Secret Sharing MPC on FPGAs in the Datacenter

Pierre-François Wolfe*, Rushi Patel†, Robert Munafo‡, Mayank Varia§, and Martin Herbordt¶
* † ‡ ¶ Dept. of Electrical and Computer Engineering & §Dept. of Computer Science,
Boston University, Boston, USA
Email: *pwolfe@bu.edu, †ruship@bu.edu, ‡rmunafo@bu.edu, §varia@bu.edu, ¶herbordt@bu.edu

***Abstract*—Multi-Party Computation (MPC) is a technique enabling data from several sources to be used in a secure computation revealing only the result while protecting the original data, facilitating shared utilization of data sets gathered by different entities. The presence of Field Programmable Gate Array (FPGA) hardware in datacenters can provide accelerated computing as well as low latency, high bandwidth communication that bolsters the performance of MPC and lowers the barrier to using MPC for many applications. In this work, we propose a Secret Sharing FPGA design based on the protocol described by Araki et al. [1]. We compare our hardware design to the original authors' software implementations of Secret Sharing and to work accelerating MPC protocols based on Garbled Circuits with FPGAs. Our conclusion is that Secret Sharing in the datacenter is competitive and when implemented on FPGA hardware was able to use at least 10× fewer computer resources than the original work using CPUs.**

***Index Terms*—Multiparty Computation, Secret Sharing, Secure Computation, FPGA, Datacenter, Cloud Service**

I. Introduction

Many organizations face the problem of wanting to perform useful computations when the underlying data is sensitive. Cryptographically secure multi-party computation (MPC) allows people to outsource encoded versions of their data to several compute parties, who can then analyze the data without reading it. As defined in pending legislation within the United States Senate, "the term 'secure multi-party computation' means a computerized system that enables different participating entities in possession of private sets of data to link and aggregate their data sets for the exclusive purpose of performing a finite number of pre-approved computations without transferring or otherwise revealing any private data to each other or anyone else" [2].

MPC has been an active area of research for about 40 years [3]–[6], and it has been deployed to protect data in the healthcare [7], [8], education [9], [10], finance [11]–[13], and technology [14], [15] sectors. Nevertheless, recent surveys reveal a few companies with specialized MPC offerings For adoption of MPC to increase, it is necessary to continue to improve the performance and ease of use of general-purpose systems. Existing work has shown that acceleration of general-purpose MPC can translate into viable systems [16].

The crux of this and related work is whether MPC is amenable to hardware acceleration. There are two main techniques for achieving MPC: Secret Sharing consumes significantly lower bandwidth but requires low latency, and Garbled Circuits are compute-bound in any environment with sufficiently high bandwidth. Between these two options, Garbled Circuits appear more amenable to hardware acceleration, which is the subject of substantial prior research, especially with FPGAs [17]–[27]. However, the overall trend of consolidating computing into data centers changes this calculus. Evans et al. note that "[b]andwidth within a data center is inexpensive" with the caveat that there are security questions that must be given careful consideration in this context [3].

Our exploration finds a compelling argument for hardware acceleration of MPC via Secret Sharing: when deployed in a datacenter, the low latency between accelerators (e.g., within a node, bump-in-the-wire, etc.) can enable Secret Sharing MPC to make more effective use of the available bandwidth than their Garbled Circuit counterparts [28]. We propose FPGAs as a hardware platform to maximize the performance of MPC in the datacenter because they provide high bandwidth and minimize latency by integrating compute and communication.

In this paper, we explore different datacenter models, consider the steps necessary to create a viable MPC cloud service, examine the trade-off between Garbled Circuits and Secret Sharing, implement Secret Sharing in hardware, test this hardware design, and assess its scalability. We conclude by proposing directions for future work toward a complete MPC cloud service.

We summarize the contributions in this work:

- We believe we are the first group to report on Secret Sharing MPC on FPGA hardware. We demonstrate that given a set of reasonable security assumptions, MPC on FPGAs in the datacenter is viable for a real service.
- We demonstrate that Secret Sharing outperforms state-of-the-art methods for implementing MPC in the datacenter.
- Using 5.5% of FPGA fabric in a consumer cloud environment, we match the throughput of an optimized 20-core CPU implementation saturating a typical 10Gbps network connection. This result scales with available bandwidth: a single FPGA is able to saturate a 200Gbs link with a throughput of ∼26 million AES operations per second.

