., 2019; Rotaru & Wood, 2019; Chaudhari et al., 2020; Patra et al., 2020) where each protocol is executed in a world where it performs the best. However, switching between different worlds necessitates efficient conversions to alternate between different (sharing) representations. In this work, we propose, to the best of our knowledge, the first robust mixed-protocol PPML framework via MPC with four parties in honest majority setting.

Related Work: The mixed-protocol framework for MPC was first shown to be practical by TASTY (Henecka et al., 2010), which combined garbled circuits with homomorphic encryption. Motivated by ABY (Demmler et al., 2015) that combines arithmetic, boolean and garbled style computation, and proposes protocols to switch between the arithmetic/boolean/garbled worlds for 2 parties, ABY3 (Mohassel & Rindal, 2018) and SecureNN (Wagh et al., 2019) extend the idea to 3 parties and apply the technique to the ML domain. The recent work of ABY2.0 (Patra et al., 2020) improves upon the ABY framework, providing a fast online phase, with applications to PPML. Relying on secret-sharing based protocols, (Wagh et al., 2020) also provides a 3PC framework for deep learning and is comparable to (Mohassel & Rindal, 2018), but is not suitable for the WAN setting. In 4PC, Trident (Chaudhari et al., 2020) provides an efficient mixed-protocol PPML framework that improves over ABY3 and provides security with fairness. Recently, SWIFT (Koti et al., 2020) presents an efficient, robust PPML framework with protocols as fast as the fair protocols of Trident and twice faster than GOD protocols of FLASH (Byali et al., 2020), but lacks support for neural network training.

1 Our Contributions

We design the first robust 4PC mixed-protocol framework, with applications to PPML training and inference, tolerating at most one malicious corruption, that efficiently combines arithmetic, boolean and garbled worlds. Towards this, we propose two robust GC-based protocols, tailor-made to act as the garbled world for our mixed protocol framework with a trade-off in online latency and preprocessing communication. Our mixed framework efficiently combines our garbled world with an improved variant of arithmetic/boolean world based on SWIFT (Koti et al., 2020), and provides efficient *end-to-end* conversions to switch between the worlds. We assume a one-time key setup phase and work in preprocessing model, which paves the way for a fast online phase.

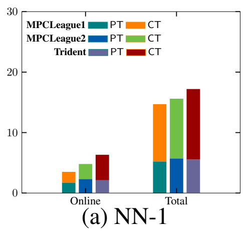
Robust and Improved Arithmetic/Boolean 4PC. Working over the ring $\mathbb{Z}_{2^\ell}$, we propose new protocols for 2,3 and 4 input multiplication, allowing multiplication of 3 and 4 inputs in one shot in the latter cases. Naively, a 3/4-input multiplication given access to 2-input multiplication will

[1]This property may allow the corrupt parties *alone* to learn the output.

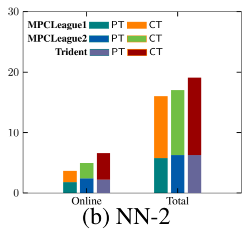
[2]This property ensures delivery of output, to all participants or none, depending on the adversarial behaviour.

inflate the communication cost and rounds by at least $2\times$. Our contribution lies in keeping the online communication cost and the online round requirement invariant of the number of inputs while trading off the offline cost. The technical core of this result is a new 2-input multiplication protocol, inspired by (Koti et al., 2020), which allows easy extension to the multi-input setting while retaining the communication cost as that in (Koti et al., 2020). Coupled with optimized parallel prefix adder circuit from (Patra et al., 2020), our multi-input multiplications bring in a $2\times$ improvement in online rounds, as well as an improvement in online communication of secure comparison protocol, thereby yielding improvements in ReLU protocol used in PPML algorithms. Further, inspired by (Mazloom et al., 2020), we show, for the first time in GOD setting, how multiplication *with truncation* can be performed without any overhead. All these bring in huge efficiency gains for PPML.

