Wave Hello to Privacy: Efficient Mixed-Mode MPC using Wavelet Transforms

José Reis
Nillion
jose.reis@nillion.com

Mehmet Ugurbil
Nillion
memo@nillion.com

Sameer Wagh
SecretBit Ventures LLC
swagh@alumni.princeton.edu

Ryan Henry
University of Calgary
ryan.henry@ucalgary.ca

Miguel de Vega
Nillion
miguel@nillion.com

ABSTRACT

This paper introduces new protocols for secure multiparty computation (MPC) leveraging Discrete Wavelet Transforms (DWTs) for computing nonlinear functions over large domains. By employing DWTs, the protocols significantly reduce the overhead typically associated with Lookup Table-style (LUT) evaluations in MPC. We state and prove foundational results for DWT-compressed LUTs in MPC, present protocols for 9 of the most common activation functions used in ML, and experimentally evaluate the performance of our protocols for large domain sizes in the LAN and WAN settings. Our protocols are extremely fast – for instance, when considering 64-bit inputs, computing 1000 parallel instances of the sigmoid function, with an error less than 2^{-24} takes only a few hundred milliseconds incurs just 29 KiB of online communication (40 bytes per evaluation).

KEYWORDS

Privacy-enhancing technologies, secure multiparty computation, lookup tables, discrete wavelet transforms

1 INTRODUCTION

Multiparty computation (MPC) enables multiple parties to jointly compute a function over their private inputs without revealing those inputs to each other. In theory, one can obliviously evaluate any computable function in this way; in practice, however, many functions of interest are prohibitively expensive to evaluate exactly in a secure MPC protocol. In the ubiquitous case of secure MPC based on additive secret-sharing, a common workaround is to transform secret function inputs into secret-shared standard basis vectors—essentially multi-server PIR queries—and then have the computation parties compute the inner products of those queries with a precomputed look-up table (LUT), obliviously "fetching" the desired function outputs from the LUT. This approach works well for evaluations over very small domains [30] or functions that admit exact piecewise-polynomial representations [25]. However, in general, LUT sizes quickly grow untenable with increasing domain sizes and accuracy requirements.

This paper explores the use of *discrete wavelet transforms* (DWTs) to produce compact LUTs that accurately approximate nonlinear functions over large domains. It proves foundational results about querying DWT-compressed LUTs efficiently in an MPC setting and presents four concrete protocols based on these results (the four protocols arise from combinations among two DWTs and two methods for preparing query vectors.) The new protocols are fast, have low round complexity, and give highly accurate function computations.

While our new methods are highly general, we focus our experimental evaluation on a specific use case: nonlinear activation functions in deep neural networks. Prior work [11, 34] underscores the need for fast, high-precision evaluation in this context, and our methods are well-suited to meet this need. We conduct a series of experiments to empirically measure the performance of our protocols in both LAN and WAN environments. Another set of experiments explores the compression-versus-accuracy curves for the most common activation functions in the neural network literature. Our findings open up a rich body of questions and potential future research directions where our techniques could be further explored and applied.

Roadmap

The rest of the paper is structured as follows: Section 2 covers essential preliminaries, including notation and fundamental concepts of DWTs and their application in secure MPC. In Section 3, we provide a detailed explanation of the theory of DWTs, focusing on their computational efficiency and utility in compressing LUTs for function evaluation in MPC protocols. Section 4 presents our first core technical contribution, fundamental results for key MPC structures under the Haar and Bior transformations. We also present a technical overview of the methodology for leveraging DWT compression in LUT-based function evaluation within an MPC framework. In Section 5 and Section 6 we present the full protocols for Haar and Bior DWT respectively. Section 7, presents a comprehensive evaluation of our protocols, highlighting the performance improvements achieved through our approach, across various network settings. We discuss related work in Section 8. Open questions and future directions are in Section 9, with conclusions in Section 10.

2 BACKGROUND

We begin by introducing the notation used throughout the work, followed by some key building blocks from literature, and state our threat model towards the end.

2.1 Notation

If $a \in \mathbb{Z}_{2^\ell}$, then $\vec{e}_a := [\,0 \cdots 0\ \ 1\ \ 0 \cdots 0\,]$ is the standard basis vector of length 2^ℓ with a 1 in its ath position (and 0s elsewhere). Notice that we can always unambiguously deduce the length of $\vec{e}_a$ if we know the domain of a. For indexing into vectors, we use an array-like notation such as $\vec{e}_a[i]$ for the ith component of $\vec{e}_a$. We write $\langle \vec{u}, \vec{v} \rangle$ for the inner product of two vectors and $\mathrm{diag}(A,B,C,\ldots)$ for a block diagonal matrix whose blocks may have varying (albeit square) dimensions. $I_N \in \mathbb{R}^{N \times N}$ is the $N \times N$ identity matrix. For an integer $a \in \mathbb{Z}_{2^\ell}$, we write $\mathrm{lsb}_k(a)$ and $\mathrm{msb}_k(a)$ respectively for the k-bit integers obtained by taking the k-least-significant and k-most-significant bits of a; that is, if $a = (a_{\ell-1}\, a_{\ell-2} \cdots a_0)_2$, then $\mathrm{lsb}_k(a) := \sum_{i=0}^{k-1} 2^i \cdot a_i$ and $\mathrm{msb}_k(a) := \sum_{i=0}^{k-1} 2^i \cdot a_{\ell-k+i}$. We write $x \in_{\mathrm{R}} X$ to denote that x is sampled uniformly at random from the finite set X.

2.1.1 Secret sharing. Throughout, we write $[a]$ for a (2,2)-additive sharing, whether of a scalar $a \in \mathbb{Z}_{2^\ell}$ or a vector $\vec{a} \in (\mathbb{Z}_{2^\ell})^N$. We frequently deal with sharings of standard basis vectors, writing $(\!|\vec{e}_a|\!)$ for a (2,2)-distributed point function (DPF) sharing of $\vec{e}_a$ (with 1-bit outputs) and $[\![\vec{e}_a]\!]$ for a (2,2)-Boolean (XOR) sharing of $\vec{e}_a$.

Finally, we employ (2,2)-distributed comparison function (DCF) sharings, writing $\{\!|(x{>}r)\ ?\ Y_0 : Y_1|\!\}$ to denote the DCF-shared step function

$$\delta_{>r,Y_0,Y_1}(x) := \begin{cases} Y_0 & \text{if } x > r \text{, and} \\ Y_1 & \text{otherwise.} \end{cases}$$

2.1.2 Fixed-point numbers. We use fixed-point numbers parameterized by a bitlength $\ell \in \mathbb{N}$ and fractional precision $f \in \mathbb{N}$ with $0 \le f \le \ell$. Reasonable values for these parameters might be, say, $\ell = 64$ and $f = 24$, with which we could approximate any real $x \in [-2^{39}, 2^{39})$ with absolute error less than 2^{-24}. In particular, we encode a real number $x \in [-2^{\ell-f-1}, 2^{\ell-f-1})$ using the integer $r = \lfloor 2^f \cdot x \rceil \in \mathbb{Z}_{2^\ell}$. The reverse transformation decodes to a real number $\tilde{x} = r \cdot 2^{-f}$ that approximates x in the sense that $|x - \tilde{x}| < 2^{-f}$. We write

$$\mathbb{R}_{\ell,f} := \{x \in \mathbb{R} \,|\, x \cdot 2^f \in \mathbb{Z} \cap [-2^{\ell-1}, 2^{\ell-1})\}$$

for the set of reals with exact representations, and

$$\mathbb{Z}_{\ell,f} := \{\lfloor 2^f \cdot x \rceil_\ell \in \mathbb{Z}_{2^\ell} \,|\, x \in \mathbb{R}_{\ell,f}\} \tag{1}$$

for the corresponding set of fixed-point integer representations. In Equation (1), the subscript in the notation $\lfloor 2^f \cdot x \rceil_\ell$ indicates that we represent $\lfloor 2^f \cdot x \rceil$ using exactly ℓ bits as an element of $\mathbb{Z}_{2^\ell}$. We refer to $\mathbb{Z}_{\ell,f}$ as the set of *(ℓ,f)-bit integers*.

2.1.3 LUTs. Suppose we are given a function $F : \mathbb{R} \to \mathbb{R}$ that we wish to approximate on an interval $[A,B)$ either using real values or (ℓ,f)-bit integers. For ease of exposition, we assume that $A, B \in \mathbb{R}_{\ell,f}$ and that $2^n \,|\, (B-A)$ for some chosen $n \in \mathbb{N}$.

We define the *real signal lookup table* (*real signal LUT*) as

$$\mathbb{L}^{\mathbb{R},n}_{A,B}(F) := [y_A, y_{A+\delta}, y_{A+2\delta}, \ldots, y_{A+(2^n-1)\delta}] \in \mathbb{R}^{2^n}, \tag{2}$$

where $\delta = 2^{-n} \cdot (B-A)$ and where each $y_x := F(x)$. Note that the parameter n determines the precision (and size) of the LUT: a higher n produces more samples of the function values and thus higher fidelity function evaluation. We call n the quantization parameter of the LUT. In cases where $n = \lg(B-A)$ so that $\delta = 1$, the resulting real signal LUT enables perfect evaluation of $F(x)$ at any $a = A + i \in \mathbb{R}$ with $i \in \{0,1,\ldots,2^n-1\}$ either via simple (scaled) array indexing or, equivalently, via a (scaled) inner product with $\vec{e}_a$

$$F(a) = \mathbb{L}^{\mathbb{R},n}_{A,B}(F))[a] = \langle \vec{e}_a, \mathbb{L}^{\mathbb{R},n}_{A,B}(F) \rangle.$$

Similarly, using fixed-point numbers, we define the *signal lookup table* (*signal LUT*) as

$$\mathbb{L}^{\ell,f,n}_{A,B}(F) := [\tilde{y}_A, \tilde{y}_{A+\delta}, \tilde{y}_{A+2\delta}, \ldots, \tilde{y}_{A+(2^n-1)\delta}] \in (\mathbb{Z}_{\ell,f})^{2^n},$$

where $\delta = 2^{-n} \cdot (B-A)$ and $\tilde{y}_x := \lfloor 2^f \cdot F(x) \rceil_\ell$ is the (ℓ,f)-bit approximation to $F(x)$. In cases where $n = \lg(B-A) + f$ so that $\delta = 2^{-f}$, the resulting signal LUT enables "perfect" fixed-point evaluation of $F(x)$ at any $a \in [A,B] \cap \mathbb{R}_{\ell,f}$ either via simple (scaled) array indexing or, equivalently, via a (scaled) inner product with $\vec{e}_{\bar{a}}$:

$$F(a) \approx 2^{-f} \cdot (\mathbb{L}^{\ell,f,n}_{A,B}(F))[\bar{a}] = 2^{-f} \cdot \langle \vec{e}_{\bar{a}}, \mathbb{L}^{\ell,f,n}_{A,B}(F) \rangle. \tag{3}$$

where $\bar{a} := \lfloor 2^f \cdot a \rceil_\ell$ is the (ℓ,f)-bit representation of a. We write $\tilde{F}_{\mathbb{L}^{\ell,f,n}}(a)$ for such an "evaluation" of $\mathbb{L}^{\ell,f,n}_{A,B}(F)$ at $a \in [-2^{\ell-1}, 2^{\ell-1})$ using Equation (3). In the common special case where $|A| = |B| = 2^{\ell-f-1}$ so that $[A,B) \cap \mathbb{R}_{\ell,f} = \mathbb{R}_{\ell,f}$, we drop the subscripts and write $\mathbb{L}^{\ell,f,n}(F)$.

2.1.4 Circular rotation. A basic operation we use repeatedly is that of *circularly rotating* vectors by some distance $x \in \mathbb{N}$; that is, applying a cyclic permutation to a length-N vector $\vec{v}$ that sends each component $\vec{v}[i]$ to $\vec{v}[(i-x) \bmod N]$. We write $\vec{v} \lll x$ as a shorthand for this mapping. We also use the inverse mapping $\vec{v} \ggg x$ which sends $\vec{v}[i]$ to $\vec{v}[(i+x) \bmod N]$.

Notice that the cyclic rotations $\vec{e}_r \lll x$ of a standard basis vector are also standard basis vectors, namely $\vec{e}_a = \vec{e}_r \lll x$ for $a := r - x \bmod N$.

2.2 Model and security assumptions

Our protocols operate in the semi-honest (2+1)-party model, wherein a trusted dealer prepares data-independent, correlated randomness in an offline phase for the players to consume in an otherwise semi-honest secure 2-party computation. Such (2 + 1)-party protocols are amenable to conversion to pure 2-party protocols with a fast online phase and (perhaps) full malicious security; however, both 2-party and malicious secure variants of our techniques are beyond the scope of this paper and left to future work.

For DPF sharings, we assume a specific construction of Boyle, Gilboa, and Ishai [3] whose security requires the existence of PRGs (and for DCF sharings from [2]); concretely, our implementation is secure provided fixed-key AES is hard to distinguish from a random permutation. With this sole exception, all cryptographic primitives we consider are unconditionally secure under a non-collusion assumption.

3 DISCRETE WAVELET TRANSFORMS

Discrete wavelet transforms (DWTs) are powerful tools for analyzing signals across different resolutions or scales, similar to discrete Fourier transforms (DFTs) but utilizing wavelets—small, localized waves—instead of indefinite sinusoidal waves.

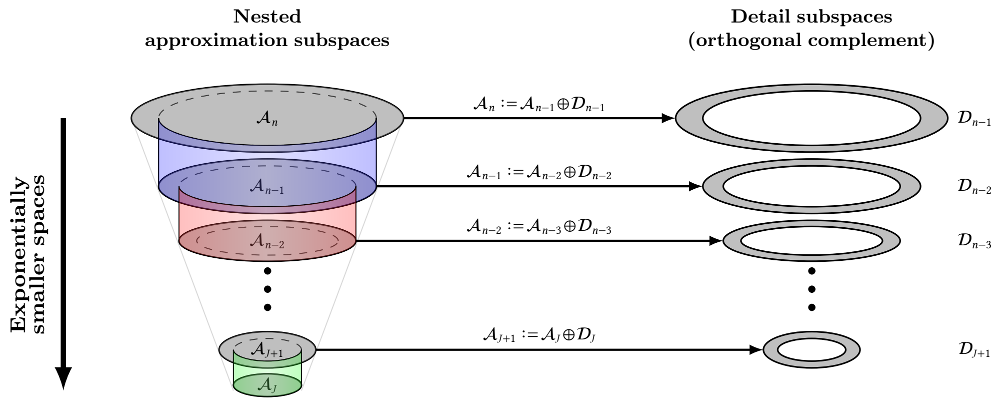
Nested approximation subspaces
Detail subspaces (orthogonal complement)
Exponentially smaller spaces
$\mathcal{A}_n$
$\mathcal{A}_{n-1}$
$\mathcal{A}_{n-2}$
$\mathcal{A}_{J+1}$
$\mathcal{A}_J$
$\mathcal{A}_n := \mathcal{A}_{n-1} \oplus \mathcal{D}_{n-1}$
$\mathcal{A}_{n-1} := \mathcal{A}_{n-2} \oplus \mathcal{D}_{n-2}$
$\mathcal{A}_{n-2} := \mathcal{A}_{n-3} \oplus \mathcal{D}_{n-3}$
$\mathcal{A}_{J+1} := \mathcal{A}_J \oplus \mathcal{D}_J$
$\mathcal{D}_{n-1}$
$\mathcal{D}_{n-2}$
$\mathcal{D}_{n-3}$
$\mathcal{D}_{J+1}$

Figure 1: In multi-resolution analysis (MRA), the signal is projected onto a sequence of nested approximation subspaces $\mathcal{A}_n, \mathcal{A}_1, \ldots, \mathcal{A}_J$ of decreasing resolution. The information lost in each successive projection is captured in an orthogonal sequence of detail subspaces $\mathcal{D}_{n-1}, \mathcal{D}_{n-2}, \ldots, \mathcal{D}_{J+1}$. The process does not lose any information and can stop at any level $0 < J \leq n$.

This localization in both time and frequency makes DWTs particularly effective for signals where properties change over time. DWTs facilitate multi-resolution analysis (MRA) by decomposing signals into increasingly finer "approximation" and "detail" coefficients. This hierarchical decomposition allows for efficient data compression and reconstruction, making DWTs ideal for constructing compact LUTs for high-fidelity univariate function approximations, balancing precision with computational efficiency—properties that can enable interesting computational tradeoffs in secure MPC.

3.1 DWT Basics

Consider a real-valued signal vector $\vec{v} \in \mathbb{R}^N$ of length $N = 2^n$. Then a DWT of $\vec{v}$ is given by the matrix-vector product $W_N \cdot \vec{v}^{\,\mathrm{T}}$, where $W_N \in \mathbb{R}^{N \times N}$ is a real-valued N-by-N matrix called a *wavelet matrix*[1] and $\vec{v}^{\,\mathrm{T}}$ is the column-vector obtained by transposing $\vec{v}$. For reasons that will be clear later, we set $W_{N,1} := W_N$. The upper and lower halves of $W_{N,1}$ respectively comprise series of low- and high-pass filters; thus, we write

$$\left[\begin{array}{c} \vec{a}_{n-1}^{(\vec{v})} \\ \hline \vec{d}_{n-1}^{(\vec{v})} \end{array}\right] := W_{N,1} \cdot \vec{v}^{\,\mathrm{T}},$$

where $\vec{a}_{n-1}^{(\vec{v})}, \vec{d}_{n-1}^{(\vec{v})} \in \mathbb{R}^{N/2}$ are called the *approximation* (or *scaling*) coefficients and *detail* (or *wavelet*) coefficients respectively arising from the low- and high-pass portions of $W_{N,1}$.

Heuristically, we expect that the approximation coefficients $\vec{a}_{n-1}^{(\vec{v})}$ contain near-complete information about $\vec{v}$, in which case we can safely discard the detail coefficients $\vec{d}_{n-1}^{(\vec{v})}$ to obtain a twofold reduction in the length of our approximation relative to $\vec{v}$. We then recurse on $\vec{a}_{n-1}^{(\vec{v})}$ by computing

$$\left[\begin{array}{c} \vec{a}_{n-2}^{(\vec{v})} \\ \hline \vec{d}_{n-2}^{(\vec{v})} \end{array}\right] := W_{N/2} \cdot \vec{a}_{n-1}^{(\vec{v})},$$

effecting another twofold reduction in length of $\vec{a}_{n-2}^{(\vec{v})}$ relative to $\vec{a}_{n-1}^{(\vec{v})}$, while capturing in $\vec{d}_{n-2}^{(\vec{v})}$ the information lost in the projection. Here $W_{N/2} \in \mathbb{R}^{N/2 \times N/2}$ is the decomposition wavelet matrix whose dimensions are halved relative to W_N. Thus, setting $\vec{y}_1 := W_{N,1} \cdot \vec{v}^{\,\mathrm{T}}$ and

$$W_{N,2} := \mathrm{diag}(W_{N/2}, I_{N/2}) = \left[\begin{array}{c|c} W_{N/2} & \mathbf{0} \\ \hline \mathbf{0} & I_{N/2} \end{array}\right],$$

we can write the depth-2 DWT as

$$\vec{y}_2 := W_{N,2} \cdot \vec{y}_1 = \left[\begin{array}{c} \vec{a}_{n-2}^{(\vec{v})} \\ \hline \vec{d}_{n-2}^{(\vec{v})} \\ \vec{d}_{n-1}^{(\vec{v})} \end{array}\right]$$

where $\vec{y}_2 \in \mathbb{R}^N$, $\vec{d}_{n-1}^{(\vec{v})} \in \mathbb{R}^{N/2}$, and $\vec{a}_{n-2}^{(\vec{v})}, \vec{d}_{n-2}^{(\vec{v})} \in \mathbb{R}^{N/4}$. Generalizing this notion, let us denote by $\vec{y}_j$ the result after a j-fold application of the DWT to a vector $\vec{v}$ of length $N = 2^n$. For instance, as seen above, we have that

$$\vec{y}_2 := W_{N,2} \cdot W_{N,1} \cdot \vec{v}^{\,\mathrm{T}}.$$

We say that $\vec{y}_j$ is the DWT with *depth* j or at *level* $J := n - j$. This process can be iterated to compute the DWT at any depth: Setting $\vec{y}_0 = \vec{v}^{\,\mathrm{T}}$, we have, for all $0 < j \leq n$, that

$$\vec{y}_j := W_{N,j} \cdot \vec{y}_{j-1}.$$

3.1.1 Formal definition. The above constructions can be formalized to provide a definition of a DWT of a vector $\vec{v} \in \mathbb{R}^N$ at depth j.

