Client-Optimized Algorithms and Acceleration for Encrypted Compute Offloading

McKenzie van der Hagen
mckenziv@andrew.cmu.edu
Carnegie Mellon University
United States

Brandon Lucia
blucia@andrew.cmu.edu
Carnegie Mellon University
United States

ABSTRACT

Homomorphic Encryption (HE) enables secure cloud offload processing on encrypted data. HE schemes are limited in the complexity and type of operations they can perform, motivating client-aided implementations that distribute computation between client (unencrypted) and server (encrypted). Prior client-aided systems optimize server performance, ignoring client costs: client-aided models put encryption and decryption on the critical path and require communicating large ciphertexts. We introduce Client-aided HE for Opaque Compute Offloading (CHOCO), a client-optimized system for encrypted offload processing. CHOCO reduces ciphertext size, reducing communication and computing costs through HE parameter minimization and through "rotational redundancy", a new HE algorithm optimization. We present Client-aided HE for Opaque Compute Offloading Through Accelerated Cryptographic Operations (CHOCO-TACO), an accelerator for HE encryption and decryption, making client-aided HE feasible for even resource-constrained clients. CHOCO supports two popular HE schemes (BFV and CKKS) and several applications, including DNNs, PageRank, KNN, and K-Means. CHOCO reduces communication by up to 2948× over prior work. With CHOCO-TACO client enc-/decryption is up to 1094× faster and uses up to 648× less energy.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**; • **Computer systems organization** → *Parallel architectures*; • **Software and its engineering** → *Designing software.*

KEYWORDS

Homomorphic Encryption, Privacy-Preserving Computation, Hardware Acceleration, IoT, Compute Offloading

ACM Reference Format:
McKenzie van der Hagen and Brandon Lucia. 2022. Client-Optimized Algorithms and Acceleration for Encrypted Compute Offloading. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '22), February 28 – March 4, 2022, Lausanne, Switzerland*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3503222.3507737

ASPLOS '22, February 28 – March 4, 2022, Lausanne, Switzerland

ACM ISBN 978-1-4503-9205-1/22/02.
https://doi.org/10.1145/3503222.3507737

1 INTRODUCTION

Data-producing client devices have a long history of decreasing in size, energy storage, and processing capability [10, 16, 25, 35, 43, 49]. The sophistication of computations on sensor data is simultaneously scaling up, often using complex machine learning (ML). Compute offloading, e.g., "inference as a service", is a way to overcome device limitations and meet rising computation demands. Clients send data to be processed by a robust server that may house aggregated datasets or a collection of ML models. Centralization makes systems easy to evolve, requiring server updates only, and avoiding the need to re-distribute large applications (such as Deep Neural Network models) to many fielded clients. *Data privacy* is the main barrier to realizing these benefits of offload computing: offloading exposes sensitive user data to a shared, potentially untrusted offload server.

Recent work offers several options for privacy-preserving computation, including trusted execution environments (TEEs)[34, 51, 68], differential privacy (DP), multi-party computation (MPC)[2, 11, 60], and homomorphic encryption (HE) [6, 8, 20, 22]. Of these, HE provides the strongest client security guarantees [8]. Unfortunately, complete HE programs are inefficient and highly limited in the type of computation that can be performed[8, 20, 22, 29, 31]. Instead, client-aided, hybrid HE-MPC protocols have seen recent success, mostly for DNN inference offloading [36, 41, 47, 55], by only using HE to apply linear operations to encrypted user data (e.g. convolution). Obfuscated intermediate results are then sent to the client, which applies non-linear operations (e.g. activation) using MPC.

Hybrid HE-MPC is promising, but currently infeasible for resource-constrained clients. Existing solutions optimize HE-MPC to benefit the offload server in both performance and privacy. These systems use large ciphertexts (MBs) and require large amounts of client-server comunication; e.g., GBs for a DNN inference. Prior work neglects to optimize HE-MPC's high compute and communication cost at the client.

This work proposes Client-aided HE for Opaque Compute Offloading (CHOCO), a new approach to client-privacy-preserving computation that minimizes client costs. CHOCO is an alternative to local compute, with its resource limitations and inability to use centralized ML models and data. CHOCO is also an alternative to the extreme client compute and communication costs of server-optimized HE-MPC. CHOCO reduces client costs by *orders of magnitude* over existing HE-MPC protocols, availing even resource-constrained client devices of the benefits of privacy-preserving compute offload.

CHOCO starts with client-aided HE, performing encrypted linear operations on the server and plaintext non-linear operations on the client. CHOCO minimizes HE parameters to minimize ciphertext communication costs, and introduces *rotational redundancy*, a

new encrypted permutation algorithm that reduces communication and resource costs. A unique facet of client-aided HE motivates CHOCO: under typical HE assumptions encryption and decryption happen once per computation, but in client-aided HE (and HE-MPC) encryption and decryption happen repeatedly on the critical path. We quantitatively show that the prohibitively high time and energy cost to encrypt and decrypt becomes the client's primary computation bottleneck. We propose Client-aided HE for Opaque Compute Offloading Through Accelerated Cryptographic Operations (CHOCO-TACO), a hardware accelerator for encryption and decryption that virtually eliminates their time cost to the client.

Our evaluation of a complete hardware-software implementation demonstrates the benefits of CHOCO for two popular HE schemes (CKKS and BFV) and for several applications *that are not limited to ML*, many of which have no prior encrypted implementation. Directly comparing to seven prior HE and/or MPC approaches, CHOCO reduces client communication costs by *orders of magnitude*, with improvements ranging from 14×–2948×. CHOCO-TACO's hardware acceleration improves active client compute time by 123.27× compared to software[1] and 54.3× compared to HEAX[59]. Our results show that with CHOCO, privacy-preserving collaborative inference offloading performs comparably to local inference in time and energy, while avoiding the limitations of both local compute and full encrypted offload. We also show the first encrypted implementation of distance-based algorithms and PageRank, demonstrating their feasibility. Our main contributions are:

- CHOCO, a client-optimized system for privacy-preserving computation enabling encrypted computing for resource-constrained devices.
- Rotational redundancy, an encrypted permutation algorithm that minimizes client-server communication.
- CHOCO-TACO, a specialized hardware accelerator for client-side HE primitives.
- A collection of CHOCO-based encrypted workloads (DNNs, distance-based algorithms, PageRank) — several of which with no prior encrypted implementation — that improve client costs by orders of magnitude compared to HE-MPC while benefitting from model centralization.

2 BACKGROUND & MOTIVATION

CHOCO is a client-privacy-preserving, encrypted computing system based on client-aided homomorphic encryption (HE). In an HE application, a client device offloads data to a server, which applies encrypted Single Instruction Multiple Data (SIMD) operations on those data using HE algorithms.

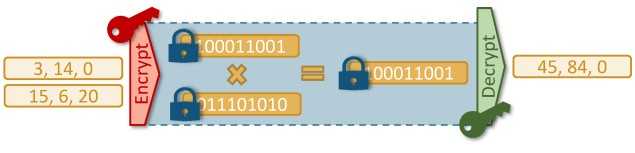

Figure 1: Homomorphic Encryption allows for computation directly on encrypted vectors of data.

2.1 Homomorphic Encryption

Homomorphic encryption is a class of cryptography schemes that allow computing on encrypted data. For messages m_1 and m_2, an operation $\oplus$, its homomorphic variant $\oplus'$, and encrypt/decrypt functions $Enc()$ and $Dec()$, the homomorphic operation on encrypted data produces a result that, when decrypted, equals the operation applied to unencrypted data:

$$Dec(Enc(m_1) \oplus' Enc(m_2)) = m_1 \oplus m_2 \quad (1)$$

Modern HE schemes [7, 13, 23] are based on ring learning with errors (RLWE). RLWE schemes encrypt a vector of values as the coefficients of a polynomial using modular arithmetic and by adding random noise. The HE operations in Table 1 manipulate vector ciphertexts, producing the encrypted result of an element-wise operation applied to the input, as Figure 1 shows. Each operation increases the ciphertext's noise level, which must remain within a *noise budget*. Some operations (multiplication) add a large amount of noise while others (addition) add little. Noise growth limits arithmetic depth. Exhausting the noise budget renders data undecryptable. An HE system must schedule operations to limit noise growth.

HE can *refresh* a ciphertext to eliminate noise, replenishing its budget. *Fully* Homomorphic Encryption (FHE) refreshes noise via specialized bootstrapping operations. This approach avoids decryption but comes at an enormous cost for vector HE schemes [7, 26]. In contrast, *Somewhat* Homomorphic Encryption (SHE) [13, 23, 36, 58] refreshes noise by decrypting and re-encrypting at known intervals.

Table 1: The operations available in HE. All operations are performed with a ciphertext, i.e. a plaintext multiply denotes the multiplication of a plaintext with a ciphertext.

Operation	Complexity	Noise Growth
Encrypt	$O(N \times \log N \times r)$	N/A
Decrypt	$O(N \times \log N \times r)$	N/A
Plaintext Add	$O(N \times r)$	Small
Ciphertext Add	$O(N \times r)$	Small
Plaintext Multiply	$O(N \times \log N \times r)$	Moderate
Ciphertext Multiply	$O(N \times \log N \times r^2)$	Large
Ciphertext Rotate	$O(N \times \log N \times r^2)$	Small

HE Schemes. HE schemes describe how to encrypt, decrypt, and operate on data. CHOCO targets the SHE variants of the two most prevalent vector HE schemes. Namely, the Brakerski/Fan-Vercauteren (BFV) scheme[7, 23] and the Cheon/Kim/Kim/Song (CKKS) scheme[13]. Both are included in SEAL[58], a highly-optimized HE library. The schemes work differently, but share many subcomputations. We focus on CHOCO for BFV here, but CHOCO works for CKKS with few additions, which we call out explicitly.