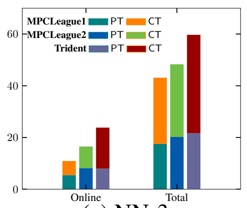
Robust 4PC Mixed-Protocol Framework. The efficiency lift of our framework compared to existing frameworks stands on the following useful observation– a large portion of computation in most of the MPC-based PPML framework is done over the arithmetic and boolean world; they switch to the garbled world to perform the non-linear operations (e.g., softmax) that are expensive in the arithmetic/boolean world and switch back to the arithmetic/boolean world immediately after. We leverage this phenomenon in two ways. First, we propose a robust tailor-made GC-based protocol, which wins over the existing works of (Ishai et al., 2015; Byali et al., 2018) when deployed in the mixed framework by offering an amortized round complexity of 1 for many instances. Since the mixed framework will need many instances of this protocol, it enjoys the impact of amortization. The construction needs 2 GC communication. When combined with our improved (compared to Koti et al. (2020)) multi-input multiplication-based arithmetic/boolean world, we obtain a mixed-protocol framework, MPCLeague1, which attains a much faster online phase than the state-of-the-art *fair* framework of Trident (Chaudhari et al., 2020). We introduce a second version of robust GC-based protocol, again tailor-made for mixed-protocol setting, that requires communicating only 1 GC at the expense of 2 (amortized) rounds. This version, when combined with our improved arithmetic/boolean world, just with the 2-input multiplication, results in a framework, MPCLeague2, that outperforms the state-of-the-art *fair* framework of Trident (Chaudhari et al., 2020) in terms of the *total* communication cost, yet with a slower online phase than MPCLeague1.[3] Second, we depart from the existing methods and provide efficient *end-to-end* conversion techniques such as Arithmetic-Garbled-Arithmetic instead of *piece-wise* combinations of *Arithmetic to Garbled* followed by a *Garbled to Arithmetic* conversion. Such a conversion benefits from not having to generate a full-fledged garbled-shared output after the garbled computation. This results in end-to-end conversions that need just a single round for the first variant of our garbled world (cf. Table 2). The above two changes bring in huge impact performance-wise.



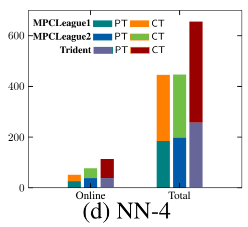
(a) NN-1



(b) NN-2



(c) NN-3



(d) NN-4

Figure 1: Training: in terms of $\mathsf{PT}_{\mathsf{on}}$ and $\mathsf{CT}_{\mathsf{on}}$ (cf. Table 3)

PPML and Benchmarking. We demonstrate the practicality of our framework by benchmarking the training and inference phases of deep Neural Networks (NN) such as LeNet (LeCun et al., 1998) (NN-3) and VGG16 (Simonyan & Zisserman, 2014) (NN-4), Convolutional Neural Networks (CNN) (NN-1, NN-2), and the inference phase of Support Vector Machines (SVM). We benchmark our protocols against SWIFT (Koti et al., 2020) 4PC (tackles only inference) and Trident (Chaudhari et al., 2020) from 4PC regime, and SWIFT 3PC (tackles only inference) and ABY3 (Mohassel & Rindal, 2018) from 3PC regime. The protocols are benchmarked over a Wide Area Network (WAN), instantiated using n1-standard-64 instances of Google Cloud[4]. The machines are equipped with 2.3 GHz Intel Xeon E5 v3 (Haswell) processors supporting hyper-threading, with 8 vCPUs, and 30 GB

[3]Recall that 3/4-input multiplication results in extra overhead in preprocessing phase compared to 2-input multiplication. This pulls down the *total* communication cost of the resultant framework. This is why MPCLeague2 resorts to standard 2-input multiplication to defeat Trident with respect to total communication cost.

[4]https://cloud.google.com/

of RAM Memory. We use a bandwidth of 50 MBps. We implement our protocols using the publicly available ENCRYPTO library (Cryptography & at TU Darmstadt, 2017) in C++17.

We evaluate the protocols for a variety of parameters as given in Table 3 which are based on runtime and communication. The two versions of our robust framework, MPCLeague1 and MPCLeague2 cater to different deployment scenarios. While MPCLeague1 is designed to provide fast online response time, MPCLeague2 provides an overall improvement in communication over the fair framework of Trident (Chaudhari et al., 2020). The asymmetric roles of the parties in our framework (2 active parties as opposed to 3 in Trident) makes it an apt fit in cloud deployment scenarios, where one pays the price for the communication and running time of the hired servers. Hence, we also consider *monetary cost* (Cost) (Miao et al., 2020) in our benchmarking, which is dependent on the communication and cumulative runtime[5] of the hired servers. We compare the monetary costs of the protocols using the pricing of Google Cloud Platform[6], where for 1 GB and 1 hour of usage, the costs are USD 0.08 and USD 3.04, respectively. To improve the performance, we also perform load balancing. Hence, for inference, we also use *online throughput* (TP) as a benchmarking parameter following Chaudhari et al. (2020); Mohassel & Rindal (2018) as it helps to analyse the effect of improved communication and round complexity in a single shot.