II. Background

A. Datacenter Model

The primary motivation of this work is to create an effective cloud datacenter that can offer MPC-as-a-service that is easy to use and has high performance. Because MPC requires multiple computing parties for security and low latency networking for performance, we consider processing hardware owned by different parties and housed within a single datacenter. This arrangement permits secure data storage across the computing

978-1-7281-9902-3/20/$31.00 ©2020 IEEE
DOI 10.1109/FPL50879.2020.00047

parties close to processing locations. Concretely, we imagine a scenario where a small number of FPGAs are connected over high-speed interconnects and have the benefit of drawing data from servers all co-located within the datacenter.

FPGA hardware acceleration has seen increasing adoption in datacenters. As described in Section II-B, FPGA hardware properties and co-location yield high throughput for MPC protocols based on Secret Sharing, which makes the most effective use of available bandwidth. Maximizing throughput is a focus for this work as this metric determines how efficiently multiple client tasks can be completed.

B. MPC Paradigms

MPC protocols support an arbitrary number N of compute parties and tolerate an arbitrary threshold T of 'bad' parties working together, where this coalition might try to break confidentiality to learn other people's data or to tamper with the integrity of the calculation. In this work, we examine a 3-party protocol tolerating 1 adversarial party who "semi-honestly" follows the protocol and only tries to break confidentiality. This matches a scenario in which a small number of FPGAs owned by different parties are co-located within a datacenter.

General-purpose MPC designs often represent the agreed-upon computation as an arithmetic or Boolean circuit, and follow the Garbled Circuit or Secret Sharing approaches. Garbled Circuits rely on one compute party generating a (large) encoded version of the entire circuit, which it then transmits to a second party who can evaluate the encoded circuit on encoded inputs in order to recover the answer. On the other hand, Secret Sharing-based MPC systems have the compute parties evaluate each gate of the circuit in parallel on their own pieces or *shares* of the data, with a small amount of network communication required for each multiplication or AND gate (none is required for addition or XOR gates).

The computation and communication overhead of MPC manifests itself differently for Garbled Circuits and Secret Sharing. Even with optimizations [29]–[33], Garbled Circuits have a small number of communication rounds but a large communication size (80-128× the size of the original data), rendering them beneficial in high-latency scenarios but detrimental when processing large datasets. Conversely, Secret Sharing approaches require a low-latency environment because they involve many rounds of communication, however they consume substantially less bandwidth per computational step.

To date, most MPC implementations are in software, and thus rely on general-purpose processing hardware and commodity networking equipment. In this scenario, Secret Sharing tends to be network latency-bound whereas Garbled Circuits are often compute-bound. Consequently, most of the prior focus in hardware acceleration has been directed toward Garbled Circuits. Our work specifically considers MPC implementations in the datacenter, where Secret Sharing systems offer higher maximum throughput and the network latency can be low enough to realize meaningful performance benefits by optimizing the computation with FPGAs.

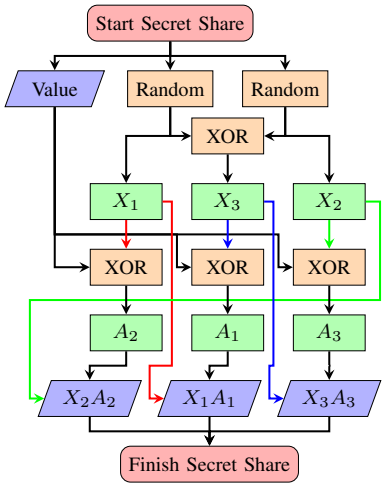


Fig. 1: Initial Secret Sharing

C. Selected MPC Protocol

Within the category of MPC protocols based on Secret Sharing, we selected a protocol by Araki et al. [1], [34] for FPGA acceleration due to its simplicity and its impressive performance in software. The Araki et al. protocol employs exactly 3 parties, and it tolerates 1 adversarial party that is presumed to follow the protocol. Also, communication occurs in a ring topology, where each party only needs to communicate with 1 of the other 2 parties.

The workflow involves 3 distinct steps. First, data holders split their data into *shares* held by the 3 compute parties. Then, the parties iteratively *compute* over these shares without revealing any secrets. Finally, the compute parties reveal their shares to the output party who can *reconstruct* the final answer.