An HE scheme is defined by a set of parameters that dictate security, computational complexity, noise budget, arithmetic depth, and ciphertext size. Table 2 summarizes these parameters for BFV and CKKS. The polynomial modulus, N, is a power of two typically between 2^{11} and 2^{15}. A fresh ciphertext with $s = 2$ is two polynomials (vectors) with N coefficients (elements) each. A smaller coefficient modulus, q, is more secure but accommodates less noise; practical q values have hundreds of bits. Operating directly on

Table 2: HE Parameters for the BFV Scheme [7, 23, 39]

Parameter	Name	Description
N	Polynomial Modulus	# of coeffs per ciphertext.
q	Coeff. Modulus	Max value of ciphertext coefficient
k	# Coprime Moduli	Number of moduli residues in RNS
$\{k\}$	Coprime Mod. Bits	Bits per coprime modulus
w	Word Size	Bytes per encrypted coefficient
t	Plaintext Modulus	Max value of plaintext coeff. (BFV only)
s	Ciphertext Components	# polynomials per ciphertext

such large values is inefficient and HE uses the Residual Number System (RNS) [4] to represent numbers using k smaller, co-prime moduli. SEAL uses up to 60-bit moduli, which fit in a 64-bit word. BFV supports integer operations modulo a plaintext modulus, t. Within a fixed q, a larger t permits larger values but less noise. CKKS supports fixed-point operations and has no plaintext modulus. Parameter selection is application-dependent. It must allow for sufficiently large quantized (BFV) or scaled (CKKS) plaintext values and sufficient noise. Table 3 shows CHOCO's parameter settings and their associated ciphertext sizes.

Table 3: HE Parameter Selections: All parameters are chosen to satisfy at least 128-bit security.

Label	Scheme	N	$\log_2 q$	$\{k\}$	$\log_2 t$	Size (Bytes)
A	BFV	8192	175	{58,58,59}	23	262,144
B	BFV	4096	109	{36,36,37}	18	131,072
C	CKKS	8192	140	{60,60,60}	N/A	262,144

HE Algorithms. The HE operations in Table 1 perform encrypted SIMD arithmetic on large vectors and are combined to create HE algorithms e.g., encrypted convolution. An HE algorithm's performance depends on the packing of its inputs into ciphertexts. *Batching* algorithms optimize for throughput and utilize SIMD processing directly. They pack a ciphertext vector with a single element (e.g. pixel) from many inputs (e.g. pictures) [22, 32]. In turn, they are highly inefficient for few inputs. *Packed* algorithms optimize for latency, packing one or more full inputs in a single ciphertext. They use permutations to align and operate on specific elements [8, 28, 36]. Consequently, packed algorithms quickly suffer from fragmentation as vectors are rearranged with each operation.
HE Applications. While FHE theoretically supports arbitrary functions (by mapping to polynomial approximations and bootstrapping at high computational cost [7, 20, 26]), practical applications use HE operations directly and are arithmetic-depth-limited. Early work on HE applications, particularly for DNN inference, operate on encrypted data only [8, 20, 22, 32]. To avoid intermediate client communication, they adapt network architectures and approximate activations with linear functions. These techniques accumulate noise quickly and require large HE parameters, resulting in multi-MB ciphertexts which are ultimately more costly to process and communicate. *Client-aided* alternatives [6, 36, 55] offload the linear convolution and fully-connected layers to an HE server, sending intermediate results to the client to perform non-linear operations. Client-aided HE has several benefits. For DNNs, Client-aided HE supports unmodified networks with arbitrary activation functions. Plaintext client operations also repack the ciphertext vectors and refresh the noise budget, mostly eliminating the arithmetic depth bound and obviating large, costly HE parameters. These benefits, however, increase the cost to the client. The client must frequently decrypt, re-encrypt, and communicate data. These client costs are the key impediment to client-aided HE. CHOCO takes a new approach, optimizing client-aided HE to *minimize client costs* through HE algorithm design and hardware acceleration.

2.2 Motivation for Client-Aware Optimization

Encrypted offloading of complete programs often requires expensive bootstrapping operations, infeasibly large parameters, and inefficient algorithm adaptations to obey the limitations of HE [8, 20, 28, 31]. Client-aided implementations circumvent these limitations. Unfortunately, client-aided HE imposes a high burden on client devices, motivating CHOCO's client-optimized design.

Client-aided HE requires the client to frequently communicate, decrypt and re-encrypt intermediate results to refresh the noise budget, repack vectors, and perform plaintext non-linear operations. We evaluate client costs for representative DNN workloads using a software client-aided prototype. This system (which Section 3 describes in detail) uses Gazelle's *server*-optimized HE algorithms [36] and SEAL's [58] default parameters. We measured image classification time for four DNN models running on the client using TFLite[1]. We then repeat the measurements using our Client-aided HE system. We find that HE client costs in both communication and computation scale with DNN complexity. Server costs are consistently high. Existing, complementary work aims to reduce server costs with HE algorithms [6, 8, 36, 41, 47, 55] and hardware [59, 63, 64, 70] but neglects client costs. In contrast, CHOCO's primary goal is to reduce the *client*'s time and communication costs.

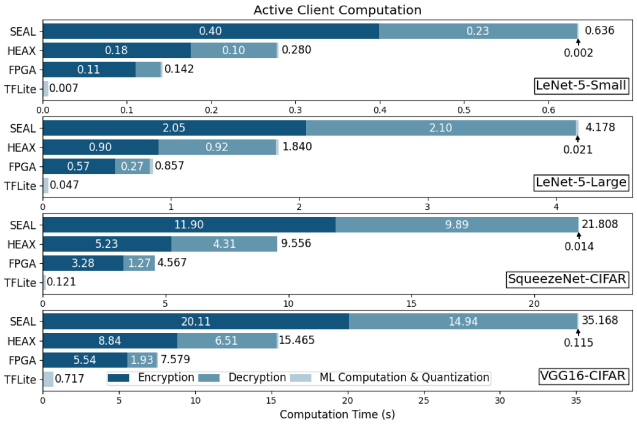

Figure 2: Characterization of active client compute time for single image DNN inference. Default SEAL encrypted inference and local TFLite versions bound versions using limited hardware support from HEAX [59] & FPGA [46].

The main client compute costs are encryption and decryption. Figure 2 breaks down client compute time: >99% of client compute is in HE operations, not application operations (i.e., activations &

quantization). The plot also shows that existing hardware and FPGA support, which primarily focuses on polynomial multiplication & Number Theoretic Transform (NTT) [46, 59, 63, 70], is insufficient to address client costs. We computed the best-case speedup with hardware by scaling our software runtime of supported operations by the reported speedups in HEAX [59] and a state-of-the-art encryption FPGA [46]. Software profiling reveals that polynomial multiplication and NTT account for only 60% of the total time in SEAL's encryption and decryption operations. Even using server-focused hardware [59, 63, 70] for these operations, client encryption and decryption time far exceeds local compute time.

A key insight in CHOCO is that in the client-aided model, client-side encryption and decryption are *not* an insignificant one-time cost; instead, they define the client's critical path, demanding optimization and acceleration.

3 CLIENT-OPTIMIZED CLIENT-AIDED HE

CHOCO is a client-optimized system model and implementation for client-aided HE offloading, partitioning work between the client and the offload device (e.g., layer-wise for DNNs) as depicted in Figure 3. We introduce *rotational redundancy*, a new HE algorithm for permuting a vector of encrypted data, which allows for smaller parameter selections and correspondingly smaller ciphertexts.

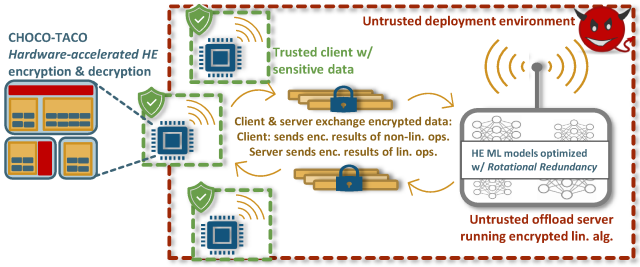

Figure 3: System architecture for CHOCO. A resource-constrained sensor device and an untrusted offload device communicate via ciphertexts to collaboratively and securely process sensitive data.

3.1 Security Assumptions

The CHOCO model assumes a resource-constrained client device and a more computationally capable, but untrusted, shared offload server. Typical of HE systems, we assume a "semi-honest adversary model" [36, 41, 48] for the offload device: the adversary may be curious about the input data, but the system is trusted to faithfully perform the specified operations. In contrast to computationally expensive MPC protocols, CHOCO does not make any attempt to hide data on the offload device from the client, including pre-trained ML model data. Rather, CHOCO prioritizes optimizations exclusively for the client, for both performance and privacy. It uses HE alone to protect sensitive client data generated by IoT devices. If server data privacy is strictly required, CHOCO's HE algorithm optimizations and hardware support also provide client benefits in HE-MPC protocols.

3.2 Selecting Efficient HE Parameters

Choosing appropriate HE parameters is a key challenge to encrypted application development [19, 20]. Parameter selection determines ciphertext size and noise budget, which together influence computation, encryption, decryption, and communication costs in client-aided HE. An application can have the same security level with different parameters. Thus, for a required security level, CHOCO selects parameters to minimize ciphertext size through aggressive 4-bit input quantization (BFV), optimal operation scheduling via the state-of-the-art EVA HE compiler [19] (CKKS), and novel encrypted algorithm optimization (both).

3.3 HE Algorithm Optimization

CHOCO implements several HE algorithms for encrypted linear algebra that support the HE applications described in Section 5. CHOCO's algorithms are inspired by existing algorithms in Gazelle [36] and LoLa [8] and target the SEAL HE library [58] for BFV and CKKS (via EVA [19]). In CHOCO, all encrypted computation executes on the offload server.

As Section 2.1 introduces, an HE algorithm must assemble data into a polynomial (vector) to be encrypted and computed on. As proposed in Gazelle and LoLa, *packed* HE algorithms require permutations to rearrange input elements in an encrypted vector. Encrypted permutations align encrypted values so that subsequent addition and multiplication operations implement an intended linear transformation, such as matrix multiplication or strided convolutions [36]. Because arbitrary permutations require a series of encrypted rotations and multiplications [28], they quickly consume the limited noise budget of a ciphertext. The depletion of a ciphertext's noise budget causes it to become undecryptable. Thus, arbitrary encrypted permutations are a severe problem for systems, like CHOCO, that prioritize small ciphertexts.