Definition 1 (Discrete Wavelet Transform [27])**.** Let $j, n \in \mathbb{N}$ with $0 < j \leq n$ and set $N := 2^n$. If $\vec{v} \in \mathbb{R}^N$ is a real-valued vector of length N and W_N a decomposition wavelet matrix, then the *discrete wavelet transform (DWT)* of $\vec{v}$ at depth j is

$$\vec{y}_j := W_{N,j} \cdot W_{N,j-1} \cdots W_{N,1} \cdot \vec{v}, \tag{4}$$

such that $W_{N,1} := W_N$ and, for each $k = 2, \ldots, j$,

$$W_{N,k} := \mathrm{diag}(W_{N/2^{k-1}}, I_{N/2^{k-1}}, I_{N/2^{k-2}}, \ldots, I_{N/2}).$$

[1] More precisely, W_N is a *decomposition wavelet matrix* in contrast with a *reconstruction wavelet matrix*; see Appendix A for details on filters and on the construction these matrices.

$$\vec{v} = [\,0\ 0\ 0\ 0\ 3\ 3\ 3\ 3\,] \qquad\qquad \vec{u} = [\,0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\,]$$

$$\vec{y}_2^{\,(\vec{v})} = [\,\underbrace{\overbrace{0\ 6}^{\vec{a}_2^{(\vec{v})}}}_{\text{approx}}\,|\,\underbrace{\overbrace{0\ 0}^{\vec{d}_2^{(\vec{v})}}\,|\,\overbrace{0\ 0\ 0\ 0}^{\vec{d}_1^{(\vec{v})}}}_{\text{detail}}\,] \qquad\qquad \vec{y}_2^{\,(\vec{u})} = [\,\underbrace{\overbrace{0\ 0.5}^{\vec{a}_2^{(\vec{u})}}}_{\text{approx}}\,|\,\underbrace{\overbrace{0\ 0.5}^{\vec{d}_2^{(\vec{u})}}\,|\,\overbrace{0\ 0\ 0.7\ 0}^{\vec{d}_1^{(\vec{u})}}}_{\text{detail}}\,]$$

Figure 2: A slow- and fast-varying signal $\vec{v}$ and $\vec{u}$ and their DWT representations at level $J=2$ under the Haar transform. From Parseval's Theorem, we know that $\langle \vec{u},\vec{v}\rangle = \langle \vec{y}_2^{(\vec{u})}, \vec{y}_2^{(\vec{v})}\rangle$. Because $\vec{v}$ is slow varying, its detail coefficients are trivial and we can ignore those terms and the inner product simplifies to $\langle \vec{a}_2^{(\vec{u})}, \vec{a}_2^{(\vec{v})}\rangle$. We show the full workings of this simple example in Appendix A.3.

Definition 2. (Approximate & Detail Coefficients [27]) Let $j,n \in \mathbb{N}$ with $0 < j \le n$, and then set $N := 2^n$ and $J := n-j$. Let $\vec{y}_j$ be the DWT of a vector $\vec{v} \in \mathbb{R}^N$ at depth j, as given by Equation (4), and write it as

$$\vec{y}_j = \begin{bmatrix} \vec{a}_J^{(\vec{v})} \\ \hline \vec{d}_J^{(\vec{v})} \\ \vec{d}_{J+1}^{(\vec{v})} \\ \vdots \\ \vec{d}_{n-1}^{(\vec{v})} \end{bmatrix} \in \mathbb{R}^N,$$

where $\vec{a}_J^{(\vec{v})}, \vec{d}_J^{(\vec{v})} \in \mathbb{R}^{2^J}$ and $\vec{d}_i^{(\vec{v})} \in \mathbb{R}^{2^i}$ for each $i \in \{J+1,\dots,n-1\}$. Then $\vec{a}_J^{(\vec{v})}$ is the *approximation coefficients* at level J and the $\vec{d}_i^{(\vec{v})}$ are the *detail coefficients* at levels $J,\dots,n-1$.

Figure 2 illustrates the Haar DWT applied to a short vector $\vec{v}$ (with $n=3$ and $j=2$, so that $N=8$ and $J=1$). We obtain an exact reconstruction of $\vec{v}$ from $\vec{y}_j$ using the inverses of the decomposition wavelet matrices

$$\vec{v} = W_{N,1}^{-1} \cdot W_{N,2}^{-1} \cdots W_{N,j}^{-1} \cdot \vec{y}_j$$

For the Haar transform, the decomposition wavelet matrices are orthogonal so that $W_{N,i}^{-1} = W_{N,i}^{\mathrm{T}}$, where $W_{N,i}^{\mathrm{T}}$ denotes the matrix transpose of $W_{N,i}$.

3.2 Haar DWT

The Haar transform is the simplest example of a DWT and while Haar DWTs' compression is poor relative to other wavelet families, it's simplicity makes it easier to analyze, and thus we use that as our starting point for exploring the use of DWTs in secure MPC. In Appendix A.1 we present a formal description of the Haar DWT as well as an explicit example of a Haar DWT matrix. In addition, in Appendix A.3 we provide a comprehensive example of the Haar DWT application and its MRA. The final result of that example is presented in Figure 2.

Parseval's Theorem for orthogonal transforms: We now state Parseval's Theorem for orthogonal DWTs: a fundamental result that establishes that orthogonal DWTs like the Haar transform preserve inner products, providing an approach to efficiently approximate inner products $\langle \vec{e}_a, \vec{v}\rangle$ when the vectors $\vec{e}_a$ and $\vec{v}$ are prohibitively large by first transforming them into the wavelet domain. Specifically, if the detail coefficients of $\vec{y}_j^{(\vec{v})}$ are all sufficiently small, a good approximation of the inner product can be achieved using only the vector of approximation coefficients $\vec{a}_J^{(\vec{v})}$, which is a factor 2^j shorter than $\vec{v}$. We defer the proof of Theorem 1 to Appendix F.

Theorem 1 (Parseval's Theorem [19] for orthogonal DWTs). *Let $j,n \in \mathbb{N}$ with $0 < j \le n$ and set $N := 2^n$ and $J := n-j$. If $\vec{u},\vec{v} \in \mathbb{R}^N$ and $W := W_{N,j} \cdot W_{N,j-1} \cdots W_{N,1} \in \mathbb{R}^{N \times N}$ is an orthogonal decomposition wavelet matrix with $\vec{y}_j^{(\vec{u})} = W \cdot \vec{u}^{\mathrm{T}}$ and $\vec{y}_j^{(\vec{v})} = W \cdot \vec{v}^{\mathrm{T}}$, then, in the notation of Definition 2, we have*

$$\langle \vec{u},\vec{v}\rangle = \langle \vec{a}_J^{(\vec{u})}, \vec{a}_J^{(\vec{v})}\rangle + \sum_{i=J}^{n-1} \langle \vec{d}_i^{(\vec{u})}, \vec{d}_i^{(\vec{v})}\rangle.$$

3.3 Biorthogonal DWT

The biorthogonal family $\mathsf{bior}(p,q)$ of DWTs achieves superior compression relative to orthogonal wavelets like Haar. Relative to Haar DWTs, the $\mathsf{bior}(p,q)$ family exhibits the following crucial differences:

(1) its decomposition wavelet matrices W_N are invertible but *not* orthogonal;
(2) they comprise low- and high-pass symmetric filters that are chosen to maximize vanishing moments for better compression. For details on symmetric filters as well as vanishing moments consult Appendices A.1 and A.2;
(3) the decomposition wavelet matrices W_N use different filters than the reconstruction wavelet matrices $\tilde{W}_N$ and they fulfill the biorthogonal property $\tilde{W}_N^{\mathrm{T}} := W_N^{-1}$.

The parameters p and q determine the lengths of the low-pass decomposition and reconstruction filters, respectively. For this initial foray into applications of DWTs to MPC, we focus on biorthogonal DWT $\mathsf{bior}(5{,}3)$. This particular choice allows an easy characterization of the DWT of a standard basis vector (see Lemma 6 in Section 4.3). Interested readers can consult Van Fleet [27, §9] for details about choosing an appropriate p and q in more general contexts. Just as we did for the Haar DWT, in Appendix A.1 we present a formal description of the biorthogonal DWT $\mathsf{bior}(5{,}3)$ as well as an explicit example of its decomposition and reconstruction matrices for $N=8$.

Parseval's Theorem for biorthogonal transforms: We now state Parseval's Theorem for biorthogonal DWTs. Analogous to the case of orthogonal DWTs, this theorem provides a strategy for efficiently approximating inner products $\langle \vec{e}_a, \vec{v}\rangle$ when the vectors $\vec{e}_a$ and $\vec{v}$ are large. We defer the proof of Theorem 2 below to Appendix F.

Theorem 2 (Parseval's Theorem [19] for biorthogonal DWTs). *Let $j,n \in \mathbb{N}$ with $0 < j \le n$ and set $N := 2^n$ and $J := n-j$. If $\vec{u},\vec{v} \in \mathbb{R}^N$ and $W := W_{N,j} \cdot W_{N,j-1} \cdots W_{N,1} \in \mathbb{R}^{N \times N}$ is a biorthogonal decomposition wavelet matrix with $\vec{y}_1^{(\vec{u})} = W \cdot \vec{u}^{\mathrm{T}}$ and $\vec{y}_1^{(\vec{v})} = \tilde{W} \cdot \vec{v}^{\mathrm{T}}$ for the reconstruction wavelet matrix $\tilde{W}_N^{\mathrm{T}} := W_N^{-1}$, then, in the* *notation of Definition 2, we have*

$$\langle \vec{u},\vec{v}\rangle = \langle \vec{a}_J^{(\vec{u})},\vec{a}_J^{(\vec{v})}\rangle + \sum_{i=J}^{n-1}\langle \vec{d}_i^{(\vec{u})},\vec{d}_i^{(\vec{v})}\rangle.$$

3.4 Effectiveness of DWTs at Compression

In general, the ability of a DWT to approximate a function, and hence to compress a LUT representing that function, depends on the number of vanishing moments (of the high-pass filter) of the DWT under consideration and the smoothness of the function where the DWT is applied. To have a more complete grasp of these relation we refer to [7, §7.4 & §8.2], [14, §10.5], and [33, §9.1 & §10.7]. Below we provide an intuition on the compressibility of polynomial functions which provide good approximations to smooth functions.

If a DWT has a high-pass filter with M vanishing moments then it will achieve "high" compression for polynomials of degree at most $M-1$. While this is not completely rigorous (in this comments we are not considering the boundary effects of using finite signals), the approximation coefficients will be a sampling of a polynomial of degree $M-1$ while the detail coefficients will be zero. Consequently, in the second application of this DWT we are applying it to a sample of a polynomial of degree $M-1$ and the properties we just mentioned are still valid. Therefore, if a DWT has M vanishing moments then it achieves "high" compression for functions that are well approximated by polynomials of degree at most $M-1$.

The DWTs we consider—Haar and bior(5,3)—have 1 and 2 vanishing moments, respectively. Applying DWTs to standard basis vectors is crucial for the MPC protocols, as the approximation coefficients play a key role. For Haar and bior(5,3), these coefficients have simple expressions that enable efficient MPC protocols (see Lemmas 3 and 6). The tradeoff between their vanishing moments manifests in two ways: (i) bior(5,3) offers better compression than Haar (cf. Table 1 and Figure 7), but (ii) it requires a more complex protocol, as shown in Figure 6, with an additional DCF and more intricate LUT evaluation compared to Haar (Figure 5). We elaborate further on this in Appendix A.2 but emphasize that many challenges remain, such as developing simpler heuristics to identify effective DWT-function pairs and discovering new DWTs that balance simple descriptions of standard basis vectors' approximation coefficients with strong compression properties.

4 NON-LINEAR FUNCTIONS IN MPC

Our protocols have 4 parameters: ℓ the bit-size used to encode MPC values, f the fixed-point precision, n a parameter that controls the precision of the original LUT and thus the fidelity of function computation[2], and J (level of the DWT) which is a dependent parameter that will control the compression offered by the DWTs. Another dependent parameter is j (depth of the DWT) is equal to $n-J$.

[2] The reader will notice that the parameter n is present in a subtle manner in Haar protocols – only appearing in the one-time pre-processing phase for the Haar protocol (in the Bior protocol, it is used in both the one-time pre-processing as well as the online phase).

Let $\bar{a}\in\mathbb{R}$ and let $a=\lfloor 2^f\cdot\bar{a}\rceil\in\mathbb{Z}_{\ell,f}$ be the (ℓ,f)-bit approximation to $\bar{a}$. Suppose two parties hold an additive sharing $[a]$ and wish to compute $[y]$ where $2^{-f}\cdot y\approx F(\bar{a})$ for some function $F:\mathbb{R}\to\mathbb{R}$. A common, function-agnostic way to do this uses LUTs; that is, we compute the signal LUT $\vec{v}:=\mathbb{L}^{\ell,f,n}(F)$ and then obliviously fetch an entry from it using $[\vec{e}_{\mathrm{msb}_n(a)}]$, which the parties compute using $[\vec{e}_r]$, $[r]$, and $[a]$. The parties rely on correlated randomness from the dealer to facilitate all of this.

We use the basic, semi-honest variant of Pika [30] as our prototypical example of such a protocol. Figure 3 sketches the steps Pika clients use to perform the desired function evaluation. The protocol is parametrized by the function F to approximate, fixed-point parameters ℓ,f for inputs and outputs, and a quantization granularity n for the LUT. The bottleneck operation in Pika is evaluating the inner product $\langle[\pm\vec{e}_{\mathrm{msb}_n(a)}],\vec{v}\rangle$, which involves vectors whose lengths are exponential in the quantization parameter n. Since the computation is proportional to the size of the LUT, it is imperative to truncate the additive shares $[a]\in\mathbb{Z}_{2^\ell}$ into $[\mathrm{msb}_n(a)]\in\mathbb{Z}_{2^n}$. Thus, the parameter n controls the precision of the LUT evaluation and introduces a trade-off: larger values of n produce approximations that are more precise, but at the cost of rapidly increasing computation cost. Wagh notes [30] that beyond $n=24$, the overhead due to this inner product quickly becomes prohibitive for an online computation.

4.1 Technical Overview

DWT compression is a powerful tool for realizing efficient LUT-based function evaluation in MPC based on linear secret sharing. Suppose that $F:\mathbb{R}\to\mathbb{R}$ is a (non-linear) univariate function, and consider the signal LUT $\mathbb{L}^{\ell,f}(F)$ for an (ℓ,f)-bit approximation to F. Intuitively, the idea is to run a Pika-like LUT evaluation protocol, but in the image of the DWT homomorphism, as enabled by Parseval's Theorem. Firstly, we use the real signal $\vec{v}:=\mathbb{L}^{\mathbb{R},n}_{A,B}(F)$ as defined in Equation (2). Note that for ease of exposition, we assume that $A,B\in\mathbb{R}_{\ell,f}$ and that $2^n\mid(B-A)$ for some chosen $n\in\mathbb{N}$. Secondly, we apply a DWT and re-write the expression as a linear combination of inner products where one of the vectors is a standard basis vector. At last, we convert the values to (ℓ,f)-bit approximations. By following this order we improve the precision of the approximations and avoid costly implementations of fixed-point precision reductions. That is, when the detail coefficients of $\vec{y}_j=W\cdot\vec{v}^{\,\mathrm{T}}$ are sufficiently small, we have

$$\begin{aligned}
F(a) &\approx \langle \vec{e}_{\tilde{a}},\vec{v}\rangle && \text{(real signal LUT)}\\
&\approx \langle \vec{a}_J^{(\vec{e}_{\tilde{a}})},\vec{a}_J^{(\vec{v})}\rangle && \text{(Parseval's Theorem)}\\
&= \sum_{i=0}^{2^J-1} c_i\langle \vec{e}_i,\vec{v}_i\rangle && \text{(re-writing)}\\
&\approx 2^{-f}\cdot\textstyle\sum_{i=0}^{2^J-1} c_i\langle \vec{e}_i,\mathbf{A}_{F,i}\rangle && \text{((}\ell,f\text{)-bit representation)} \qquad (5)
\end{aligned}$$

where $\bar{a}:=\lfloor 2^f\cdot a\rceil_\ell$ and $\mathbf{A}_{F,i}$ are, respectively, the (ℓ,f)-bit representation of a and $\vec{v}_i$, and $\tilde{a}:=\mathrm{msb}_n(\bar{a})$. In practice, for the Haar and the bior(5,3) transformations we just compute, respectively, one and two inner products.

Pika LUT protocol $\Pi^{\text{LUT-Pika}}_{n,\ell}$

One-time pre-processing: Input is a function F, fixed-point parameters $\ell, f \in \mathbb{N}$, and LUT quantization parameter $n \in \{1,2,...,\ell\}$
- Compute the signal look-up table $\vec{v} := \mathbb{L}^{\ell,f,n}(F)$

Per-run pre-processing:
- Dealer distributes sharings $[r]$ and $[\![\vec{e}_r]\!]$ for $r \in_{\mathrm{R}} \mathbb{Z}_{2^n}$, plus a Beaver triple for multiplying two secret scalars in $\mathbb{Z}_{2^\ell}$

Online phase: Input is $[a]$, $a \in \mathbb{Z}_{\ell,f}$, plus all pre-processing values

(1) Non-interactively compute $[\tilde{a}] = \mathrm{msb}_n([a])^\dagger$

(2) Interactively reconstruct $x_n := r - \tilde{a} \bmod 2^n$ from $[r]$ and $[\tilde{a}]$

(3) Non-interactively compute $[\![\vec{e}_{\tilde{a}}]\!] = [\![\vec{e}_r]\!] \lll x_n$, perform signed extension to convert this into $[\pm\vec{e}_{\tilde{a}}]$, use component-wise summation to compute $[\pm 1]$ from $[\pm\vec{e}_{\tilde{a}}]$, and evaluate $[\pm\tilde{F}_{\mathbb{L}^{\ell,f,n}}(\tilde{a})] = \langle [\pm\vec{e}_{\tilde{a}}], \vec{v} \rangle$

(4) Interactively compute the product of $[\pm\tilde{F}_{\mathbb{L}^{\ell,f,n}}(\tilde{a})]$ and $[\pm 1]$ using the Beaver triple to obtain $[\tilde{F}_{\mathbb{L}^{\ell,f,n}}(\tilde{a})]$

(5) Return $[\tilde{F}_{\mathbb{L}^{\ell,f,n}}(\tilde{a})]$

†Specifically, $\tilde{a} = \mathrm{msb}_n(a) - \delta$ for $\delta \in \{0,1\}$. We have $\delta = 1$ when the discarded low $\ell - n$ bits induce a carry-out during share reconstruction (which happens with probability 0.5), and $\delta = 0$ otherwise.

Figure 3: LUT-based function evaluation protocol from Pika [30]. The protocol uses two rounds of communication (Steps 2 and 4).

