SWIFT: SUPER-FAST AND ROBUST PRIVACY-PRESERVING MACHINE LEARNING

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

ABSTRACT

Privacy-preserving Machine Learning (PPML) has seen a visible shift towards the adoption of the Secure Outsourced Computation (SOC) paradigm due to the heavy computation that it entails. In the SOC paradigm, computation is outsourced to a set of powerful and specially equipped servers that provide service on a pay-per-use basis. In this work, we propose SWIFT, a *robust* PPML framework for a range of machine learning (ML) algorithms in SOC setting, that guarantees output delivery to the users irrespective of any adversarial behaviour. Robustness, a highly desirable feature, evokes user participation without the fear of denial of service. At the heart of our framework lies a highly-efficient, maliciously-secure, three-party computation (3PC) over rings that provides guaranteed output delivery (GOD) in the honest-majority setting. To the best of our knowledge, SWIFT is the first robust and efficient PPML framework in the 3PC setting and is as fast as (and is strictly better in some cases than) the best-known 3PC framework BLAZE (Patra et al. NDSS'20), which only achieves fairness. We extend our 3PC framework for four parties (4PC). In this regime, SWIFT is as fast as the best known *fair* 4PC framework Trident (Chaudhari et al. NDSS'20) and twice faster than the best-known *robust* 4PC framework FLASH (Byali et al. PETS'20). We demonstrate our framework's practical relevance by benchmarking ML algorithms such as Logistic Regression and deep Neural Networks such as VGG16 and LeNet, both over a 64-bit ring in a WAN setting. For deep NN, our results testify to our claims that we provide improved security guarantee while incurring no additional overhead for 3PC and obtaining $2\times$ improvement for 4PC.

Privacy-Preserving Machine Learning (PPML), a booming field of research, allows Machine Learning (ML) computations over private data of users while ensuring the privacy of the data. PPML finds applications in sectors that deal with sensitive/confidential data, e.g. healthcare, finance, autonomous driving, etc. However, PPML solutions make the already computationally heavy ML algorithms more compute-intensive. An average end-user who lacks the infrastructure required to run these tasks prefers to outsource the computation to a powerful set of specialized cloud servers and leverage their services without compromising privacy, on a pay-per-use basis. This is addressed by the Secure Outsourced Computation (SOC) paradigm, and thus is an apt fit for the need of the moment. Many recent works (Mohassel & Zhang, 2017; Riazi et al., 2018; Mohassel & Rindal, 2018; Wagh et al., 2019; Chaudhari et al., 2019; Byali et al., 2020; Chaudhari et al., 2020; Patra & Suresh, 2020) exploit Secure Multiparty Computation (MPC) techniques to realize PPML in the SOC setting where the servers enact the role of the parties. Informally, MPC enables n mutually distrusting parties to compute a function over their private inputs while ensuring privacy of the same against an adversary controlling up to t parties. Both the training and prediction phases of PPML can be realized in the SOC setting. The common approach of outsourcing followed in the PPML literature, as well as by our work, requires users to secret-share[1] their inputs between the set of hired

*This work will appear in USENIX Security'21 and the full version is available at https://eprint.iacr.org/2020/592.

[1]The threshold of the secret-sharing is decided based on the number of corrupt servers so that privacy is preserved.

(untrusted) servers, who jointly interact and compute the secret-shared output, and reconstruct it to the user.

In a bid to improve practical efficiency, many recent works (Damgård et al., 2012; Keller et al., 2016; Damgård et al., 2018; Chaudhari et al., 2019; Byali et al., 2020; Chaudhari et al., 2020; Patra & Suresh, 2020) cast their protocols into the preprocessing model where in the input-independent (yet function-dependent) phase, computationally heavy tasks are computed in advance, resulting in a fast online phase. This paradigm suits scenarios analogous to PPML setting, where functions (ML algorithms) typically need to be evaluated a large number of times, and the function description is known beforehand. To further enhance practical efficiency by leveraging CPU optimizations, recent works (Bogdanov et al., 2008; Damgård et al., 2018; Cramer et al., 2003; Demmler et al., 2015; Damgård et al., 2019) propose MPC protocols that work over 32 or 64 bit rings. Lastly, solutions for a small number of parties have received huge momentum due to the many cost-effective customizations that they permit, for instance, a cheaper realisation of multiplication through custom-made secret sharing schemes (Araki et al., 2016; 2017; Chaudhari et al., 2019; Patra & Suresh, 2020; Chaudhari et al., 2020; Byali et al., 2020).

