2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED) | 978-1-6654-8526-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/SEED55351.2022.00015

Establishing Cooperative Computation with Hardware Embassies

Alvin Oliver Glova
UC Santa Barbara
Santa Barbara, CA
aomglova@ece.ucsb.edu

Yukai Yang
University of Pennsylvania
Philadelphia, PA
yukaiy@seas.upenn.edu

Yiyao Wan
UC San Diego
San Diego, CA
y4wan@ucsd.edu

Zhizhou Zhang
UC Santa Barbara
Santa Barbara, CA
zhizhouzhang@ucsb.edu

George Michelogiannakis
Lawrence Berkeley National Laboratory
Berkeley, CA
mihelog@lbl.gov

Jonathan Balkind
UC Santa Barbara
Santa Barbara, CA
jbalkind@ucsb.edu

Timothy Sherwood
UC Santa Barbara
Santa Barbara, CA
sherwood@cs.ucsb.edu

***Abstract*—While recent cryptographic techniques enable cooperative multi-party client-server computations under mutual distrust, they also introduce an efficiency tradeoff. Hosting all of the computation from the different parties involved on one set of servers requires everyone to agree on which servers are trustworthy. On the other hand, keeping the computations truly distributed introduces significant delays because of the inherently latency-sensitive nature of the protocols involved. In this paper, we explore the architectural impact of a possible middle path to this problem: resource-poor but physically secure devices interacting with significant (but not mutually trusted) compute and storage resources. The idea is that a small and well-protected "Embassy" can serve as a plot of sovereign soil in an otherwise untrusted environment. Building on techniques from multiparty computation (MPC) we show how such an architecture, even when extremely limited in size, can leverage local network capabilities and asymmetries in cryptographic operations to perform more efficient interactive secure computations. Even with a client-side device 5× slower, we show that common MPC applications can still be accelerated by 3× on average. Moreover, we explore the potential for architectural changes to further support multi-party evaluation through the addition of dedicated evaluator hardware further improving performance 1.52×.**

I. Introduction

Through advanced cryptographic techniques, it is now possible to perform shared computations without ever fully sharing the data. For example, a class of cryptographic techniques referred to as multiparty computation (MPC) establishes secure computation *protocols* between multiple non-colluding parties that allow for functions to be iteratively computed on private inputs without revealing anything beyond the result to either party. As long as we trust those parties do not share out-of-band information with one another, these techniques allow for a mutual computation to be performed (for example a query to a database) without either side learning what the other is doing (such as keeping the query secret from the database and visa versa). Unfortunately, these protocols usually require both parties to be *active participants* in the computation to some degree. Because the computations are typically arranged as long and unbroken chains of cryptographic operations, involving multiple parties typically means a lot of *waiting around* for the other side to finish up their work and pass it back to you.

One way to deal with this is to host multiple parties on a trusted third-party platform. Co-locating the computation minimizes the time wasted transmitting parts of the computation back and forth between all parties. Of course, if you had a fully trusted third party they could just do the computation for all parties involved – no need for cryptography! However, when we "trust a server", we are trusting not only the computational and storage resources it hosts but also the **physical** and **legal** environments under which it operates. These aspects are hard to attest to remotely and only compound when multiple nation-states are involved. In reality, these cryptographic approaches are typically the most useful when the data involved is sensitive enough that we prefer to trust no one with all of the data. So, what can we do?

That introduces the new problem of where to find computational resources to host multiple parties that both parties will trust. The last decade has seen significant advances in making trusting third-party remote hardware a more reasonable choice. For example, Flickr [44] introduced a clever scheme for remote attestation built on the Trusted Platform Module (TPM) architecture which allowed the loaded system binary to be non-bypassably fingerprinted. More current approaches build on top of the capabilities of trusted execution environments (TEE) such as Intel SGX [20] to create similar "bubbles" of trust. While these and other approaches provide significant protections, the threat models one can address with ISA-level changes alone are constrained and the limits of sharing resources with an untrusted host opens up many potential side-channel attacks.

The question we attempt to answer in this paper is if and when it is *possible* to use a small island of physical security located in an otherwise very untrusted environment, to enable a broader set of physically secure computation. Moreover, we explore new architectures and machine organizations that enable such an approach to operate with higher efficiency and better performance as compared to remote computation.

Specifically, we propose an asymmetric approach to multi-party architecture with the co-location of a small physically-hardened compute element (under the control of one party) with a much larger and robust server-class system (under the control of the other). The hardened device can be physically smaller with fewer compute resources. Due in part to its small size, the small compute element can be hardened against even incredibly advanced attacks to a high degree. The small device can even be physically shipped between the guest, host, and back again as needed for initialization and decommissioning. At a high level, one can think of this idea as setting up an "*Embassy*" that serves as an island of sovereign soil in a foreign land. Just like traditional embassies, this arrangement allows for higher bandwidth and lower latency interaction facilitating joint activities even under mutual distrust. The code that lives on the device can serve to orchestrate and even *participate in* trustworthy computations in the server on behalf of the guest. Physically shipping the device adds significant setup overhead. However, there are many

978-1-6654-8526-5/22/$31.00 ©2022 IEEE
DOI 10.1109/SEED55351.2022.00015

privacy-focused applications where this one-time cost is tolerable. For instance, in hospitals or research centers, new and sensitive data are being generated constantly. The more this device is used, the more amortized the setup cost becomes.

As a first demonstration of the concept, we identify a class of cryptographic computing approaches that are *inherently asymmetric* in their needs. Building on techniques from homomorphic encryption and multiparty computation, we show how our proposed system can leverage the high bandwidth and low latency network fabric available locally to perform more efficient interactive secure computations, even when the computational abilities of these physically-smaller devices are severely limited. Specifically, we examine two important privacy-preserving applications, secure neural network inference based on Yao's Garbled Circuit (GC) [70] and private DNA matching based on Goldreich-Micali-Wigderson (GMW) [27]. In these scenarios, the Embassy (our proposed device) acts as a trusted (non-colluding) proxy for the client to perform Multi-Party Computation (MPC) with a co-located untrusted server. We show that the improvements in connectivity possible from using only systems connected by local networks more than compensate for the smaller compute resources available to this new class of device, *and* that with some simple architectural changes this gap can be extended even further. We summarize our contributions as follows:

- We propose "Hardware Embassies", a new class of devices that enable more efficient MPC by providing untrusted server co-located tamper-proof trusted hardware.
- We show how important cryptographic methods can be mapped to Hardware Embassies and, for the first time, quantitatively explore the ways in which we can take advantage of the network performance and asymmetric compute requirements of these protocols.
- Building on our experience with the above, we propose and evaluate a microarchitecture specialized in the cryptographic operations at the heart of common MPC computations.
- We show experimentally, through a mix of in-datacenter network experimentation, detailed simulation, and Verilog design, that the resulting system realizes a 4.56× improvement over more distributed computation.

II. Supporting MPC

Multi-Party Computation (MPC) is a class of cryptographic techniques that allow for the evaluation of functions without any of the participating parties learning about the inputs used in the computation [40]. The most advanced techniques support any computation expressible as a Boolean circuit, everything from neural network evaluations to bioinformatics applications, without sharing the underlying data.

A common form of MPC in practice is two-party computation (2PC) [30], which can be used as a way to securely outsource private computations to untrusted cloud machines. Yao's Garbled Circuit (GC) and Goldreich-Micali-Wigderson (GMW) are examples of 2PC protocols which have been used for applications such as privacy-preserving machine learning [58], secure genomic computations [36], and secure data search [23]. Recent algorithmic improvements to 2PC protocols, especially the transition from public-key cryptography to symmetric cryptography, have reduced the computational overhead by more than an order of magnitude but communication bottlenecks are much harder to overcome. For a single inference operation in a simple MNIST-based neural network, a strict GC approach would require a network transfer volume of 791MB [59]. While we will discuss some algorithmic ways others have found to help mitigate this problem, it remains a serious issue.

Also, the literature on MPC is largely dominated by work that optimistically assumes direct high speed and low latency connection between communicating parties. The high communication cost of GC becomes even more burdensome for the common case where a client and a server are located in different regions and therefore are using a WAN connection. There are different possible LAN and WAN assumptions one might make, but typically settings of WAN have 430× longer latency and 113× smaller bandwidth than the reported LAN configuration for AWS [46].