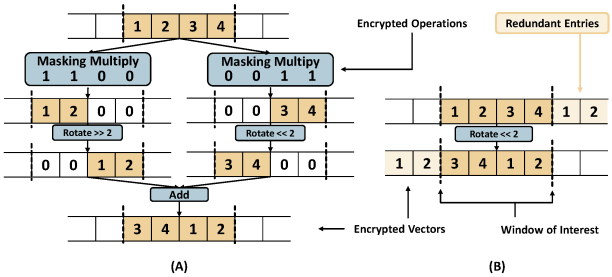

Figure 4: Encrypted windowed rotation using arbitrary permutation (A) and rotational redundancy (B). Rotational redundancy is a novel way of packing input values before encryption that allows for fewer encrypted operations.

Rotational Redundancy. CHOCO avoids accumulating excessive noise by introducing rotational redundancy to perform some encrypted permutations. The technique targets *windowed rotation* permutations that rotate the elements in a sub-range of a vector, wrapping elements around from the top of the sub-range to the bottom and vice versa. This is in contrast to standard HE rotations (Table 1) that rotate an entire ciphertext. Figure 4 (A) shows an

implementation of windowed rotation using a standard HE permutation that requires both masking multiplies and full ciphertext rotations. In comparison, Figure 4 (B) shows the same windowed rotation performed using rotational redundancy.

Rotational redundancy is a novel way of packing input vectors before encryption such that values set to wrap around during a windowed rotation are appended on either side of the window of interest. Appending these redundant values avoids expensive masking multiplications and instead requires a single, relatively low-cost encrypted rotation to achieve a complete windowed rotation. This novel technique packs vectors with the unique goal of reducing the number of encrypted operations required to run a computation. This goal is particularly relevant to client-aided HE because the client decrypts, unpacks, and repacks ciphertexts after each set of encrypted, linear operations. Any values outside the window of interest can simply be discarded from the plaintexts by the client and new values can be explicitly packed for the next operation. The goal of rotational redundancy is a contrast to LoLa[8] and other prior work that pack vectors for multi-operation compatibility on the server. Without client interaction, the output of one encrypted vector operation must be packed appropriately for the next encrypted operation to receive as input.

Table 4: Noise Budget: The initial noise budget of a ciphertext varies with different parameter selections. The noise budget remaining after a single rotation versus an arbitrary permutation with masking is also contrasted. Rotational Redundancy eliminates masking multiplies and therefore has noise behavior synonymous with just a single rotation.

Parameters	Noise		
N, $\log_2 t$, $\{k\}$	Initial	Post-Rotate	Post-Permute
8192, 20, {58,58,59}	68	66	42
8192, 23, {58,58,59}	62	59	33
8192, 28, {58,58,59}	52	50	18
4096, 16, {36,36,37}	33	31	12
4096, 18, {36,36,37}	29	26	5
4096, 20, {36,36,37}	25	22	0

Although the optimization reduces the density of useful input values in a ciphertext, the amount of redundancy required depends on the amount of rotation required, which is typically a small fraction of the total vector size. Rotational redundancy trades the use of more space in an encrypted vector for a slower depletion of ciphertext noise. The optimization thus enables the use of better HE parameter selections that permit smaller ciphertexts; Table 4 quantifies these benefits. Even within the same ciphertext size, dictated by N and k, the use of rotational redundancy allows for a larger BFV plaintext modulus t. This increased capacity enables larger quantization bitwidths and more encrypted accumulation. Recall that the same security guarantees can be achieved with different parameter selections (Table 3) and vectors are packed *before* encryption. Therefore, rotational redundancy has no impact on the security guarantee of the ciphertext. Ultimately, the benefit of rotational redundancy in reducing ciphertext size is witnessed by the client as greatly reduced computation and communication.

Applying Rotational Redundancy in CHOCO. We applied rotational redundancy in all of our applications that required windowed rotations, and it provided substantial benefits. We implemented several DNN image classifiers (Section 5) in which a ciphertext vector is the concatenation of a vector per channel. Convolution requires windowed rotation within each channel. Based on the amount of rotation, CHOCO packs each channel of an image with sufficient redundancy to keep values aligned throughout the encrypted convolution. CHOCO then stacks the redundant channel vectors into evenly-spaced, power-of-two-sized slots in the final ciphertext vector. With this redundant, stacked packing of channels, all elements can be properly aligned for convolution without any arbitrary permutations or masking multiplies. Alignment requires *only* simple encrypted rotations. Using rotational redundancy, convolution is achieved with optimal multiplication efficiency - a single multiplication of the weights with the inputs.

Algorithmic optimization, including rotational redundancy, and CHOCO's client-focused HE parameter selection provide a substantial reduction in costs to the client. For encrypted DNN computations, CHOCO computes on ciphertexts with only 2 prime residues, which is a 50% reduction in ciphertext size compared to SEAL's default with $N = 8192$. Half of this improvement — eliminating an entire RNS residue — comes from rotational redundancy alone. As Section 5 shows, ciphertext size reduction results in substantial client benefits.

4 HARDWARE ACCELERATION

CHOCO-TACO is a hardware accelerator for homomorphic encryption and decryption operations, designed for client-aided HE. Recall from Figure 2 that accelerating NTT/INTT and polynomial multiplication alone insufficiently reduces the dominant client costs of encryption and decryption. CHOCO-TACO, instead, accelerates all of the component sub-operations of HE encryption and decryption, virtually eliminating these client costs.

4.1 HE Encryption

$$Enc([P0, P1], m) = ([\Delta m + P0u + e1]_q, [P1u + e2]_q)$$
$$where\ u \xleftarrow{\$} R_2\ and\ e1, e2 \leftarrow \chi \tag{2}$$

CHOCO-TACO targets asymmetric encryption and decryption in BFV and CKKS. Equation 2 [39] is BFV's encryption kernel, where m is a message to encrypt, Δm is the encoded message, $P0, P1$ are public keys, $u, e1, e2$ are vectors of randomly sampled numbers, and $[]_q$ denotes modulation by the coefficient modulus q. Figure 5 diagrams the RNS implementation of equation 2 from SEAL [58]. The algorithm first encrypts the value zero by combining randomly sampled vectors with the public encryption keys via polynomial multiplication and addition. The algorithm then encrypts Δm by adding it to the encrypted zero to produce the final ciphertext.

4.2 Accelerator Architecture

CHOCO-TACO is a straightforward, parallel, pipelined hardware mapping of the sub-computations used in BFV and CKKS encryption and decryption. Figure 6 shows the full encryption and decryption accelerator, including datapath arcs concretely illustrating the BFV functions. The design's main *modules* are Pseudo-Random

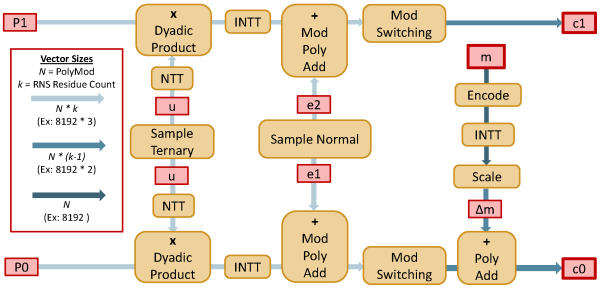

Figure 5: Pipeline of the BFV encryption operation to sample random noise, perform polynomial arithmetic, scale the input message, and ultimately create a ciphertext of two polynomials, each in RNS form. [39, 70]

Number Generation, Polynomial Multiplication, Polynomial Addition, Modulus Switching, and Message Encoding. Each module contains memories and *functional blocks*. Within each block, *Processing elements* carry out a computation on individual coefficients. Both functional blocks and processing elements are pipelined and replicated for data parallelism.

Pseudo-Random Number Generation. CHOCO-TACO has a dedicated PRNG module that implements the Blake3 [50] cryptographic hash. The CHOCO-TACO configuration in Figure 6, requries 565 MB/s of random values at peak and 201 MB/s on average, drawn from either a ternary or normal distribution and represented in RNS form. We modified SEAL's software to also use Blake3 instead of Blake2, providing consistent performance increases to both CHOCO-TACO and the baseline.

Polynomial Multiplication. CHOCO-TACO includes a module for polynomial multiplication of two polynomials, (i.e., u and each public key). Like prior work [46, 59, 63, 70], the module converts inputs to NTT form, computes element-wise dyadic products, and converts results back to a polynomial via INTT. CHOCO-TACO uses iterative NTT and INTT modules [46, 57, 59], performing pipelined, SIMD memory accesses, following a butterfly dataflow pattern.

Polynomial Addition and Modulus Switching. The polynomial addition module performs coefficient-wise addition, passing results to the modulus switching module. Modulus switching runs a series of modular multiplications and reductions, which removes the key-prime residue from the RNS encoding, resulting in $k-1$ polynomials. Modulus switching is the only operation that requires interaction across RNS residues, which precludes straightforward data parallelism across residues.

Message Encoding. The encode/decode module includes a pair of small NTT and INTT blocks. The encoder computes each coefficient modulus using the plaintext modulus t, and reorders coefficients into plaintext slots. The hardware scales and converts the encoded message into RNS with $k-1$ residues only. A polynomial addition block adds the result to the $k-1$ intermediate residues of $c0$.

Memory. Each module includes embedded SRAM to store inputs and ouputs. All modules except NTT accept a stream of inputs/outputs. The optimal size of their SRAM buffers is empirically found to be sub-1kb (Section 4.4). NTT and INTT algorithmically operate on a full polynomial, requiring their SRAM to match the full polynomial size, e.g., 64kB with $N = 8192$ and $k = 3$. We model memories using Destiny, ported for single-reader, single-writer 64-byte data accesses.

Parallelism. CHOCO-TACO exploits pipelining and data parallelism of independent RNS residues and coefficients. CHOCO-TACO processes a polynomial's RNS residues in parallel in multiplication and addition, using full replicas of these operations' modules. RNS data parallelism eliminates the need to buffer large random vectors throughout an execution. Instead, $u, e1$, and $e2$ are immediately consumed and distributed to all residues in parallel as they are generated. Figure 6 shows RNS residue parallelism through layering. Within an RNS layer, thousands of coefficients per polynomial afford data parallelism. A key accelerator design parameter is the degree to which each module exploits this data parallelism with repeated processing elements. Within power and area limitations, CHOCO-TACO can grow a module's blocks, sizing memories to match, to enable higher coefficient processing throughput.