In the *share phase*, anyone holding a secret value $v \in \{0, 1\}$ can split this secret among the 3 compute parties as follows.

- The data holder selects x_1, x_2, x_3 uniformly from $\{0, 1\}$ subject to the constraint that $x_1 \oplus x_2 \oplus x_3 = 0$.
- The data holder sends (x_i, a_i) to each compute party, where $a_i = x_{i-1} \oplus v$ is a one-time pad of the secret.

The one-time pad hides the secret value v from any single party P_i. See Figure 1 for an illustration.

In the *compute phase*, the parties work together to compute shares of the result of each XOR or AND gate in a privacy-preserving manner. It is easiest to imagine fan-in 2 operations proceeding sequentially with inputs (x_i, a_i) and (y_i, b_i), though we stress that this process is embarrassingly parallel.

- *XOR operation*: Each share of the XOR of the two values simply equals the XOR of the two input shares (see Figure 2) because a one-time pad is homomorphic under the $\oplus$ operation. The parties do not need to communicate.
- *AND operation*: Calculating shares of the result of an AND gate is more complex; it requires each compute party to compute a non-trivial amount of Boolean logic and transmit one bit of information to one other compute party. First, the parties produce *correlated random values* α_1, α_2, and α_3 that XOR to 0 but are independent of any

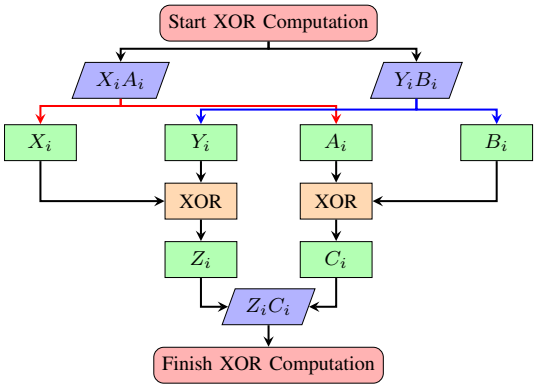


Fig. 2: Party i's contribution toward computing an XOR gate

secret values; if the parties distribute short keys before the computation, they can generate correlated randomness quickly using a PRF such as AES (Figure 3a). Second, each compute party follows the circuit shown in Figure 3b that consumes the correlated randomness generated above; Araki et al. show that the resulting values R_i have the property that $R_1 \oplus R_2 \oplus R_3$ equals the result of the AND gate. Finally, each party P_i transmits R_i to another party P_{i+1}, and then re-builds shares of the result in our desired format $\{(z_i, c_i)\}$ (Figure 3c).

In the *reconstruction* phase, we presume that the compute parties have calculated shares $\{(x'_i, a'_i)\}$ corresponding to the output value v'. Then, the parties can reconstruct v' by revealing their shares and computing $\bigoplus_{i=1}^{3} a'_i = \bigoplus_{i=1}^{3}(x'_i \oplus v) = v$.

There exist extensions of the Araki et al. protocol that permit additional parties or provide stronger security against a malicious attacker [35]–[37]. An FPGA implementation of this protocol provides an ideal starting point from which to explore the benefits of acceleration for related schemes with different features, and the possibility of dynamically switching between them to improve performance further [38], [39].

D. FPGA Models

Several FPGA deployment models are possible with varying trade-offs. Options from lowest to highest performance include: (1) co-processor, (2) bump-in-the-wire, (3) single-node cluster, (4) enclave/silo on FPGA. The enclave/silo approach where one FPGA is allocated into several regions for different parties is appealing from a performance perspective as it would enable near zero latency and near infinite bandwidth. Such an arrangement does raise many difficult questions about the isolation of the parties which go beyond the scope of the current work. Within this hierarchy, the Amazon AWS F1 instances we consider fall into the single-node cluster category. Amazon describes two different inter-board communication approaches.The F1.4xlarge, and F1.16xlarge instances should have a 400Gbps serial ring link but support is only planned in a future release. Communication between FPGAs is possible at 12Gbps over PCIe. In testing the proposed Secret Sharing block, an AWS F1.2xlarge instance was used rather than the the 4x or 16x as initial testing only required a single FPGA.