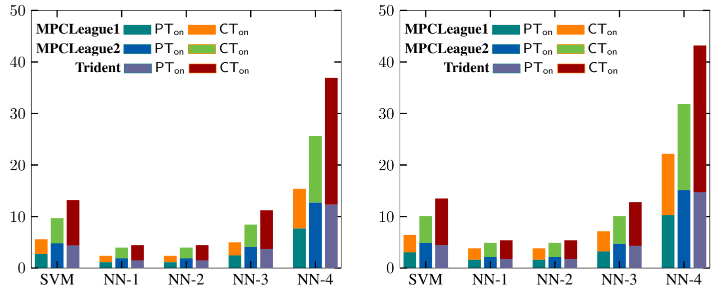
Performance Comparison. We compare MPCLeague1 and MPCLeague2 with Trident in Table 1 based on runtime, communication and monetary cost (cf. Table 3 for parameters and Table 7 for concrete values). As per the table, MPCLeague1 has the best runtime (both online and total) when compared to the others. This is attributed to the savings in round brought in by multi-input multiplication. Particularly for training, the savings in rounds is also due to the use of our customised garbled world which results in efficient conversions with a $2\times$ improvement (over Trident) in online rounds (cf. Table 2). This has a huge impact on the round complexity of training. Thus, for protocols which demand a fast response-time, MPCLeague1 is best suited due to its fast online phase. On the other hand, MPCLeague2 slightly underperforms compared to Trident (loss $< 1\%$) due to additional rounds incurred in verification. However, this gap closes-in for deeper networks. Comparison of our framework with Trident with respect to online runtime for training and inference is summarised in Fig. 1 and Fig. 2, respectively.

Table 1: Comparison of Trident (Chaudhari et al., 2020) with the two versions of **MPCLeague** for deep neural networks (NN-4).

Ref	Training & Inference				Training	Inference	
	$\mathsf{Com}_{\mathsf{on}}$	$\mathsf{Time}_{\mathsf{on}}$	$\mathsf{Com}_{\mathsf{tot}}$	$\mathsf{Time}_{\mathsf{tot}}$	Cost	Cost	TP
Trident	○	◐	◐	○	◐	○	○
MPCLeague1	●	●	○	●	○	●	●
MPCLeague2	◐	○	●	◐	●	◐	◐

– 'Com' - Communication, 'Time' - Runtime, 'Cost' - Overall Monetary Cost, 'TP' - Online Throughput, on - online, tot - total (cf. Table 3)
– ● - best, ◐ - 2$^{\text{nd}}$ best, ○ - least best (w.r.t parameter considered).

With respect to online communication, MPCLeague1 performs best due to improvements brought in by multi-input multiplication. However, with respect to the total communication, MPCLeague2 outperforms as it requires 1 less GC compared to MPCLeague1 and also due to the absence multi-input multiplication (which introduces extra overhead in the preprocessing communication).



(a) Online RunTime (b) Total RunTime

Figure 2: Inference in terms of $\mathsf{PT}_{\mathsf{on}}$ and $\mathsf{CT}_{\mathsf{on}}$ (cf. Table 3)

With respect to monetary cost (cf. §A.2.4 for concrete estimations), for training, MPCLeague1 has the highest cost (16% increase over Trident) because of the higher preprocessing communication

[5]Cumulative runtime is the sum of the up-time of all the hired servers.

[6]See https://cloud.google.com/vpc/network-pricing for network cost and https://cloud.google.com/compute/vm-instance-pricing for computation cost.

as mentioned earlier. Further, although MPCLeague2 is on par with Trident in terms of communication, its monetary cost is less (saving of 12.5%), owing to the presence of only 2 active parties compared to the 3 in Trident, which helps in reducing the combined runtime. As opposed to training, MPCLeague1 outperforms the rest for inference because the weightage given to communication is very less over the weightage given to the runtime while estimating the monetary cost Since for the case of inference, MPCLeague1 witnesses a 30% decrease in runtime but an 84% increase in communication when compared to MPCLeague2, the effect of increased communication is nullified by the savings in rounds while estimating the monetary cost. This results in MPCLeague1 having the best cost (28% less than Trident). MPCLeague2 also improves over Trident, for inference, by 24%.