We propose the use of a low-resource device under the direct control of an entity cooperating with the co-located server that takes advantage of the high-speed LAN performance. This is made possible by *physically co-locating this device* with the cooperating agents while taking advantage of the inherent asymmetry of client and server computational requirements for most common cryptographic techniques. For example, as shown in Table I, the evaluation phase in GC (typically performed at the client-side) has 2× smaller compute requirements [72] than the garbling phase (typically performed on the server). This difference in compute load between the client and server becomes more asymmetric for the hybrid protocols we consider in this work.

Unfortunately, co-location inherently creates a trade off between performance and security as one party now has physical access to both sides of the computation. In situations with mutual trust between all parties, this does not pose a security challenge, however, under these assumptions it is often unnecessary to utilize a co-located device. When the embassy device is under the physical control of an untrusted entity then that entity could potentially break the non-collusion assumptions that MPC and other cryptographic protocols rely on. It becomes necessary to ensure the embassy is secure against physical attacks.

While physical security is not the focus of this work, NIST provides a standard for the security of cryptographic hardware in untrusted environments called FIPS 140. The latest versions, 140-2 [47] and 140-3 [48], categorize hardware into four categories. For FIPS 140-3 levels 1 and 2 have no physical security requirements, and so such devices would be unfit for an embassy device. Levels 3 and 4 require strong enclosures with tamper detection that causes either an automatic zeroisation or a module shutdown. While both level 3 and 4 devices are sufficient to implement Embassy, these tamper detection techniques introduce overhead proportional to the original chip area. Also, there is rarely a single technique that is able to provide catch-all tamper detection [53]. For instance, silicon light sensors have been used to detect active optical attacks [52], but cannot detect other attacks. Due to this, most FIPS 140 level 3 or 4 devices tend to be small, such as USB drives, security cards, and hardware security modules. While these devices are too small to support the computation necessary for an Embassy, the techniques used can be expanded to cover a larger device.

Using a small computing device for the Embassy gives us the following advantages: (i) better defence against physical tampering because of a smaller attack surface; (ii) better protection if the server gets compromised, given the Embassy has a different hardware configuration and security guarantees (an attack on the server does not automatically compromise the Embassy); and (iii) this setup relaxes the need for the client to be online because it allows precomputations that can reduce ad-hoc runtime.

In this paper, we study two different applications to demonstrate the practicality of our solution. The first application is secure NN inference

Properties	Garbled Circuit	GMW
XOR Gate	free	free
AND Gate - Setup computation - Setup communication [bits] - Online computation - Online communication [bits]	 - - C: 2×AES; S: 4×AES C from S: 2×κ	 Client/Server: 6×AES C to S and C from S: 2×κ negligible C to S and C from S: 4
wire storage [bits]	κ	1

TABLE I: Comparison between Garbled Circuit and GMW. The security level κ is usually fixed at 128 bits. XOR gates require no communications for both protocols and can be computed locally. For each AND gate, the garbled circuit computes more AES on the server-side (2×) while work is evenly split in GMW. In general, GMW requires more AES computations while GC consumes a greater memory footprint. Both protocols have the same communication overhead.

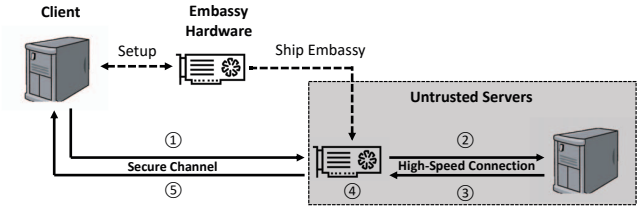

Fig. 1: Protocol Flow. A trust setup phase is first performed with a client machine before being sent to the co-located server. The client machine sends inputs and receives results from the Embassy through a secure channel. The actual secure computation happens inside the co-located server between the Embassy and an untrusted server.

using a hybrid protocol (HE + GC) [37]. The second application is secure DNA matching using a private set intersection with the GMW protocol [23]. While we use these two specific MPC approaches to evaluate this approach, there is nothing application-specific about the architecture we propose.

For the two applications considered in this work, we follow the threat model of [23], [37]. In secure neural network inference, we assume that the network model is available as plain text in the server, similar to past work [24], [37], [42]. We assume that the cryptographic protocols which we make use of in this work are correct and that the adversary is computationally-bound, i.e. brute-force attacks are infeasible. We assume that the Embassy is resistant to physical tampering and that any attempt to pry open the device results in irretrievably corrupting the data in the device as per FIPS 140-3 level 3 and 4 devices.

III. Hardware Embassy Approach

The general protocol governing the use of an Embassy consists of three main phases.

I. Key Setup. Unique among other approaches, a client can begin with direct physical control of the device to be embedded in the co-located server. The client generates a random symmetric encryption key which is then stored in the Embassy. This key can be used to securely communicate back with the client from the co-located server.

II. Program Select and Compute. After the Embassy is installed in the co-located server, the client can send a request to the device consisting of an input and a program for the computation. This is done through a secure channel using the key that was generated by the client. To initiate the compute operation, the Embassy sends a request to an untrusted server in the system. For example, to perform GC, the untrusted server sends the garbled tables of the program to the Embassy and performs OT for input wires. Note that the garbled tables can be precomputed offline for certain programs. The Embassy can then evaluate the garbled tables and obtain the result of the computation.

III. Result Retrieval. As results are generated, the Embassy can send them to the client. Alternatively, the client can batch a request of computations and query the results stored in the Embassy. Results are sent back to the client using a secure channel as the data leaves the co-located server.

A. Embassy Design

In designing an Embassy, we consider a spectrum of specifications with the highest performance being a server-class machine. Given the highly-advanced threat model we are considering, it would be advantageous to use a device that has small enough dimensions to be physically protected using the most aggressive tamper-resistant methods known [35]. At the lowest end that may be a simple USB-sized package similar to those used for edge neural network acceleration [9], [12]. However, we are interested in using a standalone device that does not need a host, for security and performance reasons. The closest commercial device on the market is Intel's compute stick which was first released in 2015 [11]. These devices have USB ports that can be used to connect to either a USB-based NIC or a switch that incorporates USB connectivity, however more complicated wire connectivity and better networking capabilities could be possible.

One of the main challenges in using such small devices as an Embassy is the lower performance they provide compared to server-class machines. However, as we will show later, with some creativity this level of device can still provide sufficient compute for secure computation due to the fact that most of these protocols have an inherent compute asymmetry – most of the compute-intensive actions can be carried out on a powerful but untrusted server.

B. Co-locating with Untrusted Servers

Here we present one possible design for a co-located server that supports Embassy. We use the concept of a *disaggregated co-located server* that allows different computing devices or accelerators to be separated and individually addressed instead of relying on host machines [29]. A sample co-located server configuration with Embassy is shown in Figure 2. This design presents advantages in terms of cost, performance, and security. An Embassy without a host machine yields significant cost savings and lower maintenance costs. In terms of performance, it has fewer network and software layers to traverse since it does not communicate through a host machine. Using a compute-stick class device also ensures we are not over-provisioning for workloads that work on these devices. As for security, the potential attack surface is reduced since data does not need to pass through host machines, thus making side-channel attacks harder.

Dedicated and physically-separated machines for clients might be less cost-efficient than co-located servers. Nevertheless, recent attacks on virtualized environments [39], [41], [64] have made it clear that it is sometimes better to use separate machines if security is important since they can be more easily isolated in physically secure spaces. We also draw inspiration from the rise of baremetal servers which allow companies to have physically separate machines in the co-located server instead of using virtualized environments.

While we have shown that there are advantages in introducing third-party hardware such as Embassy into co-located servers, an understandable concern from server operators is if the device itself is malicious. While this has been an ongoing trend [18], here we discuss further potential safeguards to protect servers from malicious

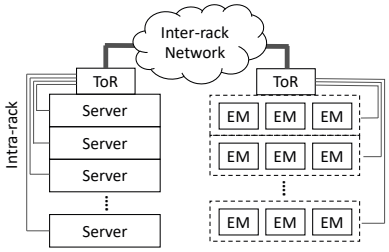

Fig. 2: Model of Embassies with Co-located Servers. Embassies are host-less network-connected computing devices. Each one can connect to a server on other racks through Top of Rack (ToR) switches or through other Embassies.

Embassies. One protection is to add a firewall using a switch exploiting software-defined networking to provide software-controlled protection of the broader co-located servers from potentially malicious traffic produced by errant Embassies. Another safeguard would be for the provider to release an open-source reference design for the Embassy with auditing by developers and the potential to perform attestation using Physical Unclonable Function (PUF) or Zero Knowledge Proof (ZKP) so that the provider can confirm that the Embassy can be trusted. The OpenTitan project shows a potential proof of concept in the related space of providing an open-source silicon root-of trust [43].