We now motivate the need for robustness, aka guaranteed output delivery (GOD) over fairness, or even abort security, in the domain of PPML. Robustness provides the guarantee of output delivery to all protocol participants, no matter how the adversary misbehaves. Robustness is extremely crucial for real-world deployment and usage of PPML techniques. Consider the following scenario wherein an ML model owner wishes to provide inference service. The model owner shares the model parameters between the servers, while the end-users share their queries. A protocol that provides security with abort or fairness will not suffice as in both cases, a malicious adversary can lead to the protocol aborting, resulting in the user not obtaining the desired output. This leads to denial of service and heavy economic losses for the service provider. For data providers, as more training data leads to more accurate models, collaboratively building a model enables them to provide better ML services, and consequently, attract more clients. A robust framework encourages active involvement from multiple data providers. Hence, for seamless adoption of PPML solutions in the real-world, robustness of the protocol is of utmost importance. The hall-mark result of Cleve (1986) suggests that honest-majority amongst the servers is necessary to achieve robustness. Consequent to the discussion above, in this work we design a PPML framework in the 3,4-party setting with honest majority, both of which have drawn enormous attention recently (Mohassel et al., 2015; Araki et al., 2016; Furukawa et al., 2017; Araki et al., 2017; Byali et al., 2018; Chaudhari et al., 2019; Boneh et al., 2019; Patra & Suresh, 2020; Chaudhari et al., 2020; Byali et al., 2020; Boyle et al., 2019). Our protocols work over rings, are cast in the preprocessing model, and achieve GOD.

Table 1: 3PC and 4PC: Comparison of SWIFT with its closest competitors in terms of Communication and Round Complexity

Building Blocks	3PC					4PC				
	Ref.	Pre.	Online		Security	Ref.	Pre.	Online		Security
		Comm. (ℓ)	Rounds	Comm. (ℓ)			Comm. (ℓ)	Rounds	Comm. (ℓ)	
Multiplication	Boneh et al. (2019)	1	1	2	Abort					
	Boyle et al. (2019)	-	3	3	GOD	Trident	3	1	3	Fair
	BLAZE	3	1	3	Fair	FLASH	6	1	6	GOD
	SWIFT	**3**	**1**	**3**	GOD	**SWIFT**	**3**	**1**	**3**	GOD
Dot Product	BLAZE	$3\mathsf{n}$	1	3	Fair	Trident	3	1	3	Fair
	SWIFT	**3**	**1**	**3**	GOD	FLASH	6	1	6	GOD
						SWIFT	**3**	**1**	**3**	GOD
Dot Product with Truncation	BLAZE	$3\mathsf{n}+2$	1	3	Fair	Trident	6	1	3	Fair
	SWIFT	**15**	**1**	**3**	GOD	FLASH	8	1	6	GOD
						SWIFT	**4**	**1**	**3**	GOD
Bit Extraction	BLAZE	9	$1+\log \ell$	9	Fair	Trident	≈ 8	$\log \ell + 1$	≈ 7	Fair
	SWIFT	**9**	$1+\log \ell$	**9**	GOD	FLASH	14	$\log \ell$	14	GOD
						SWIFT	$\approx$ **7**	$\log \ell$	$\approx$ **7**	GOD
Bit to Arithmetic	BLAZE	9	1	4	Fair	Trident	≈ 3	1	3	Fair
	SWIFT	**9**	**1**	**4**	GOD	FLASH	6	1	8	GOD
						SWIFT	$\approx$ **3**	**1**	**3**	GOD
Bit Injection	BLAZE	12	2	7	Fair	Trident	≈ 6	1	3	Fair
	SWIFT	**12**	**2**	**7**	GOD	FLASH	8	2	10	GOD
						SWIFT	$\approx$ **6**	**1**	**3**	GOD

– Notations: ℓ - size of ring in bits, n - size of vectors for dot product.