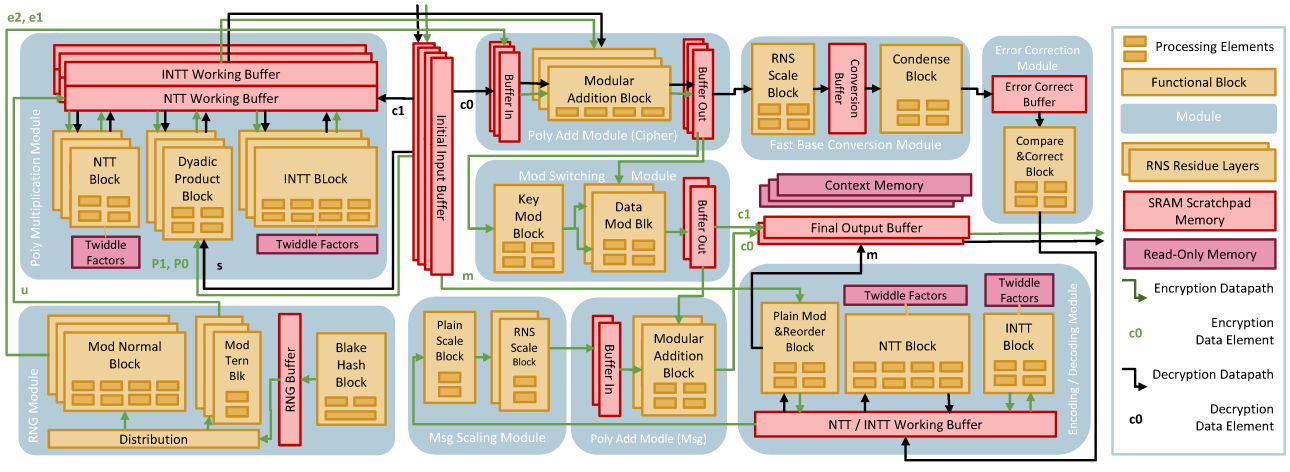

Figure 6: Specialized architecture to support encryption & decryption for $k = 3$ coprime residues

4.3 Encryption Operation Example

BFV encryption generates an encrypted "zero" which is then added to the scaled message to produce the final ciphertext. To start, the accelerator samples N bytes from the PRNG according to a ternary distribution and stores them in the NTT working buffer as u. Polynomial multiplication with the first public key can then begin. The NTT block produces the NTT of u in place, and the value becomes the input to the dyadic product block. Both ciphertext components ($c0$ and $c1$) use the same NTT encoding of u, allowing it to remain in the NTT working buffer throughout encryption. The other input to the dyadic product block comes from the accelerator's input buffer, which software initializes with the NTT-transformed residues of $P1$. Dyadic multiply produces the element-wise product of u and $P1$ in the INTT working buffer, which the INTT block then processes in place.

In parallel with the multiplication of u and $P1$, the PRNG starts producing $e2$, a sequence of 8-byte, normally distributed samples. These are sent through a small buffer to the cipher addition module. The results of the completed INTT operation are then streamed in as the second input to this module. Element-wise addition is performed, and the results are sent, via another small streaming buffer, through the modulus switching module as $c1$. As one component of the final ciphertext, $c1$ is then output to the CPU's host memory.

Meanwhile, the accelerator begins encoding the input message and producing $c0$ in the polynomial multiplication module. Computing $c0$ reuses the NTT of u, performing element-wise multiplication with public key $P0$. The accelerator samples a sequence of normally-distributed 8-byte $e1$ values (like $e2$) and adds them element-wise with the INTT transformed product of u and $P0$. The result is a partially-computed version of $c0$ which again undergoes modulus switching. The last step is the polynomial addition of the encoded input message with $c0$. This final sum, together with $c1$, completes the final ciphertext.

4.4 Design Space Exploration

With abundant parallelism available, it is important to optimally allocate resources within the modest constraints of IoT client devices. We explore this design space of the CHOCO-TACO hardware using a custom simulation infrastructure. The hardware model captures the effects of parallelism and pipelining and estimates time, power, area, and energy. We implemented individual hardware components in RTL and synthesized them with Cadence Genus in a generic 45nm technology node. To model memory, we used Destiny [53], modeling SRAMs using its aggressive wire technology, optimized for read energy with 8-word, 64-byte, memory accesses. The access latency of our energy-optimized memories limits clock frequency, and we clocked the design at 100 MHz.

We quantified the tradeoff of area, time, and power, with a systematic exploration of the CHOCO-TACO hardware design space. Using our simulator, we swept across 31,340 different architectural configurations. For each configuration, the study assessed power (leakage & average dynamic), area, energy and compute time for a single encryption operation. Results from the design exploration are presented in Figure 7. Overall, the design space shows significant variation in power and area, with a marked Pareto Frontier along which power, time, and area balance. We selected an operating point for CHOCO-TACO by limiting power to 200 mW, and choosing the smallest design that had a run time within 1% of the optimal time (and energy). The chosen configuration has 19.3 mm^2 area and consumes .1228 mJ to perform a single encryption in .66 ms for $N = 8192$ and $k = 3$. Figure 6 illustrates this configuration.

4.5 Scalable Benefits of CHOCO-TACO

We evaluated the benefit in time and energy of CHOCO-TACO for encryption compared to a software encryption baseline and found that hardware support provides substantial improvements across a range of HE parameter settings. For larger N, NTT and INTT working buffers are expanded to hold all N elements of a single polynomial. More "layers" are added for larger k values. Pipeline logic and sub-1kb streaming SRAM buffers within a single CHOCO-TACO layer remain the same regardless of parameter selection because they operate element-wise.

Results are shown in Figure 8 compared to a software baseline of 100 SEAL encryption operations running on our IMX6 hardware (Section 5.2). We omit baseline data for the (N,k)=(32768,16) parameter setting because the IMX6 board does not have enough memory to encrypt data with these parameters. Notably, this configuration with its prohibitive memory requirements is not uncommon in existing encrypted inference solutions [8, 20].

For the CHOCO (8192, 3) configuration, CHOCO-TACO provides an improvement over the software baseline of 417× in time and 603× in energy. The data also show a performance scaling trend: With hardware support, encryption time scales up directly with N, while software scales up with both N and k. The scalability benefits

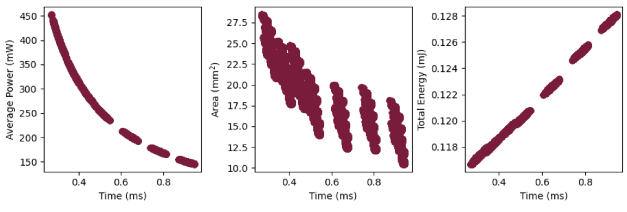

Figure 7: Design space for encryption hardware with respect to power, area, and energy. Parallelism tradeoffs are available in multiple dimensions at each stage of encryption.

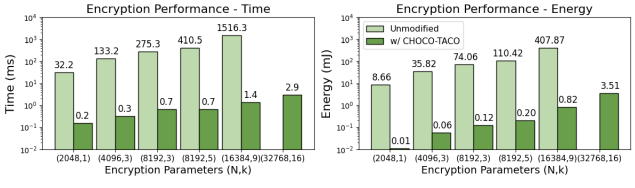

Figure 8: Logarithmic comparison of time & energy across varying encryption parameters (N, k) between the CHOCO-TACO architecture presented in Figure 6 and a 528MHz IoT device without dedicated hardware support.

come from parallelism in the accelerator architecture: Replicated modules process k independent RNS residues in parallel.

Overall, CHOCO-TACO provides up to 1094× and 648× savings in time and energy, respectively, with consistent gains across all HE parameter settings.

4.6 Decryption Support

BFV decryption is operationally very similar to encryption. Equation 3 shows decryption mathematically.

$$Dec(sk, [c0, c1]) = \left[\left\lfloor \frac{t}{q}[c0 + c1sk]_q \right\rceil\right]_t \quad (3)$$

Figure 6 shows the flow of control and data for decryption with black lines. Decryption requires a few additional hardware components, but reuses the existing polynomial multiplication and addition modules to process $c1$, the secret key sk, and $c0$. After addition, these intermediate results undergo base conversion and error correction. At that point the message need only be decoded. Decoding uses the message encoding module to perform NTT and take the plaintext modulus t of each coefficient. The result is the decrypted message, which the hardware conveys to the CPU's memory.

Again, we compare hardware acceleration to our IMX6 SEAL baseline. Decryption sees less benefit from hardware acceleration than encryption, with only a 125× speedup over software, taking .65 ms for the (8192,3) CHOCO parameter selection. This decrease in speedup is contributed to limited parallelism because decryption operates on only a single polynomial at this parameter selection.

4.7 CKKS Support

Leveraging the underlying similarities in both encryption schemes, most importantly the RNS polynomial representation of ciphertexts, the BFV hardware presented in Figure 6 can be modified to support CKKS encryption and decryption as well. Namely, the same hardware modules are utilized in both schemes, but for CKKS an additional datapath has to be added to route intermediate results through the modules in a different order.

Software profiling of the CKKS functions reveals that 59% of CKKS encoding and 46% of decoding are NTT and INTT operations, respectively. These portions can also be accelerated via the existing hardware. The remaining portions of CKKS encoding and decoding require processing of complex conjugates. We leave these parts un-accelerated in software.

Overall, 95% (56%) of CKKS encrypt & encode (decrypt & decode) execution time is supported by the previously presented hardware with an additional datapath. For these portions we assume speedups proportional to those witnessed for BFV. Using this methodology, we find that hardware support for CKKS reduces encrypt & encode time by 18× from 310 ms to 18 ms and decrypt & decode by 2.3× from 37 ms to 16 ms over our IMX6 client baseline. Additional speedup could be expected if additional modules were incorporated to cover the CKKS operations in their entirety.