IV. Applications

While Embassy can be used for a wide range of applications, in this paper we investigate two representative applications commonly outsourced to third-party co-located servers that highly demand privacy guarantees: neural network inference and DNA matching. Below we describe how we can adapt these applications to leverage Embassy.

A. Embassies in Secure Computation

Embassy for Secure NN Inference: The protocol followed in this work is broadly similar to the hybrid protocol used in Gazelle [37], as shown in Figure 3. After securely receiving the input data from the client, Embassy encrypts the input data sent by the client (e.g., image) using packed additively HE (PAHE). The linear layers (convolution and fully-connected) are then processed using PAHE operations. Non-linear layers such as ReLU and MaxPool are performed using a garbled circuit. The ReLU circuit that is evaluated is shown in Figure 4, where s_x and s_y are shares from the server and c_x comes from the Embassy or client, and p is the prime parameter selected in PAHE. Conversion from PAHE to GC is done using secret sharing (adding a blinding random number). These steps are repeated in the series of linear/non-linear layers of the neural network until the final result (prediction) is obtained which is still in an encrypted form. This is sent back to the Embassy where it is decrypted and sent back to the client in a secure channel or stored for a later query by the client device.

Unlike in Gazelle, we assign the untrusted server as the garbler and the Embassy as the evaluator. In this way, we are taking advantage of the workload asymmetry that exists between the evaluator (less work) and garbler (more work). Note that compared to using pure GC, this hybrid protocol results in reduced online execution time and communication cost. This means that for cases where we are interested in similar runtimes, there is a larger margin for the performance degradation range of the Embassy.

Embassy for Private Set Intersection: By improving an application using a generic circuit protocol alone (GMW), we demonstrate that

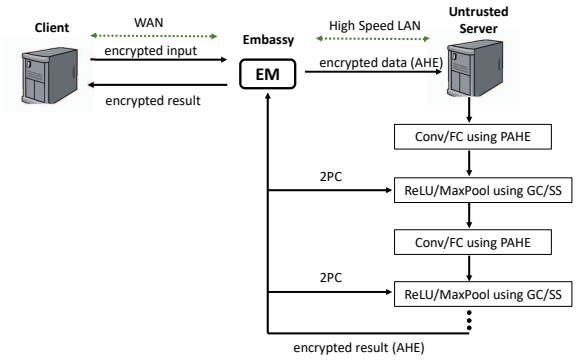

Fig. 3: Hybrid secure neural network inference flow using Embassy. This flow is adapted from Gazelle [37] which combines Additive HE and Garbled Circuits to evaluate linear and non-linear parts of the neural network, respectively

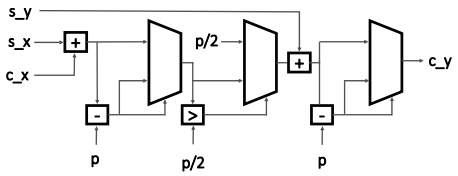

Fig. 4: ReLU Gadget Unit to be evaluated in the GC phase of the Hybrid Secure Neural Network [37]

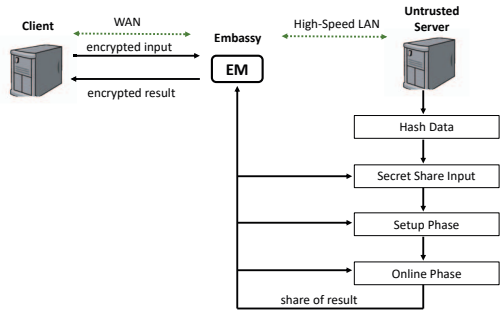

Fig. 5: PSI-GMW Flow using Embassy. This is adpated from [50] that uses GMW to perform pairwise comparison for each bucket of the hash tables.

similar results can be obtained in the most secure 2PC applications as GMW can be rapidly adapted to a different program by constructing a new corresponding circuit. We thus adapt the PSI pairwise-comparison-circuit using GMW [50] to Embassy. Dessouky et al. proposed a look-up table circuit protocol [23] that outperforms GMW in PSI, but since the protocol reduces the communication overhead at the cost of increased computation, it performs poorly in the LAN network and is thus not considered. For comparison, we evaluate a dedicated PSI protocol using Oblivious Transfer [51], one of the fastest PSI protocols in the literature, on the Embassy. For simplicity, we name the two protocols PSI-GMW and PSI-OT.

PSI-GMW computes the intersection between two sets by mapping elements from both parties into hash tables and evaluates a pairwise comparison circuit between each bucket of the hash tables, as shown in Figure 5. The complexity of PSI-GMW scales linearly with the product of the entry bit width and the set size [50]. The process begins with the client hashing its private data locally (e.g. genomic

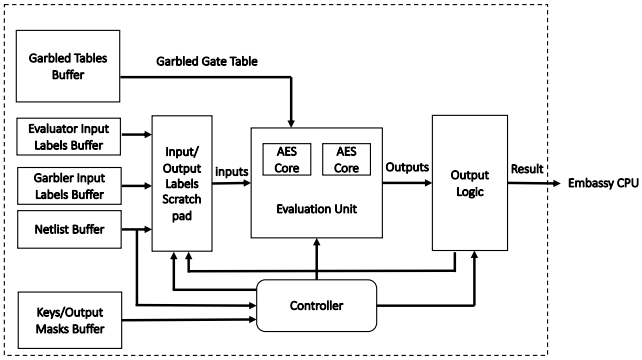

Fig. 6: Embassy GC Evaluator Architecture. All the inputs stored in the buffers are sent directly from the Garbler except for the Evaluator input labels which are obtained via oblivious transfer (OT) protocol.

data in a VCF file) and sending that data to the Embassy in the co-located server. We adopt the same hashing technique in [50] that maps data to fewer bits, which reduces the one-time communication overhead over the WAN network and the storage requirement in the Embassy. The setup phase of GMW is dominated by AES operations in OT. To balance the computation workload, the server and the client switch roles in 1-out-of-2 OT after computing multiplication triples for half of the AND gates [63]. Because of the resource constraints of the Embassy, we remove this optimization and make Embassy always play the receiver in OT. Removing role switching also reduces computation intensity by reducing two base-OT computations to one. We still keep the two-thread implementation in [50]. Note that it is possible to improve performance by using more threads in the setup phase to further take advantage of the fast High-Speed LAN.

The Embassy enables the client to stay online only during the transfer of the input and output data. The multiplication triples can be generated as long as the size of the circuit is known. The actual program (e.g. PSI) does not have to be known in advance. Because the Embassy can always stay online, the Embassy can precompute a certain number of multiplication triples (say 2^{30}) with the server when it is idle. After the client makes a request to execute a program, it needs to query a multiplication triple for each AND gate in the circuit and uses them directly in the online phase. As a result, the ad-hoc runtime can be reduced by more than 99% for a set size of 100K. After the set intersection is computed, the intersection results will be stored in the Embassy in a bitmap format, which can be later queried by the remote client.

For the PSI-OT flow, the same hashing process is still required. Instead of evaluating a circuit, both parties perform a random 1-out-of-N OT for each bucket of the hash tables. As a result, both parties obtain a randomly generated mask for all of their own table entries. Then the server sends a randomly permuted set of all of its masks to the Embassy. The Embassy finally computes the intersection by comparing the masks, and the results will be stored in the same bitmap format. The complexity of PSI-OT is independent of the entry bit width and scales linearly with the set size [51]. However, 1-out-of-N OT requires more base-OT computations, which can easily become compute-bound in a fast network.

B. The "Ambassador" Garbled Circuit Evaluator Accelerator

While the algorithmic mappings described above take advantage of computational asymmetries, if you make the hardware embassy resource-constrained *enough*, eventually it begins limiting performance again. However, after examining the way these devices are exercised by real code, we observed that *much* of the work is well-structured cryptographic operations amenable to hardware acceleration. We propose that a cryptographic co-processor designed to sit alongside the main Embassy CPU and perform common MPC operations can be used to further improve performance or, more usefully, provide even further computational asymmetry allowing even smaller and more resource-limited devices to be useful in this context. Figure 6 details the high level architecture of the Ambassador Garbled Circuit Evaluator module.