This is notable because $\vec{a}_J^{(\vec{e}_a)}$ and $\vec{a}_J^{(\vec{v})}$ are a factor 2^j shorter than $\vec{e}_a$ and $\vec{v}$ (recall that $J := n - j$ so that $j = n - J$), providing a substantial reduction in the cost of evaluating the inner product in the bottom. We write $\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}$ and $\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}$ for the approximation functions that arise by respectively applying the Haar and bior(5,3) transforms at level J in this manner; see Equation (7) and Equation (8).

Thus, our primary insight is that using DWTs to compress LUTs may provide significant computational advantages with only a comparatively modest impact on approximation accuracy. The exact computational savings, of course, hinge on how one arrives at $\vec{a}_J^{(\vec{e}_{\tilde{a}})}$ (note that $\vec{a}_J^{(\vec{v})}$ will be pre-computed once and for all). The naïve method (given by Equation (4)) requires constructing secret-shared basis vectors of length 2^n, and then using massive vector-matrix products to "lower" those vectors into length-2^J approximation coefficients. In the sequel, we present encouraging results for the Haar transform (Subsection 4.2) and the bior(5,3) transform (Subsection 4.3); specifically, we give exact (and efficient) expressions for $\vec{a}_J^{(\vec{e}_r)}$ in terms of r and for $\vec{a}_J^{(\vec{e}_{\tilde{a}})}$ in terms of $\vec{a}_J^{(\vec{e}_r)}$, r, and a.

4.1.1 A note on computing $\vec{a}_J^{(\vec{e}_{\tilde{a}})}$. As with Pika, we consider a setting where inputs are additively shared (ℓ, f)-bit integers, but where the signal LUT uses an n-bit quantization of F. DWT compression then compresses the real signal LUT from 2^n down to just 2^J entries. Consequently, computing a sharing $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$ of $\vec{e}_{\mathrm{msb}_J(a)}$ from an additive sharing $[a]$ of $a \in \mathbb{Z}_{\ell,f}$ is necessary.[3] As illustrated in Figure 3, Pika employs a probabilistic truncation approach for this step that we want to avoid; i.e., the value $\tilde{a}$ in Pika is not necessarily equal to $\mathrm{msb}_n(a)$, resulting in an off-by-one error with probability 0.5.

To avoid "stacking" the approximation errors from probabilistic truncation and DWT compression, we replace Pika's probabilistic approach to computing $[\![\vec{e}_{\mathrm{msb}_n(a)}]\!]$ with a deterministic one. This way, our protocols are guaranteed to give the exact approximation obtained via plaintext evaluation with the same DWT-compressed LUT. We showcase two distinct approaches to deterministically producing $[\![\vec{e}_{\mathrm{msb}_n(a)}]\!]$, one using DCFs to perform deterministic truncation in a Pika-like protocol, and the other entirely side-stepping the need for truncation by using Grotto's segment-parity approach [25, Theorem 1]. Both methods give $[\![\vec{e}_{\mathrm{msb}_n(a)}]\!]$ exactly, but they differ in their performance profiles. We can employ either method in conjunction with either DWT, yielding four potential pairings: "Haar+Pika", "Haar+Grotto", "bior(5,3)+Pika", and "bior(5,3)+Grotto".)

In the sequel, we make the editorial decision to present our Haar-based protocol with the DCF-based computation ("Haar+Pika") and our bior(5,3)-based protocol with the Grotto-based computation ("bior(5,3)+Grotto"). This choice of pairings is arbitrary and intended only to ensure that both methods feature in the main body; for completeness, we present the other two pairings in Appendices H and I.

4.2 Haar-transformed basis vectors

The Haar transform's simplicity yields an expression for approximation coefficients of $\vec{e}_a$ that is incredibly easy to evaluate. Note that all results in this section hold whenever $2^j \mid N$, but we only consider the case of $N = 2^n$. Our first lemma states that the Haar approximation coefficients of a standard basis vector at depth j are just a (much) shorter standard basis vector, only scaled by $2^{-j/2}$.

Lemma 3 (Haar transform for basis vectors)**.** *Let $j, n, \ell \in \mathbb{N}$ with $0 < j \le n \le \ell$, let $a \in \mathbb{Z}_{2^\ell}$, and set $J := n - j$ and $\bar{a} := \mathrm{msb}_n(a)$. The approximation coefficients $\vec{a}_J^{(\vec{e}_{\bar{a}})}$ of $\vec{e}_{\bar{a}}$ at level J under the Haar transform are given by*

$$\vec{a}_J^{(\vec{e}_{\bar{a}})} = 2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(a)} \in \mathbb{R}^{2^J}.$$

The next lemma characterizes the impact of applying a circular rotation (see Step 2 of Figure 3) to a standard basis vector under the Haar transform.

Lemma 4 (Haar transform for shifted basis vectors)**.** *Let $j, n, \ell \in \mathbb{N}$ with $0 < j \le n \le \ell$, let $r, a \in \mathbb{Z}_{2^\ell}$, and set $J := n - j$, $x := r - a \bmod 2^\ell$, $\bar{a} := \mathrm{msb}_n(a)$ and $\bar{r} := \mathrm{msb}_n(r)$. The approximation coefficients $\vec{a}_J^{(\vec{e}_{\bar{a}})}$ of $\vec{e}_{\bar{a}}$ at level J under the Haar transform are given by*

$$\vec{a}_J^{(\vec{e}_{\bar{a}})} = 2^{-j/2} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \lll x_J),$$

where $x_J := \mathrm{msb}_J(x) + \big(\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x)\big) \bmod 2^J$.

We defer the proofs of Lemmas 3 and 4 to Appendix B. Figure 4 illustrates the implications of Lemma 3 and Lemma 4 as they relate to leveraging Theorem 1 to evaluate approximate inner products in a Pika-like protocol (bullets 1 and 2) or Grotto-like protocol (bullet 3). The final result of this section rephrases the first two results from a computational perspective and introduces a third (trivial) observation.

[3] Jumping ahead, the protocol for bior(5,3) additionally requires us to compute $[\mathrm{lsb}_j(\mathrm{msb}_n(a))]$ from $[a]$; for this, we always use a DCF-based approach.

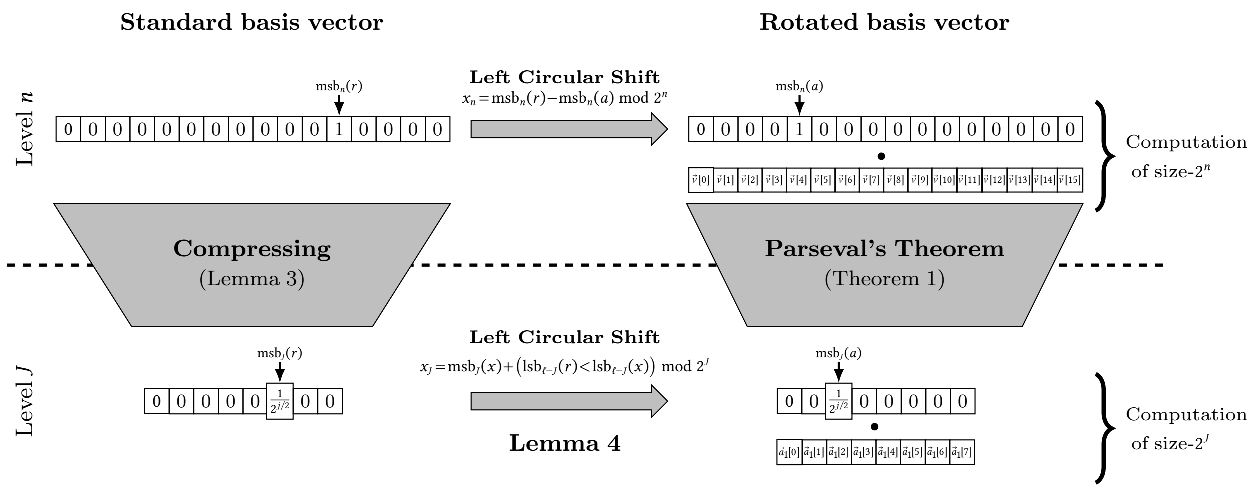

Figure 4: Implications of Lemmas 3 and 4 as they relate to leveraging Theorem 1 for approximate inner-product evaluations. One option is to apply Lemma 3 to compress $\vec{e}_{\mathrm{msb}_n(r)}$ into $2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(r)}$, and then to apply Lemma 4 to cyclically rotate $2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(r)}$ into $2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(a)}$; the other option is to apply a cyclic rotation to obtain $\vec{e}_{\mathrm{msb}_n(a)}$ from $\vec{e}_{\mathrm{msb}_n(r)}$, and then to apply Lemma 3 to compress $\vec{e}_{\mathrm{msb}_n(a)}$ into $2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(a)}$.

Corollary 5. *Let $j,n,\ell \in \mathbb{N}$ with $0 < j \le n \le \ell$ and $a,r \in \mathbb{Z}_{2^\ell}$, and set $J := n-j$, $x := r-a$, $\bar{a} := \mathrm{msb}_n(a)$, and $\bar{r} := \mathrm{msb}_n(r)$. Then the following expressions compute the approximation coefficients $\vec{a}_J^{(\vec{e}_{\bar{a}})}$ of $\vec{e}_{\bar{a}}$ at level J under the Haar transform:*

(1) $\vec{a}_J^{(\vec{e}_{\bar{a}})} = 2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(a)}$;
(2) $\vec{a}_J^{(\vec{e}_{\bar{a}})} = 2^{-j/2} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \lll x_J)$; and
(3) $\vec{a}_J^{(\vec{e}_{\bar{a}})} = 2^{-j/2} \cdot \left[\bigoplus_{i=k \cdot 2^j}^{(k+1) \cdot 2^j - 1} \vec{e}_a[i] \,\middle|\, k = 0,\ldots,2^J - 1 \right]$,

where $x_J := \mathrm{msb}_J(x) + \big(\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x)\big) \bmod 2^J$.

4.3 Bior-transformed basis vectors

As explained in Subsection 3.3, the bior(p,q) family of transforms use different sets of low- and high-pass filters for decomposition and reconstruction. This is in contrast to the orthogonal wavelets of the Haar transform, where the filters used for decomposition and reconstruction are identical. The upshot of this relaxation is that it enables the use of symmetric filters of length larger than 2, yielding superior compression.

The following lemma and corollary for the bior(5,3) transform are analogous to the preceding results for the Haar transform. We defer their proofs to Appendix C.

Lemma 6 (bior(5,3) transform for basis vectors)**.** *Let $j,n,\ell \in \mathbb{N}$ with $0 < j \le n \le \ell$, let $a \in \mathbb{Z}_{2^\ell}$, and set $J := n-j$ and $\bar{a} := \mathrm{msb}_n(a)$. The approximation coefficients $\vec{a}_J^{(\vec{e}_{\bar{a}})}$ of $\vec{e}_{\bar{a}}$ at level J under the* bior(5,3) *transform are given by*

$$\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0} \cdot \vec{e}_{\mathrm{msb}_J(a)} + c_{J,1} \cdot (\vec{e}_{\mathrm{msb}_J(a)} \ggg 1) \in \mathbb{R}^{2^J},$$

where $c_{J,0} := \big(2^j - \mathrm{lsb}_j(\bar{a})\big) \cdot 2^{-3j/2}$ and $c_{J,1} := \mathrm{lsb}_j(\bar{a}) \cdot 2^{-3j/2}$.

The next lemma characterizes the impact of applying a circular rotation to a standard basis vector under the bior(5,3) transform.

Lemma 7 (bior(5,3) transform for shifted basis vectors)**.** *Let $j,n,\ell \in \mathbb{N}$ with $0 < j \le n \le \ell$, let $r,a \in \mathbb{Z}_{2^\ell}$, and set $J := n-j$, $x := r-a \bmod 2^\ell$ and $\bar{a} := \mathrm{msb}_n(a)$. The approximation coefficients $\vec{a}_J^{(\vec{e}_{\bar{a}})}$ of $\vec{e}_{\bar{a}}$ at level J under the* bior(5,3) *transform are given by*

$$\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \lll x_J) + c_{J,1} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \lll x_J - 1) \in \mathbb{R}^{2^J}, \qquad (6)$$

where $c_{J,0}, c_{J,1} \in \mathbb{R}^{2^J}$ are as defined in Lemma 6 and where $x_J := \mathrm{msb}_J(x) + (\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x)) \bmod 2^J$.

The final result of this section rephrases the first two results from a computational perspective and introduces a third (trivial) observation.

Corollary 8. *Let $j,n \in \mathbb{N}$ with $0 < j \le n$, let $a,r \in \mathbb{Z}_{2^n}$, and set $J := n-j$, $x := r-a \bmod 2^\ell$ and $\bar{a} := \mathrm{msb}_n(a)$. Then the following expressions compute the approximation coefficients $\vec{a}_J^{(\vec{e}_{\bar{a}})}$ of $\vec{e}_{\bar{a}}$ at level J under the Haar transform:*

(1) $\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0} \cdot \vec{e}_{\mathrm{msb}_J(a)} + c_{J,1} \cdot (\vec{e}_{\mathrm{msb}_J(a)} \ggg 1)$;
(2) $\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \lll x_J) + c_{J,1} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \lll x_J - 1)$; and
(3) $\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0} \cdot \tilde{e} + c_{J,1} \cdot (\tilde{e} \ggg 1)$,

where $x_J := \mathrm{msb}_J(x) + \big(\mathrm{lsb}_j(r) < \mathrm{lsb}_j(x)\big) \bmod 2^J$, $c_{J,0}, c_{J,1} \in \mathbb{R}^{2^J}$ are as defined in Lemma 6 and

$$\tilde{e} := \left[\bigoplus_{i=k \cdot 2^j}^{(k+1) \cdot 2^j - 1} \vec{e}_a[i] \,\middle|\, k = 0,\ldots,2^J - 1 \right].$$

5 LUT USING HAAR DWT

We now shift our focus to this work's main contribution: fast and efficient secure (2+1)-party evaluation of non-linear functions using DWT-based LUTs. In this section, we describe our protocols for the Haar transform before tackling the more complicated—yet, in many cases, concretely superior—protocols for the bior(5,3) transform in the next section.

Recall that given a function $F : \mathbb{R} \to \mathbb{R}$ we wish to approximate it on an interval $[A,B)$ using (ℓ,f)-bit integers. Thanks to Equation (5), Theorem 1 and Lemma 3 we have

$$\begin{aligned} F(a) &\approx \langle \vec{e}_{\bar{a}}, \vec{v} \rangle \approx \langle \vec{a}_J^{(\vec{e}_{\bar{a}})}, \vec{a}_J^{(\vec{v})} \rangle = \langle 2^{-j/2} \cdot \vec{e}_{\mathrm{msb}_J(\bar{a})}, \vec{a}_J^{(\vec{v})} \rangle \\ &= \langle \vec{e}_{\mathrm{msb}_J(\bar{a})}, 2^{-j/2} \cdot \vec{a}_J^{(\vec{v})} \rangle \approx 2^{-f} \cdot \langle \vec{e}_{\mathrm{msb}_J(\bar{a})}, \mathbf{A}_F \rangle \end{aligned}$$

Haar+Pika LUT protocol $\Pi^{\text{LUT-Haar-Pika}}_{n,\ell,J,\vec{v}}$

One-time pre-processing: Input is a function F, fixed-point parameters $\ell,f\in\mathbb{N}$, LUT quantization parameter $n\in[1..\ell)$, and DWT depth j (so that $J:=n-j$)

- Compute the real-valued compressed LUT $\vec{a}_J\in\mathbb{R}^{2^J}$ as the DWT of $\vec{v}:=\mathbb{L}^{\mathbb{R},n}(F)$ at level J under the Haar transform, and then from it compute the fixed-point LUT $\mathbf{A}_F:=\lfloor 2^{f-j/2}\cdot\vec{a}_J^{(\vec{v})}\rceil_\ell\in(\mathbb{Z}_{\ell,f})^{2^J}$

Per-run pre-processing $\mathcal{F}^{\text{pre-Haar-Pika}}_{n,\ell,J,\vec{v}}$:

- Dealer distributes sharings $[r]$ and $(\!|\vec{e}_{\text{msb}_J(r)}|\!)$ for $r\in_{\text{R}}\mathbb{Z}_{2^\ell}$, a DCF $\{\!|(x>\text{lsb}_{\ell-J}(r))\,?\,1\!:\!0|\!\}$, plus a Beaver triple for multiplying two secret scalars in $\mathbb{Z}_{2^\ell}$

Online phase: Input is $[a]$, $a\in\mathbb{Z}_{\ell,f}$, plus all pre-processing values

(1) Interactively reconstruct $x_{\neg J}:=\text{lsb}_{\ell-J}(r-a)$ and the carry-out bit $carry$ that arises in the reconstruction

(2) Non-interactively compute $[borrow]=DCFEval(\{\!|(x>\text{lsb}_{\ell-J}(r))\,?\,1\!:\!0|\!\},x_{\neg J})$

(3) Interactively reconstruct $x_J\in\mathbb{Z}_{2^J}$ from $carry$, $[r]$, and $[a]$ using $\text{msb}_J([r]-[a])+carry+[borrow]$

(4) Non-interactively compute $[\![\vec{e}_{\text{msb}_J(r)}]\!]$ via full-domain evaluation of $(\!|\vec{e}_{\text{msb}_J(r)}|\!)$ and $[\![\vec{e}_{\text{msb}_J(a)}]\!]=[\![\vec{e}_{\text{msb}_J(r)}]\!]\lll x_J$.

(5) Non-interactively perform signed extension to convert this into $[\pm\vec{e}_{\text{msb}_J(a)}]$, use component-wise summation to compute $[\pm1]$ from $[\pm\vec{e}_{\text{msb}_J(a)}]$, and then evaluate $[\pm\mathbf{A}_F[\text{msb}_J(a)]]=\langle[\pm\vec{e}_{\text{msb}_J(a)}],\mathbf{A}_F\rangle$

(6) Interactively compute the product of $[\pm\mathbf{A}_F[\text{msb}_J(a)]]$ and $[\pm1]$ using the Beaver triple to obtain $[\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)]$

(7) Return $[\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)]$

Figure 5: LUT-based function evaluation using the Haar DWT. The protocol uses three rounds of communication (Steps 1, 3 and 6)

where $\bar{a}:=\lfloor 2^f\cdot a\rceil_\ell\in\mathbb{Z}_{\ell,f}$, $\mathbf{A}_F:=\lfloor 2^f\cdot 2^{-j/2}\cdot\vec{a}_J^{(\vec{v})}\rceil_\ell\in(\mathbb{Z}_{\ell,f})^{2^J}$ and $\tilde{a}:=\text{msb}_n(\bar{a})$. We write $\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)$ for such approximation of $F(a)$ at $a\in[-2^{\ell-1},2^{\ell-1})$:

$$\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a):=\langle\vec{e}_{\text{msb}_J(\tilde{a})},\mathbf{A}_F\rangle. \tag{7}$$

Corollary 5 suggests three potential strategies to compute this inner product. Our first protocol uses Bullet 2 from the result, namely

$$\vec{a}_J^{(\vec{e}_a)}=2^{-j/2}\cdot(\vec{e}_{\text{msb}_J(r)}\lll x_J),$$

to construct a direct analog of Pika with deterministic (errorless) truncation. Figure 5 describes all of the protocol steps; we elaborate on and justify non-obvious aspects below.

One-time pre-processing phase: In the one-time pre-processing phase, some quasi-trusted entity produces the fixed-point representation of a DWT-compressed (and scaled) LUT to be used in the online phase. This phase only needs to be completed once and for all for any given function F and set of parameters j,ℓ,f,n; however, we stress that, since the entire process is imminently feasible, deterministic, and based on public values, it can be easily reproduced by anyone at any time and, consequently, does *not* require a trusted initializer.