Related Work In the PPML domain, MPC has been used for various ML algorithms such as Decision Trees (Lindell & Pinkas, 2002), Linear Regression (Du & Atallah, 2001; Sanil et al., 2004), k-

means clustering (Jagannathan & Wright, 2005; Bunn & Ostrovsky, 2007), SVM Classification (Yu et al., 2006; Vaidya et al., 2008), Logistic Regression (Slavkovic et al., 2007). In the 3PC SOC setting, the works of ABY3 (Mohassel & Rindal, 2018) and SecureNN (Wagh et al., 2019) provide security with abort. This was followed by ASTRA (Chaudhari et al., 2019), which improves upon ABY3 and achieves security with fairness. ASTRA presents primitives to build protocols for Linear Regression and Logistic Regression inference. Recently, BLAZE improves the efficiency of ASTRA and additionally tackles training for the above ML tasks, which requires building additional PPML building blocks, such as truncation and bit to arithmetic conversions. In the 4PC setting, the first robust framework for PPML was provided by FLASH (Byali et al., 2020) which proposed efficient building blocks for ML such as dot product, truncation, MSB extraction, and bit conversion. The works of (Mohassel & Zhang, 2017; Mohassel & Rindal, 2018; Wagh et al., 2019; Chaudhari et al., 2019; Patra & Suresh, 2020; Chaudhari et al., 2020; Byali et al., 2020) work over rings to garner practical efficiency. In terms of efficiency, BLAZE and respectively FLASH and Trident are the closest competitors of this work in 3 and 4 party settings. We now present our contributions and compare them with these works.

1 Our Contributions

We propose, **SWIFT**, a robust maliciously-secure framework for PPML in the SOC setting, with a set of 3 and 4 servers having an honest-majority. At the heart of our framework lies highly-efficient, maliciously-secure, 3PC and 4PC over rings (both $\mathbb{Z}_{2^\ell}$ and $\mathbb{Z}_{2^1}$) that provide GOD in the honest-majority setting. We cast our protocols in the preprocessing model, which helps obtain a fast online phase. To the best of our knowledge, SWIFT is the first robust and efficient PPML framework in the 3PC setting and is as fast as the best known *fair* 3PC framework BLAZE (Patra & Suresh, 2020). We extend our 3PC framework for 4 parties. In this regime, SWIFT is as fast as the best known *fair* 4PC framework Trident (Chaudhari et al., 2020) and twice faster than best known *robust* 4PC framework FLASH (Byali et al., 2020). We detail our contributions next.

Robust 3/4PC PPML frameworks. The framework consists of a range of primitives realized in a privacy-preserving way which is ensured via running computation in a secret-shared fashion. We use secret-sharing over both $\mathbb{Z}_{2^\ell}$ and its special instantiation $\mathbb{Z}_{2^1}$ and refer them as *arithmetic* and respectively *boolean* sharing. Our framework consists of realizations for all primitives needed for general MPC and PPML such as multiplication, dot-product, truncation, bit extraction (given arithmetic sharing of a value v, this is used to generate boolean sharing of the most significant bit (msb) of the value), bit to arithmetic sharing conversion (converts the boolean sharing of a single bit value to its arithmetic sharing), bit injection (computes the arithmetic sharing of $\mathsf{b} \cdot \mathsf{v}$, given the boolean sharing of a bit b and the arithmetic sharing of a ring element v) and above all, input sharing and output reconstruction in the SOC setting. A highlight of our 3PC framework, which, to the best of our knowledge is achieved for the first time, is a robust dot-product protocol whose (amortized) communication cost is independent of the vector size, which we obtain by extending the techniques of (Boneh et al., 2019; Boyle et al., 2019). The performance comparison in terms of concrete cost for communication and rounds, for PPML primitives in both 3PC and 4PC setting, appear in Table 1. As claimed, SWIFT is on par with BLAZE for most of the primitives (while improving security from fair to GOD) and is strictly better than BLAZE in case of dot product and dot product with truncation. For 4PC, SWIFT is on par with Trident in most cases (and is slightly better for dot product with truncation and bit injection), while it is doubly faster than FLASH. Since BLAZE outperforms the 3PC abort framework of ABY3 (Mohassel & Rindal, 2018) while Trident outperforms the known 4PC with abort (Gordon et al., 2018), SWIFT attains robustness with better cost than the know protocols with weaker guarantees. No performance loss coupled with the strongest security guarantee makes our robust framework an opt choice for practical applications, including PPML.