The main component of this module is the evaluation unit which houses 2 AES cores designed to accept one gate per cycle. The garbled tables, garbler/evaluator input labels, and other necessary data such as output masks are obtained from the garbler and are stored in their respective buffer memory. Note that in order to maintain the privacy of input, the evaluator input labels are obtained from the garbler using oblivious transfer (OT). Each pair of input labels is processed in the evaluation unit to obtain the output label which is then sent back to the label scratchpad memory. It will be used in subsequent gate evaluation as the evaluator goes through the netlist gates one by one. Each of the AES core consists of a 10-stage pipeline performing AES-128 on ECB mode. Therefore it improves throughput but introduces potential dependency issues when evaluating the gates whose inputs have not been processed yet. This is the same issue as arises in FASE [32], the project we extended to evaluate the Ambassador. Note that the main operations in Evaluation is the opposite of Garbling where the goal is to use Garbled tables to generate and evaluate a circuit whereas the goal in Garbling is to produce garbled tables. Because of the Half-Gates optimization [72] the amount of work needed to be done by the garbler is 2× more than the evaluator. This explains why our Evaluator unit only needs 2 AES cores instead of 4 to achieve the same throughput performance.

V. Evaluation

With the application mapping and inherent algorithmic asymmetry described, the most pressing question is how well a compute-restricted device might actually perform on these MPC applications. Rather than rely on a simulation of the system, we perform direct system experimentation with two machines running the full application stack connected point-to-point. By tuning *down* the performance of the compute and network from this base "1:1" system we can explore the relative impact of network and compute asymmetry on the workload under evaluation.

A. Methodology

1) Hardware and Software Setup: To simulate the server-Embassy connection, we use two Equinix c3.small [8] bare metal nodes connected with a 10 Gbps LAN. Both machines have an Intel Core E-2278G 3.4 GHz (8C/16T) with 32 GB of memory and a top frequency of 5 GHz. Both are running Ubuntu 18.04.

We emulate a slower machine for the Embassy by scaling down from the maximum operating frequency of the client machine. We achieve this by setting the appropriate *max_perf_pct* Intel p-state parameter that corresponds to the percentage of maximum processor frequency. The particular machine we used for evaluation can be tuned from 800 MHz to 5 GHz (6.25× tuning range). Throughout the evaluation, we make use of 1 GHz (5× slowdown) as our representative Embassy performance. This roughly corresponds to the single-core benchmark performance gap between a typical server-class processor and processor (Intel Celeron N4100) from a commercially-available compute-stick [3].

Network	Description
SNN-MNIST_NetC	1-Conv, 2-FC, ReLU activation [59]
SNN-MNIST_NetD	2-Conv, 2-FC, ReLU and MaxPool [42]
SNN-CIFAR10	7-Conv, 1-FC, ReLU and MeanPool [42]

TABLE II: Neural Network Architectures for SNN Workloads

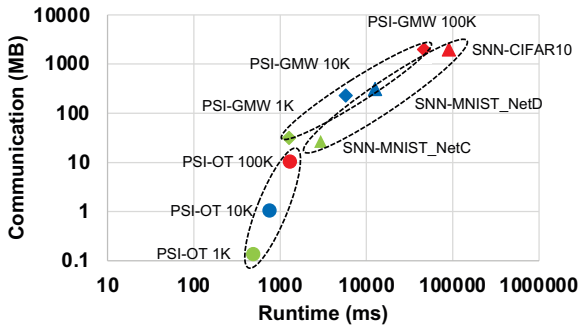

Fig. 7: Communication volume versus total application runtime. The color-coding indicates the change in input size for each application.

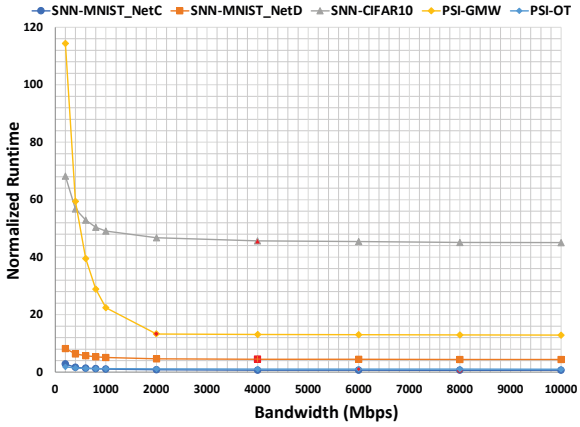

Fig. 8: Bandwidth limit analysis. Runtime is normalized to the smallest runtime of all applications. Latency is fixed at 0.6 ms (High Speed LAN). The red marker in each line indicates saturation in runtime, where the change in runtime starts to fall below 2% as bandwidth improves.

To accurately simulate a wide sweep of network transfer parameters between Embassy and the server over the LAN connection, we use the Linux *tc* tool. With this tool, we can add artificial delays to simulate latency and throttle bandwidth. We measure the effective network bandwidth and latency using *iperf3* and *ping*, respectively. The default network setting between Embassy and server has an average bandwidth of 9.42 Gbps and an average round trip latency (RTT) of 0.6 ms. We use the available secure neural network implementation from Gazelle [4] and the private set intersection implementations in ABY [6] and PSI [7] frameworks. Both applications are written in C++ and were adapted for our Embassy evaluation.

2) Parameter Selection: We consider two network settings: WAN and High-Speed LAN representing the baseline operation and the Embassy operation, respectively. We set the bandwidth/latency configuration for WAN as 200 Mbps/40 ms [15], [74] and High-Speed LAN as 10 Gbps/0.6 ms, which is typical in datacenters. Note that we refrain from selecting extreme network speeds to achieve overly optimistic results although modern datacenters have far more improved network infrastructure reaching bandwidths of 100 Gbps and 400 Gbps [2]. For both applications, we fix the *security parameter* κ to 128 bits. For secure neural network inference, we evaluate the performance overhead of two groups of neural networks designed for MNIST and CIFAR10, respectively. The network architectures are described in Table II. For PSI, we use a 32-bit entry size and fix the number of entries to 100 thousand elements for both client and server, which is a moderate size in DNA matching applications [38]. We include all one-time transfer, offline phase, and online phase costs in our timing measurements. Timing results were averaged over 10 execution iterations.

B. Baseline Embassy Results

Application Communication Cost: Figure 7 shows communication volume in MB as a function of the runtime for different applications with various input parameters. We can see that all applications show larger communication costs as the input size increases but they show different characteristics indicated by the slopes of their trend lines. PSI-OT shows little communication and scales well to large input sizes compared to PSI-GMW and SNN. The slope of PSI-OT is also steeper than PSI-GMW indicating that it is less sensitive to communication network improvement. PSI-GMW and SNN have comparable communication that is at least two orders of magnitude more than PSI-OT, while SNN has the highest runtime. PSI-GMW and SNN scale poorly in communication and runtime as input size increases and are thus ideal for Embassy.

Network Bandwidth Limit: Applications can be characterized by their communication-to-computation ratio which is determined by their underlying algorithm and protocol. This property can determine how much performance improvement the application can achieve by improving network bandwidth, i.e., the larger this ratio the larger the potential speedup. Figure 8 shows the normalized runtime of the applications for various bandwidth configurations at a High Speed LAN latency of 0.6 ms. PSI-GMW is more sensitive to changes in bandwidth compared to SNN applications below 2 Gbps. Since SNN applications have slightly greater communication-to-computation ratios, they saturate at higher bandwidths (indicated by the red markers in the figure) compared to PSI applications. A key observation is that most applications do not utilize the full bandwidth improvement and become compute-bound before reaching the High-Speed LAN bandwidth. There is no further speedup for SNN-CIFAR10 and SNN-MNIST_NetD after 4000 Mbps. The runtime of PSI-GMW stops decreasing as early as 2000 Mbps. As we will show later in the multithreaded experiments, the reason for relatively low saturation is due to the unoptimized use of threads in the applications. PSI-OT is more dominanted by public-key cryptography computations in base OT and thus shows the least benefit from bandwidth improvement. Note that bandwidth can also be better utilized when we have contention with multiple Embassies in the system.

Network Latency Sweep: Figure 9 shows the normalized runtime of the applications for various latency configurations at a High Speed LAN bandwidth of 10 Gbps. Compared to PSI, SNN applications are more sensitive to changes in latency, which are characterized by their greater slopes. This is intuitive because despite Garbled Circuit being a constant-round protocol, a large number of ReLU layers in SNN stacks up communication rounds, while data transfers in PSI can be efficiently batched. The runtimes of all 5 applications scale linearly with latency and show improvement throughout the entire latency range.