Since our goal is to evaluate $\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)$ using Equation (7); we precompute and publish the LUT

$$\mathbf{A}_F:=\lfloor 2^f\cdot 2^{-j/2}\cdot\vec{a}_J^{(\vec{v})}\rceil_\ell\in(\mathbb{Z}_{\ell,f})^{2^J}.$$

Online evaluation phase: The online phase is where the actual computation takes place, using the pre-processed data and the shared inputs to securely evaluate the function $\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)$ on the input shares. This phase is designed to have low communication cost and round complexity, leveraging the pre-computed values to complete the function evaluation with just $2\ell+J$ bits sent by each party across three communication rounds.

In the protocol, Steps 1–3 compute the rotation distance x_J at level J using a two-round protocol; the remaining steps complete the calculation exactly as in Pika, using the above-computed LUT. The computation of x_J proceeds in two steps because the parties must reconstruct $\text{lsb}_{\ell-J}(x)$ before they can compute and reconstruct $x_J=\text{msb}_J(x)+\big(\text{lsb}_{\ell-J}(x)<\text{lsb}_{\ell-J}(r)\big)$.

Cost analysis: The per-run preprocessing values have size about $\lambda\cdot(J-\lg\lambda)$ bits for the DPF and $2(\ell-J)\cdot(\lambda+\ell-J)$ bits for the DCF, plus 4ℓ values for the additive sharing and Beaver triple, where $\lambda\in\mathbb{N}$ is the seed length of the PRG. This gives a total precomputation size in $O\big(\ell\cdot(\lambda+\ell)\big)$. Computation cost for the dealer is a modest $\Theta(\ell-\lg\lambda)$ PRG invocations and XORs.

As for the online phase, the parties exchange one ℓ-bit value across Steps 1 and 3, and two ℓ-bit values in Step 6, for a total online communication of just 3ℓ bits in either direction. The DCF evaluation in Step 2 requires $\Theta(\ell)$ work and the full-domain evaluation in Step 4 requires $\Theta(2^{J-\lg\lambda})$ work. Essentially all of the remaining computation cost come from the inner product in Step 5, which uses at most 2^J many ℓ-bit additions.

Theorem 9. *Protocol $\Pi^{LUT\text{-}Haar}_{n,\ell,J,\vec{v}}$ securely evaluates $\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)$ in the presence of semi-honest non-colluding parties in the $\mathcal{F}^{pre\text{-}Haar\text{-}Pika}_{n,\ell,J,\vec{v}}$-hybrid model.*

We defer proof of Theorem 9 to Appendix D. We highlight that all the approximation of the DWTs happen outside the secure computation protocol (refer to the ideal functionality in Figure 8). The approximation can be independently studied with the error being a tunable parameter as a function of J,n. Section 7.1 contains the experimental numbers for specific parameter choices.

6 LUTS USING bior(5,3) DWT

Similar to Section 5, we approximate a function $F:\mathbb{R}\to\mathbb{R}$ on an interval $[A,B)$ by using (ℓ,f)-bit integers and making use of the bior(5,3) transformation. Indeed, following Equation (5), Theorem 2 and Lemma 6 we have

$$\begin{aligned}F(a)&\approx\langle\vec{e}_{\tilde{a}},\vec{v}\rangle\approx\langle\vec{a}_J^{(\vec{e}_{\tilde{a}})},\vec{a}_J^{(\vec{v})}\rangle=\langle c_{J,0}\cdot\vec{e}_{\text{msb}_J(\bar{a})}+c_{J,1}\cdot(\vec{e}_{\text{msb}_J(\bar{a})}\ggg 1),\vec{a}_J^{(\vec{v})}\rangle\\&=\langle\big(2^j-\text{lsb}_j(\tilde{a})\big)\cdot 2^{-3j/2}\cdot\vec{e}_{\text{msb}_J(\bar{a})}+\text{lsb}_j(\tilde{a})\cdot 2^{-3j/2}\cdot(\vec{e}_{\text{msb}_J(\bar{a})}\ggg 1),\vec{a}_J^{(\vec{v})}\rangle\\&=\langle\vec{e}_{\text{msb}_J(\bar{a})},2^{-j/2}\cdot\vec{a}_J^{(\vec{v})}\rangle+\text{lsb}_j(\tilde{a})\cdot\langle\vec{e}_{\text{msb}_J(\bar{a})},2^{-3j/2}\cdot((\vec{a}_J^{(\vec{v})}\lll 1)-\vec{a}_J^{(\vec{v})})\rangle\\&\approx 2^{-f}\cdot\big(\langle\vec{e}_{\text{msb}_J(\bar{a})},\mathbf{A}_{F,c_0}\rangle+\text{lsb}_j(\tilde{a})\cdot\langle\vec{e}_{\text{msb}_J(\bar{a})},\mathbf{A}_{F,c_1}\rangle\big).\end{aligned}$$

where $\bar{a}:=\lfloor 2^f\cdot a\rceil_\ell\in\mathbb{Z}_{\ell,f}$, $\mathbf{A}_{F,c_0}:=\lfloor 2^f\cdot 2^{-j/2}\cdot\vec{a}_J^{(\vec{v})}\rceil_\ell$, and $\mathbf{A}_{F,c_1}:=\lfloor 2^f\cdot 2^{-3j/2}\cdot((\vec{a}_J^{(\vec{v})}\lll 1)-\vec{a}_J^{(\vec{v})})\rceil_\ell\in(\mathbb{Z}_{\ell,f})^{2^J}$ and $\tilde{a}:=\text{msb}_n(\bar{a})$. We bior(5,3)+**Grotto LUT protocol** $\Pi^{\text{LUT-bior(5,3)-Grotto}}_{n,\ell,J,\vec{v}}$

One-time pre-processing: Input is a function F, fixed-point parameters $\ell, f \in \mathbb{N}$, LUT quantization parameter $n \in [1..\ell)$, and DWT depth j (so that $J := n - j$)

- Compute the real-valued compressed LUT $\vec{a}_J \in \mathbb{R}^{2^J}$ as the DWT of $\vec{v} := \mathbb{L}^{\mathbb{R},n}(F)$ at level J under the bior(5,3) transform, and then from it compute the fixed-point LUTs
  - $\mathbf{A}_{F,c_0} := \lfloor 2^{f-j/2} \cdot \vec{a}_J^{(\vec{v})} \rceil_\ell \in (\mathbb{Z}_{\ell,f})^{2^J}$
  - $\mathbf{A}_{F,c_1} := \lfloor 2^{f-3j/2} \cdot \big((\vec{a}_J^{(\vec{v})} \lll 1) - \vec{a}_J^{(\vec{v})} \big) \rceil_\ell \in (\mathbb{Z}_{\ell,f})^{2^J}$

Per-run pre-processing $\mathcal{F}^{\textbf{pre-}\text{bior(5,3)}\textbf{-Grotto}}_{n,\ell,J,\vec{v}}$**:**

- Dealer distributes sharings $[r]$, $[\mathrm{lsb}_j(\mathrm{msb}_n(r))]$ and $(\!|\vec{e}_r|\!)$ for $r \in_{\mathrm{R}} \mathbb{Z}_{2^\ell}$, two DCFs $\{\!\!\{(x > \mathrm{lsb}_{\ell-n}(r))\,?\,1:0\}\!\!\}$ and $\{\!\!\{(x > \mathrm{lsb}_{\ell-J}(r))\,?\,2^j:0\}\!\!\}$, plus a generalized Beaver tuple for equations of the form $[u]\cdot([c_1]\cdot[x]+[c_0])$ with arithmetic in $\mathbb{Z}_{2^\ell}$.

Online phase: Input is $[a]$, $a \in \mathbb{Z}_{\ell,f}$, plus all pre-processing values

(1) Interactively reconstruct $x := r - a$ from $[r]$ and $[a]$, and then non-interactively compute $x_{\neg J} := \mathrm{lsb}_{\ell-J}(x)$ and $x_{\neg n} := \mathrm{lsb}_{\ell-n}(x)$ from x

(2) Non-interactively compute
  - $[borrow_n] = DCFEval(\{\!\!\{(x > \mathrm{lsb}_{\ell-n}(r))\,?\,1:0\}\!\!\}, x_{\neg n})$ and
  - $[borrow_J] = DCFEval(\{\!\!\{(x > \mathrm{lsb}_{\ell-J}(r))\,?\,2^j:0\}\!\!\}, x_{\neg J})$ and
  - $[\mathrm{lsb}_j(\mathrm{msb}_n(a))] := [\mathrm{lsb}_j(\mathrm{msb}_n(r))] - \big(\mathrm{lsb}_j(\mathrm{msb}_n(x)) + [borrow_n] - [borrow_J]\big) \in \mathbb{Z}_{2^\ell}$,

  and then compute
  - $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$ using Grotto's segment-parity algorithm on input $(\!|\vec{e}_r|\!)$, $(P_i := i \cdot 2^j \mid i,\ldots,2^J - 1)$, and x,
  - perform signed extension to convert this into $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$,
  - use component-wise summation to compute $[\pm 1]$ from $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$, and then
  - evaluate $[\pm\mathbf{A}_{F,c_0}[\,\mathrm{msb}_J(a)]] = \langle [\pm\vec{e}_{\mathrm{msb}_J(a)}], \mathbf{A}_{F,c_0} \rangle$ and $[\pm\mathbf{A}_{F,c_1}[\mathrm{msb}_J(a)]] = \langle [\pm\vec{e}_{\mathrm{msb}_J(a)}], \mathbf{A}_{F,c_1} \rangle$

(3) Interactively compute the sign-corrected linear evaluation $[\pm 1]\cdot([\pm\mathbf{A}_{F,c_1}[\mathrm{msb}_J(a)]]\cdot[\mathrm{lsb}_j(\mathrm{msb}_n(a))] + [\pm\mathbf{A}_{F,c_0}[\mathrm{msb}_J(a)]])$ using the generalized Beaver triple to obtain $[\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)]$

(4) Return $[\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)]$

Figure 6: LUT-based function evaluation using the bior(5,3) **DWT. The protocol uses two rounds of communication (Steps 1 and 3)**

write $\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)$ for such approximation of $F(a)$ at $a \in [-2^{\ell-1}, 2^{\ell-1})$. That is,

$$\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a) := \langle \vec{e}_{\mathrm{msb}_J(\tilde{a})}, \mathbf{A}_{F,c_0} \rangle + \mathrm{lsb}_j(\tilde{a}) \cdot \langle \vec{e}_{\mathrm{msb}_J(\tilde{a})}, \mathbf{A}_{F,c_1} \rangle. \tag{8}$$

Corollary 8 suggests three potential strategies compute Equation (8). The protocol we present here uses Bullet 3 from the result, namely

$$\vec{a}_J^{(\vec{e}_{\tilde{a}})} = c_{J,0} \cdot \tilde{e}_J + c_{J,1} \cdot (\tilde{e}_J \ggg 1), \tag{9}$$

where $c_{J,0}, c_{J,1} \in \mathbb{R}^{2^J}$ are as defined in Lemma 6, and where

$$\tilde{e}_J := \big[\bigoplus_{i=k\cdot 2^j}^{(k+1)\cdot 2^j - 1} \vec{e}_a[i] \,\big|\, k = 0,\ldots,2^J - 1 \big],$$

with Grotto's parity-segment algorithm to construct a protocol that avoids the need for truncation altogether. Figure 6 describes all of the protocol steps; we elaborate on and justify non-obvious aspects of these steps below.

One-time pre-processing phase: As with the Haar transform, the pre-processing phase of out bior(5,3) protocol consists of a quasi-trusted entity producing fixed-point representations of DWT-compressed LUTs to be used in the online phase. Recall that our goal is to evaluate $\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)$ (Equation (8)) using Equation (9).

Online evaluation phase: The online phase diverges from the Haar protocol in two main ways. The first significant difference is that it uses Grotto's segment-parity algorithm to arrive at $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$. The segment-parity algorithm takes as inputs a DPF key dpf, a monotone increasing sequence S of segment endpoints in the domain of dpf, and a cyclic rotation distance x. It simulates the effect of invoking full-domain evaluation to obtain $[\![\vec{e}_r]\!]$, applying the rotation $[\![\vec{e}_a]\!] = [\![\vec{e}_r]\!] \lll x$, and then computing the parity of every segment defined by consecutive endpoints in S in the resulting vector. However, it computes this using $o(\ell \cdot 2^J)$ PRG evaluations and bitwise operations [25, Theorem 4.1].[4]

Cost analysis: The per-run preprocessing values have size about $\lambda \cdot (\ell - \lg\lambda)$ bits for the DPF, $2\cdot(\ell - n)\cdot(\lambda + \ell - n)$ and $2\cdot(\ell - J)\cdot(\lambda + \ell - J)$ for the two DCFs, and 10ℓ bits additive shares and generalized Beaver tuple, where $\lambda \in \mathbb{N}$ is the seed length of the PRG. This gives a total precomputation size in $O\big(\ell\cdot(\lambda+\ell)\big)$. Computation cost for the dealer is a modest $\Theta(\ell - \lg\lambda)$ PRG invocations and XORs.

As for the online phase, the parties exchange one ℓ-bit value in Step 1 and an additional four ℓ-bit values in Step 3, for a total online communication of just 5ℓ bits in either direction. In Step 2, the DCF evaluations require $\Theta(\ell - J)$ and $\Theta(\ell - n)$ work, respectively, while the segment-parity computation requires $\Theta(j \cdot 2^{J - \lg\lambda})$ work. Essentially all of the remaining computation cost comes from the inner products in Step 5, which is $\Theta(2^J)$ work. Below we state our main theorem and defer the proof to Appendix E.

Theorem 10. *Protocol* $\Pi^{LUT\text{-bior(5,3)}}_{n,\ell,J,\vec{v}}$ *securely evaluates* $\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)$ *in the presence of semi-honest non-colluding parties in the* $\mathcal{F}^{pre\text{-bior(5,3)-}Grotto}_{n,\ell,J,\vec{v}}$*-hybrid model.*

6.1 Methods for computing $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$

The Haar protocol in Figure 5 uses an errorless variant of Pika's approach to computing $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$, whereas the bior(5,3) protocol in Figure 6 opts for the parity-segment approach from Grotto. Here, we briefly explain how the two methods work and differ.

For the errorless Pika-like variant, we employ the formula for x_J as given in Lemmas 4 and 7, using a dealer-provided DCF to evaluate the required integer comparison. The benefit of this approach is that it uses small DPFs with computation costs that are concretely lower than the Grotto approach. The drawback is that the comparison must occur *before* the reconstruction of $\mathrm{msb}_J(x)$, necessitating a dedicated round of interaction. The resulting protocol uses three rounds in total.

For the Grotto-like variant, we instead employ the segment-parity algorithm to compute $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$ directly from $(\!|\vec{e}_r|\!)$ and x. The benefit of this approach is that it avoids the need for

[4]In our case, the cost works out to $\Theta(j \cdot 2^J)$ PRG evaluations and bitwise operations.

an additional round of interaction, which is significant in high-latency WAN scenarios. The result is a protocol with just two rounds of interaction. The drawback is that while asymptotically similar, the parity-segment approach is concretely slower, and this increased computation cost must be weighed against the lower round complexity.

6.2 Large Domain Sizes

The Grotto-based variants of both protocols naturally extend to efficient evaluation over massive domains, at least for certain functions of interest, notably including most non-linear activation functions like the ones we consider in our experimental evaluation. In particular, given a function $f : \mathbb{R} \to \mathbb{R}$ that admits a piecewise approximation in which most "pieces" are well-approximated by low-degree polynomials, we can adapt the segment-parity computation by inputting an augmented list of segment endpoints. This list consolidates each polynomial-approximatable piece into a single segment while dividing the non-polynomial segment(s) into equidistant points (cf. Corollary 5, Bullet 3 and Corollary 8, Bullet 3). By doing so, we extend Grotto's approach, which works well for polynomial segments, and complement it with our DWT approach to accurately model any highly non-linear portions beyond the reach of low-degree polynomial approximations. In particular, for a DWT at level J the augmented segment-parity computation grows from 2^J segments to 2^J+2 segments, a nominal overhead of a factor $1+2^{-J}$. When the tails are constant (as in sigmoid), there is essentially no additional overhead; for functions that converge to $f(x)=x$, the LUT returns coefficients of a linear polynomial, and there is slight additional communication overhead to obliviously evaluate that polynomial. In our experiments, the cost difference between evaluating on the full, 64-bit domain versus over a small domain centered at the origin were consistently dwarfed by the variance in running times.

7 EVALUATION

We run our experiments on a server equipped with an Intel® Core™ i9-13900KS processor and 128 GiB of RAM, running Ubuntu 23.04 (lunar) in headless mode. We run the dealer and each online party in its own Docker container and control simulated network parameters between them using traffic control (`tc`). The source code to reproduce the experiments is open sourced at https://github.com/NillionNetwork/WaveHelloToPrivacy. All numbers are generated after averaging for a 100 trials. All LAN experiments are run with 5000 Mbit bandwith and 0.2 ms latency, whereas WAN experiments are run with 300 Mbit bandwith and 70 ms latency. A brief overview of our experimental evaluation is given below:

(1) We analyze the performance of Haar and Biorthogonal (bior(5,3)) DWTs compared to traditional quantization techniques – Section 7.1 and Table 1
(2) The trade-off between compression and error by studying a range of compression levels (J) – Section 7.2 and Figure 7
(3) Computation run-time for varying compression levels (J) and fixed input bit-widths (n) (in both LAN and WAN) – Section 7.3, Figure 7, and Appendix J
(4) Comparison against existing methods such as Curl, Pika, and Ripple – Appendix K

A note on DWT implementation: To implement DWTs we used the PyWavelets package [16] for Python. In this package the Haar DWT is referred to as "haar" while the bior(5,3) DWT corresponds to "bior2.2". The difference in the notation occurs because we consider the lengths of the low and high-pass filters (5 and 3) while in the PyWavelets package they consider the number of vanishing moments of those filters (2 and 2).

7.1 Activation Functions

In Table 1, we compare Haar and bior(5,3) DWT with quantization, i.e., truncating trailing bits of the number and evaluating the function at the remaining value. The precision is $f=24$, and all LUT outputs have $l=64$ bits. The input bit width n depends on the function's domain. For a real-valued univariate function $g(x)$ over domain $[a,b]$, we sample 2^n equidistant values to construct quantized, Haar, and Biorthogonal LUTs. We report the Mean and Max Absolute Errors attained using these compression techniques, focusing on ML activation functions (https://paperswithcode.com/methods/category/activation-functions) that are not piece-wise linear. We exclude ReLU and similar activations as they can be calculated accurately via splines, focusing on more complex non-linearities. To set lookup table domains and sizes, we use a threshold $T=2^{-18}\approx 3.81\times10^{-6}$, choosing domains as powers of 2 so errors outside fall under this threshold.

For j-bit compression in quantization, the quantized LUT is built by evaluating the function at values after truncating the j trailing bits of the input. That is, we evaluate the function at 2^J equidistant points, where $J=n-j$, each point with j trailing bits equal to 0. On the other hand, for j-bit compression in DWT, we apply the DWT transform j times. We end up with 2^J approximation coefficients, which becomes our LUT. We then normalize the LUT. In Haar, this normalization consists of multiplication with $2^{-j/2}$. This value comes from normalizing to 1 the approximation coefficients of the one-hot vector after Haar DWT is applied j times.

For GeLU and tanh, we focus on the domain $[-8,8]$. The maximum absolute error we get outside of this domain for these functions are 5.33×10^{-15} and 2.25×10^{-7} respectively. For sigmoid, SiLU, softplus and Mish we use the domain $[-16,16]$ resulting in errors of 1.13×10^{-7}, 1.80×10^{-6}, 1.13×10^{-7}, and 1.80×10^{-6} respectively. In the case of SELU, we limit our attention to $[-16,0]$ as it is linear for positive numbers, getting an error of 1.98×10^{-7} for values less than -16. For exponential, it is also enough to consider $[-16,0]$ as $e^{-16}\approx 1.13\times10^{-7}$, which is the maximum absolute error incurred. For reciprocal, we only consider the domain $[1,64]$ because this resembles the range of input that it's going to take in a softmax function of up to 64 values.