For inference, throughput (cf. Table 4 for concrete numbers) helps to capture the combined effect of runtime and communication for multiple executions. Here, MPCLeague1 is a clear cut winner owing to the minimal online communication, better round complexity and presence of only 2 active parties. The reduced number of online parties also help MPCLeague2 to win over Trident. Concretely, both MPCLeague1 and MPCLeague2 have at least $2\times$ improvement over Trident.

Conclusion

We presented robust mixed protocol-framework for 4PC with one corruption and demonstrated its practicality by benchmarking for NN, CNN and SVM. We provide two variants of our framework –MPCLeague1 and MPCLeague2– with a trade-off in terms of the online runtime and overall communication. While MPCLeague1 has a better online performance over Trident, MPCLeague2 has a better overall performance. Further, our framework outperforms the 3PC framework (with abort) of ABY3 by several orders of magnitude where we trade the strongest security and better efficiency for sub-optimal resilience.

Acknowledgments

Nishat Koti would like to acknowledge financial support from Cisco PhD Fellowship 2020. Arpita Patra would like to acknowledge financial support from SERB MATRICS (Theoretical Sciences) Grant 2020 and Google India AI/ML Research Award 2020. Ajith Suresh would like to acknowledge financial support from Google PhD Fellowship 2019. The authors would also like to acknowledge the financial support from Google Cloud to perform the benchmarking.

References

Abdelrahaman Aly, Emmanuela Orsini, Dragos Rotaru, Nigel P. Smart, and Tim Wood. Zaphod: Efficiently combining LSSS and garbled circuits in SCALE. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC@CCS 2019, London, UK, November 11-15, 2019*, pp. 33–44. ACM, 2019.

Carsten Baum, Ivan Damgård, Tomas Toft, and Rasmus Winther Zakarias. Better preprocessing for secure multiparty computation. In *ACNS*, pp. 327–345, 2016.

Megha Byali, Arun Joseph, Arpita Patra, and Divya Ravi. Fast secure computation for small population over the internet. In *ACM CCS*, pp. 677–694, 2018.

Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. FLASH: fast and robust framework for privacy-preserving machine learning. *PETS*, 2020. URL `https://eprint.iacr.org/2019/1365`.

Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. In *ACM CCSW@CCS*, 2019. URL `https://eprint.iacr.org/2019/429`.

Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning. *NDSS*, 2020. URL `https://eprint.iacr.org/2019/1315`.

David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, 1988.

Cryptography and Privacy Engineering Group at TU Darmstadt. ENCRYPTO Utils. https://github.com/encryptogroup/ENCRYPTO_utils, 2017.

Ivan Damgård, Claudio Orlandi, and Mark Simkin. Yet another compiler for active security or: Efficient MPC over arbitrary rings. In *CRYPTO*, pp. 799–829, 2018.

Ivan Damgård, Daniel Escudero, Tore Kasper Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning. *IEEE S&P*, 2019.

Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, 1987.

S. Dov Gordon, Samuel Ranellucci, and Xiao Wang. Secure computation with low communication from cross-checking. In *ASIACRYPT*, pp. 59–85, 2018.

Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: Tool for automating secure two-party computations. IACR Cryptology ePrint Archive, 2010. https://eprint.iacr.org/2010/365.

Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In *CRYPTO*, pp. 359–378, 2015.

Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In *EUROCRYPT*, pp. 158–189, 2018.

Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning. *Cryptology ePrint Archive, Report 2020/592*, 2020. URL https://eprint.iacr.org/2020/592.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Ali Madani, Ramy Arnaout, Mohammad R. K. Mofrad, and Rima Arnaout. Fast and accurate classification of echocardiograms using deep learning. *CoRR*, 2017.

Eleftheria Makri, Dragos Rotaru, Nigel P. Smart, and Frederik Vercauteren. EPIC: efficient private image classification (or: Learning from the masters). In *CT-RSA*, pp. 473–492, 2019.

Sahar Mazloom, Phi Hung Le, Samuel Ranellucci, and S Dov Gordon. Secure parallel computation on national scale volumes of data. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 2487–2504, 2020.

Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In *CRYPTO*, 2020.

Payman Mohassel and Peter Rindal. ABY^3: A mixed protocol framework for machine learning. In *ACM CCS*, pp. 35–52, 2018.

Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S&P*, pp. 19–38, 2017.