Embassy Performance: We illustrate application speedup using

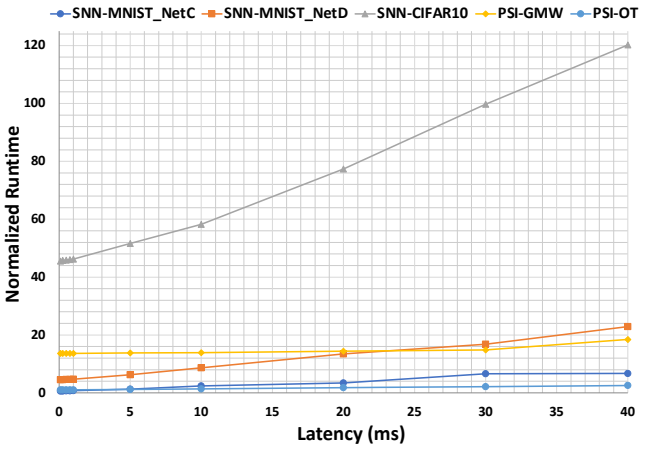

Fig. 9: Latency sweep analysis. Runtime is normalized based on the smallest runtime of all applications. Bandwidth is fixed at 10 Gbps (High Speed LAN).

Embassy in Figure 10 as a function of the performance of the Embassy scaled relative to the server. As discussed in Section V-A , we use core frequency as our performance scaling metric. A speedup of 1 (no speedup) indicates that the Embassy and the server have similar performance and that the speedup is gained from the network improvement from using an intra-system network. This speedup is gradually reduced as the Embassy is slowed down because any benefits from the network are lost from the slower computation. The dotted line represents the *slowdown margin* as this is the point where the speedup from network improvement is exhausted (speedup = 1) from continued Embassy performance slowdown. Note that since in our setup we can only test for a slowdown of 6.25×, we are unable to check the actual slowdown margin for some of the applications.

The slowdown margin of SNN applications is generally higher compared to PSI applications owing to the larger asymmetry in computations (more of the compute-intensive portions of the protocol happen in the untrusted server). For example, for an Embassy that has a slowdown of 5× we can get a speedup of as much as 2.33× in SNN-MNIST_NetD as opposed to 1.97× in PSI-GMW. Shallower networks for MNIST have greater speedup at the same slowdown rate. Within SNN applications, the gap between slowdown margins of the different network architectures comes from the communication composition of the workload. Since SNN-MNIST_NetC uses a significantly shallower neural network compared to SNN-MNIST_NetD and SNN-CIFAR10, communication takes a larger chunk of the overall runtime hence we can get a greater speedup. Note that for PSI-OT there is no speedup at 5× because it has the least communication-to-computation ratio among all applications, meaning that Embassy can barely have any improvement in terms of runtime performance for relatively more compute-bound applications.

Multithreading to Improve Bandwidth Utilization: The previous results show the default unoptimized configuration for the applications with limited thread usage. Since our setup largely alleviates the communication overhead, most applications become compute-bound, as shown in the bandwidth limit evaluation. In Figure 11, we illustrate that thread-level parallelism that takes advantage of the available to compute resources can improve the bandwidth utilization for those applications and in turn the performance of Embassy, which is not possible in the traditional WAN setting [51] with its limited bandwidth. The underlying GMW protocol can be parallelized evenly by dividing the multiplication triple generation in the setup phase to each individual thread [51]. At 5× device slowdown, the speedup grows by 1.96× by increasing from 2 threads to only 4 threads. As the number of threads increases from 2 to 8, the speedup increases by 3.42× from 1.97 to 6.73. Note that reducing the significant overhead of server-side homomorphic encryption in SNN algorithms can achieve a similar effect in the Embassy setting.

Energy Evaluation: One of the key advantages of using Embassy is the energy savings from performing secure multi-party computation locally within a co-located server, because that keeps communication within the co-located server instead of across a WAN. Figure 12 shows the estimated energy savings from using Embassy (client-Embassy-server) instead of baseline direct WAN (client-server) computation. Energy consumption is computed as a sum of the total network transfer energy and total computation energy. The network transfer energy gap is conservatively assumed to be 5× [5]. The computation energy is computed from a client and server TDP of 95W. Typical energy savings is around 15×. This is mostly due to the use of lower-energy local data movement compared to WANs. Note that Embassy still needs WAN transfers for the client communication but the High-Speed LAN communication still dominates the transfers. For the PSI-OT application, it is less affected by Embassy but 8× savings is still beneficial compared to the WAN setting.

As shown, the power reduction from adding Embassies far outweighs the power increase of the Embassies themselves, because those are more efficient than general-purpose untrusted servers. This is intuitive since the power consumption is 8× per virtual machine (VM) in modern datacenters which span from tens of Watts to hundreds of Watts [60]. Therefore, the additional 6W for an Embassy is comfortably outweighed by less expensive data movement and computation reduction on general-purpose servers.

C. Ambassador Evaluator Results

An Embassy implemented only as a compute stick-class processor is likely to see a significant performance slowdown as compared to the co-located servers it is connecting to. However, much of the cryptographic calculation that is performed in the MPC setting is amenable to hardware acceleration and so we propose to include such hardware accelerators as part of the Embassy in order to boost both performance and energy efficiency. Here, we investigate the performance improvement available for Embassy if we use dedicated hardware-accelerated implementation of the GC evaluator module to improve GC operations. Our Verilog implementation of the Ambassador evaluator is based on the garbler accelerator provided as part of FASE [32]. We show a comparison of the GC evaluation performance of the Ambassador Evaluator accelerator compared to a system without such an accelerator. Since we are interested in using Embassy for SNN workloads, we focus our evaluation on workloads with non-linear SNN operations like ReLU.

Table III shows the Ambassador Evaluator's estimated evaluation time and speedup compared to the CPU implementation. We implemented the ReLU circuit shown in Fig. 4 in Verilog and obtained the optimized gate count (shown in Table III) from synthesis using Synopsys Design Compiler using the TinyGarble Circuit Synthesis Library [1]. From this gate count, we make use of the similarity in architecture between our evaluator accelerator and the garbler accelerator provided by FASE [32] in order to estimate the expected performance of our evaluator accelerator. We conservatively estimate a range for a processing rate of 2-5.5 cycles/gate based on the simulation results of various circuits reported with the FASE garbler accelerator. We use this cycles/gate to estimate the range of evaluation time and

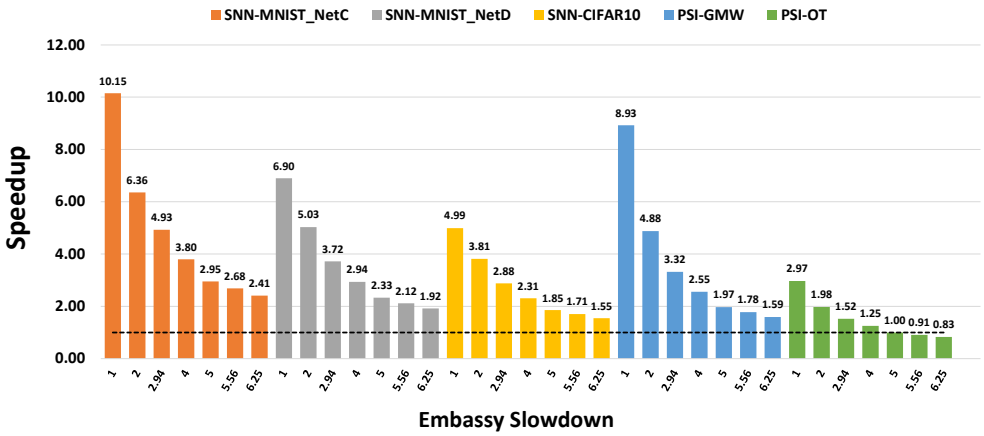

Fig. 10: Speedup as a function of Embassy slowdown. The runtime speedup for each application is obtained by improving the network bandwidth and latency from a WAN setting (baseline) to the Embassy with High-Speed LAN. The slowdown represents the Embassy performance slowdown relative to a server-class machine. The dotted line shows the slowdown margin for Embassy where the speedup from network improvement is exhausted (speedup = 1).

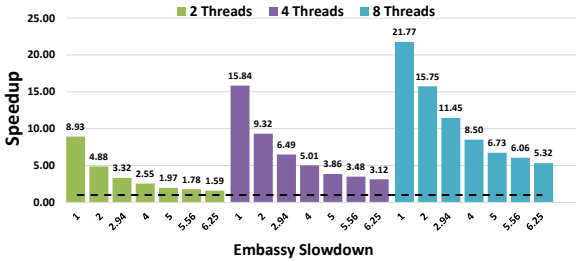

Fig. 11: Speedup as a function of Embassy slowdown for PSI-GMW with multithreading. The runtime speedup is obtained by improving the network bandwidth and latency from a WAN setting (baseline) to the Embassy setting that uses the High Speed LAN network.