5 EVALUATION

We evaluate CHOCO, comparing directly to prior work and showing substantial benefits from architectural acceleration and algorithmic improvements. The main result is that CHOCO reduces communication costs by up to three orders of magnitude over state-of-the-art privacy-preserving computation solutions. CHOCO-TACO provides an additional 28.6× speedup for client compute over limited client-side support in existing FPGA solutions [46, 59]. The benefits apply to both BFV and CKKS and a variety of applications. We ultimately show that CHOCO's *client*-focused optimizations, largely neglected by prior work, provide significant end-to-end benefits.

5.1 Applications

We fully developed several client-aided encrypted computing applications. We built four neural networks in BFV that are comparable to or larger than encrypted DNNs from prior work [8, 20, 22, 36]. Using CKKS, we implemented distance-based algorithms — K-Nearest Neighbors (KNN) and K-Means clustering — and PageRank. We are aware of no prior implementation of the latter applications, and we developed several algorithmic variants to study their efficiency.

Deep Neural Networks. We implemented four image classification DNN inference models in BFV, which we list in Table 5. Consistent with existing literature , the LeNet variants operate on MNIST data [40], and the other, larger networks classify CIFAR-10 images [38]. We trained the DNNs on unencrypted data using standard quantization-aware training in Tensorflow 2.2.0 [1]. Each experiment utilizes client-aided HE to run inference on a single image. All linear layers are performed over encrypted data on the server using HE, while the client computes all non-linear operations locally on plaintext data. Encrypted intermediate results are communicated at layer boundaries. As mentioned in Section 2.1, this client-aided implementation allows for unmodified networks with unbounded depth that would normally be prohibited in a server-only encrypted inference implementation.

Table 5: Neural Networks used for system evaluation

Network	# Layers				MACs	% Acc.			Mod. Sz. (MB)		Comm.
	Cnv	FC	Act	Pl	($\times 10^6$)	Float	8b	4b	Float	4b	(MB)
LeNetSm[24]	2	1	2	2	0.24	99.0	94.9	93.8	0.02	0.01	0.66
LeNetLg [69]	2	2	3	2	12.27	98.7	97.2	96.4	8.22	2.07	2.6
SqzNet [17]	10	0	10	3	32.60	76.5	74.0	15.0	0.57	0.16	13.8
VGG16 [42]	13	2	14	5	313.26	70.0	66.0	21.0	56.40	14.13	22.2

PageRank. We implemented encrypted PageRank algorithms in both BFV and CKKS. Exploiting similarities in ciphertext construction, this demonstrates the generality of encrypted algorithms to both schemes. The PageRank algorithm relies completely on linear algebra operations that can run entirely in encrypted space. For continuous encrypted operation, we use alternating sparse and dense dot-product representations from LoLa [8]. In addition to continuous encrypted execution, we also consider a client-aided version of the algorithm in which the client periodically decrypts and re-encrypts ciphertexts to refresh their noise budget.

Distance-based Algorithms. We implemented K-Nearest Neighbors (KNN) and K-Means clustering, both using encrypted distance calculations in CKKS. We modify their Euclidean distance kernel to use a simple summation of squared differences, eliminating the square root and enabling offloaded calculation on an HE server. The client handles only classification data and newly collected (e.g.

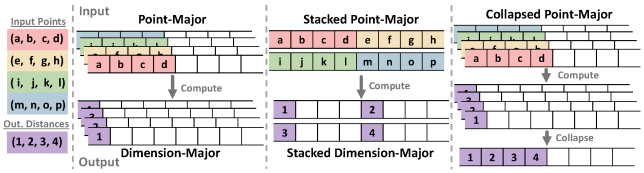

Figure 9: Encrypted vector packing for distance calculations.

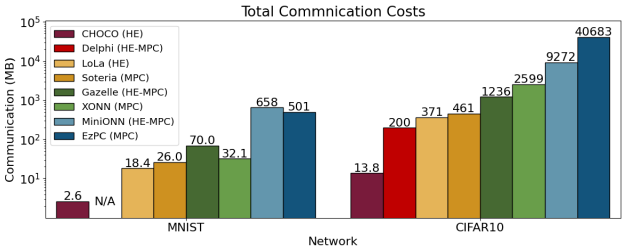

Figure 10: Comparison of total communication requirements to perform single image inference via the Lenet-5-Large (MNIST) and SqueezeNet (CIFAR-10) implementations in CHOCO and comparable networks in several state of the art privacy-preserving DNN protocols.

new point in KNN) or computed (e.g. updated K-Means centroids) coordinate data. The client sends new coordinate data to the server, which runs encrypted distance calculations. The client decrypts the results and performs algorithm-specific non-linear operations, i.e. `min()`. K-Means iterates client-server interaction until convergence. In KNN, classifying a new point requires just a single interaction. A key advantage of HE for distance-based algorithms is that the server can store and query against points from many clients, providing the client with results unavailable to a local computation.

For KNN and K-Means, encrypted distance calculations require one-to-many interactions between a point or centroid, and all other points. For each pairwise calculation, *all* dimensions of the two points interact and are accumulated. This follows the same access pattern as matrix-vector multiplication. We implement five variants of the distance calculation kernel, inspired by the matrix-vector products from LoLa [8]. As shown in Figure 9, point(dimension)-major packs vectors with a single point (dimension). For small inputs, stacked variations include multiple input points (dimensions) in one ciphertext. With the exception of the collapsed algorithm, which packs both its input and output as point-major, point-major inputs produce dimension-major outputs and vice versa.

5.2 Methodology

Consistent with our target of resource-constrained IoT clients, we perform baseline client evaluations for software running on an NXP IMX6 evaluation kit with an ARM Cortex-A7 CPU at 528MHz, 32/128 kB of L1/L2 cache, and 4 GB DDR3L SDRAM. We estimate power and energy using an average power characterization (running Dhrystone) of 269.5 mW from the manufacturer's Application Note AN5345 [66]. Server results are collected from an Intel Xeon processor running at 2.50 GHz. We follow the methodology from Section 2 to estimate acceleration from existing FPGA implementations and use the hardware configuration from Figure 6 to model CHOCO-TACO acceleration. We compute time and energy savings for client computation by counting the number of encryption and decryption operations necessary to run the application. Counts are then multiplied by the hardware cost of each operation as presented in Sections 4.4 (BFV) and 4.7 (CKKS).

5.3 CHOCO Reduces Communication

The algorithmic optimizations presented in Section 3.3, including rotational redundancy, minimize noise growth to enable smaller parameter selections and correspondingly smaller ciphertexts. As discussed in Section 2.1, all of the networks included in Table 5 can be evaluated in CHOCO using ciphertexts with no more than 8192 elements ($N = 8192$). This is in contrast to existing HE solutions [8, 20, 22, 36, 41, 55] which commonly use ciphertexts of 16 or even 32 thousand elements. By eliminating unnecessary prime residues, CHOCO further reduces ciphertext size by another 50% over SEAL's default parameters at $N = 8192$.

CHOCO's reduction of both ciphertext size and quantity directly translates to improvements in server runtime, client responsibility (Figure 12), and communication overhead. Figure 10 demonstrates the communication improvement over several state-of-the-art privacy-preserving DNN protocols. Comparisons are evaluated between the CHOCO implementations of Lenet-5-Large and SqueezeNet and comparable networks performing MNIST and CIFAR-10 single-image inference, respectively. They include communication for both offline preprocessing and online computation. Although the networks evaluated in this work are substantially larger (more model parameters) in all cases, CHOCO outperforms existing protocols by up to three orders of magnitude. In the worst cases, many MPC protocols manipulate single values, not vectors, already giving an advantage to vector HE schemes, like the ones CHOCO targets. Even compared to LoLa[8], a *non*-client-aided encrypted inference protocol with complete HE offloading, benefits are witnessed because of the smaller parameter selections and more efficient ciphertext packing. For the most closely comparable protocol, namely Gazelle [36], CHOCO still provides a nearly 90× improvement in communication overhead. This reduction dramatically reduces end-to-end latency, especially for IoT devices often communicating over low bandwidth channels such as Bluetooth.

5.4 CHOCO Optimizes for Clients

Within a single encrypted algorithm, there can be many variations and alternate approaches. Prioritization of the client versus the server can lead to divergent implementation decisions. These trade-offs are evaluated using the five variations of distance calculations presented in Section 5.1. Figure 11 presents results for several representative dimension-point pairs in the context of KNN.

For small dimensions the stacked algorithms allow more data to be packed, communicated, and processed with fewer ciphertexts. This high ciphertext utilization produces favorable results across all three components (server time, client time, and communication) of the collaborative algorithm. However, for a client-optimized implementation it is clear that the collapsed point-major implementation is the best choice. Unlike the other algorithms which trade high

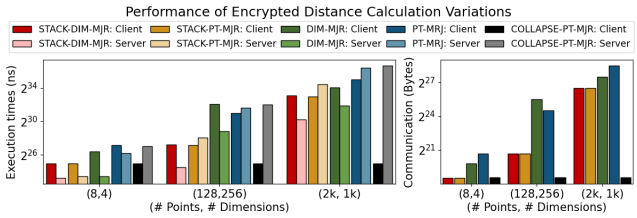

Figure 11: Performance tradeoffs for encrypted distance calculations with varying vector packing and encrypted operations using CHOCO. Results include server execution time, client execution time, and communication.

ciphertext utilization in the input for low utilization in the output ciphertext(s), i.e. few input ciphers for many output ciphers and vice versa, the collapsed algorithm both produces and consumes high-utilization ciphers. This is because it employs an extra round of masking multiplications and accumulations to combine results from several ciphers before communicating just a single cipher. This approach places extra work on the offload server, and would therefore likely be disregarded in a server-optimized system. However, from the client perspective it dramatically reduces the number of ciphertexts being communicated and processed.

5.5 CHOCO-TACO Accelerates Client Compute

We evaluate our hardware setup running single-image inference. We compare it to both the software optimized baseline and the baseline equipped with limited hardware support from existing FPGA solutions. The software optimized baseline includes the algorithmic optimizations of CHOCO, namely rotational redundancy, and already demonstrates an average 1.7× improvement over the SEAL baseline software with standard permutations and default parameter selections. A baseline that runs local inference on the ARM Cortex-A7 CPU using TensorFlow Lite (TFLite) software is also included as a lower-bound. Figure 12 reports the resulting execution times, breaking down the total time into its constituent components. We assume that the time for client computation and quantization, including ReLu and Pooling, stays the same across all implementations. The data show an average speedup of 121× over the optimized software baseline for computations on the client, which is consistent with the 417× and 125× speedups observed for encryption and decryption, respectively.