Arpita Patra and Ajith Suresh. BLAZE: Blazing Fast Privacy-Preserving Machine Learning. *NDSS*, 2020. URL https://eprint.iacr.org/2020/042.

Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. Aby2.0: Improved mixed-protocol secure two-party computation. Cryptology ePrint Archive, Report 2020/1225, 2020. https://eprint.iacr.org/2020/1225.

M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *AsiaCCS*, pp. 707–721, 2018.

Dragos Rotaru and Tim Wood. Marbled circuits: Mixing arithmetic and boolean circuits with active security. In *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*, volume 11898, pp. 227–249, 2019.

Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sameer Wagh, Divya Gupta, and Nishanth Chandran. Securenn: 3-party secure computation for neural network training. *PETS*, pp. 26–49, 2019.

Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *arXiv preprint*, 2020. https://arxiv.org/abs/2004.02229v1.

Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5934–5938. IEEE, 2018.

Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, 1982.

A Appendix

A.1 Comparison of conversion protocols with existing frameworks

Comparison of the two versions of our framework with respect to ABY3 (Mohassel & Rindal, 2018) and Trident (Chaudhari et al., 2020) with respect to the conversion protocols appears in Table 2.

A.2 Benchmarking Results

A.2.1 Benchmarking Parameters

Benchmarking parameters used for comparing the performance of our framework with the existing ones appear in Table 3.

A.2.2 Detailed Benchmarking Result

Here we provide the detailed benchmarking results with respect to the latency, communication cost and monetary cost considering the parameters in Table 3. The results are summarized in Table 7.

A.2.3 Online Throughput for Inference

Table 4 summarises the online throughput for inference.

A.2.4 Monetary Cost

The monetary costs of the protocols are given in Table 5, Table 6. MPCLeague1 provides the best price for inference while MPCLeague2 provides the best price for training. This demonstrates that although MPCLeague1 has a higher preprocessing communication, the savings in online rounds play an important role in reducing the total price to be paid for inference. Whereas, for training, MPCLeague2 does not incur any extra overhead in communication (it does not rely on multi-input multiplication and uses only 1 GC), which leads to reducing the overall price compared to Trident. Also, since we pay the price for using only 2 active parties, this helps in obtaining better prices compared to Trident for inference and training. Concretely, our savings range up to 12.5% for inference and 28.3% for training over Trident.

Table 2: Sharing conversions of ABY3 Mohassel & Rindal (2018), Trident Chaudhari et al. (2020), **MPCLeague1**, and **MPCLeague2**

Protocol	Reference	Communication (Preprocessing)	Rounds (Online)	Communication (Online)
Arithmetic to Garbled to Arithmetic	ABY3	$\lvert\mathsf{C}^{\mathsf{S3,A3,F}}\rvert + \ell\kappa$	2	$4\ell\kappa$
	Trident	$\lvert\mathsf{C}^{\mathsf{S2,S2+,F}}\rvert + 2\ell\kappa + \ell$	2	$\ell\kappa + 3\ell$
	MPCLeague1	$2\lvert\mathsf{C}^{\mathsf{F}}\rvert + 6\ell\kappa + \ell$	1	$2\ell\kappa + \ell$
	MPCLeague2	$\lvert\mathsf{C}^{\mathsf{F}}\rvert + 3\ell\kappa + \ell$	2	$\ell\kappa + 2\ell$
Arithmetic to Garbled to Boolean	ABY3	$\lvert\mathsf{C}^{\mathsf{A3,F}}\rvert + \kappa$	2	$3\ell\kappa$
	Trident	$\lvert\mathsf{C}^{\mathsf{S2,F}}\rvert + 2\ell\kappa + \ell$	2	$\ell\kappa + 3\ell$
	MPCLeague1	$2\lvert\mathsf{C}^{\mathsf{F}}\rvert + 6\ell\kappa + \ell$	1	$2\ell\kappa + \ell$
	MPCLeague2	$\lvert\mathsf{C}^{\mathsf{F}}\rvert + 3\ell\kappa + \ell$	2	$\ell\kappa + 2\ell$
Boolean to Garbled to Arithmetic	ABY3	$\lvert\mathsf{C}^{\mathsf{S3,F}}\rvert + \ell\kappa$	2	$4\ell\kappa$
	Trident	$\lvert\mathsf{C}^{\mathsf{S2+,F}}\rvert + 2\ell\kappa + \ell$	2	$\ell\kappa + 3\ell$
	MPCLeague1	$2\lvert\mathsf{C}^{\mathsf{F}}\rvert + 6\ell\kappa + \ell$	1	$2\ell\kappa + \ell$
	MPCLeague2	$\lvert\mathsf{C}^{\mathsf{F}}\rvert + 3\ell\kappa + \ell$	2	$\ell\kappa + 2\ell$
Boolean to Garbled to Boolean	ABY3	$\lvert\mathsf{C}^{\mathsf{F}}\rvert + \ell\kappa$	2	$3\ell\kappa$
	Trident	$\lvert\mathsf{C}^{\mathsf{F}}\rvert + 2\ell\kappa + \ell$	2	$\ell\kappa + 3\ell$
	MPCLeague1	$2\lvert\mathsf{C}^{\mathsf{F}}\rvert + 6\ell\kappa + \ell$	1	$2\ell\kappa + \ell$
	MPCLeague2	$\lvert\mathsf{C}^{\mathsf{F}}\rvert + 3\ell\kappa + \ell$	2	$\ell\kappa + 2\ell$
Arithmetic to Boolean	ABY3	$12\ell\log_2\ell + 12\ell$	$1 + \log_2\ell$	$9\ell\log_2\ell + 9\ell$
	Trident	$3\ell\log_2\ell + 2\ell$	$1 + \log_2\ell$	$3\ell\log_2\ell + \ell$
	MPCLeague	$3\ell\log_2\ell + \ell$	$\log_2\ell$ *	$3\ell\log_2\ell + \ell$
Boolean to Arithmetic	ABY3	$12\ell\log_2\ell + 12\ell$	$1 + \log_2\ell$	$9\ell\log_2\ell + 9\ell$
	Trident	$3\ell^2 + \ell$	1	3ℓ
	MPCLeague	$3\ell^2 + \ell$	1	3ℓ