TABLE I: Araki et al. Result Analysis

Araki et al. Results		Verification		
Cores	AES/sec	Gbps/serv.	Gbps/serv. w/over.	Error
1	100103 ± 1632	0.572	0.559	2.19%
5	530408 ± 7219	2.99	2.96	0.85%
10	975237 ± 3049	5.47	5.45	0.35%
16	1242310 ± 4154	6.95	6.94	0.10%
20	1324117 ± 3721	7.38	7.40	0.28%

III. System Design and Implementation

A. Analysis of Original Implementation

Obtaining the secure operation metrics enables comparison of the FPGA design to the original results. Inspection of the original results of this Secret Sharing implementation for secure AES [35] reveals the use of the Bristol Fashion Key Expanded AES [40] requiring 5440 secure AND operations. The test described by Araki et al. is embarrassingly parallel, simultaneously running 12800 independent secure AES computations per core in each node. The total AES operations performed can be used to verify the number of bits communicated. Runtime is obtained from the AES/sec rate and total number of AES operations. Including a reasonable overhead for TCP/IP of 2.74% [41] the verified network rates closely match the reported results with less than 2.5% error (Table I). The FPGA design can be reasonably compared to this system using the 5440 AND/AES conversion.

B. FPGA Implementation

Here we cover some FPGA implementation details for the chosen MPC protocol. Two OpenCores projects were used, one for AES [42] and one for a RNG [43] for faster development. Amazon Web Services (AWS) reference designs [44] and hardware were used for testing the preliminary scheme.

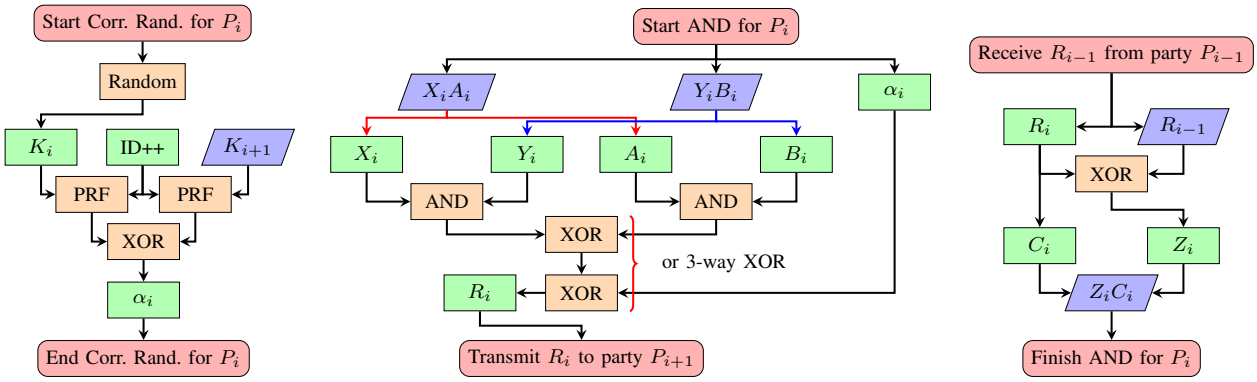
At startup, each party generates a random key for the PRF and shares it with one other party. Currently, each party uses one RNG module [43] to generate the key. The security of this RNG block was not examined; a deployed version might use a physically unclonable function (PUF) or other secure hardware RNG.

Each party contains one PRF instance that is alternately evaluated in counter mode, using each of the two keys the party holds and the same counter. Each output pair is then XORed to produce a new correlated random number. As the keys are only set at startup, the 21 clock cycle pipeline delay for the selected PRF was only experienced at initialization.

The MPC AND module itself consists of a few bitwise operations that produce the intermediate R_i values (Figure 3b). Most latency occurs in the transmission of the R_i values, since the final step (Figure 3c) cannot begin without those values.

C. Analysis of FPGA Implementation

The MPC AND hardware operation must be fed data and triggered by external logic. The first implementation uses an Arria 10 and NIOS II softcore. A NIOS custom instruction was used to load data and start the operation. The custom instruction enables simple software control of the hardware



(a) correlated random value (b) initial computation and exchange (c) final computation

Fig. 3: Party i's contribution toward computing an AND gate

MPC AND operation. The second implementation uses Amazon Web Service (AWS) FPGAs available through its Elastic Compute Cloud (Amazon EC2). Specifically, Xilinx Virtex UltraScale+ VU9P FPGAs are accessible via a virtual machine in EC2 F1 instances. Amazon includes a hardware shell for software/hardware co-design between the node CPU (Intel Xeon E5-2686 v4) and FPGA. Software to control the FPGA uses provided DMA functions and PCIe function templates. This furnishes the mechanism for loading data, controlling operations, and retrieving results.