The data clearly show that encryption and decryption are the client bottleneck. Even with limited hardware support for polynomial multiplication and NTT, these cryptographic operations for a client-aided protocol are still 14.5× slower on average than computing the full network locally with TF Lite.

Comprehensive hardware acceleration for encryption and decryption is imperative. CHOCO-TACO recognizes this and uses an optimal allocation of compute resources, minimal buffering, tightly integrated memories, and multiple levels of parallelism to address the remaining 40% of computation. With the acceleration provided by CHOCO-TACO, the active client computation time in client-aided encrypted DNN inference becomes 2.2× faster on average than local inference.

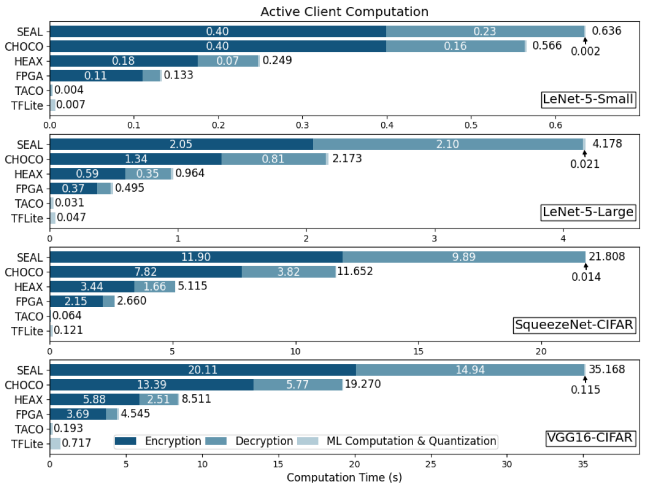

Figure 12: Characterization of active computation time on the client for DNN inference, extended from Figure 2. CHOCO includes software optimizations only. TACO includes comprehensive hardware acceleration.

5.6 CHOCO-TACO Enables General Algorithms

With the configuration depicted in Figure 6, CHOCO-TACO supports encryption and decryption for parameters sets with $N \leq 8192$ and $k \leq 3$. In the following evaluation, centered on our client-aided PageRank, we find that this limitation on parameter selection does *not* limit application selection.

Unlike DNN implementations which require communication at pre-determined locations because of non-linear ReLu operations, the client-aided PageRank algorithm presents optional opportunities for communication after each iteration. We exploit this flexibility to determine client-optimal implementations and their corresponding parameter selections. Recognizing the correlation between communication and client responsibility, we use communication as our evaluation metric.

For a given number of total PageRank iterations, we allow that total to be achieved via any suitable combination of fully encrypted iteration sets. For example, a total of 24 iterations can be achieved entirely in encrypted space with a single set of 24, or it can be done with two smaller sets of 12 iterations each and a round of communication in between. In the extreme case, communication, and corresponding noise refresh, is performed between each iteration. With less frequent communication, one must use larger parameters to achieve more continuous encrypted iterations, and vice versa.

Figure 13 shows the *total* communication (y-axis) necessary to achieve a variety of *total* PageRank iterations (x-axis). Each dot represents a unique combination of encrypted iteration sets, i.e. 1 set of 24, 2 sets of 12, etc. It is found that CKKS can achieve the same encrypted iteration set with smaller parameters, compared to BFV. This leads to reduced communication across the board when using CKKS.

Although the PageRank algorithm allows for continuous encrypted operation with larger parameter sets, it is found that frequent communication of smaller ciphertexts is consistently favorable for both schemes. The client-aided implementations are

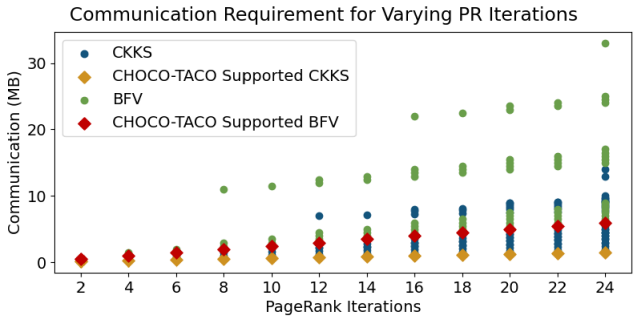

Figure 13: Client-aided encrypted PageRank implementations in both BFV and CKKs with varying frequency of communication between iterations

counter-intuitively more efficient than the fully-offloaded server-only alternatives. These optimal client-aided combinations, highlighted with red and gold diamonds in Figure 13, are synonymous with the combinations supported by CHOCO-TACO because they use parameter sets with $N \leq 8192$ and $k \leq 3$. Hardware support for these parameter sets thus provides a synergy, not a limitation, with client-aided encrypted algorithm optimization.

This conclusion is further recognized in both the DNN and distance-based algorithms as well. The LoLa results in Figure 10 show very high communication costs even though it is a non-client aided protocol. Furthermore, *all* of the CHOCO DNN and distance-based implementations, and their corresponding results, are achieved with hardware compatible parameters.

5.7 CHOCO is Competitive w/ Local Compute

To understand the end-to-end benefits, we study a reference implementation of CHOCO that communicates between the client and the offload server using 10 mW Bluetooth communication at 22 Mbps[45]. Timing and energy results follow analytically from the data communication requirements, included in Table 5, of each network. End-to-end time and energy results are compared against the TFLite baseline in Figure 14. In a full implementation communication time begins to dominate. For low-power, low-data-rate protocols such as Bluetooth, communication presents a 24× average time overhead compared to local compute. However, in small devices battery preservation often outweighs the need for fast compute. In energy consumption, CHOCO is competitive with the TFLite baseline. For VGG, the largest and most complex of the DNNs evaluated, CHOCO earns up to a 37% end-to-end energy savings.

This data shows that intentional client-aware optimizations are essential to bring encrypted computation to resource-constrained IoT clients. Although communication remains a key bottleneck in time and energy, the algorithmic optimizations of CHOCO reduce this cost by up to three orders of magnitude. Furthermore, our hardware support accelerates the entire encryption and decryption computation, driving its cost down, eliminating it as the bottleneck, and making CHOCO feasible. For the first time, this dramatic reduction makes client time and energy competitive with local inference, even displaying the possibility of end-to-end gains.

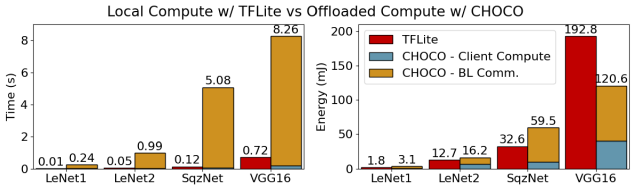

Figure 14: Client execution time & energy for single image inference via local compute using TFLite software and offloaded compute using a full hardware-software reference implementation of CHOCO-TACO using Bluetooth communication.

5.8 Workload Dependent Benefits of CHOCO

Different workloads see different benefit from the CHOCO model, due to the varying rates of computation and communication required by each workload. Using CHOCO with Bluetooth communication, VGG sees substantial performance and energy improvements, while SqueezeNet sees a break-even or loss. We performed a microbenchmarking study to evaluate this influence of workload structure. We constructed workloads with a variety of different convolutional DNN layers of different dimensions. The structure of the convolutional layers varies the number of multiply-accumulate (MAC) operations performed by each layer, as well as the amount of communication required to send and receive the ciphertexts that contain each layer's inputs.

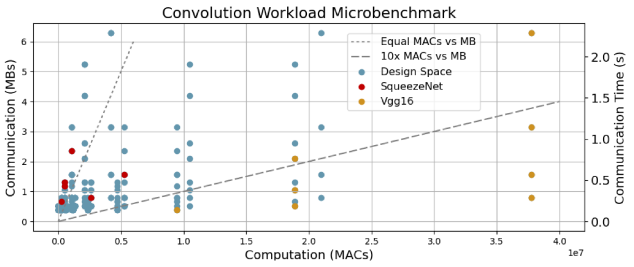

Figure 15: Communication vs Computation for Convolution Layers of a DNN with different parameters

Figure 15 shows the results of this study, plotting these microbenchmark convolution points, as well as each of the layers from VGG and from SqueezeNet. For the microbenchmark points, we varied image size from 2 to 32 by powers of two, varied image channel values from 32 to 512 by powers of 2. Following the implementations of SqueezeNet and VGG16, we used filter sizes of 3 or 1. Many different layer configurations will use the same number of MACs (x-axis) but vary in the amount of communication required to send and receive intermediate results (y-axis). The data show that workloads like VGG (which are likely to see its same energy benefits) are ones that maximize the number of MACs per MB of communication. One way to do this is to use larger filters, as those contribute to more MACs, more classification power, but have no impact on communication with the client. Workloads like SqueezeNet (which are likely to see a break-even or overhead in costs) are ones that have fewer MACs per MB of communication.

These data provide two main benefits in interpreting CHOCO. First, the data show that a quick analytical comparison of computation (MACs) versus communication (MBs) per layer helps an application designer decide if their DNN application will see an energy benefit in the CHOCO client-aided model. Second, the data point to an opportunity for future work, optimizing DNN structure to maximize compute per communication for the CHOCO model.

6 RELATED WORK

Several classes of work relate to CHOCO: low-power ML, HE hardware, hardware security, and privacy-preserving ML.

Privacy Preserving DNN Inference. ML offloading requires data privacy. Recent work optimized server-centric metrics, including usability [20, 22], training [48], throughput (via batching) [6, 22, 32], latency (via packing) [8, 20, 36, 61], network complexity [6, 36, 41], performance [59, 63, 70], and model privacy [2, 11, 36, 41, 47, 55, 60]. Unlike prior work that focused on the server, to the best of our knowledge, CHOCO is the first work optimizing for resource-constrained client devices in client-aided HE. Client-aided HE is quickly favorable over complete HE offload because it can circumvent fundamental depth and operation limitations of HE without employing expensive bootstrapping operations. Furthermore, as shown in Sections 5.3 and 5.6, frequent communication of smaller ciphertexts results in substantially less communication overall while gaining the crucial ability to pack and repack ciphertext vectors for efficient encrypted computation.