Applications and Benchmarking. We demonstrate the practicality of our protocols by benchmarking PPML, particularly, Logistic Regression (training and inference) and popular Neural Networks (inference) such as (Mohassel & Zhang, 2017) (NN-1), LeNet (LeCun et al., 1998) (NN-2) and VGG16 (Simonyan & Zisserman, 2014) (NN-3) having millions of parameters. The NN training requires mixed-world conversions (Demmler et al., 2015; Mohassel & Rindal, 2018; Chaudhari et al., 2020), which we leave as future work. Our PPML blocks can be used to perform training and inference of Linear Regression, Support Vector Machines, and Binarized Neural Networks (as demonstrated in (Chaudhari et al., 2019; 2020; Patra & Suresh, 2020; Byali et al., 2020)).

We use a 64-bit ring ($\mathbb{Z}_{2^{64}}$). The benchmarking is performed over a WAN that was instantiated using n1-standard-8 instances of Google Cloud[2]. The machines are equipped with 2.3 GHz Intel Xeon E5 v3 (Haswell) processors supporting hyper-threading, with 8 vCPUs, and 30 GB of RAM Memory and with a bandwidth of 50 Mbps. We implement our protocols using the publicly available ENCRYPTO library (Cryptography & at TU Darmstadt, 2017) in C++17. We use *throughput* (TP) as the benchmarking parameter following BLAZE and ABY3 (Mohassel & Rindal, 2018) to analyse the effect of improved communication and round complexity in a single shot. Here, TP denotes the number of operations ("iterations" for the case of training and "queries" for the case of inference) that can be performed per minute. Due to our protocols' asymmetric nature, we perform *load balancing*, where we run several parallel execution threads, each with roles of the servers changed. We report communication and runtime of protocols for online phase and total.

Table 2: Logistic Regression training and inference. TP is given in (#it/min) for training and (#queries/min) for inference.

Setting	Ref.	Online (TP in $\times 10^3$)			Total	
		Latency (s)	Com [KB]	TP	Latency (s)	Com [KB]
3PC Training	BLAZE	0.74	50.26	4872.38	0.93	203.35
	SWIFT	1.05	50.32	4872.38	1.54	203.47
3PC Inference	BLAZE	0.66	0.28	7852.05	0.84	0.74
	SWIFT	0.97	0.34	6076.46	1.46	0.86
4PC Training	FLASH	0.83	88.93	5194.18	1.11	166.75
	SWIFT	0.83	41.32	11969.48	1.11	92.91
4PC Inference	FLASH	0.76	0.50	7678.40	1.04	0.96
	SWIFT	0.75	0.27	15586.96	1.03	0.57

The results for logistic regression are summarized in Table 2. We observe that the online TP for the case of 3PC inference is slightly lower compared to that of BLAZE. This is because the total number of rounds for the inference phase is slightly higher in our case due to the additional rounds introduced by the verification mechanism (aka *verify* phase which also needs broadcast). This gap becomes less evident for protocols with more number of rounds, as is demonstrated in the case of NN (presented next), where verification for several iterations is clubbed together, making the overhead for verification insignificant. For the case of 4PC, our solution outperforms FLASH in terms of communication as well as throughput. Concretely, we observe a $2\times$ improvement in TP for inference and a $2.3\times$ improvement for training. For Trident (Chaudhari et al., 2020), we observe a drop of 15.86% in TP for inference due to the extra rounds required for verification to achieve GOD. This loss is, however, traded-off with a stronger security guarantee. For training, we are on par with Trident as the effect of extra rounds becomes less significant for more number of rounds, as will also be evident from the comparisons for NN inference. As a final remark, note that our 4PC sees roughly $2.5\times$ improvement compared to our 3PC for logistic regression.

Table 3: 3PC NN Inference. TP is given in (#queries/min).