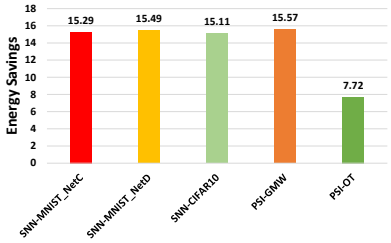

Fig. 12: Energy savings estimation comparing Embassy to direct WAN computation. TDP is assumed to be 95W for a server and 6W for Embassy. For conventional computation, both client's and server's TDP are the same. The average number of hops is assumed to be 16 end-to-end for WAN while local co-located server hops are assumed to be 5 as for a typical fat tree. Energy consumption is computed as a sum of the network transfer energy and computation energy.

the speedup compared to the software Gazelle implementation of the evaluator. At the FPGA's 100MHz clock frequency, we calculate a performance improvement ranging from 1.57x to 4.31x. Note that even though we demonstrate the advantage of the Ambassador Evaluator accelerator as implemented on FPGA in this study, an ASIC implementation could certainly be used and would result in further improved performance and energy efficiency.

	#XOR	#Non-XOR	#Total	Eval Time	Speedup
ReLU Unit	564	189	753	1506 - 4141 (cc) 15.06 - 41.41 (us)	1.57x -4.31x

TABLE III: Ambassador Evaluator Performance compared to Gazelle Evaluate function on CPU [37] for the ReLU unit in Fig. 4.

Resource Overhead: Our Ambassador Evaluator resource estimation exploits the FPGA infrastructure provided by the FASE Garbler implementation [32]. The FASE Garbler was implemented on a Xilinx Virtex UltraScale VCU108 FPGA while our Evaluator is implemented on a Xilinx Zynq ZCU104 FPGA with lesser system resources. Table IV shows the estimated resource utilization with a clock frequency of 100MHz. As expected, our Ambassador Evaluator accelerator consumes fewer resources than the FASE garbler as it only needs two AES cores compared to the garbler's four.

	Total	%Util
LUT	42472	18.43
Registers	11886	2.58
BRAM	37.5	12.02

TABLE IV: Resource Utilization

Overall SNN Workload Speedup: In order to estimate the overall improvement of introducing a dedicated evaluator accelerator into the Embassy, we profile the SNN applications for the percentage of execution time spent on non-linear layers versus the total runtime. Figure 13 shows the percentage of runtime spent on non-linear layers. It shows that the amount of time spent on the total runtime increases when the network becomes deeper and when the network increases in the number of non-linear layers such as ReLU and MaxPool.

We use this profiled non-linear execution time and the improvement we obtained from the individual ReLU unit running on the Ambassador Evaluator accelerator to calculate the overall speedup for running the whole network which we observed to range from **1.19**× to **1.52**× in a larger network like CIFAR10. The speedup available is dependent on the type of network and this setup favors deeper networks with more and wider non-linear layers like CIFAR10 as compared to MNIST networks. Note that although it is tempting to think of further speeding

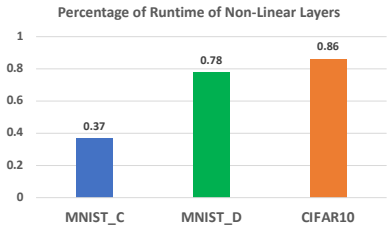

Fig. 13: Percentage of Non-Linear Layers in SNN Workloads

up the operation by adding support to Embassy for homomorphic encryption in the linear layers, in the design of Hybrid SNNs, the HE evaluation is done in the server and not in the client/Embassy, thus server support, rather than Embassy support, would be required.

VI. Related Work

In considering the viability of our approach in a co-located server setting, we look for inspiration from two trends in co-located server infrastructure design. The first is disaggregated datacenter networks [26], which increase the efficiency and lower the total cost of ownership (TCO) of datacenters using network-attached host-less accelerators. For example, Facebook recently rolled out their F16 [14] datacenter fabric design. The second trend is the adoption of bare metal cloud services [71], where providers allocate dedicated servers for customers. Unlike typical virtual machine-based cloud providers like AWS and Google, doing away with layers of virtualization and dedicating the use of hardware resources results in performance improvements. Further, because clients do not have to share the same physical machines (single tenant), there are fewer potential security risks from recent cross-VM side-channel attacks [39], [41], [64], [69]. Embassies are host-less network-connected computing devices that are exclusively used by clients to perform MPC with co-located untrusted servers.

A. Trusted Hardware

Trusted hardware such as Intel SGX has been used to support privacy-preserving machine learning [28], [34], [49], [68]. SGX creates enclaves for isolated execution environments and supports remote attestation. However, SGX is fundamentally limited because the trusted execution environment and an untrusted CPU share the same computing resources resulting in a switching overhead. Furthermore, it has limited memory resources (90MB), leading to paging overheads for larger applications [20]. These make these solutions not feasible for evaluating much larger networks, not to mention recent side-channel attacks in SGX [69]. To meet demands for larger workloads, Intel recently released a PCIe interface-based SGX Card with three SGX-equipped CPUs [19]. This hardware with its discrete processors would incur significantly more power than our specialized solution and is a band-aid solution with the same fundamental performance and security flaws of SGX.

Trusted hardware has also been used to support secure multiparty computation. Bahmani et al. [16] make use of code running in the SGX as a trusted third-party and parties which are represented as SGX enclaves perform function evaluation during the online phase. Sartakov et al. [61] extend this by adding support for fast inter-enclave communication. For both works, because of SGX limitations, evaluated applications are very simple such as summations, unlike the applications we consider. Demmler et al. [22] used a trusted secure card in a mobile phone to speed up the generation of multiplication triples in the offline phase of GMW. Our trusted Embassy is a much more capable device that participates in the online phase of the computation. Embassy also physically decouples the computation and does not share any resources with the host. This reduces the number of avenues for side-channel attacks, but does require physical security mechanisms for the Embassy device. Additionally the Embassy device can be flexible in the amount of compute resources it has, allowing the device to be designed to fit the workload.

Bugiel et al. [17] proposed a Twin Clouds model which represents the closest work to our protocol but has many significant differences. First, they make a strong assumption of non-collusion between the two cloud machines. This is not the case for our work since the Embassy is considered a trusted proxy of the client. Second, Twin Clouds' high bandwidth channel is not aimed to improve the network overhead of secure communication but instead, it is used for quick bulk file transfers. Third, they don't describe potential hardware implementation and evaluations.

Eguro et al. [25] proposed FPGA-based secure computation hardware aimed at emulating homomorphic encryption. Our solution, on the other hand, involves no host for the trusted device and can be used to make MPC more efficient.

More recently, Telekine [31] was proposed to mitigate side-channel attacks when clients use cloud-deployed GPUs with TEEs. HETEE [73] was designed to manage all compute units in a server rack by using the PCIe switch fabric to securely allocate accelerators. Unlike Telekine and HETEE, Embassy only considers the security of one single type of portable device.

B. Privacy-Preserving NN Inference

CryptoNets [24] is the first work on privacy-preserving neural network inference we are aware of. It is used as a leveled homomorphic encryption scheme for evaluating all layer which resulted in significant performance overhead and lower accuracy from using square activation functions. DeepSecure [59] used an all garbled circuit approach which improved the computation efficiency of CryptoNets but in turn had worse communication overhead. For example, to perform an MNIST-based inference operation, DeepSecure needs to transfer 791MB per single inference compared to CryptoNets's 595MB for a batch size of 8129. To address this problem, Gazelle [37] proposed a hybrid protocol composed of HE and MPC for NN inference. In this scheme, HE is used for linear operations (e.g., matrix-vector multiplication in convolutional layers) while MPC is used for non-linear operations (ReLU and max pooling functions). This improved the overall compute and communication overheads since HE performs better than GC when the computation has small multiplicative depth (linear function Boolean circuit) and GC is better suited for non-linear functions which can be represented as simple linear-size circuits. However, it still suffers from significant communication overhead because of non-linear layers making it difficult to scale to much larger networks. Our work uses this hybrid protocol for neural network inference but improves on the communication overhead using the Embassy protocol.

XONN [56] proposed the use binarized neural network (BNN) with garbled circuits to speedup linear layers. BNNs use XNOR for multiplication which is considered free when using GC (FreeXOR). This allowed them to make evaluate much larger networks such as VGG. However, as they still use GC for the non-linear layers, there is still significant communication overhead. Furthermore, despite being more efficient computationally, BNNs show significantly lower accuracy. Chameleon [57] proposed the use of a trusted third party to generate multiplication triples during the offline phase. They adopt a seed expansion technique for multiplication triples to save communication at the expense of more computation in random number

generation. However, our solution allows the efficient use of the original beaver triple generation with less communication overhead.