The PCIe packets are translated through the Amazon shell and utilize multiple AXI bus configurations to send and receive data with the software system. We use the general purpose AXI bus supporting a 512 bit data packet to provide a single message containing two secret share vectors (4×128-bits) prior to starting the hardware operations. The HDL design takes each AXI bus message, parses the information, and relays data to the desired AND module.

D. Design Improvements and Additions

Based on the minimal data dependencies and flow in Figure 3, in principle a fully pipelined MPC AND module (128 Boolean MPC ANDs) can execute one AND operation per clock cycle. Such a design would saturate a 10Gbps network connection when operated at 78.13 MHz. Operating at the higher frequencies used commonly by FPGAs would require higher bandwidth. For example, at 200 MHz, a single MPC module of this type saturates a 25.6Gbps link.

IV. Results

A. Testing and Data

The MPC AND module on FPGA was assessed in regards to its resource utilization with varying levels of duplication and based on the latency of evaluation.

Initial testing targeted Intel FPGAs, such as the Arria 10, with the the NIOS II softcore executing test software to load data and trigger the MPC AND hardware. The limitations of the Avalon Bus width and the latency a simple softcore design imposes encouraged us to consider other options. Note, there are still circumstances where a softcore is viable such as in a design using local storage and other techniques to overcome the limits of loading individual data and running an operation in sequence. Regardless, with the single AND design synthesized in Quartus, targeting an Arria 10 (10AX115S2F45I1SG) provided initial insights. The most constrained resource for a single AND was the 704.5Kb of M20K block memory consumed post-synthesis, making is possible to estimate the utilization to be ∼1.32% based on the total 53.260Mb of M20K available on the Arria 10. With perfect utilization this would permit ∼76 instances. More realistically perhaps 70% of the fabric might be used allowing for ∼54 instances of the MPC AND. As implemented, the MPC AND requires 6 clock cycles between operation which means that 48 MPC AND instances makes it possible for 8 operations to occur each cycle. Even with only 8 AND operations per cycle at 200Mhz the design is able to saturate ∼205Gbps, far more than the 10Gbps link in the original paper.

With these synthesis results from Quartus but seeking to avoid the limitations of the NIOS II and to find a more fitting cloud target we looked to Amazon Web Services (AWS). While the Amazon AWS F1 instances do not currently offer the promised high-speed serial ring [44], targeting the boards available provides hardware utilization insights, and leaves the possibility of more easily using higher speed communication when support materializes. Furthermore, the PCIe option, while lower performance, remains available for future tests.

For a single 1-party block post-routing, the Virtex Ultrascale+ utilizes ∼3.20% of its resources. In order to verify that all three parties functioned together properly, a 3×party design with 1 AND per party was made to target a single FPGA. This made it possible to pass data and trigger operations without having to immediately spend the development time to bring-up the AXI4-Stream between FPGAs over PCIe. Furthermore, since each 1 party block contains more control logic than just a single MPC AND it serves as an adequate conservative estimate of how many AND blocks might fit on one FPGA. Continuing this line of testing led to duplicating groups of the 3×party block the results of which are summarized in Table II. The number of AND modules are used to determine the number of bits per

TABLE II: AWS Implementation Result Analysis

AND Cores	Bits	Gbps	AES (millions op.)/sec
1	128	2.67	0.490
3	384	8.00	1.47
12	1536	32.0	5.89
24	3072	64.0	11.8
48	6144	128	23.5
60	7680	160	29.4

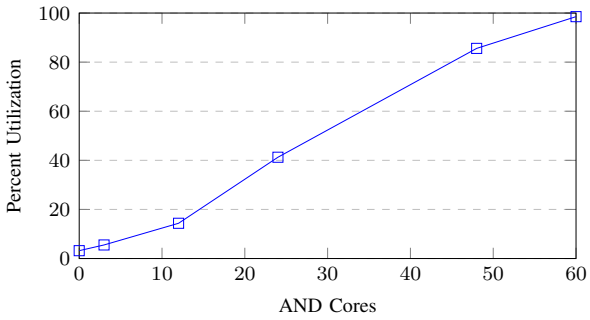


Fig. 4: AWS FPGA fabric total utilization

clock cycle that are processed. The F1 instance was clocked at 125Mhz producing the listed Gbps results. The equivalent number of AES/sec was determined by dividing by the 5440 AND/AES used in Araki et al. A plot of the FPGA utilization in Figure 4 shows a fairly linear relationship between number of AND modules and utilization.