HE Hardware Support. Some prior work used hardware to accelerate kernels for lattice-based cryptography schemes [21, 44, 57, 62], including current state-of-art HE schemes [7, 13, 23, 26]. Others accelerate HE directly [37, 46, 55, 59, 64, 70], focusing primarily on hardware NTT and server-side operations. As we show in Figure 2, NTT acceleration helps but is insufficient. Our work is the first to *comprehensively* optimize *client-side* HE cryptographic primitives, which is crucial in client-aided HE. Furthermore, unlike prior work targeting large, high-power GPUs [3, 18, 54] and FPGAs [46, 59, 63, 70], CHOCO-TACO empirically optimizes for a small ASIC implementation, directly addressing the need for low-power, energy-efficient operation at the client device.

Hardware Security. Recent architectures offer privacy-preserving offloaded computation. Data privacy techniques include Trusted Execution Environments (TEEs) [34, 51, 68], as well as memory access control and obfuscation [33, 56, 65, 67]. While these prior techniques are vulnerable to side channel attacks, HE is not. Data remain provably hidden while offloaded. HE is favorable thanks to its strong, proven privacy guarantee. Moreover, client-aided HE welcomes interactions between the client and server, e.g. to provide proprietary services, that are not allowed by TEEs.

Low-Power ML Acceleration. Client DNN inference performance is improving through software [1, 52] and hardware optimization [12, 30]. One alternative to HE for private inference is to simply outfit IoT devices with ML acceleration and compute locally. However, as we argue in Section 2, local compute imposes tight resource limits and requires maintaining (i.e., updating) models on a potentially very large number of client devices, rather than an offload server's centralized model. In contrast, CHOCO targets encrypted offload of ML (and other) computations, imposing few restrictions on centrally managed models. Furthermore, CHOCO's support straightforwardly generalizes beyond ML: outfitting a device with a HE cryptographic acclerator, rather than specialized DNN hardware, enables participating in *any* client-aided, encrypted computation, not only ML. Encrypted applications research is an emerging area [5, 9, 29, 31, 48, 71, 72]; CHOCO-TACO benefits existing and future applications.

7 CONCLUSION AND FUTURE WORK

CHOCO is a *client*-optimized system for collaborative encrypted compute offloading. We showed that minimizing client costs requires new HE algorithms (rotational redundancy) and client-friendly selection of HE parameters. Owing to the ability to use smaller ciphertexts, CHOCO reduces communication costs — a dominant end-to-end cost — by *three orders of magnitude* over existing client-aided privacy-preserving schemes. The pivot to client-focus puts encryption and decryption on the critical path, necessitating our CHOCO-TACO hardware accelerator for HE encryption primitives beyond the partial hardware support of prior work. Compared to software, CHOCO-TACO's hardware yields 417× speedup and a 603× energy savings for encryption. CHOCO enabled development of several full-scale, real-world DNN inference applications, as well as several new HE applications — KNN, KMeans, and PageRank — that have not been explored in the literature previously. CHOCO's support for client-aided HE thus makes *possible* efficient new applications of encrypted offload computing and enables participation from IoT clients.

Future Work. The future is bright for encrypted computing as system support matures. While FHE was once many orders of magnitude too slow to be practical [26], the performance gap has narrowed to the point of practicality with new work on hardware acceleration, such as F1 [64] (which was concurrent with our work). Moving forward, our community is likely to see improvements on many fronts. Client-aided HE, as CHOCO enables, evades many downfalls of complete vector HE offload (e.g. expensive bootstrapping, operation limitations, and fragmented vector packing) and makes possible the participation of edge devices in encrypted computing. Partitioning encrypted workloads between client and server and managing communication of encrypted data remains a key compilers, software, architecture, and networks challenge for future systems. New schemes lead to new applications by eliminating the cost of bootstrapping [14, 73] and enabling more complex operations [15, 27], but new schemes require new hardware support and a restructuring of applications to fit these schemes. Hardware for existing operations is likely to continue to improve in energy, performance, and scale. Together, these changes mark an exciting shift toward the practical utility of encrypted computing.

ACKNOWLEDGEMENTS

We thank the ASPLOS 2022 reviewers and members of the ABSTRACT group at CMU for the helpful feedback in improving our ideas and manuscript. McKenzie van der Hagen was funded by an Apple Ph.D. Fellowship and this project was funded in part by NSF CAREER Award #1751029.

REFERENCES

[1] M. Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (release 2.2). https://www.tensorflow.org/.

[2] Anshul Aggarwal, Trevor E. Carlson, Reza Shokri, and Shruti Tople. 2020. SOTERIA: In Search of Efficient Neural Networks for Private Inference. arXiv:2007.12934 [cs.CR]

[3] Ahmad Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun, and Khin Mi Mi Aung. 2018. High-Performance FV Somewhat Homomorphic Encryption on GPUs: An Implementation using CUDA. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 2 (May 2018), 70–95. https://doi.org/10.13154/tches.v2018.i2.70-95

[4] Jean Claude Bajard, Nicolas Meloni, and Thomas Plantard. 2005. Efficient RNS Bases for Cryptography.

[5] Foteini Baldimtsi and Olga Ohrimenko. 2014. Sorting and Searching Behind the Curtain: Private Outsourced Sort and Frequency-Based Ranking of Search Results Over Encrypted Data. Cryptology ePrint Archive, Report 2014/1017. https://eprint.iacr.org/2014/1017.

[6] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski. 2019. nGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data. arXiv:1908.04172v2.

[7] Zvika Brakerski. 2012. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*. Springer-Verlag, Berlin, Heidelberg, 868–886. https://doi.org/10.1007/978-3-642-32009-5_50

[8] Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. 2019. Low Latency Privacy Preserving Inference. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, California, USA, 812–821. https://proceedings.mlr.press/v97/brutzkus19a.html

[9] Gizem S. Çetin, Yarkın Doröz, Berk Sunar, and Erkay Savaş. 2015. Low Depth Circuits for Efficient Homomorphic Sorting. Cryptology ePrint Archive, Report 2015/274. https://eprint.iacr.org/2015/274.

[10] Hee-Jin Chae, Daniel Yeager, Joshua Smith, and Kevin Fu. 2013. Wirelessly Powered Sensor Networks and Computational RFID. https://doi.org/10.1007/978-1-4419-6166-2_10

[11] Nishanth Chandran, Divya Gupta, Aseem Rastogi, Rahul Sharma, and Shardul Tripathi. 2017. EzPC: Programmable, Efficient, and Scalable Secure Two-Party Computation for Machine Learning. Cryptology ePrint Archive, Report 2017/1109. https://eprint.iacr.org/2017/1109.

[12] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. 2017. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits* 52, 1 (2017), 127–138. https://doi.org/10.1109/JSSC.2016.2616357

[13] J. H. Cheon, A. Kim, M. Kim, and Y. Song. 2016. Homomorphic Encryption for Arithmetic of Approximate Numbers. Cryptology ePrint Archive, Report 2016/421. https://eprint.iacr.org/2016/421.

[14] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. August 2016. TFHE: Fast Fully Homomorphic Encryption Library. https://tfhe.github.io/tfhe/.

[15] Ilaria Chillotti, Marc Joye, and Pascal Paillier. 2021. Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks. https://whitepaper.zama.ai/.

[16] Alexei Colin, Emily Ruppel, and Brandon Lucia. 2018. A Reconfigurable Energy Storage Architecture for Energy-Harvesting Devices. *SIGPLAN Not.* 53, 2 (March 2018), 767–781. https://doi.org/10.1145/3296957.3173210

[17] D. Corvoysier. 2017. SqueezeNet for CIFAR-10. https://github.com/kaizouman/tensorsandbox/tree/master/cifar10/models/squeeze.

[18] W. Dai and B. Sunar. 2015. cuHE: A Homomorphic Encryption Accelerator Library. In *Cryptography and Information Security in the Balkans*. Springer Ineternational Publishing, Koper, Slovenia, 169–186.

[19] Roshan Dathathri, Blagovesta Kostova, Olli Saarikivi, Wei Dai, Kim Laine, and Madan Musuvathi. 2020. EVA: An Encrypted Vector Arithmetic Language and Compiler for Efficient Homomorphic Computation. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (London, UK) *(PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 546–561. https://doi.org/10.1145/3385412.3386023

[20] Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. 2019. CHET: An Optimizing Compiler for Fully-Homomorphic Neural-Network Inferencing. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Phoenix, AZ, USA) *(PLDI 2019)*. Association for Computing Machinery, New York, NY, USA, 142–156. https://doi.org/10.1145/3314221.3314628

[21] Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2015. Efficient Software Implementation of Ring-LWE Encryption. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (Grenoble, France) *(DATE '15)*. EDA Consortium, San Jose, CA, USA, 339–344.

[22] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. 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 (ICML'16)*. JMLR.org, New York, NY, USA, 201–210.

[23] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144. https://eprint.iacr.org/2012/144.

[24] E. Freiman. 2018. Digit Recognizer for MlPack. https://https://github.com/mlpack/models/tree/master/Kaggle.

[25] J. D. Garside, S. B. Furber, S. Temple, and J. V. Woods. 2009. The Amulet chips: Architectural development for asynchronous microprocessors. In *2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009)*. IEEE, Yasmine Hammamet, Tunisia, 343–346. https://doi.org/10.1109/ICECS.2009.5411006

[26] C. Gentry. 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing* (Bethesda, Maryland) *(STOC 41)*. Association for Computing Machinery, New York, NY, USA, 169–178.

[27] Shruthi Gorantala, Rob Springer, Sean Purser-Haskell, William Lam, Royce Wilson, Asra Ali, Eric P. Astor, Itai Zukerman, Sam Ruth, Christoph Dibak, Phillipp Schoppmann, Sasha Kulankhina, Alain Forget, David Marn, Cameron Tew, Rafael Misoczki, Bernat Guillen, Xinyu Ye, Dennis Kraft, Damien Desfontaines, Aishe Krishnamurthy, Miguel Guevara, Irippuge Milinda Perera, Yurii Sushko, and Bryant Gipson. 2021. *A General Purpose Transpiler for Fully Homomorphic Encryption*. Technical Report. Google LLC.