– Notations: ℓ - size of ring in bits, κ - computational security parameter.
– GC notations: C^{S2} - 2-input garbled subtraction circuit; $\mathsf{C}^{\mathsf{S2+}}$ - 2-input garbled subtraction circuit with its decoding information; C^{S3} - 3-input garbled subtraction circuit (with input: $\mathsf{x}, \mathsf{y}, \mathsf{z}$, output: $\mathsf{x} - \mathsf{y} - \mathsf{z}$); C^{A3} - 3-input garbled addition circuit; $\mathsf{C}^{1,\ldots,\mathsf{i}}$ - set of GCs $\mathsf{C}^1, \ldots, \mathsf{C}^{\mathsf{i}}$; $\lvert\mathsf{C}^{1,\ldots,\mathsf{i}}\rvert$ - size of $\mathsf{C}^{1,\ldots,\mathsf{i}}$.
*: can be reduced to $\log_4\ell$.

Table 3: Benchmarking parameters

Notation	Description
$\mathsf{T}_{\mathsf{on,i}}$	Online runtime of party P_i.
$\mathsf{T}_{\mathsf{tot,i}}$	Total runtime of party P_i.
$\mathsf{PT}_{\mathsf{on}}$	Protocol online runtime; $\mathsf{max}_{\mathsf{i}}\{\mathsf{T}_{\mathsf{on,i}}\}$.
$\mathsf{PT}_{\mathsf{tot}}$	Protocol total runtime; $\mathsf{max}_{\mathsf{i}}\{\mathsf{T}_{\mathsf{tot,i}}\}$.
$\mathsf{CT}_{\mathsf{on}}$	Cumulative online runtime; $\Sigma_i \mathsf{T}_{\mathsf{on,i}}$.
$\mathsf{CT}_{\mathsf{tot}}$	Cumulative total runtime; $\Sigma_i \mathsf{T}_{\mathsf{tot,i}}$.
$\mathsf{Comm}_{\mathsf{on}}$	Online communication.
$\mathsf{Comm}_{\mathsf{tot}}$	Total communication.
TP	Online throughput; (no. of queries handled per minute in the online phase)

Table 4: Online Throughput (#predictions/minute) for Inference

Ref.	SVM	NN-1	NN-2	NN-3	NN-4
ABY3	247.10	752.94	118.06	52.96	0.39
SWIFT (3PC)	1613.45	4129.03	4122.27	1877.75	342.27
SWIFT (4PC)	3331.89	9035.29	9031.45	3898.48	1022.15
Trident	1749.43	5154.36	5153.64	2064.52	511.07
MPCLeague1	5585.45	13356.52	13352.51	6269.39	1067.62
MPCLeague2	3206.68	8126.98	8123.92	3728.16	1022.15