We observe that Haar DWT is able to compress a single extra bit more than quantization with about the same error, while Biorthogonal DWT approximately halves the bit size of the LUT for less error. This reduction in LUT size leads to considerable speedup of the lookup table evaluation and the key-size required.

Function	Method	Bitlength		Absolute Error	
		original	LUT	mean	max
GeLU	**Q**	28	23	5.12×10^{-7}	2.14×10^{-6}
	H		22	5.11×10^{-7}	2.18×10^{-6}
	B		12	9.36×10^{-8}	1.02×10^{-6}
sigmoid	**Q**	29	22	1.48×10^{-7}	1.95×10^{-6}
	H		21	1.39×10^{-7}	1.96×10^{-6}
	B		11	1.41×10^{-7}	2.00×10^{-6}
tanh	**Q**	28	22	2.64×10^{-7}	3.81×10^{-6}
	H		22	1.39×10^{-7}	1.94×10^{-6}
	B		12	8.17×10^{-8}	1.06×10^{-6}
SiLU/Swish	**Q**	29	24	5.01×10^{-7}	2.09×10^{-6}
	H		23	5.04×10^{-7}	2.12×10^{-6}
	B		12	1.30×10^{-7}	2.54×10^{-6}
softplus	**Q**	29	23	9.69×10^{-7}	3.81×10^{-6}
	H		23	4.88×10^{-7}	1.94×10^{-6}
	B		12	1.06×10^{-7}	1.27×10^{-6}
SELU	**Q**	28	23	1.31×10^{-7}	3.31×10^{-6}
	H		22	1.26×10^{-7}	3.36×10^{-6}
	B		12	7.71×10^{-8}	2.11×10^{-6}
Mish	**Q**	29	24	5.07×10^{-7}	2.07×10^{-6}
	H		23	5.05×10^{-7}	2.10×10^{-6}
	B		12	1.28×10^{-7}	3.27×10^{-6}
exponential	**Q**	28	22	1.47×10^{-7}	3.81×10^{-6}
	H		22	8.19×10^{-8}	1.94×10^{-6}
	B		12	5.39×10^{-8}	1.21×10^{-6}
reciprocal	**Q**	29	23	4.74×10^{-8}	3.76×10^{-6}
	H		22	4.05×10^{-8}	1.94×10^{-6}
	B		13	3.64×10^{-8}	2.72×10^{-6}
Legend:	**Q**: Quantization; **H**: Haar; **B**: bior(5,3)				

Table 1: Accuracy by function and approx method for select LUT sizes.

7.2 Compression Error

In Figure 7, we plot the mean absolute error vs. compression level for the selected functions for quantization, Haar DWT, and Biorthogonal DWT. This data is plot on the primary axis (left hand y-axis). We observe that Haar DWT is usually better than quantization by a constant factor, but Biorthogonal DWT is asymptotically better. Using Biorthogonal DWT, the lookup table can be shrunk to around half the bit-width with error in the order of the precision ($2^{-24} \approx 10^{-7}$). For small depths (large lookup table sizes – J close to n), Biorthogonal DWT experiences boundary effects which explains the increased error. We see similar patterns for all the functions plotted, however, there are minute differences like reciprocal function experiences higher border effects for small compression depth.

7.3 Runtime Experiments

In Table 1, we empirically report the compression for a given threshold of acceptable error. In practice however, the error threshold depends on the application and thus in Figure 7 we use a dual axis to present the trade-off in our system. Using a smaller value of J (thus the LUT table is compressed a lot) results in really efficient protocols (right-hand y-axis) but at the cost of increased absolute error (left-hand y-axis). On the other extreme, using large values of J (closer to n) result in a very accurate function evaluation (close to 8 decimal digits precision) at the cost of a higher runtime. Our implementation processes messages in a streaming fashion to effectively hide latency; this is evident in the plots, where the WAN running times initially are dominated by communication latency, while the clear exponential trend in the computation cost eventually dominates in both the LAN and WAN settings.

We restrict the value of J between 10 and 24 (as representative compression levels from Table 1). However, we present the full data by varying n between 28 and 32, for $\ell=64$ and the same range for J in Appendix I. For the experiment, we first create pre-processing material and measure the total time for 1000 function evaluations (average time for 100 runs is reported) for each n and J. We report the total time of the runs. Note that while the runtime changes, the online communication is always the same. For Haar, it is 24 bytes per evaluation and for bior(5,3) it is 40 bytes for all n and J.

In Appendix K, we compare our approach relative to prior works such as Curl [24], Pika [30], and Ripple [10].

8 RELATED WORK

This work builds upon two closely related systems, Pika [30] and Grotto [25], both of which have contributed to the advancement of secure multi-party computation (MPC) using function secret sharing (FSS). Pika leverages FSS to evaluate a lookup table (LUT) within a (2+1) party setup and extends this semi-honest protocol to achieve malicious security by stating and proving a generalization of the Schwartz-Zippel Lemma. Despite its innovative approach, Pika faces scalability challenges due to the computational overhead associated with the naïve FSS EvalAll method, which incurs an $O(N)$ complexity where N is the size of the LUT. Addressing this limitation, Grotto introduces the parity-segment tree, a novel data structure optimized for efficiently answering parity queries over substrings of a binary string. This structure reduces the computational cost for certain operations, enabling efficient computation at the cost of a single distributed point function (DPF) rather than multiple distributed correlated functions (DCF).

There has been a plethora of work in the (2+1) party and 3-party settings focusing on non-linear functions [4, 21, 28, 29, 31, 32]. Particularly, there has been significant progress in improving the cost of FSS-based protocols – notable examples include SIRNN [23], LLAMA [12] and [2]. SIRNN, in particular, employs LUTs differently than this work. It uses LUTs to achieve an initial approximation of mathematical functions, which is then refined using iterative algorithms such as Goldschmidt's iterations. While larger LUTs yield more precise results, the communication overhead in secure protocols increases linearly with the LUT size. Furthermore, the rise of large language models (LLMs) and the transformer architecture has spurred interest in the secure evaluation of these models, with works

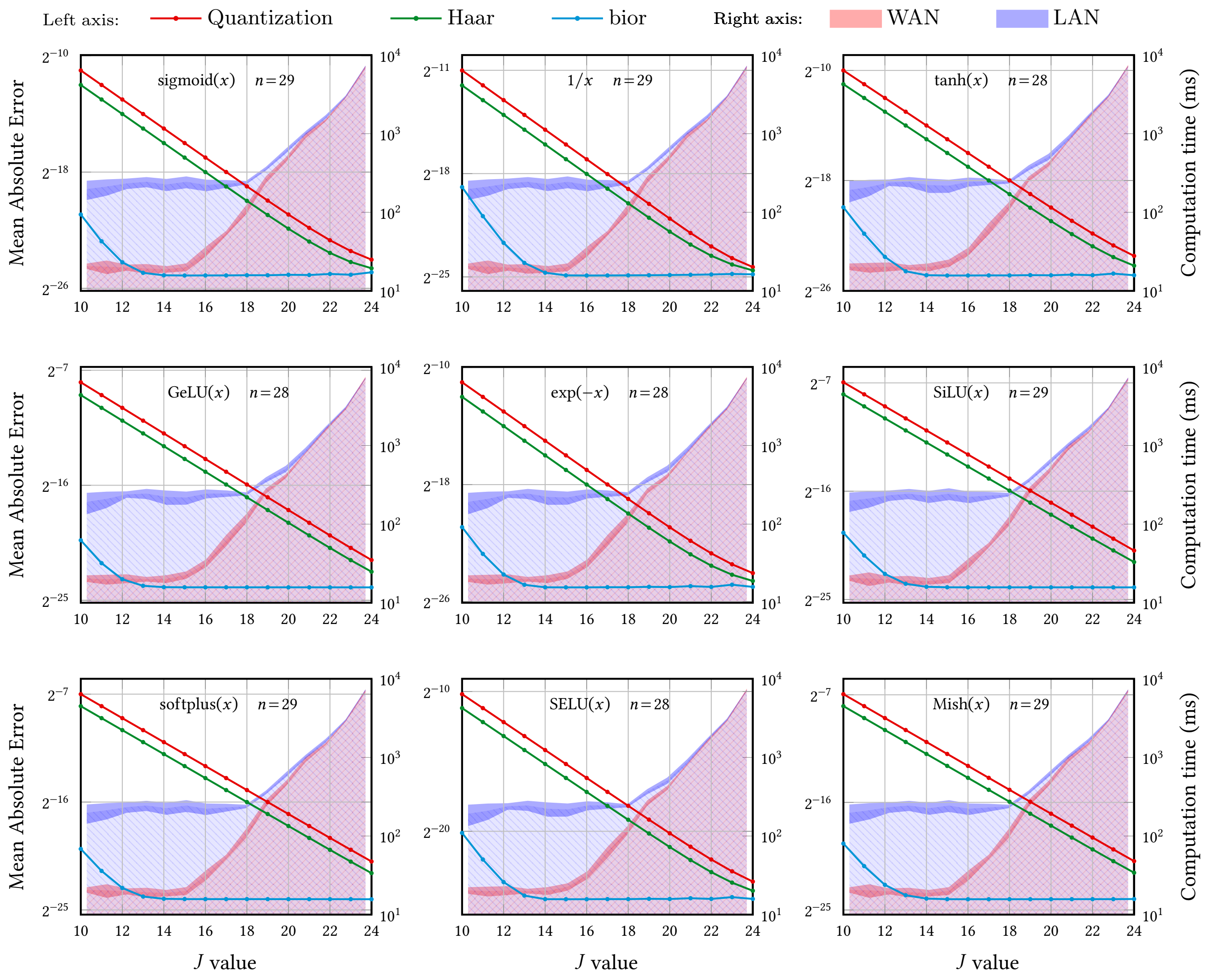

Figure 7: This figure shows the inherent accuracy-performance trade-off in this work. The x-axis is the DWT level J (from 10 to $n{-}1$) with each subplot for a single function. On the primary axis, we have the mean absolute error (line plots) and on the secondary axis (area charts) we have the total time required for 1000 function evaluations using the Haar DWT (averaged over 100 trials). The dark shaded band indicates one standard deviation above and below the average. We use Table 1 as the guide for the choice of n values for functions as well as the range of J values considered. As the primary and secondary axis are independent there is no meaning to the intersection of line-plots with the area charts.

such as SIGMA [11], MPCFormer [17], Puma [8], and Bolt [18] focusing on this area. These efforts aim to adapt and optimize secure computation techniques to efficiently handle the complexities and non-linearities of transformers.

An orthogonal yet related line of research explores the use of GPUs to accelerate secure computation. Systems such as CryptGPU [26], Piranha [34], and Orca [13] have investigated this approach, with Orca specifically focusing on enhancing FSS-based 2PC protocols using GPUs. This exploration highlights the potential for leveraging parallel processing capabilities to improve the performance of secure computation.

Furthermore, two concurrent works, Ripple [10] and Curl [24], have explored the use of discrete wavelet transforms (DWTs) in secure computation. Ripple applies similar techniques within the context of homomorphic encryption, utilizing programmable bootstrapping, while Curl builds upon Crypten to implement more efficient LUTs using DWTs. Curl demonstrates the effectiveness of this approach by achieving secure inference on a range of LLMs but lacks the use of DPFs, DCFs, and the parity segmentation that is critical in extending these techniques to large domain use-cases.

9 FUTURE DIRECTIONS

In this study, we concentrate on the fundamentals of DWT within the context of LSSS based secure computation. However, this research raises several open questions and suggests multiple avenues for future investigation. We outline these briefly here, with the hope that subsequent studies will provide precise constructions, conduct empirical evaluations, and develop into substantial, concrete bodies of work.

Improved Adversarial Models. Extending DWT-based protocols to withstand malicious adversarial models is crucial for ensuring the robustness and security of cryptographic systems. Investigating methodologies akin to Pika [30] for enhancing protocol security under malicious adversaries or MPC with a friend adversarial models [15] could contribute to the development of more resilient and trustworthy secure computation frameworks. Future research directions may include concrete protocol constructions, other Rings and Fields.

Families of Wavelet Transforms. Apart from the Haar and bior(5,3) wavelets considered in this work, other wavelet families may offer interesting possibilities for improving efficiency or addressing specific application requirements. Exploring alternative families, such as Daubechies or Symlet wavelets, in terms of their computational properties and suitability for secure computation tasks could provide valuable insights into their potential advantages and limitations.

Function Evaluations over Non-linearly Spaced Intervals. While our study focuses on discrete, equally spaced evaluation of functions, exploring techniques for more adaptive sampling based on the characteristics of non-linear functions could significantly enhance computation efficiency. Investigating the integration of *packet DWT* techniques [22] to achieve this is a promising avenue for future research, potentially involving the development of algorithms or heuristics to dynamically adjust sampling intervals based on local function behavior.

PIR Protocols using N-D Wavelet Theory. Adapting N-Dimensional wavelet theory to Private Information Retrieval (PIR) protocols presents a direction for enhancing privacy-preserving data retrieval mechanisms. Chor's protocol [6], with 2^k servers can benefit from a DWT based look-up where a standard basis vector is secret shared across each of the k dimensions to select a single entry in the k-dimension hypercube. In other words, leveraging DWT-based lookup tables within PIR frameworks could lead to more efficient and secure solutions for accessing distributed data while preserving user privacy.

Integration with Chinese Remainder Theorem (CRT). Combining DWT techniques with the CRT may enable possibilities for efficient computation over large domains represented by composite moduli. Partitioning large domain computations into smaller DWT LUTs of arithmetic functions could offer significant efficiency and scalability gains for CRT-based cryptographic protocols. Future research could focus on specific algorithms or protocols to combine DWT with the CRT and the experimental validation of their performance in cryptographic applications.

Applications to FHE/TFHE. Exploring the integration of DWT techniques with Fully Homomorphic Encryption (FHE) or TFHE schemes offers avenues for enhancing the efficiency and scalability of secure computation protocols. Investigating how DWT-based approaches can complement or improve existing encryption techniques could lead to advancements in privacy-preserving computation across various domains. Key directions include designing DWT-enhanced FHE or TFHE protocols and evaluating their real-world performance.

Applications to Secure Matrix Multiplications. Extending the application of DWT-based techniques to secure matrix multiplications could yield efficient solutions for cryptographic protocols relying on such operations [5, 35]. Leveraging DWT for sampling across various dimensions within matrix operations holds potential for optimizing computations, particularly in scenarios involving high-dimensional data processing.

Extensions to Large Domains. While our method is well suited for piecewise linear functions, the technique applies generally. The Segment-parity algorithm works for arbitrary splitting of functions and thus generalizes well. However, the lower the degree of the polynomial which is used to approximate a piece, the smaller is the relative contribution to the overall run-time. Future research directions may focus on these types of non-linear functions and study this problem of evaluation over large domains more generally.

10 CONCLUSION

In conclusion, this work explores the novel approach to secure multiparty computation by leveraging discrete wavelet transforms (DWTs) to create compact and high-fidelity lookup tables (LUTs) for approximating nonlinear functions over large domains. The proposed protocols demonstrate significant improvements in run-time and accuracy, particularly in the context of evaluating nonlinear activation functions popular in deep neural networks. The experimental results confirm the practical benefits of these methods, showing multiple order of magnitude run-time improvements for similar LUT sizes. This work not only enhances the feasibility of secure MPC for complex functions but also opens new avenues for further research and application of DWTs in secure computation.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support and resources of Nillion for this research (and the resources provided by Ryan's machine). Ryan Henry acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) through its Discoverg Grants program, and the National Cybersecurity Consortium (NCC) through award 2024-1346.

REFERENCES

[1] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Annual International Cryptology Conference*. Springer, 1991, pp. 420–432.
[2] E. Boyle, N. Chandran, N. Gilboa, D. Gupta, Y. Ishai, N. Kumar, and M. Rathee, "Function secret sharing for mixed-mode and

fixed-point secure computation," in *Advances in Cryptology—EUROCRYPT*, 2021.

[3] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing: Improvements and extensions," in *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[4] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, "Astra: High throughput 3pc over rings with application to secure prediction," in *ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2019.

[5] H. Chen, M. Kim, I. P. Razenshteyn, D. Rotaru, Y. Song, and S. Wagh, "Maliciously secure matrix multiplication with applications to private deep learning," in *Advances in Cryptology—ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 12493. Springer, 2020, pp. 31–59. [Online]. Available: https://doi.org/10.1007/978-3-030-64840-4_2

[6] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM (JACM)*, 1998.

[7] I. Daubechies, *Ten Lectures on Wavelets.* Society for Industrial and Applied Mathematics, 1992. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611970104

[8] Y. Dong, W. jie Lu, Y. Zheng, H. Wu, D. Zhao, J. Tan, Z. Huang, C. Hong, T. Wei, and W. Chen, "Puma: Secure inference of llama-7b in five minutes," 2023. [Online]. Available: https://arxiv.org/abs/2307.12533

[9] W. Du and M. J. Atallah, "Protocols for secure remote database access with approximate matching," in *E-Commerce Security and Privacy (Part II)*, ser. Advances in Information Security, vol. 2, February 2001, pp. 87–111.

[10] C. Gouert, M. Ugurbil, D. Mouris, M. de Vega, and N. G. Tsoutsos, "Ripple: Accelerating Programmable Bootstraps for FHE with Wavelet Approximations," in *International Conference on Information Security.* Springer, 2024, pp. 1–20.

[11] K. Gupta, N. Jawalkar, A. Mukherjee, N. Chandran, D. Gupta, A. Panwar, and R. Sharma, "SIGMA: Secure GPT inference with function secret sharing," in *Privacy Enhancing Technologies Symposium (PETS)*, 2024.

[12] K. Gupta, D. Kumaraswamy, N. Chandran, and D. Gupta, "LLAMA: A low latency math library for secure inference," in *Privacy Enhancing Technologies Symposium (PETS)*, 2022.

[13] N. Jawalkar, K. Gupta, A. Basu, N. Chandran, D. Gupta, and R. Sharma, "Orca: FSS-based secure training and inference with GPUs," in *IEEE Symposium on Security and Privacy (S&P)*, 2024.

[14] A. Jensen and A. Cour-Harbo, *Ripples in Mathematics: The Discrete Wavelet Transform.* Springer Berlin Heidelberg, 2011. [Online]. Available: https://books.google.pt/books?id=nMIPBwAAQBAJ

[15] B. Karmakar, N. Koti, A. Patra, S. Patranabis, P. Paul, and D. Ravi, "Asterisk: Super-fast MPC with a friend," *IACR Cryptol. ePrint Arch.*, p. 1098, 2023. [Online]. Available: https://eprint.iacr.org/2023/1098

[16] G. R. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O&#8217;Leary, "Pywavelets: A python package for wavelet analysis," *Journal of Open Source Software*, vol. 4, no. 36, p. 1237, 2019. [Online]. Available: https://doi.org/10.21105/joss.01237

[17] D. Li, R. Shao, H. Wang, H. Guo, E. P. Xing, and H. Zhang, "Mpcformer: fast, performant and private transformer inference with mpc," 2023. [Online]. Available: https://arxiv.org/abs/2211.01452

[18] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "BOLT: Privacy-preserving, accurate and efficient inference for transformers," Cryptology ePrint Archive, Paper 2023/1893, 2023, https://eprint.iacr.org/2023/1893. [Online]. Available: https://eprint.iacr.org/2023/1893

[19] M.-A. Parseval, *Sur la représentation des fonctions d'une variable par les sommes finies.* Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, 1899, vol. 129.