B. Analysis

The tests performed with different quantities of MPC AND blocks on Amazon AWS provide sufficient data points to establish that Secret Sharing MPC can be competitive when implemented on FPGA hardware in the datacenter. The fabric utilization in the design scales relatively linearly with increases in the quantity of AND cores.

The original work utilized a software implementation of the protocol executed on general purpose processors [1], [34]. Specifically, each party used varying numbers of cores from one or two Xeon processors. The authors were able to nearly saturate their 10Gbps link (7.38Gbps) between parties when using all cores in each node. This required about 50% of the processor's time. While the authors were limited by multiple cores causing queuing congestion at the Network Interface Card (NIC) the use of a CPU appears to have more limited scaling potential. Using the reported number of AES/sec and network communication for 1 core, scaling from 73.3% CPU usage to 100% would appear to show 1 core being capable of at most $\sim$130 thousand AES/sec, saturating a $\sim$0.780Gbps connection. Multiplying for 20 cores that would amount to a peak of $\sim$2.7 million AES/sec and $\sim$15.6Gbps.

In comparison, the FPGA AND block we tested for performing Secret Sharing only requires 3 AND cores per party to exceed the 7.38Gbps reached with 20 CPU cores, instead being capable of 8.00Gbps. This uses $\sim$5% of the fabric available on the FPGA targeted, a $10\times$ improvement vs the CPU utilization.

Attempting to fully employ the available fabric it is possible to implement 60 AND cores which would permit saturation of 160Gbps links while performing 29.4 million AES/sec.

These results demonstrate preferable scaling properties supporting the selection of FPGAs for acceleration. Based on the results, targeting the anticipated 200Gbps links in the Amazon F1 would require less than 25% fabric utilization to reach full saturation if just the pipeline improvement is made. With a frequency improvement, even less fabric would be required. The remaining available fabric is beneficial as it allows for work distribution and additional secure computations.

V. Related Work

There exists earlier research exploring hardware accelerated MPC, but the efforts have focused on Garbled Circuits rather than Secret Sharing. The hardware considered has included GPUs [19], [45]–[47]; most efforts, however, employ FPGAs. The earliest of these efforts dates to 2010 [17], [18], with more work recently [23]–[25], and some considering Amazon AWS [25], [27]. Other work explored garbling entire processors [20], [22] and specialized problem acceleration [21].

The studies from researchers at Northeastern University are most relevant here. Their overlay architecture [24] and identification of datacenters as an ideal place to perform such computations [26] matches our decisions. With respect to overlays, they implement blocks to accelerate the garbling of AND and XOR operations and that do not require the FPGA image to be recreated and programmed. Instead data is passed to these processing elements which is much more efficient. We follow a similar scheme with Secret Sharing.

VI. Conclusion

In this paper, we describe one approach to implementing the underlying MPC AND operation described by Araki et al. [35] in hardware. We demonstrate the viability of Secret Sharing MPC in a low latency environment and test the design on an FPGA in the cloud highlighting greater potential scalability of the design compared to alternatives. With these insights, we plan to pursue improvements to this design to increase the performance further and to implement the higher level controls necessary to use our Secret Sharing building block in a complete MPC cloud service.

Some specific future work includes HDL implementation optimizations while maintaining the same scheme. FPGA to FPGA communication will be evaluated. Additional research directions include different viable MPC security models and hardware security considerations on FPGAs.

VII. Acknowledgements

Supported by Red Hat and NSF Grants 1718135, 1739000, and 1931714. DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

REFERENCES

[1] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 805–817. [Online]. Available: https://doi.org/10.1145/2976749.2978331

[2] R. Wyden, "Student right to know before you go act of 2019," https://www.congress.gov/bill/116th-congress/senate-bill/681/all-info, 2019.