There have been proposals to combine GC with other secure computation primitives, such as secret sharing using two untrusted servers, which can be housed by the same cloud and connected in a high bandwidth and low latency channel [21], [36], [62]. These solutions, however, make a strong assumption that two untrusted servers are *non-colluding*.

HEAX [55] proposed the first hardware accelerator implementation for CKKS HE on FPGAs. Cheetah [54] significantly accelerates HE in Gazelle for deeper neural networks by optimizing HE parameters tuning and operator scheduling, while proposing a custom hardware accelerator for server-side HE. The results of HEAX and Cheetah are orthogonal to the contribution of this paper since our solution tackles the communication bottleneck in MPC.

DELPHI [45] improved upon Gazelle by moving expensive cryptographic operations over LHE cipher-texts to the offline phase and proposed to use quadratic polynomials to approximate ReLU, which reduces communication cost. However, DELPHI had to settle with a hybrid approach because of severe accuracy degradation from quadratic approximations.

C. Tamper-Resistant Hardware

With the rise of mobile and IoT devices, there is greater risk for more sophisticated physical tampering and side-channel attacks. To address this, Google released a tamper-resistant security module [10] used starting from Pixel 2 phones while ARM released Cortex-M35P processor [13] for embedded IoT applications. These solutions can protect against physical penetration and most side-channel attacks (power, timing, electromagnetic).

Two examples of tamper-resistant USB device available in the market are IronKey and Kanguru. IronKey is a FIPS 140-2 Level 3-certified device which zeroizes data or makes the device unusable by applying a wear level current on the device memory after a configurable number of break-in attempts. Kanguru, on the other hand, has a casing that is protected with an epoxy compound, which when removed, destroys the flash chip making the device unusable.

Recently, Immler et al. [35] presented tamper-resistant secure physical enclosure for PCBs. This work allowed for a more practical battery-less physical tampering solution and also proposed the use of PUFs for determining the structural integrity of the device. This tamper resistance mechanism is particularly useful for Embassy.

D. Hardware Support for MPC

There have been a few works related to hardware support for secure multi-party computation. Songhori et al. proposed TinyGarble [66] to convert big combinational circuits to smaller sequential circuits which is run on multiple clock cycles. The compact circuit results in smaller memory footprint which can fit in the processor cache. As a result, cache misses are minimized during garbling while accessing wire tokens improving garbling performance. This smaller footprint makes it more useful for embedded devices which have limited compute and memory resources.

Implementation and acceleration of the garbling operation have been shown in various hardware platforms [33], [65], [67]. Since they only tackle the issue of GC computation (garbling), overall performance of the protocol is not significantly improved since the bottleneck of the protocol is communication (network transfer) especially with larger applications. This is in contrast to our work which focuses on the communication overhead of secure computation.

VII. Conclusion

In this paper we explore supporting collections of small but physically secure devices embedded closely with more traditional compute and storage resources. The use of co-located trusted hardware helps resolve the inefficiency of conventional two-party secure computation protocols with surprisingly little compute. We evaluate the ability of such devices to participate in trustworthy computations physically among co-located servers on behalf of a remote client. This general approach could be useful in many different scenarios, but we evaluate one of the most integrated ways one might think to apply such trusted elements: as an active party in a multiparty computation. We show how this Hardware Embassy can leverage a local high bandwidth and low latency network connection to enable more efficient and robust interactive secure computations. We further show that two important privacy-preserving applications, secure neural network inference, and private DNA matching based on Yao's Garbled Circuit (GC) and Goldreich-Micali-Wigderson (GMW), are both amenable to this heterogeneous architecture even without any application specialization. Our experiments indicate that even when the Embassy is 5× slower than external compute resources available, the total system performance is higher due to this increased connectivity. This advantage can be further pressed with addition of specialized hardware, bringing the total performance improvement up to over 4.5×.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants No. 1763699, 1730309, 1717779, 1563935. We thank our shepherd and the anonymous reviewers for their feedback.

References

[1] "Aceslabucsd/tinygarblecircuitsynthesis: Circuit synthesis for yao's garbled circuit by tinygarble," https://github.com/ACESLabUCSD/TinyGarbleCircuitSynthesis, (Accessed on 06/09/2022).
[2] "AWS Direct Connect," https://aws.amazon.com/directconnect/.
[3] "Azulle Access3," https://azulletech.com/product/access3/.
[4] "chiraag/gazelle_mpc," https://github.com/chiraag/gazelle_mpc, (Accessed on 06/09/2022).
[5] "Cost of Click," https://energyzarr.typepad.com/energyzarrnationalcom/2008/08/the-true-cost-o.html.
[6] "encryptogroup/aby: Aby - a framework for efficient mixed-protocol secure two-party computation," https://github.com/encryptogroup/ABY, (Accessed on 06/09/2022).
[7] "encryptogroup/psi: Implementations of private set intersection protocols," https://github.com/encryptogroup/PSI, (Accessed on 06/09/2022).
[8] "Equinix Metal," https://metal.equinix.com/.
[9] "Google Edge TPU," https://cloud.google.com/edge-tpu/.
[10] "How the Pixel 2's security module delivers enterprise-grade security," https://www.blog.google/products/android-enterprise/how-pixel-2s-security-module-delivers-enterprise-grade-security/.
[11] "Intel Compute Stick," https://www.intel.com/content/www/us/en/products/boards-kits/compute-stick.html.
[12] "Intel Unveils the Intel Neural Compute Stick 2 at Intel AI Devcon Beijing for Building Smarter AI Edge Devices," https://newsroom.intel.com/news/intel-unveils-intel-neural-compute-stick-2/.
[13] "New Arm IP Helps Protect IoT Devices from Increasingly Prevalent – Arm," https://www.arm.com/company/news/2018/05/new-arm-ip-helps-protect-iot-devices-from-increasingly-prevalent-physical-threats.
[14] "Reinventing our data center network with F16, Minipack - Facebook Code," https://code.fb.com/data-center-engineering/f16-minipack/.
[15] "Verizon IP Latency Statistics," https://www.verizon.com/business/terms/latency/.
[16] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, and B. Warinschi, "Secure Multiparty Computation from SGX," in *Financial Cryptography and Data Security*, A. Kiayias, Ed. Cham: Springer International Publishing, 2017, pp. 477–497.

[17] S. Bugiel, S. Nürnberger, A.-R. Sadeghi, and T. Schneider, "Twin clouds: Secure cloud computing with low latency," in *Proceedings of the 12th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security*, ser. CMS'11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 32–44.

[18] A. Caulfield, P. Costa, and M. Ghobadi, "Beyond smartnics: Towards a fully programmable cloud: Invited paper," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, 2018, pp. 1–6.

[19] S. Chakrabarti, M. Hoekstra, D. Kuvaiskii, and M. Vij, "Scaling intel® software guard extensions applications with intel® sgx card," in *Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy*, ser. HASP '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3337167.3337173

[20] V. Costan and S. Devadas, "Intel sgx explained," *IACR Cryptology ePrint Archive*, vol. 2016, p. 86, 2016.

[21] D. Demmler, K. Hamacher, T. Schneider, and S. Stammler, "Privacy-preserving whole-genome variant queries," in *Cryptology and Network Security*, S. Capkun and S. S. M. Chow, Eds. Cham: Springer International Publishing, 2018, pp. 71–92.

[22] D. Demmler, T. Schneider, M. Zohner, and T. Universit, "Ad-Hoc Secure Two-Party Computation on Mobile Devices using Hardware Tokens," *USENIX Security*, pp. 893–908, 2014. [Online]. Available: https://dl.acm.org/citation.cfm?id=2671282

[23] G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner, "Pushing the Communication Barrier in Secure Computation using Lookup Tables," in *Proceedings 2017 Network and Distributed System Security Symposium*. Reston, VA: Internet Society, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/pushing-communication-barrier-secure-computation-using-lookup-tables/

[24] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 201–210. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045413

[25] K. Eguro and R. Venkatesan, "Fpgas for trusted cloud computing," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2012, pp. 63–70.

[26] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network Requirements for Resource Disaggregation," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 249–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=3026877.3026897

[27] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. New York, NY, USA: ACM, 1987, pp. 218–229. [Online]. Available: http://doi.acm.org/10.1145/28395.28420

[28] K. Grover, S. Tople, S. Shinde, R. Bhagwan, and R. Ramjee, "Privado: Practical and secure dnn inference with enclaves," 2019.