[20] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved mixed-protocol secure two-party computation," in *USENIX Security Symposium (USENIX)*, 2021.

[21] A. Patra and A. Suresh, "Blaze: Blazing fast privacy-preserving machine learning," in *Symposium on Network and Distributed System Security (NDSS)*, 2020.

[22] D. B. Percival and A. T. Walden, *The Discrete Wavelet Packet Transform*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2000, p. 206–254.

[23] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi, "Sirnn: A math library for secure rnn inference," in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[24] M. B. Santos, D. Mouris, M. Ugurbil, S. Jarecki, J. Reis, S. Sengupta, and M. de Vega, "Curl: Private LLMs through Wavelet-Encoded Look-Up Tables," in *Proceedings of the Conference on Applied Machine Learning in Information Security*, Arlington, Virginia, USA, October 24–25, 2024, pp. 1–31.

[25] K. Storrier, A. Vadapalli, A. Lyons, and R. Henry, "Grotto: Screaming fast (2+1)-PC for $\mathbb{Z}_{2^n}$ via (2,2)-DPFs," in *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[26] S. Tan, B. Knott, Y. Tian, and D. J. Wu, "Cryptgpu: Fast privacy-preserving machine learning on the GPU," in *IEEE Symposium on Security and Privacy (S&P)*, 2021.

[27] P. J. Van Fleet, *Discrete Wavelet Transformations: An Elementary Approach with Applications (Second Edition).* Wiley, April 2019.

[28] S. Wagh, "New Directions in Efficient Privacy Preserving Machine Learning," Ph.D. dissertation, Princeton University, 2020.

[29] ——, "BarnOwl: Secure Comparisons using Silent Pseudorandom Correlation Generators," Cryptology ePrint Archive, Report 2022/800, 2022, https://eprint.iacr.org/2022/800.pdf.

[30] ——, "Pika: Secure computation using function secret sharing over rings," in *Privacy Enhancing Technologies Symposium (PETS)*, 2022.

[31] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-Party Secure Computation for Neural Network Training," in *Privacy Enhancing Technologies Symposium (PETS)*, 2019.

[32] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning," in *Privacy Enhancing Technologies Symposium (PETS)*, 2021.

[33] D. Walnut, *An Introduction to Wavelet Analysis*, ser. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2002. [Online]. Available: https://books.google.pt/books?id=atOStnsp9a0C

[34] J.-L. Watson, S. Wagh, and R. Ada Popa, "Piranha: A GPU Platform for Secure Computation," in *USENIX Security Symposium (USENIX)*, 2022.

[35] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *IEEE Symposium on Security and Privacy (S&P)*, 2019.

A CONSTRUCTING WAVELET MATRICES

A.1 Filters and Wavelet Matrix

For the construction of the Wavelet Matrices we follow the approach given in [27]. The main idea is to define low- and high-pass filters and then construct a matrix by considering those filters. To show which filters determine a wavelet matrix is beyond the scope of this appendix.

Definition 3. A bi-infinite sequence b is a sequence on the integers that we shall write as $b=(...,b[-1],b[0],b[1],...)$.

Definition 4. Let h and x be two bi-infinite sequences and assume that for all $n\in\mathbb{Z}$ the sum $\sum_{k=-\infty}^{\infty}h[k]x[n-k]$ converges.[5] Then, the convolution product $h*x$ is the bi-infinite sequence y such that $y[n]=\sum_{k=-\infty}^{\infty}h[k]x[n-k]$.

Definition 5. A bi-infinite sequence h is called a finite impulse response (FIR) filter if there exist $l,L\in\mathbb{N}$ such that for $k<-l$ and $k>L$, $h[k]=0$ and $h[-l]$, $h[L]\neq 0$. We write $h=(h[-l],h[-l+1],...,h[L])$.

Moreover, a FIR filter h of length $N=L+l+1$ is symmetric if

- $h[k]=h[-k]$ for all $k\in\mathbb{Z}$, when N is odd,
- $h[k]=h[1-k]$ for all $k\in\mathbb{Z}$, when N is even.

[5]In this paper we assume that h always has a finite number of nonzero terms. This ensures the convergence of the sum.

Definition 6. Let $h=(h[l],\dots,h[L])$ be a FIR filter, u and v be bi-infinite sequences such that $u[k]=1$ and $v[k]=(-1)^k$. We say that h is a

- low-pass filter if $h*u\neq 0$ and $h*v=0$,
- high-pass filter if $h*u=0$ and $h*v\neq 0$.[6]

As mentioned before, a DWT matrix is built by considering low- and high-pass filters. For the construction of filters that determine a DWT matrix we refer the reader to [27, §9]. As for vectors, for indexing into matrices we use an array-like notation such as $M[i,j]$ for the entry in row i column j of a matrix M.

Definition 7. Let $h=\left(h[-l_1],\dots,h[0],\dots,h[L_1]\right)$ and $g=\left(g[-l_2],\dots,g[0],\dots,g[L_2]\right)$ be, respectively, low- and high-pass filters that determine a DWT. For an even number N let

$$W_N[i,j]:=\begin{cases} h_{j-2i} & \text{if } 0\leq i<N/2,\ j\in\{-l_1+2i,\dots,L_1+2i\} \\ g_{j-2i} & \text{if } N/2\leq i<N,\ j\in\{-l_1+2i,\dots,L_2+2i\} \\ 0 & \text{otherwise} \end{cases}.$$

The matrix W_N is called the DWT matrix associated to the filter (h,g).

Note that by applying a DWT to a real-valued signal vector $\vec{v}\in\mathbb{R}^N$, with $N=2^n$, one decomposes this vector into approximation $\vec{a}^{(\vec{v})}_{n-1}\in\mathbb{R}^{N/2}$ and details $\vec{d}^{(\vec{v})}_{n-1}\in\mathbb{R}^{N/2}$ coefficients as shown in Subsection 3.1:

$$\left[\begin{array}{c} \vec{a}^{(\vec{v})}_{n-1} \\ \hline \vec{d}^{(\vec{v})}_{n-1} \end{array}\right]:=W_N\cdot\vec{v}^{\,\mathrm{T}}.$$

Hence, we refer to W_N as the decomposition matrix of a DWT. One recovers $\vec{v}$ from the approximation and detail coefficients by doing the following computation:

$$W_N^{-1}\cdot\left[\begin{array}{c} \vec{a}^{(\vec{v})}_{n-1} \\ \hline \vec{d}^{(\vec{v})}_{n-1} \end{array}\right]=\vec{v}^{\,\mathrm{T}}.$$

Recall that W_N is determined by a pair of low and high-pass filters (h,g). Assume that $W_N^{-1}=W_N^{\mathrm{T}}$, i.e. the DWT matrix is orthogonal. In this case we say that W_N is (also) the reconstruction matrix for the same DWT. Moreover, a DWT satisfying this condition is said to be orthogonal.

Now, assume that $W_N^{-1}=\tilde{W}_N^{\mathrm{T}}$ where $\tilde{W}_N$ is the DWT matrix defined by a pair of low- and high-pass filters $(\tilde{h},\tilde{g})$. In this case, we say that $\tilde{W}_N$ is the reconstruction matrix for the same DWT. Moreover, a DWT satisfying this condition is said to be biorthogonal. Note that both matrices W_N and $\tilde{W}_N$ are DWT matrices. Note that orthogonal DWTs are a particular case of biorthogonal DWTs, just consider the case where $(\tilde{h},\tilde{g})=(h,g)$.

We can now define the Haar and Biorthogonal DWT.

A.1.1 Explicit Matrix Constructions. Next we look at the explicit matrix constructions for $N=8=2^3$ for Haar and Biorthogonal transforms.

Haar transform. We present the formal definition followed by the explicit matrix construction.

[6] We usually use the letter h to denote a low-pass filter and letter g to denote a high-pass filter.

Definition 8. Consider the pair of filters (h,g) with $h=\left(h[0],h[1]\right)=\left(\frac{\sqrt{2}}{2},\frac{-\sqrt{2}}{2}\right)$ and $h=\left(g[0],g[1]\right)=\left(\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2}\right)$. The DWT defined by the DWT matrix associated to the filters (h,g) is called the Haar DWT.

Explicitly, for $N=8=2^3$ we have that the Haar wavelet matrix is as follows:

$$W_8=\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix}.$$

It is clear that the Haar wavelet matrix is orthogonal, i.e., $W_8^{-1}=W_8^{\mathrm{T}}$. For an example of application of the Haar DWT we refer to Appendix A.3.

Definition 9. Consider the following filters:

$$h=(h[-2],h[-1],h[0],h[1],h[2])=\left(-\frac{\sqrt{2}}{8},\frac{\sqrt{2}}{4},\frac{3\sqrt{2}}{4},\frac{\sqrt{2}}{4},-\frac{\sqrt{2}}{8}\right),$$

$$\tilde{h}=(\tilde{h}[-1],\tilde{h}[0],\tilde{h}[1])=\left(\frac{\sqrt{2}}{4},\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{4}\right),$$

$$g=(g[0],g[1],g[2])=\left(\frac{\sqrt{2}}{4},-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{4},\right),$$

$$\tilde{g}=(\tilde{g}[-1],\tilde{g}[0],\tilde{g}[1],\tilde{g}[2],\tilde{g}[3])=\left(\frac{\sqrt{2}}{8},\frac{\sqrt{2}}{4},-\frac{3\sqrt{2}}{4},\frac{\sqrt{2}}{4},\frac{\sqrt{2}}{8}\right).$$

Let W_N and $\tilde{W}_N$ be, respectively, the DWT matrix associated to the pair of filters (h,g) and $(\tilde{h},\tilde{g})$. The DWT whose decomposition matrix is W_N and the reconstruction matrix is $\tilde{W}_N$ is called the biorthogonal bior(5,3) DWT.

Explicitly, for $N=8=2^3$, we have that the decomposition matrix W_8 is

$$W_8=\begin{bmatrix} \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{8} & 0 & 0 & 0 & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{4} \\ \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{8} & 0 & 0 & 0 \\ 0 & 0 & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{8} & 0 \\ \frac{-\sqrt{2}}{8} & 0 & 0 & 0 & \frac{-\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 \\ \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & \frac{-\sqrt{2}}{2} \end{bmatrix},$$

and the reconstruction matrix $\tilde{W}_8$ of the bior(5,3) transform is

$$\tilde{W}_8 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} \\ 0 & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & \frac{-3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} & 0 & 0 & 0 & \frac{\sqrt{2}}{8} \\ 0 & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{-3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{-3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} & 0 & 0 & 0 & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{-3\sqrt{2}}{4} \end{bmatrix}.$$

One can easily verify that $\tilde{W}_8^{\mathrm{T}} = W_8^{-1}$.

A.2 Theory of Vanishing Moments

In this subsection, we mainly follow [14, §10.5] and [7, §7.4].

Definition 10. Let g be an high-pass filter. Then, g has M-vanishing moments if $\sum_{n\in\mathbb{Z}} n^k g[n] = 0$, for all $k = 0,\ldots,M-1$.

An important consequence is that the convolution of an high-pass filter g having M vanishing moments with a infinite length signal obtained from sampling (choosing equidistant points) a polynomial of degree at most $M-1$ is zero. That is, applying a DWT whose corresponding high-pass filter has M vanishing moments to a signal in the above conditions produces detail coefficients equal to zero. Moreover, it is also shown that the convolution of any filter h of finite length takes a sampled polynomial of degree at most $M-1$ into another sampled polynomial of degree at most $M-1$, consult [14, §10.5.2]. Again, this assumes that the signal obtained by sampling a polynomial is infinite.

As a consequence, DWT matrices defined by considering high-pass filters with a high number of vanishing moments lead to high compressibility of polynomials. As an example, take a signal sampling a polynomial of degree at most $M-1$ and consider a DWT whose high-pass filter has M vanishing moments. In the first application of the DWT to the original signal the approximation coefficients will still be a sampling of a polynomial of degree at most $M-1$ and the detail coefficients will be zero, since the high-pass filter has M vanishing moments. So, we are in the same conditions as in the beginning and we can apply the DWT again. Hence, if we take a signal that is well approximated by a polynomial of degree $M-1$, in the first application of the DWT we will obtain approximation coefficients that are well approximated by another polynomial of degree $M-1$ and the detail coefficient will be close to zero. Intuitively speaking this occurs because we are approximating something that is close to a polynomial of degree $M-1$ and hence it has a somewhat similar behavior. We can repeat the application of the DWT in this manner to get high compression. Note that this cannot go on indefinitely because the "gap" between the polynomial approximation and the signal (or approximate coefficients) gets progressively worse with each application of the DWT. For further details on this, we refer the reader to [7, §7.4 & §8.2] and [33, §9].

When one considers finite signals, as we do in this paper, boundary problems arise. Indeed, this can be seen from the construction of the matrix wavelet. If the filters are long enough the top and bottom samples of a signal will be combined to obtain the top (as well as the bottom) samples of the approximation (and detail) coefficients. There are some options to deal with this problem. In this work we choose to consider symmetric filters, see [27, 7.3] and [33, §10.7.3]. Other methods can be seen in [14, §10].

A.3 Full Example using the Haar DWT

We now show all the computations that lead to the example in Figure 2 where we present the application of the Haar DWT to two different vectors. Consider Haar DWT matrix W_8

$$W_8 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix},$$

and vectors $\vec{v}$, $\vec{u} \in \mathbb{R}^8$:

$$\vec{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix} \quad \text{and} \quad \vec{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^8.$$

We start by determining the approximation and detail coefficients of both $\vec{v}$ and $\vec{u}$ at level $J=2$ (or depth $j=1$, since $N = 8 = 2^3$):

$$W_8 \cdot \vec{v} = \left[\begin{array}{c} 0 \\ 0 \\ 3\sqrt{2} \\ 3\sqrt{2} \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right] = \left[\begin{array}{c} \vec{a}_2^{(\vec{v})} \\ \hline \vec{d}_2^{(\vec{v})} \end{array}\right], \quad \text{and} \quad W_8 \cdot \vec{u} = \left[\begin{array}{c} 0 \\ 0 \\ \frac{\sqrt{2}}{2} \\ 0 \\ \hline 0 \\ 0 \\ \frac{\sqrt{2}}{2} \\ 0 \end{array}\right] = \left[\begin{array}{c} \vec{a}_2^{(\vec{u})} \\ \hline \vec{d}_2^{(\vec{u})} \end{array}\right].$$

To determine the approximation and detail coefficients of both $\vec{v}$ and $\vec{u}$ at level $J=1$ (or depth $j=2$) we need to consider the Haar DWT matrix

$$W_4 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix},$$

and apply it to $\vec{a}_2^{(\vec{v})}$ and $\vec{a}_2^{(\vec{u})}$. That is:

$$W_4 \cdot \vec{a}_2^{(\vec{v})} = \begin{bmatrix} 0 \\ 6 \\ \hline 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{a}_1^{(\vec{v})} \\ \hline \vec{d}_1^{(\vec{v})} \end{bmatrix}, \quad \text{and} \quad W_4 \cdot \vec{a}_2^{(\vec{u})} = \begin{bmatrix} 0.5 \\ 0 \\ \hline 0.5 \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{a}_1^{(\vec{u})} \\ \hline \vec{d}_1^{(\vec{u})} \end{bmatrix}.$$

We now write the DWT vector of both $\vec{v}$ and $\vec{u}$ at depth $j=2$, cfr. Definition 2.

$$\vec{y}_2^{(\vec{v})} = \begin{bmatrix} \vec{a}_1^{(\vec{v})} \\ \hline \vec{d}_1^{(\vec{v})} \\ \vec{d}_2^{(\vec{v})} \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \text{and} \quad \vec{y}_2^{(\vec{u})} = \begin{bmatrix} \vec{a}_1^{(\vec{u})} \\ \hline \vec{d}_1^{(\vec{u})} \\ \vec{d}_2^{(\vec{u})} \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \\ \hline 0 \\ 0.5 \\ 0 \\ 0 \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}.$$

Let us now focus on $\vec{u}$. We point out that these computation also portrait what happens in Lemma 3. For instance, for $\vec{u} = \vec{e}_4 \in \mathbb{R}^8$ one has $\vec{a}_2^{(\vec{u})} = 2^{-1/2}\vec{e}_{\text{msb}_2(4)} = 2^{-1/2}\vec{e}_2 \in \mathbb{R}^4$ and $\vec{a}_1^{(\vec{u})} = 2^{-2/2}\vec{e}_{\text{msb}_1(4)} = 0.5\vec{e}_1 \in \mathbb{R}^2$ as predicted by Lemma 3. To exemplify Lemma 4 one would have to consider another standard bases and do some simple subtractions.

A.4 Can DFTs be used instead of DWTs?

In this paper we decided to use DWTs instead of the more standard Discrete Fourier Transformation (DFT). In this small section we elaborate on the reasons for this decision.

We start by noting that a DFT is defined as a linear combination of sine and cosine functions which have values ranging from −1 to 1 in the entire domain of the function. Consequently, applying a DFT to a signal spreads the information relatively evenly to the next approximation. This adds two different challenges for its application in this context.

One challenge is the encoding of the vector obtained by applying the DFT to a standard basis vector. In the case of the (Haar and Bior) DWT we have that the approximation coefficients are encoded in a straightforward manner: it is at most the linear combination of two standard basis vector (cfr. Lemma 3 and Lemma 6). If we were to apply a DFT to a standard basis vector, then we would obtain a vector that has many non-zero entries. As a result, this would affect efficiency, since increases the difficulty to encode the approximation coefficients using compact FSS primitives like DPFs or DCFs. What happens to the standard basis vector is crucial because are those results that allow an efficient evaluation of the compressed LUT. Note that here we could reach a more extreme situation than the one we already point out when one considers DWTs with more vanishing moments. Thus, if we were to find a solution to this difficulty, then we could think about using the DFTs in a similar fashion to the way we used DWTs. However, even assuming that the encoding of the approximation coefficients of a standard bases vector under the application of DFTs was simple and efficiency, there would still exist some challenges regarding the approximation.

The second related challenge is that since the application of a DFT spreads the information relatively evenly to the next approximation, the effect of assuming some values to be zero may affect more the approximations. An extra aspect to have in mind is that the application of DFTs produces complex values and we are working with real values. One idea is to make use of real DFTs but it comes with it's set of challenges. To convey these challenges, we reproduce part of the example in Figure 2 but instead of using DWT, we use a real DFT once. After applying the DFT to vectors $\vec{v}$ and $\vec{u}$ of Figure 2, zeroing values and applying the inverse of the DFT we obtain vectors: $\vec{v}'$ and $\vec{u}'$

$$\vec{v} = \begin{bmatrix} 0.750 \\ -0.311 \\ -0.311 \\ 0.750 \\ 2.250 \\ 3.310 \\ 3.310 \\ 2.250 \end{bmatrix} \quad \text{and} \quad \vec{u} = \begin{bmatrix} -0.125 \\ -0.052 \\ 0.125 \\ 0.302 \\ 0.375 \\ 0.302 \\ 0.125 \\ -0.052 \end{bmatrix} \in \mathbb{R}^8,$$

whose inner product is approximately 2.25, and not 3. Thus, the equivalent of a single MRA level in the DFT case leads to an error of 25%. While these arguments just give a glimpse of the complexity, there is no conclusive answer to this. After all, both DWTs and DFTs share some important properties (such as Parseval's theorem) and thus it remains to be seen if there is a analogous uses of DFTs in the context of MPC.

B LEMMA 3 AND LEMMA 4 PROOFS

In this appendix we prove Lemma 3 and Lemma 4.

Proof of Lemma 3. The proof is by induction on j.