[3] D. Evans, V. Kolesnikov, and M. Rosulek, *A Pragmatic Introduction to Secure Multi-Party Computation*. NOW Publishers, 2018.

[4] A. C. Yao, "Protocols for Secure Computations." *Annual Symposium on Foundations of Computer Science - Proceedings*, pp. 160–164, 1982.

[5] A. C. C. Yao, "How To Generate and Exchange Secrets." *Annual Symposium on Foundations of Computer Science (Proceedings)*, no. 1, pp. 162–167, 1986.

[6] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[7] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From keys to databases - real-world applications of secure multi-party computation," *Comput. J.*, vol. 61, no. 12, pp. 1749–1771, 2018.

[8] T. Giannopoulos and D. Mouris, "Privacy preserving medical data analytics using secure multi party computation. an end-to-end use case." Ph.D. dissertation, National and Kapodistrian University of Athens, 09 2018.

[9] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, and R. Talviste, "Students and taxes: a privacy-preserving social study using secure computation," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1159, 2015.

[10] J. Feigenbaum, B. Pinkas, R. Ryger, and F. Saint-Jean, "Secure computation of surveys," in *EU Workshop on Secure Multiparty Protocols*, 2004, pp. 2–14. [Online]. Available: https://www.cs.yale.edu/homes/jf/SMP2004.pdf

[11] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft, "Secure multiparty computation goes live," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 5628. Springer, 2009, pp. 325–343.

[12] I. Damgård, K. Damgård, K. Nielsen, P. S. Nordholt, and T. Toft, "Confidential benchmarking based on multiparty computation," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 9603. Springer, 2016, pp. 169–187.

[13] A. Abidin, A. Aly, S. Cleemput, and M. A. Mustafa, "An mpc-based privacy-preserving protocol for a local electricity trading market," in *CANS*, ser. Lecture Notes in Computer Science, vol. 10052, 2016, pp. 615–625.

[14] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.

[15] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, M. Raykova, S. Saxena, K. Seth, D. Shanahan, and M. Yung, "On deploying secure computing commercially: Private intersection-sum protocols and their business applications," *IACR Cryptology ePrint Archive*, vol. 2019, p. 723, 2019.

[16] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS*. The Internet Society, 2012.

[17] K. Järvinen, V. Kolesnikov, A. R. Sadeghi, and T. Schneider, "Embedded SFE: Offloading server and network using hardware tokens," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6052 LNCS, pp. 207–221, 2010.

[18] ——, "Garbled circuits for leakage-resilience: Hardware implementation and evaluation of one-time programs," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6225 LNCS, pp. 383–397, 2010.

[19] T. K. Frederiksen, T. P. Jakobsen, and J. B. Nielsen, "Faster maliciously secure two-party computation using the GPU," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8642, no. grant 61061130540, pp. 358–379, 2014.

[20] E. M. Songhori, S. Zeitouni, G. Dessouky, T. Schneider, A. R. Sadeghi, and F. Koushanfar, "GarbledCPU: A MIPS processor for secure computation in hardware," *Proceedings - Design Automation Conference*, vol. 05-09-June, 2016.

[21] S. U. Hussain, B. D. Rouhani, M. Ghasemzadeh, and F. Koushanfar, "MAXelerator: FPGA Accelerator for Privacy Preserving Multiply-Accumulate (MAC) on Cloud Servers," *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

[22] E. M. Songhori, M. S. Riazi, S. U. Hussain, A. R. Sadeghi, and F. Koushanfar, "ARM2GC: Succinct garbled processor for secure computation," *Proceedings - Design Automation Conference*, 2019.

[23] S. U. Hussain and F. Koushanfar, "FASE: FPGA acceleration of secure function evaluation," *Proceedings - 27th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2019*, pp. 280–288, 2019.

[24] X. Fang, S. Ioannidis, and M. Leeser, "Secure function evaluation using an FPGA overlay architecture," *FPGA 2017 - Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 257–266, 2017.

[25] ——, "SIFO: Secure computational infrastructure using FPGA overlays," *International Journal of Reconfigurable Computing*, vol. 2019, 2019.