Table 5: Monetary Costs for Training (1 mini-batch, 10^3 iterations)

Ref.	NN-1	NN-2	NN-3	NN-4
ABY3	318\$	2530\$	10200\$	1350000\$
Trident	28\$	37\$	175\$	2880\$
MPCLeague1	61\$	70\$	248\$	3340\$
MPCLeague2	26\$	34\$	160\$	2520\$

Table 6: Monetary Costs for Inference (10^5 predictions)

Ref.	SVM	NN-1	NN-2	NN-3	NN-4
ABY3	2750\$	1100\$	2610\$	6220\$	546000\$
SWIFT (3PC)	919\$	448\$	451\$	964\$	3930\$
SWIFT (4PC)	868\$	418\$	420\$	902\$	3390\$
Trident	1150\$	456\$	457\$	1130\$	4340\$
MPCLeague1	568\$	324\$	327\$	688\$	3110\$
MPCLeague2	796\$	347\$	347\$	828\$	3290\$

Table 7: Benchmarking results for ML training and inference with comparisons with Trident Chaudhari et al. (2020), ABY3 Mohassel & Rindal (2018), SWIFT Koti et al. (2020). Parameters considered are $\mathsf{PT}_{\mathsf{on}}$, $\mathsf{PT}_{\mathsf{tot}}$, $\mathsf{CT}_{\mathsf{on}}$, $\mathsf{CT}_{\mathsf{tot}}$, $\mathsf{Comm}_{\mathsf{on}}$ and $\mathsf{Comm}_{\mathsf{tot}}$(cf. Table 3 for details). Time in given in seconds and communication in MB.