[28] S. Halevi and V. Shoup. 2014. Algorithms in HElib. Cryptology ePrint Archive, Report 2014/106. https://eprint.iacr.org/2014/106.

[29] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. 2019. Logistic Regression on Homomorphic Encrypted Data at Scale. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 9466–9471. https://doi.org/10.1609/aaai.v33i01.33019466

[30] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. *SIGARCH Comput. Archit. News* 44, 3 (jun 2016), 243–254. https://doi.org/10.1145/3007787.3001163

[31] HElib. 2019. HElib BGV Country Database Lookup Example. https://github.com/homenc/HElib/tree/master/examples/BGV{_}country{_}db{_}lookup.

[32] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2019. Deep Neural Networks Classification over Encrypted Data. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy* (Richardson, Texas, USA) *(CODASPY '19)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/3292006.3300044

[33] Casen Hunger, Lluis Vilanova, Charalampos Papamanthou, Yoav Etsion, and Mohit Tiwari. 2018. DATS - Data Containers for Web Applications. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems* (Williamsburg, VA, USA) *(ASPLOS '18)*. Association for Computing Machinery, New York, NY, USA, 722–736. https://doi.org/10.1145/3173162.3173213

[34] Intel Corporation 2020. *Intel 64 and IA-32 Architectures Software Developer's Manual*. Intel Corporation.

[35] Neal Jackson and Prabal Dutta. 2020. Permacam: A Wireless Camera Sensor Platform For Multi-Year Indoor Computer Vision Applications. https://conix.io/wp-content/uploads/pubs/3113/jackson_permacam_conix_2020.pptx.pdf.

[36] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. 2018. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In *Proceedings of the 27th USENIX Conference on Security Symposium* (Baltimore, MD, USA) *(SEC'18)*. USENIX Association, USA, 1651–1668.

[37] S. Kim, K. Lee, W. Cho, Y. Nam, J. H. Cheon, and R. A. Rutenbar. 2020. Hardware Architecture of a Number Theoretic Transform for a Bootstrappable RNS-based Homomorphic Encryption Scheme. In *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, Fayetteville, Arkansas, USA, 56–64. https://doi.org/10.1109/FCCM48280.2020.00017

[38] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report. University of Toronto.

[39] K. Laine. 2017. *Simple Encrypted Arithmetic Library 2.3.1*. Microsoft Research.

[40] Y. LeCun, C. Cortes, and CJ. Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010), 1.

[41] J. Liu, M. Juuti, Y. Lu, and N. Asokan. 2017. Oblivious Neural Network Predictions via MiniONN Transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 619–631. https://doi.org/10.1145/3133956.3134056

[42] S. Liu and W. Deng. 2015. Very deep convolutional neural network based image classification using small training sample size. , 730-734 pages.

[43] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. 2011. Flikker: Saving DRAM Refresh-Power through Critical Data Partitioning. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Newport Beach, California, USA) *(ASPLOS XVI)*. Association for Computing Machinery, New York, NY, USA,

213–224. https://doi.org/10.1145/1950365.1950391

[44] P. Longa and M. Naehrig. 2016. "Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *Cryptology and Network Security*. Springer, Milan, Italy, 124–139.

[45] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. 2017. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys Tutorials* 19, 4 (2017), 2322–2358. https://doi.org/10.1109/COMST.2017.2745201

[46] A. Mert, E. Ozturk, and E. Savas. 2020. Design and Implementation of Encryption/Decryption Architectures for BFV Homomorphic Encryption Scheme. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 02 (feb 2020), 353–362. https://doi.org/10.1109/TVLSI.2019.2943127

[47] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Boston, MA, 2505–2522. https://www.usenix.org/conference/usenixsecurity20/presentation/mishra

[48] P. Mohassel and Y. Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, San Jose, CA, USA, 19–38.

[49] Matteo Nardello, Harsh Desai, Davide Brunelli, and Brandon Lucia. 2019. Camaroptera: A Batteryless Long-Range Remote Visual Sensing System. In *Proceedings of the 7th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems* (New York, NY, USA) *(ENSsys'19)*. Association for Computing Machinery, New York, NY, USA, 8–14. https://doi.org/10.1145/3362053.3363491

[50] Jack O'Connor, Samuel Neves, Jean-Philippe Aumasson, and Zooko Wilcox-O'Hearn. 2019. BLAKE3. https://github.com/BLAKE3-team/BLAKE3.

[51] Joongun Park, Naegyeong Kang, Taehoon Kim, Youngjin Kwon, and Jaehyuk Huh. 2020. Nested Enclave: Supporting Fine-Grained Hierarchical Isolation with SGX. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA '20)*. IEEE Press, Virtual Event, 776–789. https://doi.org/10.1109/ISCA45697.2020.00069

[52] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., Red Hook, NY, USA, 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[53] Matt Poremba, Sparsh Mittal, Dong Li, Jeffrey S. Vetter, and Yuan Xie. 2015. DESTINY: A Tool for Modeling Emerging 3D NVM and EDRAM Caches. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (Grenoble, France) *(DATE '15)*. EDA Consortium, San Jose, CA, USA, 1543–1546.

[54] A. Qaisar Ahmad Al Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, and K. Rohloff. 2019. Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme. *IEEE Transactions on Emerging Topics in Computing* 9, 2 (2019), 1–1. https://doi.org/10.1109/TETC.2019.2902799

[55] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T. Lee, Gu-Yeon Wei, Hsien-Hsin S. Lee, and David Brooks. 2021. Cheetah: Optimizing and Accelerating Homomorphic Encryption for Private Inference. In *2021 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Seoul, South Korea, 26–39. https://doi.org/10.1109/HPCA51647.2021.00013

[56] Ling Ren, Xiangyao Yu, Christopher W. Fletcher, Marten van Dijk, and Srinivas Devadas. 2013. Design Space Exploration and Optimization of Path Oblivious RAM in Secure Processors. In *Proceedings of the 40th Annual International Symposium on Computer Architecture* (Tel-Aviv, Israel) *(ISCA '13)*. Association for Computing Machinery, New York, NY, USA, 571–582. https://doi.org/10.1145/2485922.2485971

[57] C. Renteria-Mejia and J. Velasco-Medina. 2017. High-Throughput Ring-LWE Cryptoprocessors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 08 (aug 2017), 2332–2345. https://doi.org/10.1109/TVLSI.2017.2697841

[58] Microsoft Research. 2019. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL.

[59] M. S. Riazi, K. Laine, B. Pelton, and W. Dai. 2020. 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* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 1295–1309. https://doi.org/10.1145/3373376.3378523

[60] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin Lauter, and Farinaz Koushanfar. 2019. XONN: XNOR-based Oblivious Deep Neural Network Inference. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1501–1518. https://www.usenix.org/conference/usenixsecurity19/presentation/riazi

[61] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea) *(ASIACCS '18)*. Association for Computing Machinery, New York, NY, USA, 707–721. https://doi.org/10.1145/3196494.3196522

[62] S.S. Roy, F. Vercauteren, N. Mentens, D.D. Chen, and I. Verbauwhede. 2014. Compact Ring-LWE cryptoprocessor. In *Proceedings of the 16th International Workshop on Cryptographic Hardware Embedded Systems (CHES)*. Springer, Busan, South Korea, 371–391.

[63] Sujoy Sinha Roy, Furkan Turan, Kimmo Jarvinen, Frederik Vercauteren, and Ingrid Verbauwhede. 2019. FPGA-based High-Performance Parallel Architecture for Homomorphic Computing on Encrypted Data. Cryptology ePrint Archive, Report 2019/160. https://eprint.iacr.org/2019/160.

[64] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Srinivas Devadas, Ronald Dreslinski, Christopher Peikert, and Daniel Sanchez. 2021. *F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption*. Association for Computing Machinery, New York, NY, USA, 238–252. https://doi.org/10.1145/3466752.3480070

[65] Hiroshi Sasaki, Miguel A. Arroyo, M. Tarek Ibn Ziad, Koustubha Bhat, Kanad Sinha, and Simha Sethumadhavan. 2019. Practical Byte-Granular Memory Blacklisting Using Califorms. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 558–571. https://doi.org/10.1145/3352460.3358299

[66] NXP Semiconductors. 2016. *IMX6ULL Power Consumption Application Note*. Technical Report AN5345-2. arm. https://www.nxp.com/docs/en/application-note/AN5345.pdf.

[67] A. Shafiee, R. Balasubramonian, M. Tiwari, and F. Li. 2018. Secure DIMM: Moving ORAM Primitives Closer to Memory. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Vienna, Austria, 428–440. https://doi.org/10.1109/HPCA.2018.00044

[68] Pramod Subramanyan, Rohit Sinha, Ilia Lebedev, Srinivas Devadas, and Sanjit A. Seshia. 2017. A Formal Foundation for Secure Remote Execution of Enclaves. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 2435–2450. https://doi.org/10.1145/3133956.3134098

[69] Tensorflow. 2016. Lenet-5-like Convolutional MNIST Model Example. https://github.com/tensorflow/models/blob/v1.9.0/tutorials/image/mnist/convolutional.py.

[70] F. Turan, S. S. Roy, and I. Verbauwhede. 2020. HEAWS: An Accelerator for Homomorphic Encryption on the Amazon AWS FPGA. *IEEE Trans. Comput.* 69, 8 (2020), 1185–1196. https://doi.org/10.1109/TC.2020.2988765

[71] Jun Wang, Afonso Arriaga, Qiang Tang, and Peter Y.A. Ryan. 2018. Facilitating Privacy-Preserving Recommendation-as-a-Service with Machine Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 2306–2308. https://doi.org/10.1145/3243734.3278504

[72] y. GUO, X. Yuan, X. Wang, C. Wang, B. Li, and X. Jia. 2019. Enabling Encrypted Rich Queries in Distributed Key-Value Stores. *IEEE Transactions on Parallel and Distributed Systems* 30, 6 (2019), 1283–1297. https://doi.org/10.1109/TPDS.2018.2885519

[73] Zama-AI. 2022. Concrete Operates oN Cyphertexts Rapidly by Extending TfhE. https://github.com/zama-ai/concrete.