Base case ($j=1$ and $J=n-1$): Let $W_{N,1}$ be the decomposition Haar matrix with $N=2^n$. By construction (see Appendix A), we have for $i \in [0..N/2-1]$ and $k \in [0..N-1]$ that

$$W_{N,1}[i,k] = \begin{cases} 2^{-1/2} & \text{if } i = \lfloor k/2 \rfloor, \text{ and} \\ 0 & \text{otherwise;} \end{cases}$$

consequently, $\vec{a}_J^{(\vec{e}_{\tilde{a}})} = 2^{-1/2} \cdot \vec{e}_{\lfloor \tilde{a}/2 \rfloor} = 2^{-1/2} \cdot \vec{e}_{\text{msb}_{n-1}(\tilde{a})} = 2^{-1/2} \cdot \vec{e}_{\text{msb}_{n-1}(a)}$. This proves the base case.

Inductive step: Suppose that $\vec{a}_J^{(\vec{e}_{\tilde{a}})} = 2^{-j/2} \cdot \vec{e}_{\text{msb}_J(a)}$ and consider the product of $\vec{y}_{j+1} = W_{N,j+1} \cdot \vec{y}_j$. By construction, the first 2^{J-1} rows of $W_{N,j+1}$ satisfy $W_{N,j+1}[i,k] = 2^{-1/2}$ if $i = \lfloor k/2 \rfloor$ and 0 otherwise. Consequently, $\vec{a}_{J-1}^{(\vec{e}_{\tilde{a}})} = 2^{-j/2} \cdot 2^{-1/2} \cdot \vec{e}_{\lfloor \text{msb}_J(a)/2 \rfloor} = 2^{-(j+1)/2} \cdot \vec{e}_{\text{msb}_{J-1}(a)}$.

□

Proof of Lemma 4. From Lemma 3, we know that $\vec{a}_J^{(\vec{e}_{\tilde{a}})} = 2^{-j/2} \cdot \vec{e}_{\text{msb}_J(a)}$ and $\vec{a}_J^{(\vec{e}_{\tilde{r}})} = 2^{-j/2} \cdot \vec{e}_{\text{msb}_J(r)}$ both hold; moreover, we have that

$$\vec{e}_{\text{msb}_J(a)} = \vec{e}_{\text{msb}_J(r)} \lll (\text{msb}_J(r) - \text{msb}_J(a) \bmod 2^J).$$

Suppose $r \geq x$ and recall that $a \equiv r - x \bmod 2^{\ell}$. In this case,

$$\begin{aligned} x_J &= \mathrm{msb}_J(r) - \mathrm{msb}_J(a) \bmod 2^J \\ &= \mathrm{msb}_J(r) - \big(\mathrm{msb}_J(r) - \mathrm{msb}_J(x) \\ &\qquad - (\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x))\big) \bmod 2^J \\ &= \mathrm{msb}_J(x) + \big(\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x)\big) \bmod 2^J. \end{aligned}$$

In case $r < x$, we write $a \equiv r - x \equiv 2^{\ell} + r - x \bmod 2^{\ell}$ and the reasoning is the same:

$$\begin{aligned} x_J &= \mathrm{msb}_J(r) - \mathrm{msb}_J(a) \bmod 2^J \\ &= \mathrm{msb}_J(r) - \big(2^J + \mathrm{msb}_J(r) - \mathrm{msb}_J(x) \\ &\qquad - (\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x))\big) \bmod 2^J \\ &= \mathrm{msb}_J(x) + (\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x)) \bmod 2^J. \end{aligned}$$

This completes the proof. □

C LEMMA 6 AND LEMMA 7 PROOFS

In this appendix we prove Lemma 6 and Lemma 7 that are related to the application of the bior(5,3) to a standard basis vector.

Proof of Lemma 6. Let $\tilde{W}_N$ be the reconstruction bior(5,3) matrix with $N = 2^n$. The proof is by induction on j.

Base case ($j = 1$ and $J = n-1$): By construction (see Definition 9), we have for $i \in [0..N/2-1]$ and $k \in [0..N-1]$ that

$$\tilde{W}_N[i,k] = \begin{cases} 2^{-1/2} & \text{if } k = 2i, \text{ and} \\ 2^{-3/2} & \text{if } k = 2i \pm 1 \bmod N, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

If $\bar{a}$ is even, then $\bar{a} = 2\mathrm{msb}_{n-1}(\bar{a})$, $\mathrm{lsb}_1(\bar{a}) = 0$, and

$$\begin{aligned} \tilde{W}_N \cdot \vec{e}_{\bar{a}}^{\,\mathrm{T}} &= 2^{-1/2} \vec{e}_{\bar{a}/2} = (2-0) 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})} \\ &= \big(2 - \mathrm{lsb}_1(\bar{a})\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})} \\ &\quad + \mathrm{lsb}_1(\bar{a}) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})+1} \\ &= \big(2 - \mathrm{lsb}_1(\bar{a})\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(a)} \\ &\quad + \mathrm{lsb}_1(\bar{a}) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(a)+1}. \end{aligned}$$

If $\bar{a}$ is odd, then $\bar{a} = 2 \cdot \mathrm{msb}_{n-1}(\bar{a}) + 1$, and

$$\begin{aligned} \tilde{W}_N \cdot \vec{e}_{\bar{a}}^{\,\mathrm{T}} &= 2^{-3/2} \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})} + 2^{-3/2} \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})+1} \\ &= \big(2 - \mathrm{lsb}_1(\bar{a})\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})} \\ &\quad + \mathrm{lsb}_1(\bar{a}) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(\bar{a})+1} \\ &= \big(2 - \mathrm{lsb}_1(\bar{a})\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(a)} \\ &\quad + \mathrm{lsb}_1(\bar{a}) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{n-1}(a)+1}. \end{aligned}$$

This proves the base case.

Inductive step: Suppose that

$$\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0} \cdot \vec{e}_{\mathrm{msb}_J(a)} + c_{J,1} \cdot (\vec{e}_{\mathrm{msb}_J(a)} \ggg 1) \in \mathbb{R}^{2^J}$$

and consider the product $\tilde{W}_{2^J}$ with $\vec{e}_{\mathrm{msb}_J(a)}$ and with $\vec{e}_{\mathrm{msb}_J(a)} \ggg 1 = \vec{e}_{\mathrm{msb}_J(a)+1}$. Let $a' = \mathrm{msb}_J(a)$ and apply the base case. Then

$$\begin{aligned} \tilde{W}_{2^J} \cdot \vec{e}_{a'} &= \big(2 - \mathrm{lsb}_1(a')\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a')} \\ &\quad + \mathrm{lsb}_1(a') \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a')+1} \\ &= \left(2 - \mathrm{lsb}_1\big(\mathrm{msb}_J(a)\big)\right) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(\mathrm{msb}_J(a))} \\ &\quad + \mathrm{lsb}_1\big(\mathrm{msb}_J(a)\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(\mathrm{msb}_J(a))+1} \\ &= \left(2 - \mathrm{lsb}_1\big(\mathrm{msb}_J(a)\big)\right) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \mathrm{lsb}_1\big(\mathrm{msb}_J(a)\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1} \end{aligned}$$

and

$$\begin{aligned} \tilde{W}_{2^J} \cdot \vec{e}_{a'+1} &= \big(2 - \mathrm{lsb}_1(a'+1)\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a'+1)} \\ &\quad + \mathrm{lsb}_1(a'+1) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a'+1)+1} \\ &= \left(2 - \mathrm{lsb}_1\big(\mathrm{msb}_J(a)+1\big)\right) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(\mathrm{msb}_J(a)+1)} \\ &\quad + \mathrm{lsb}_1\big(\mathrm{msb}_J(a)+1\big) \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(\mathrm{msb}_J(a)+1)+1}. \end{aligned}$$

Now, assume that $a' = \mathrm{msb}_J(a)$ is even, that is $\mathrm{lsb}_1(a') = \mathrm{lsb}_1(\mathrm{msb}_J(a)) = 0$. Hence,

$$\tilde{W}_{2^J} \cdot \vec{e}_{\mathrm{msb}_J(a)+1} = 2^{-3/2} \cdot \vec{e}_{msb_{J-1}(a)} + 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1}.$$

Thus, under the assumption that a' is even, we have:

$$\begin{aligned} \vec{a}_{J-1}^{(\vec{e}_{\bar{a}})} &= \big(2^j - \mathrm{lsb}_j(\bar{a})\big) \cdot 2^{-3j/2} \cdot 2^{-3/2} \cdot 2 \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \mathrm{lsb}_j(\bar{a}) \cdot 2^{-3j/2} \cdot 2^{-3/2} \cdot \big(\vec{e}_{msb_{J-1}(a)} + \vec{e}_{\mathrm{msb}_{J-1}(a)+1}\big) \\ &= \big(2^{j+1} - \mathrm{lsb}_j(\bar{a})\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \mathrm{lsb}_j(\bar{a}) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1} \\ &= \big(2^{j+1} - \mathrm{lsb}_{j+1}(\bar{a})\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \mathrm{lsb}_{j+1}(\bar{a}) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1}. \end{aligned}$$

This finishes the proof when a' is even.
Assume that a' is odd, that is $\mathrm{lsb}_1(a') = \mathrm{lsb}_1(\mathrm{msb}_J(a)) = 1$. Then,

$$\tilde{W}_{2^J} \cdot \vec{e}_{a'+1} = 2 \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1}.$$

and we obtain

$$\begin{aligned} \vec{a}_{J-1}^{(\vec{e}_{\bar{a}})} &= \big(2^j - \mathrm{lsb}_j(\bar{a})\big) \cdot 2^{-3j/2} \cdot 2^{-3/2} \cdot \big(\vec{e}_{\mathrm{msb}_{J-1}(a)} + \vec{e}_{\mathrm{msb}_{J-1}(a)+1}\big) \\ &\quad + \mathrm{lsb}_j(\bar{a}) \cdot 2^{-3j/2} \cdot 2 \cdot 2^{-3/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1} \\ &= \big(2^j - \mathrm{lsb}_j(\bar{a})\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \big(2^j + \mathrm{lsb}_j(\bar{a})\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1} \\ &= \big(2^j - \mathrm{lsb}_{j+1}(\bar{a}) + 2^j\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \big(2^j + \mathrm{lsb}_{j+1}(\bar{a}) - 2^j\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1} \\ &= \big(2^{j+1} - \mathrm{lsb}_{j+1}(\bar{a})\big) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)} \\ &\quad + \mathrm{lsb}_{j+1}(\bar{a}) \cdot 2^{-3(j+1)/2} \cdot \vec{e}_{\mathrm{msb}_{J-1}(a)+1}. \end{aligned}$$

This finishes the proof.

□

Ideal Functionality for approximation $\tilde{F}_{\mathbb{X}\ell,f,n,J}(\cdot)$

Public parameters:

- Real-valued univariate function $F: \mathbb{R} \to \mathbb{R}$
- Transform $\mathbb{X} \in \{\mathbb{H}, \mathbb{B}\}$ (i.e., $\mathbb{H}$aar or $\mathbb{B}$iorthogonal)
- Fixed-point parameters $(\ell, f) \in \mathbb{N} \times \mathbb{N}$ with $0 \le f \le \ell$ and $0 < \ell$
- Quantization parameter $n \in \mathbb{N}$ and DWT compression level $J \in \mathbb{N}$ with $0 < J < n \le \ell$

Input:

- $[a] = ([a]_0, [a]_1)$, where $[a]_0$ and $[a]_1$ are the input shares from the two parties

Output:

- $[y] = ([y]_0, [y]_1)$ for $y := \tilde{F}_{\mathbb{X}\ell,f,n,J}([a]_0 + [a]_1)$ using Equation (7) if $\mathbb{X} = \mathbb{H}$ or (8) if $\mathbb{X} = \mathbb{B}$

Figure 8: Ideal functionality for all 4 variants of our protocol – Figure 5, Figure 6, Figure 10, and Figure 11.

Proof of Lemma 7. Let $\bar{r} := \mathrm{msb}_n(r)$. From Lemma 6, we know that

$$\vec{a}_J^{(\vec{e}_{\bar{a}})} = c_{J,0}^{\bar{a}} \cdot \vec{e}_{\mathrm{msb}_J(a)} + c_{J,1}^{\bar{a}} \cdot (\vec{e}_{\mathrm{msb}_J(a)} \ggg 1),$$
$$\vec{a}_J^{(\vec{e}_{\bar{r}})} = c_{J,0}^{\bar{r}} \cdot \vec{e}_{\mathrm{msb}_J(r)} + c_{J,1}^{\bar{r}} \cdot (\vec{e}_{\mathrm{msb}_J(r)} \ggg 1) \in \mathbb{R}^{2^J},$$

where

$$c_{J,0}^{\bar{a}} := \left(2^j - \mathrm{lsb}_j(\bar{a})\right) \cdot 2^{-3j/2},$$
$$c_{J,1}^{\bar{a}} := \mathrm{lsb}_j(\bar{a}) \cdot 2^{-3j/2},$$
$$c_{J,0}^{\bar{r}} := \left(2^j - \mathrm{lsb}_j(\bar{r})\right) \cdot 2^{-3j/2}, \text{ and}$$
$$c_{J,1}^{\bar{r}} := \mathrm{lsb}_j(\bar{r}) \cdot 2^{-j/2}.$$

hold. Moreover, we have that

$$\vec{e}_{\mathrm{msb}_J(a)} = \vec{e}_{\mathrm{msb}_J(r)} \lll (\mathrm{msb}_J(r) - \mathrm{msb}_J(a) \bmod 2^J).$$

From the proof of Lemma 4 we have

$$\begin{aligned} x_J &= \mathrm{msb}_J(r) - \mathrm{msb}_J(a) \bmod 2^J \\ &= \mathrm{msb}_J(x) + (\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x)) \bmod 2^J. \end{aligned}$$

□

Note that in the above conditions,

$$\begin{aligned} \mathrm{lsb}_j(\mathrm{msb}_n(a)) = \mathrm{lsb}_j(\mathrm{msb}_n(r)) - \big(&\mathrm{lsb}_j(\mathrm{msb}_n(x)) \\ &+ (\mathrm{lsb}_{\ell-n}(r) < \mathrm{lsb}_{\ell-n}(x)) \\ &- 2^j \cdot (\mathrm{lsb}_{\ell-J}(r) < \mathrm{lsb}_{\ell-J}(x))\big) \bmod 2^j. \end{aligned}$$

D SECURITY PROOF

Note that the ideal functionality for the 4 protocol variants are very similar and Figure 8 describes them agnostic to the Haar or Bior transform. While the protocols however differ slightly in their constructions, their security proofs are very similar and below is the proof for the Pika+Haar variant (Theorem 9). Note that the ideal functionality is an exact computation and thus all the approximations lie "outside" the MPC protocol which makes the simulation straightforward.

Proof of Theorem 9. To prove correctness, note that thanks to Bullets 1 and 2 of Corollary 5, we have that in the Step 4 of the Online Phase of Figure 5 each party actually obtains shares $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$ and lifts it into $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$. So, we get $[\langle \pm\vec{e}_{\mathrm{msb}_J(a)}, \mathbf{A}_F \rangle] = [\pm\mathbf{A}_F[\mathrm{msb}_J(a)]]$. Then, the Beaver triple multiplication corrects the sign. This proves the correctness.

Figure 9 describes the simulator to simulate the protocol transcript from the adversarial view. The three rounds of interaction can be simulated as described in Steps(1), (2), and (3). Noting that all outgoing messages written to the transcript are deterministically computed exactly as specified in the real protocol; incoming messages are all sampled uniformly from their domains, ensuring their distributions match those in the real protocol (which are all blinded via uniform random shares). □

Simulator $\mathcal{S}$ for Pika+Haar $\mathcal{S}_{n,\ell,J,\vec{v}}^{\textbf{LUT-Haar-Pika}}$

Inputs:

- Haar-compressed LUT, $\mathbf{A}_F$
- Additive shares, $[r]_b$ and $[a]_b$;
- DPF share, $\llparenthesis \vec{e}_{\mathrm{msb}_J(r)} \rrparenthesis_b$;
- DCF share, $\{\!\{(x > r)\ ?\ 0 : 1\}\!\}$; and
- Beaver triple shares, $([X]_b, [Y]_b, [Z]_b)$ for $Z = [X]_0 \cdot [Y]_1 + [X]_1 \cdot [Y]_0$.

Simulation steps:

(1) Compute shares of the full-length rotation distance (difference of a and r): $[x]_b = [r]_b - [a]_b$
  - Write $\mathrm{lsb}_{\ell-J}([x]_b)$ to the transcript to simulate "sending" it to the other party
  - Sample $\mathrm{lsb}_{\ell-J}([x]_{1-b}) \in_{\mathrm{R}} \{0,1\}^{\ell-J}$, and then write it to the transcript to simulate "receiving" it from the other party

(2) Compute $\tilde{x} = \mathrm{lsb}_{\ell-J}\,[x]_0 + \mathrm{lsb}_{\ell-J}\,[x]_1$ over $\mathbb{Z}$; parse it as $(carry, x_{\neg J}) \in \mathbb{Z}_2 \times \mathbb{Z}_{2^{\ell-J}}$ where $carry$ is the carry-out ($(\ell-J+1)$th bit) of the summation over $\mathbb{Z}$. Likewise, compute the borrow share $[borrow]_b = DCFEval(\{\!\{(x > \mathrm{lsb}_{\ell-J}(r))\ ?\ 1 : 0\}\!\}, x_{\neg J})$. Also compute $[x_J]_b = \mathrm{msb}_J([x]_b) + b \cdot carry + [borrow]_b$.
  - Write $[x_J]_b$ to the transcript to simulate "sending" it to the other party.
  - Sample $[x_J]_{1-b} \in_{\mathrm{R}} \{0,1\}^J$, and then write it to the transcript to simulate "receiving" it from the other party.

(3) Compute $x_J = [x_J]_b + [x_J]_{1-b}$, $[\![\vec{e}_{\mathrm{msb}_J(r)}]\!]_b = EvalFull(\llparenthesis \vec{e}_{\mathrm{msb}_J(r)} \rrparenthesis_b)$, and $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]_b = [\![\vec{e}_{\mathrm{msb}_J(r)}]\!]_b \lll x_J$, and then sign-extend this into $[\pm\vec{e}_{\mathrm{msb}_J(a)}]_b$ and use component-wise summation to compute $[\pm 1]_b$ from $[\pm\vec{e}_{\mathrm{msb}_J(a)}]_b$, and then evaluate $[\pm\mathbf{A}_F[\,\mathrm{msb}_J(a)]]_b = \langle [\pm\vec{e}_{\mathrm{msb}_J(a)}]_b, \mathbf{A}_F \rangle$. Also compute $[\pm 1]_b + [X]_b$ and $[\pm\vec{e}_{\mathrm{msb}_J(a)}]_b + [Y]_b$.
  - Write $[\pm 1]_b + [X]_b$ and $[\pm\vec{e}_{\mathrm{msb}_J(a)}]_b + [Y]_b$ to the transcript to simulate "sending" them to the other party
  - Sample $[\pm 1]_{1-b} + [X]_{1-b} \in_{\mathrm{R}} \{0,1\}^{\ell}$ and $[\pm\vec{e}_{\mathrm{msb}_J(a)}]_{1-b} + [Y]_{1-b} \in_{\mathrm{R}} \{0,1\}^{\ell}$, and then write them to the transcript to simulate "receiving" them from the other party

(4) Finally, compute $[\tilde{F}_{\mathbb{X}\ell,f,n,J}([a]_0 + [a]_1)]_b = [\pm 1]_b \cdot \big([\pm\vec{e}_{\mathrm{msb}_J(a)}]_b + ([\pm\vec{e}_{\mathrm{msb}_J(a)}]_{1-b} + [Y]_{1-b}) - [Y]_b \cdot ([\pm 1]_{1-b} + [X]_{1-b})\big) + [Z]_b$.