[29] A. Guleria, J. Lakshmi, and C. Padala, "Quadd: Quantifying accelerator disaggregated datacenter efficiency," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 2019, pp. 349–357.

[30] A. Herzberg and H. Shulman, "Oblivious and fair server-aided two-party computation," in *2012 Seventh International Conference on Availability, Reliability and Security*, 2012, pp. 75–84.

[31] T. Hunt, Z. Jia, V. Miller, A. Szekely, Y. Hu, C. J. Rossbach, and E. Witchel, "Telekine: Secure computing with cloud gpus," in *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 2020, pp. 817–833.

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

[33] S. U. Hussain, B. D. Rouhani, M. Ghasemzadeh, and F. Koushanfar, "MAXelerator: FPGA Accelerator for Privacy Preserving Multiply-Accumulate (MAC) on Cloud Servers," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, jun 2018, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8465770/

[34] N. Hynes, R. Cheng, and D. Song, "Efficient Deep Learning on Multi-Source Private Data," *arXiv e-prints*, p. arXiv:1807.06689, Jul. 2018.

[35] V. Immler, J. Obermaier, K. Ng, F. Ke, J. Lee, Y. Lim, W. Oh, K. Wee, and G. Sigl, "Secure physical enclosures from covers with tamper-resistance," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 51–96, Nov. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7334

[36] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, and G. Bejerano, "Deriving genomic diagnoses without revealing patient genomes." *Science (New York, N.Y.)*, vol. 357, no. 6352, pp. 692–695, aug 2017. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/28818945

[37] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A Low Latency Framework for Secure Neural Network Inference," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18. Berkeley, CA, USA: USENIX Association, 2018, pp. 1651–1668. [Online]. Available: http://dl.acm.org/citation.cfm?id=3277203.3277326

[38] M. Kim, Y. . C. Song, and J.H, "Secure searching of biomarkers through hybrid homomorphic encryption scheme," in *BMC Med Genomics*, 2017. [Online]. Available: https://bmcmedgenomics.biomedcentral.com/articles/10.1186/s12920-017-0280-3

[39] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[40] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, vol. 64, no. 1, p. 86–96, Dec. 2020. [Online]. Available: https://doi.org/10.1145/3387108

[41] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[42] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious Neural Network Predictions via MiniONN Transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 619–631. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134056

[43] lowRISC C.I.C., "OpenTitan," https://opentitan.org/.

[44] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: An execution infrastructure for tcb minimization," in *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*, ser. Eurosys '08. New York, NY, USA: ACM, 2008, pp. 315–328. [Online]. Available: http://doi.acm.org/10.1145/1352592.1352625

[45] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2505–2522. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/mishra

[46] P. Mohassel and Y. Zhang, "SecureML: A System for Scalable Privacy-Preserving Machine Learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, may 2017, pp. 19–38. [Online]. Available: http://ieeexplore.ieee.org/document/7958569/

[47] NIST, "Fips pub 140-2: Security requirements for cryptographic modules," 2001.

[48] NIST, "Fips pub 140-3: Security requirements for cryptographic modules," 2019.

[49] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 619–636. [Online]. Available: http://dl.acm.org/citation.cfm?id=3241094.3241143

[50] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 515–530. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas

[51] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Trans. Priv. Secur.*, vol. 21, no. 2, Jan. 2018. [Online]. Available: https://doi.org/10.1145/3154794

[52] M. T. Rahman, Q. Shi, S. Tajik, H. Shen, D. L. Woodard, M. Tehranipoor, and N. Asadizanjani, "Physical inspection attacks: New frontier in hardware security," in *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, 2018, pp. 93–102.

[53] N. Rangarajan, S. Patnaik, J. Knechtel, S. Rakheja, and O. Sinanoglu, "Tamper-proof hardware from emerging technologies," in *The Next Era in Hardware Security*. Springer, 2021, pp. 195–209.

[54] B. Reagen, W. Choi, Y. Ko, V. Lee, G.-Y. Wei, H.-H. S. Lee, and D. Brooks, "Cheetah: Optimizing and accelerating homomorphic encryption for private inference," in *2021 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2021. [Online]. Available: http://ieeexplore.ieee.org/document/8327042/

[55] M. S. Riazi, K. Laine, B. Pelton, and W. Dai, "Heax: An architecture for computing on encrypted data," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1295–1309. [Online]. Available: https://doi.org/10.1145/3373376.3378523

[56] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. E. Lauter, and F. Koushanfar, "XONN: XNOR-based Oblivious Deep Neural Network Inference," in *Proceedings of the 28th USENIX Conference on Security Symposium*, ser. SEC'19. USENIX Association, 2019. [Online]. Available: http://arxiv.org/abs/1902.07342https://eprint.iacr.org/2019/171.pdf

[57] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security - ASIACCS '18*. New York, New York, USA: ACM Press, 2018, pp. 707–721. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196494.3196522

[58] B. D. Rouhani, S. U. Hussain, K. Lauter, and F. Koushanfar, "ReDCrypt: Real-Time Privacy-Preserving Deep Learning Inference in Clouds Using FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 11, no. 3, pp. 1–21, dec 2018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3299999.3242899

[59] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable Provably-Secure Deep Learning," in *Proceedings of the 55th Annual Design Automation Conference on - DAC '18*. New York, New York, USA: ACM Press, 2018, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3195970.3196023

[60] P. Ruiu, C. Fiandrino, P. Giaccone, A. Bianco, D. Kliazovich, and P. Bouvry, "On the energy-proportionality of data center networks," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 197–210, 2017.

[61] V. A. Sartakov, S. Brenner, S. Ben Mokhtar, S. Bouchenak, G. Thomas, and R. Kapitza, "Eactors: Fast and flexible trusted computing using sgx," in *Proceedings of the 19th International Middleware Conference*, ser. Middleware '18. New York, NY, USA: ACM, 2018, pp. 187–200. [Online]. Available: http://doi.acm.org/10.1145/3274808.3274823

[62] T. Schneider and O. Tkachenko, "EPISODE: Efficient Privacy-PreservIng Similar Sequence Queries on Outsourced Genomic DatabasEs," in *Proceedings of the 2019 on Asia Conference on Computer and Communications Security - ASIACCS '19*, 2019. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196494.3196522

[63] T. Schneider and M. Zohner, "GMW vs. Yao? Efficient secure two-party computation with low depth circuits," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7859 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 275–292. [Online]. Available: http://link.springer.com/10.1007/978-3-642-39884-1{_}23

[64] M. Schwarz, M. Lipp, D. Moghimi, J. Van Bulck, J. Stecklina, T. Prescher, and D. Gruss, "Zombieload: Cross-privilege-boundary data sampling," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 753–768. [Online]. Available: https://doi.org/10.1145/3319535.3354252

[65] E. M. Songhori, T. Schneider, S. Zeitouni, A. Sadeghi, G. Dessouky, and F. Koushanfar, "Garbledcpu: A mips processor for secure computation in hardware," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6.

[66] E. M. Songhori, S. U. Hussain, A. R. Sadeghi, T. Schneider, and F. Koushanfar, "TinyGarble: Highly compressed and scalable sequential Garbled Circuits," *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2015-July, pp. 411–428, 2015.

[67] E. M. Songhori, M. S. Riazi, S. U. Hussain, A.-R. Sadeghi, and F. Koushanfar, "Arm2gc: Succinct garbled processor for secure computation," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317777

[68] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJVorjCcKQ

[69] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-order Execution," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18. Berkeley, CA, USA: USENIX Association, 2018, pp. 991–1008. [Online]. Available: http://dl.acm.org/citation.cfm?id=3277203.3277277

[70] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, oct 2008, pp. 162–167. [Online]. Available: http://ieeexplore.ieee.org/document/4568207/

[71] S. Yeoman, "How secure are bare metal servers?" *Network Security*, vol. 2019, no. 2, pp. 16 – 17, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485819300248

[72] S. Zahur, M. Rosulek, and D. Evans, "Two Halves Make a Whole," in *Advances in Cryptology - EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 220–250.

[73] J. Zhu, R. Hou, X. Wang, W. Wang, J. Cao, B. Zhao, Z. Wang, Y. Zhang, J. Ying, L. Zhang, and D. Meng, "Enabling rack-scale confidential computing using heterogeneous trusted execution environment," 2020.

[74] R. Zhu, D. Cassel, A. Sabry, and Y. Huang, "NANOPI: Extreme-Scale Actively-Secure Multi-Party Computation," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18*. New York, New York, USA: ACM Press, 2018, pp. 862–879. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3243734.3243850