Parameter	Protocol	ML Training				ML Inference				
		NN-1	NN-2	NN-3	NN-4	SVM	NN-1	NN-2	NN-3	NN-4
Protocol Execution Time Online ($\mathsf{PT}_{\mathsf{on}}$)	ABY3	1.24×10^1	7.27×10^1	2.87×10^2	3.84×10^4	1.55×10^1	5.10×10^0	5.48×10^0	1.38×10^1	1.56×10^2
	SWIFT (3PC)					4.76×10^0	1.86×10^0	1.86×10^0	4.09×10^0	1.27×10^1
	SWIFT (4PC)					4.61×10^0	1.70×10^0	1.70×10^0	3.94×10^0	1.25×10^1
	Trident	2.11×10^0	2.20×10^0	7.93×10^0	3.79×10^1	4.39×10^0	1.49×10^0	1.49×10^0	3.72×10^0	1.23×10^1
	MPCLeague1	1.71×10^0	1.79×10^0	5.41×10^0	2.57×10^1	2.75×10^0	1.15×10^0	1.15×10^0	2.45×10^0	7.67×10^0
	MPCLeague2	2.31×10^0	2.40×10^0	8.13×10^0	3.81×10^1	4.79×10^0	1.89×10^0	1.89×10^0	4.12×10^0	1.27×10^1
Protocol Execution Time Total ($\mathsf{PT}_{\mathsf{tot}}$)	ABY3	3.36×10^1	2.25×10^2	9.85×10^2	1.28×10^5	1.57×10^1	5.47×10^0	6.52×10^0	1.65×10^1	4.90×10^2
	SWIFT (3PC)					5.19×10^0	2.42×10^0	2.42×10^0	4.96×10^0	1.54×10^1
	SWIFT (4PC)					4.89×10^0	2.17×10^0	2.17×10^0	4.71×10^0	1.51×10^1
	Trident	5.53×10^0	6.26×10^0	2.16×10^1	2.57×10^2	4.49×10^0	1.77×10^0	1.77×10^0	4.31×10^0	1.47×10^1
	MPCLeague1	5.21×10^0	5.76×10^0	1.75×10^1	1.85×10^2	3.04×10^0	1.61×10^0	1.61×10^0	3.22×10^0	1.03×10^1
	MPCLeague2	5.71×10^0	6.27×10^0	2.02×10^1	1.98×10^2	4.89×10^0	2.17×10^0	2.17×10^0	4.71×10^0	1.51×10^1
Cumulative Protocol Execution Time Online ($\mathsf{CT}_{\mathsf{on}}$)	ABY3	2.85×10^1	1.54×10^2	4.76×10^2	5.92×10^4	3.12×10^1	1.03×10^1	1.13×10^1	2.85×10^1	3.76×10^2
	SWIFT (3PC)					9.61×10^0	3.81×10^0	3.81×10^0	8.27×10^0	2.54×10^1
	SWIFT (4PC)					9.30×10^0	3.50×10^0	3.50×10^0	7.96×10^0	2.51×10^1
	Trident	6.33×10^0	6.60×10^0	2.38×10^1	1.14×10^2	1.32×10^1	4.46×10^0	4.46×10^0	1.12×10^1	3.69×10^1
	MPCLeague1	3.50×10^0	3.68×10^0	1.09×10^1	5.14×10^1	5.59×10^0	2.38×10^0	2.38×10^0	4.99×10^0	1.54×10^1
	MPCLeague2	4.80×10^0	4.99×10^0	1.65×10^1	7.64×10^1	9.70×10^0	3.97×10^0	3.97×10^0	8.43×10^0	2.56×10^1
Cumulative Protocol Execution Time Total ($\mathsf{CT}_{\mathsf{tot}}$)	ABY3	8.04×10^1	5.15×10^2	2.13×10^3	2.80×10^5	3.16×10^1	1.12×10^1	1.37×10^1	3.47×10^1	1.20×10^3
	SWIFT (3PC)					1.07×10^1	5.29×10^0	5.29×10^0	1.05×10^1	3.22×10^1
	SWIFT (4PC)					1.02×10^1	4.94×10^0	4.95×10^0	1.01×10^1	3.20×10^1
	Trident	1.72×10^1	1.91×10^1	5.97×10^1	6.56×10^2	1.35×10^1	5.39×10^0	5.39×10^0	1.28×10^1	4.32×10^1
	MPCLeague1	1.47×10^1	1.60×10^1	4.31×10^1	4.46×10^2	6.45×10^0	3.83×10^0	3.83×10^0	7.15×10^0	2.22×10^1
	MPCLeague2	1.56×10^1	1.70×10^1	4.83×10^1	4.47×10^2	1.01×10^1	4.90×10^0	4.90×10^0	1.01×10^1	3.18×10^1
Communication Online ($\mathsf{Comm}_{\mathsf{on}}$)	ABY3	1.22×10^3	1.07×10^4	4.30×10^4	5.77×10^6	4.26×10^0	8.23×10^0	7.62×10^1	1.70×10^2	2.33×10^4
	SWIFT (3PC)					0.95×10^0	0.04×10^0	0.16×10^0	3.54×10^0	5.26×10^1
	SWIFT (4PC)					0.59×10^0	0.03×10^0	0.10×10^0	2.33×10^0	3.52×10^1
	Trident	7.74×10^0	4.55×10^1	5.62×10^2	1.17×10^4	0.59×10^0	0.03×10^0	0.11×10^0	2.33×10^0	3.52×10^1
	MPCLeague1	7.98×10^0	4.53×10^1	5.47×10^2	1.15×10^4	0.56×10^0	0.03×10^0	0.10×10^0	2.23×10^0	3.37×10^1
	MPCLeague2	7.74×10^0	4.55×10^1	5.62×10^2	1.17×10^4	0.59×10^0	0.03×10^0	0.11×10^0	2.33×10^0	3.52×10^1
Communication Total ($\mathsf{Comm}_{\mathsf{tot}}$)	ABY3	3.06×10^3	2.56×10^4	1.03×10^5	1.35×10^7	9.95×10^0	1.93×10^1	1.78×10^2	4.01×10^2	5.44×10^4
	SWIFT (3PC)					2.20×10^0	0.11×10^0	0.47×10^0	9.59×10^0	1.48×10^2
	SWIFT (4PC)					1.37×10^0	0.06×10^0	0.25×10^0	5.59×10^0	8.42×10^1
	Trident	1.60×10^2	2.51×10^2	1.52×10^3	2.84×10^4	1.42×10^0	0.06×10^0	0.26×10^0	5.59×10^0	8.42×10^1
	MPCLeague1	5.93×10^2	6.93×10^2	2.58×10^3	3.62×10^4	2.85×10^0	0.11×10^0	0.45×10^0	1.03×10^1	1.51×10^2
	MPCLeague2	1.59×10^2	2.40×10^2	1.46×10^3	2.62×10^4	1.42×10^0	0.06×10^0	0.25×10^0	5.47×10^0	8.20×10^1