Network	Ref.	Online			Total	
		Latency (s)	Com [MB]	TP	Latency (s)	Com [MB]
NN-1	BLAZE	1.92	0.04	49275.19	2.35	0.11
	SWIFT	2.22	0.04	49275.19	2.97	0.11
NN-2	BLAZE	4.77	3.54	536.52	5.61	9.59
	SWIFT	5.08	3.54	536.52	6.22	9.59
NN-3	BLAZE	15.58	52.58	36.03	18.81	148.02
	SWIFT	15.89	52.58	36.03	19.29	148.02

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

Table 3 summarises the results for 3PC NN inference. As illustrated, the performance of our 3PC framework is on par with BLAZE while providing a better security guarantee. Table 4 summarises NN inference for 4PC setting. Here, we outperform FLASH in every aspect, with the improvement in TP being at least $2.5\times$ for each NN architecture. Further, we are on par with Trident (Chaudhari et al., 2020) because the extra rounds required for verification get amortized with an increase in the number of rounds required for computing NN inference. This establishes the practical relevance of our work.

Table 4: 4PC NN Inference. TP is given in (#queries/min).

Network	Ref.	Online			Total	
		Latency (s)	Com [MB]	TP	Latency (s)	Com [MB]
NN-1	FLASH	1.70	0.06	59130.23	2.17	0.12
	SWIFT	1.70	0.03	147825.56	2.17	0.06
NN-2	FLASH	3.93	5.51	653.67	4.71	10.50
	SWIFT	3.93	2.33	1672.55	4.71	5.40
NN-3	FLASH	12.65	82.54	43.61	15.31	157.11
	SWIFT	12.50	35.21	110.47	15.14	81.46

As a final remark, our 4PC sees roughly $3\times$ improvement over our 3PC for NN inference. This reflects improvements brought in by the additional honest server in the system.

CONCLUSION

In this work, we presented an efficient framework for PPML that achieves the strongest security of GOD or robustness. Our 3PC protocol builds upon the recent work of BLAZE (Patra & Suresh, 2020) and achieves almost similar (in some cases, better) performance albeit improving the security guarantee. For the case of 4PC, we outperform the best-known– (a) robust protocol of FLASH (Byali et al., 2020) by $2\times$ performance-wise and (b) fair protocol of Trident (Chaudhari et al., 2020) by uplifting its security. We leave the problem of extending our framework to support mixed-world conversions as well as to design protocols to support algorithms like Decision Trees, k-means Clustering etc. as open problem.

ACKNOWLEDGMENTS

Nishat Koti would like to acknowledge financial support from Cisco PhD Fellowship 2020. Mahak Pancholi would like to acknowledge financial support from Cisco MTech 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

Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM CCS*, pp. 805–817, 2016.

Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and Or Weinstein. Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier. In *IEEE S&P*, pp. 843–862, 2017.

Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, pp. 192–206, 2008.

Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear pcps. In *CRYPTO*, pp. 67–97, 2019.

Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In *ACM CCS*, pp. 869–886, 2019.

Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *ACM CCS*, pp. 486–497, 2007.

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.

Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *ACM STOC*, pp. 364–369, 1986.

Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pp. 596–613, 2003.

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

Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pp. 643–662, 2012.

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.

Wenliang Du and Mikhail J. Atallah. Privacy-preserving cooperative scientific computations. In *IEEE (CSFW-14*, pp. 273–294, 2001.

Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *EUROCRYPT*, pp. 225–255, 2017.

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

Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *ACM SIGKDD*, pp. 593–599, 2005.

Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *ACM CCS*, pp. 830–842, 2016.

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

Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *J. Cryptology*, pp. 177–206, 2002.

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.

Payman Mohassel, Mike Rosulek, and Ye Zhang. Fast and secure three-party computation: The garbled circuit approach. In *ACM CCS*, pp. 591–602, 2015.

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

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.

Ashish P. Sanil, Alan F. Karr, Xiaodong Lin, and Jerome P. Reiter. Privacy preserving regression modelling via distributed computation. In *ACM SIGKDD*, pp. 677–682, 2004.

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

Aleksandra B. Slavkovic, Yuval Nardi, and Matthew M. Tibbits. Secure logistic regression of horizontally and vertically partitioned distributed databases. In *ICDM*, pp. 723–728, 2007.

Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving SVM classification. *Knowl. Inf. Syst.*, pp. 161–178, 2008.

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

Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving SVM classification on vertically partitioned data. In *PAKDD*, pp. 647–656, 2006.