[26] K. Huang, M. Gungor, X. Fang, S. Ioannidis, and M. Leeser, "Garbled circuits in the cloud using FPGA enabled nodes," *2019 IEEE High Performance Extreme Computing Conference, HPEC 2019*, pp. 1–6, 2019.

[27] M. Leeser, M. Gungor, K. Huang, and S. Ioannidis, "Accelerating large garbled circuits on an FPGA-enabled cloud," *Proceedings of H2RC 2019: 5th International Workshop on Heterogeneous High-Performance Reconfigurable Computing - Held in conjunction with SC 2019: The International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 19–25, 2019.

[28] T. Schneider and M. Zohner, "GMW vs. Yao? Efficient secure two-party computation with low depth circuits," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7859 LNCS, pp. 275–292, 2013.

[29] D. H. U. Beaver, S. M. Micali, and P. M. Rogaway, "The Round Complexity of Secure Protocols," *ACM*, 1990.

[30] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," *ACM International Conference Proceeding Series*, pp. 129–139, 1999.

[31] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5126 LNCS, no. PART 2, pp. 486–498, 2008.

[32] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole - reducing data transfer in garbled circuits using half gates," in *EUROCRYPT (2)*, ser. Lecture Notes in Computer Science, vol. 9057. Springer, 2015, pp. 220–250.

[33] S. Yakoubov, "A Gentle Introduction to Yao's Garbled Circuits," 2017, http://web.mit.edu/sonka89/www/papers/2017ygc.pdf.

[34] T. Araki, A. Barak, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "Demo: High-throughput secure three-party computation of kerberos ticket generation," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1841–1843. [Online]. Available: https://doi.org/10.1145/2976749.2989035

[35] T. Araki, A. Barak, J. Furukawa, T. Lichter, Y. Lindell, A. Nof, K. Ohara, A. Watzman, and O. Weinstein, "Optimized Honest-Majority MPC for Malicious Adversaries - Breaking the 1 Billion-Gate per Second Barrier," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 843–862, 2017.

[36] Furukawa, Jun and Lindell, Yehuda and Nof, Ariel and Weinstein, Or, "High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority," in *Advances in Cryptology – EUROCRYPT 2017*, Coron, Jean-Sébastien and Nielsen, Jesper Buus, Ed. Cham: Springer International Publishing, 2017, pp. 225–255.

[37] J. Furukawa and Y. Lindell, "Two-thirds honest-majority MPC for malicious adversaries at almost the cost of semi-honest," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1557–1571, 2019.

[38] D. Demmler, T. Schneider, and M. Zohner, "ABY - A framework for efficient mixed-protocol secure two-party computation," in *NDSS*. The Internet Society, 2015.

[39] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *ACM Conference on Computer and Communications Security*. ACM, 2018, pp. 35–52.
[40] "'bristol fashion' mpc circuits," https://homes.esat.kuleuven.be/~nsmart/MPC/.
[41] S. Iveson, "Tcp over ip bandwidth overhead," Sep 2013. [Online]. Available: https://packetpushers.net/tcp-over-ip-bandwidth-overhead/
[42] H. Hsing, "tiny_aes," https://opencores.org/projects/tiny_aes, 2012. [Online]. Available: https://opencores.org/ocsvn/tiny_aes/tiny_aes/trunk
[43] J. Castillo, "systemc_rng," https://opencores.org/projects/systemc_rng, 2004. [Online]. Available: https://opencores.org/ocsvn/systemc_rng/systemc_rng/trunk
[44] A. W. Services, "aws_fpga," https://github.com/aws/aws-fpga, 2016. [Online]. Available: https://github.com/aws/aws-fpga.git
[45] S. Pu, P. Duan, and J.-C. Liu, "Fastplay-A Parallelization Model and Implementation of SMC on CUDA based GPU Cluster Architecture," *IACR Cryptology ePrint Archive*, vol. 2011, p. 97, 2011.
[46] S. Pu and J. Liu, "Computing Privacy-Preserving Edit Distance and Smith-Waterman Problems on the GPU Architecture." *IACR Cryptology ePrint Archive*, 2013. [Online]. Available: http://eprint.iacr.org/2013/204.pdf
[47] N. Husted, S. Myers, A. Shelat, and P. Grubbs, "GPU and CPU parallelization of honest-but-curious secure two-party computation," *ACM International Conference Proceeding Series*, pp. 169–178, 2013.