Figure 9: Simulator for Pika+Haar variant (Figure 5)

E CORRECTNESS OF THEOREM 10

Security argument for Theorem 10 follows along similar lines as that of Theorem 9 described in Appendix D. In order to prove correctness, note that Step 3 precisely computes the r.h.s of the expression from Equation (8) (following Bullet 1 of Corollary 8) and thus computes additive shares of the required LUT value $\tilde{F}_{\mathbb{B}\ell,f,n,J}(a)$. Furthermore, the correctness of $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$ follows from the correctness of Grotto's segment-parity algorithm and thus the correctness of "bior(5,3)+Grotto" protocol hinges on the correctness of the computation of $\mathrm{lsb}_j(\mathrm{msb}_n(a))$. Let us define b_k for $k\in\{J,n\}$

$$b_k=\begin{cases}1 \text{ if } \mathrm{lsb}_{\ell-k}(x)>\mathrm{lsb}_{\ell-k}(r)\\ 0 \text{ otherwise.}\end{cases} \tag{10}$$

Note that $borrow_n=b_n$ and $borrow_J=2^j\cdot b_J$ as defined in Step 2 of the online phase of Figure 6. Recall that $a\equiv r-x \pmod{2^\ell}$ and thus, for $k\in\{J,n\}$ we have:

$$\mathrm{lsb}_{\ell-k}(a)=\mathrm{lsb}_{\ell-k}(r)-\mathrm{lsb}_{\ell-k}(x)+b_k\cdot 2^{\ell-k} \tag{11}$$

Subtracting Equation (11) for $k=n$ from $k=J$ and rearranging terms, we get that

$$\begin{aligned}(\mathrm{lsb}_{\ell-J}(a)-\mathrm{lsb}_{\ell-n}(a))=&(\mathrm{lsb}_{\ell-J}(r)-\mathrm{lsb}_{\ell-n}(r))\\ &+(\mathrm{lsb}_{\ell-J}(x)-\mathrm{lsb}_{\ell-n}(x))\\ &+b_J\cdot 2^{\ell-J}-b_n\cdot 2^{\ell-n}\end{aligned} \tag{12}$$

Now note that for any value $y\in\{r,a,x\}$, we have that $\mathrm{lsb}_{\ell-J}(y)-\mathrm{lsb}_{\ell-n}(y)=\mathrm{lsb}_j(\mathrm{msb}_n(y))\cdot 2^{\ell-n}$. Substituting this into the last equation and dividing by $2^{\ell-n}$, we have that:

$$\begin{aligned}\mathrm{lsb}_j(\mathrm{msb}_n(a))=&\mathrm{lsb}_j(\mathrm{msb}_n(r))-\mathrm{lsb}_j(\mathrm{msb}_n(x))\\ &-b_J\cdot 2^{n-J}+b_n\\ =&\mathrm{lsb}_j(\mathrm{msb}_n(r))-\mathrm{lsb}_j(\mathrm{msb}_n(x))\\ &-borrow_n+borrow_J\end{aligned} \tag{13}$$

This completes the proof. □

F THEOREM 1, 2 PROOFS

Theorem 1 Proof. For orthogonal DWTs, we have that

$$\begin{aligned}\langle\vec{u},\vec{v}\rangle&=\vec{u}^{\mathrm{T}}\cdot\vec{v}\\ &=\vec{u}^{\mathrm{T}}\cdot(W^{-1}\cdot W)\cdot\vec{v}\\ &=\vec{u}^{\mathrm{T}}\cdot\big((W_{N,1}^{-1}\cdots W_{N,J}^{-1})\cdot W\big)\cdot\vec{v}\\ &=\vec{u}^{\mathrm{T}}\cdot\big((W_{N,1}^{\mathrm{T}}\cdots W_{N,J}^{\mathrm{T}})\cdot W\big)\cdot\vec{v} && \text{(orthogonality)}\\ &=\vec{u}^{\mathrm{T}}\cdot\big((W_{N,J}\cdots W_{N,1})^{\mathrm{T}}\cdot W\big)\cdot\vec{v} && \text{(transpose)}\\ &=\vec{u}^{\mathrm{T}}\cdot(W^{\mathrm{T}}\cdot W)\cdot\vec{v}\\ &=[\,\vec{a}_J^{(\vec{u})}\ \vec{d}_J^{(\vec{u})}\ \vec{d}_{J+1}^{(\vec{u})}\ \cdots\ \vec{d}_{n-1}^{(\vec{u})}\,]\cdot[\,\vec{a}_J^{(\vec{v})}\ \vec{d}_J^{(\vec{v})}\ \vec{d}_{J+1}^{(\vec{v})}\ \cdots\ \vec{d}_{n-1}^{(\vec{v})}\,]^{\mathrm{T}}\\ &=\langle\vec{a}_J^{(\vec{u})},\vec{a}_J^{(\vec{v})}\rangle+\textstyle\sum_{i=J}^{n-1}\langle\vec{d}_i^{(\vec{u})},\vec{d}_i^{(\vec{v})}\rangle.\end{aligned}$$

□

Theorem 2 Proof. For biorthogonal DWTs, we have that

$$\begin{aligned}\langle\vec{u},\vec{v}\rangle&=\vec{u}^{\mathrm{T}}\cdot\vec{v}\\ &=\vec{u}^{\mathrm{T}}\cdot(W^{-1}\cdot W)\cdot\vec{v}\\ &=\vec{u}^{\mathrm{T}}\cdot\big((W_{N,1}^{-1}\cdots W_{N,J}^{-1})\cdot W\big)\cdot\vec{v}\\ &=\vec{u}^{\mathrm{T}}\cdot\big((\tilde{W}_{N,1}^{\mathrm{T}}\cdots \tilde{W}_{N,J}^{\mathrm{T}})\cdot W\big)\cdot\vec{v} && \text{(biorthogonality)}\\ &=\vec{u}^{\mathrm{T}}\cdot\big((\tilde{W}_{N,J}\cdots \tilde{W}_{N,1})^{\mathrm{T}}\cdot W\big)\cdot\vec{v} && \text{(transpose)}\\ &=\vec{u}^{\mathrm{T}}\cdot(\tilde{W}^{\mathrm{T}}\cdot W)\cdot\vec{v}\\ &=[\,\vec{a}_J^{(\vec{u})}\ \vec{d}_J^{(\vec{u})}\ \vec{d}_{J+1}^{(\vec{u})}\ \cdots\ \vec{d}_{n-1}^{(\vec{u})}\,]\cdot[\,\vec{a}_J^{(\vec{v})}\ \vec{d}_J^{(\vec{v})}\ \vec{d}_{J+1}^{(\vec{v})}\ \cdots\ \vec{d}_{n-1}^{(\vec{v})}\,]^{\mathrm{T}}\\ &=\langle\vec{a}_J^{(\vec{u})},\vec{a}_J^{(\vec{v})}\rangle+\textstyle\sum_{i=J}^{n-1}\langle\vec{d}_i^{(\vec{u})},\vec{d}_i^{(\vec{v})}\rangle.\end{aligned}$$

□

G (GENERALIZED) BEAVER MULTIPLICATION

Beaver multiplication triples [1] enable the efficient multiplication of additively shared secrets. Each Beaver comprises three additive sharings $\big([X],[Y],[Z]\big)$, where $X,Y\in_{\mathrm{R}}\mathbb{Z}_{2^\ell}$ and

$$Z:=[X]_0\cdot[Y]_1+[X]_1\cdot[Y]_0.$$

In a (2+1)-party protocol, triples can be provided to the shareholders for "free" by the dealer [9]. This is possible because triples consist of nothing more than data-independent correlated randomness.

Given a pair of shared values $[x]$ and $[y]$ and a Beaver triple $([X],[Y],[Z])$, each party b sends

$$\big([x]_b+[X]_b,[y]_b+[Y]_b\big)$$

to its peer, and then it outputs

$$\begin{aligned}[z]_b=&[x]_b\cdot\big([y]_b+([y]_{1-b}+[Y]_{1-b})\big)\\ &-[Y]_b\cdot([x]_{1-b}+[X]_{1-b})+[Z]_b.\end{aligned}$$

A mechanical derivation establishes that $[z]_0+[z]_1=x\cdot y$.

Beaver triples are ephemeral, each enabling just a single multiplication. The multiplication itself is agnostic as to whether x and y represent "actual" integers or fixed-point numbers. They naturally generalize to scalar-vector products and to vector-vector inner products.

Moreover, Patra, Schneider, Suresh, and Yalame [20] describe a generalization for single-round *n-way products*; we employ Patra et al.'s approach with the *sign-corrected linear function evaluation* formula derived by Storrier, Lyons, Vadapalli, and Henry [25] to evaluate expressions of the form $[y]=[u]\cdot([c_1]\cdot[x]+[c_0])$ with $u=\pm1$. To facilitate this evaluation, the dealer provides an 8-tuple of correlated random shares to the peers; specifically, the tuple consists of shares $\big([U],[C_1],[X],[C_0];[U\cdot C_1],[U\cdot X],[C_1\cdot X-C_0],[U\cdot(C_1\cdot X-C_0)]\big)$. The first four values are blinding factors used respectively to mask u, c_1, x, and c_0; the other four values are correction values used for cancellations in the secret reconstruction.

The protocol proceeds in a single round wherein the peers reconstruct blinded copies of each secret, namely $\bar{u}=u+U$, $\bar{c}_1=c_1+C_1$, $\bar{x}=x+X$, and $\bar{c}_0=c_0+C_0$. Finally, for $b=0,1$, peer

Haar+Grotto LUT protocol $\Pi^{\text{LUT-Haar-Grotto}}_{n,\ell,J,\vec{v}}$

One-time pre-processing: Input is a function F, fixed-point parameters $\ell, f \in \mathbb{N}$, LUT quantization parameter $n \in [1..\ell)$, and DWT depth j (so that $J := n-j$)

- Compute the real-valued compressed LUT $\vec{a}_J^{(\vec{v})} \in \mathbb{R}^{2^J}$ as the DWT of $\vec{v} := \mathbb{L}^{\mathbb{R},n}(F)$ at level J under the Haar transform, and then from it compute the fixed-point LUT $\mathbf{A}_F := \lfloor 2^{f-j/2} \cdot \vec{a}_J^{(\vec{v})} \rceil_\ell \in (\mathbb{Z}_{\ell,f})^{2^J}$

Per-run pre-processing $\mathcal{F}^{\textbf{pre-Haar-Grotto}}_{n,\ell,J,\vec{v}}$**:**

- Dealer distributes sharings $[r]$ and $\llparenthesis \vec{e}_r \rrparenthesis$ for $r \in_R \mathbb{Z}_{2^\ell}$, plus a Beaver triple for multiplying two secret scalars in $\mathbb{Z}_{2^\ell}$

Online phase: Input is $[a]$, $a \in \mathbb{Z}_{\ell,f}$, plus all pre-processing values

(1) Interactively reconstruct $x := r-a$ from $[r]$ and $[a]$

(2) Non-interactively compute
  - $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!]$ from $\llparenthesis \vec{e}_r \rrparenthesis$ using Grotto's parity-segmentation on partition $[\,P_i := i\cdot 2^j \,|\, i,\ldots,2^J-1\,] \ggg x$
  - perform signed extension to convert this into $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$
  - use component-wise summation to compute $[\pm 1]$ from $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$
  - and then evaluate $[\pm\mathbf{A}_F[\mathrm{msb}_J(a)]] = \langle [\pm\vec{e}_{\mathrm{msb}_J(a)}], \mathbf{A}_F \rangle$

(3) Interactively compute the product of $[\pm\mathbf{A}_F[\mathrm{msb}_J(a)]]$ and $[\pm 1]$ using the Beaver triple to obtain $[\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)]$

(4) Return $[\tilde{F}_{\mathbb{H}^{\ell,f,n,J}}(a)]$

Figure 10: Secure LUT-based function evaluation using the Haar DWT in conjunction with Grotto for vector preparation ("Haar+Grotto" variant).

b outputs

$$\begin{aligned}[y]_b := \bar{u}\cdot\big(b\cdot(\bar{c}_1\cdot\bar{x}+\bar{c}_0) - \bar{c}_1\cdot[X]_b - \bar{x}\cdot[C_1]_b + [C_1\cdot X – C_0]_b\big)\\ -(\bar{c}_1\cdot\bar{x}+\bar{c}_0)\cdot[U]_b + \bar{c}_1\cdot[U\cdot X]_b + \bar{x}\cdot[U\cdot C_1]_b - [U\cdot(C_1\cdot X – C_0)]_b.\end{aligned}$$

H GROTTO-HAAR

Figure 10 contains our Haar+Grotto pairing.

I PIKA-BIOR

Figure 11 contains our bior(5,3)+Pika pairing.

J FUNCTION EVALUATIONS

We set bit-width to 64 and vary n between 28 and 32, while J between 10 and 24. For the experiment, we first create pre-processing material and then run the online phase 1000 times for each n and J. We report the total time of the runs. Note that while the runtime changes, the online communication is always the same. For Haar, it is 24 bytes per evaluation and for bior(5,3) it is 40 bytes for all n and J.

We run experiments both in the LAN and WAN settings. Tables 2 and 3 have the results for the LAN setting while Tables 4 and 5 show the results for the WAN setting. The WAN runtimes are dominated by communication time hence the numbers do not vary as much as in the LAN setting, where an exponential trend in the runtime is observed.

K COMPARISON WITH PRIOR WORKS

We present our comparison with related work in Table 6. We run comparison experiments with Pika and Curl by fixing n and

bior(5,3)+**Pika LUT protocol** $\Pi^{\text{LUT-bior(5,3)-Grotto}}_{n,\ell,J,\vec{v}}$

One-time pre-processing: Input is a function F, fixed-point parameters $\ell, f \in \mathbb{N}$, LUT quantization parameter $n \in [1..\ell)$, and DWT depth j (so that $J := n-j$)

- Compute the real-valued compressed LUT $\vec{a}_J^{(\vec{v})} \in \mathbb{R}^{2^J}$ as the DWT of $\vec{v}\mathbb{L}^{\mathbb{R},n}(F)$ at level J under the bior(5,3) transform, and then from it compute the fixed-point LUTs
  - $\mathbf{A}_{F,c_0} := \lfloor 2^{f-j/2} \cdot \vec{a}_J^{(\vec{v})} \rceil_\ell \in (\mathbb{Z}_{\ell,f})^{2^J}$
  - $\mathbf{A}_{F,c_1} := \lfloor 2^{f-3j/2} \cdot \big((\vec{a}_J^{(\vec{v})} \lll 1) - \vec{a}_J^{(\vec{v})}\big) \rceil_\ell \in (\mathbb{Z}_{\ell,f})^{2^J}$

Per-run pre-processing $\mathcal{F}^{\textbf{pre-}\text{bior(5,3)}\textbf{-Pika}}_{n,\ell,J,\vec{v}}$**:**

- Dealer distributes sharings $[r]$, $[\mathrm{lsb}_j(\mathrm{msb}_n(r))]$ and $\llparenthesis \vec{e}_{\mathrm{msb}_J(r)} \rrparenthesis$ for $r \in_R \mathbb{Z}_{2^\ell}$, two DCFs $\{\!\![(x > \mathrm{lsb}_{\ell-n}(r))\ ?\ 1 : 0]\!\!\}$ and $\{\!\![(x > \mathrm{lsb}_{\ell-J}(r))\ ?\ 2^j : 0]\!\!\}$, plus a generalized Beaver tuple for equations of the form $[u]\cdot([c_1]\cdot[x]+[c_0])$ with arithmetic in $\mathbb{Z}_{2^\ell}$

Online phase: Input is $[a]$, $a \in \mathbb{Z}_{\ell,f}$, plus all pre-processing values

(1) Interactively reconstruct $x_{\neg J} := \mathrm{lsb}_{\ell-J}(r-a)$ and the carry-out bit $carry$ that arises in the reconstruction as well as $x_{\neg n} := \mathrm{lsb}_{\ell-n}(r-a)$ from $\mathrm{lsb}_{\ell-J}([r])$ and $\mathrm{lsb}_{\ell-J}([a])$

(2) Non-interactively compute $[borrow_n] = DCFEval(\{\!\![(x > \mathrm{lsb}_{\ell-n}(r))\ ?\ 1 : 0]\!\!\}, x_{\neg n})$ and $[borrow_J] = DCFEval(\{\!\![(x > \mathrm{lsb}_{\ell-J}(r))\ ?\ 2^j : 0]\!\!\}, x_{\neg J})$

(3) Interactively reconstruct $x_J = \mathrm{msb}_J(r-a) \in \mathbb{Z}_{2^J}$ from $[r]$ and $[a]$ using $\mathrm{msb}_J([r]-[a]) + carry$

(4) Non-interactively compute $[\mathrm{lsb}_j(\mathrm{msb}_n(a))] := [\mathrm{lsb}_j(\mathrm{msb}_n(r))] - \big(\mathrm{lsb}_j(\mathrm{msb}_n(x)) + [borrow_n] - [borrow_J]\big) \in \mathbb{Z}_{2^\ell}$, compute $[\![\vec{e}_{\mathrm{msb}_J(r)}]\!]$ via full-domain evaluation of $\llparenthesis \vec{e}_{\mathrm{msb}_J(r)} \rrparenthesis$ and $[\![\vec{e}_{\mathrm{msb}_J(a)}]\!] = [\![\vec{e}_{\mathrm{msb}_J(r)}]\!] \lll x_J$, perform signed extension to convert this into $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$, use component-wise summation to compute $[\pm 1]$ from $[\pm\vec{e}_{\mathrm{msb}_J(a)}]$, and then evaluate $[\pm\mathbf{A}_{F,c_1}[\mathrm{msb}_J(a)]] = \langle [\pm\vec{e}_{\mathrm{msb}_J(a)}], \mathbf{A}_{F,c_1} \rangle$ and $[\pm\mathbf{A}_{F,c_0}[\mathrm{msb}_J(a)]] = \langle [\pm\vec{e}_{\mathrm{msb}_J(a)}], \mathbf{A}_{F,c_0} \rangle$

(5) Interactively compute the sign-corrected linear evaluation $[\pm 1] \cdot ([\pm\mathbf{A}_{F,c_1}[\mathrm{msb}_J(a)]] \cdot [\mathrm{lsb}_j(\mathrm{msb}_n(a))] + [\pm\mathbf{A}_{F,c_0}[\mathrm{msb}_J(a)]])$ using the generalized Beaver triple to obtain $[\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)]$

(6) Return $[\tilde{F}_{\mathbb{B}^{\ell,f,n,J}}(a)]$

Figure 11: Secure LUT-based function evaluation using the bior(5,3) **DWT in conjunction with our DCF-based vector preparation ("**bior(5,3)**+Pika" variant).**

picking a J value that gives a DWT compression error less than 0.001 and run experiments in both LAN and WAN network settings. It can be seen that for n past 20, Curl and Wave both outperform Pika with the gains exponentially increasing with n. This is to be expected as Pika is meant for running exact table lookups and does not provide any approximations other than the conversion to fixed point arithmetic. This becomes very costly as n increases and as acknowledged in the paper, fails to outperform state of the art work past $n \geq 24$ since the dominant cost is the full-sized table lookup has to be evaluated.

Our work (and Curl) provide a way to do an approximate lookup through DWTs, therefore eliminating this problem of explosion in the size of the lookup table. This leads to a much improved lookup time as n increases. Curl, on the other hand, uses arithmetic shares which results in large communication

n \ J	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
32	10	9	10	9	10	15	36	60	103	218	384	762	1580	3189	5709
31	9	11	9	9	11	14	34	57	101	200	369	745	1588	3188	5739
30	9	9	9	9	10	18	28	56	111	208	375	744	1592	3187	5685
29	9	9	9	9	9	17	27	61	109	196	379	741	1588	3197	5714
28	9	9	9	9	10	16	27	55	104	198	361	794	1934	2651	5715

Table 2: Wall-clock running time (in ms) for 1000 Haar-compressed LUT evaluations on a LAN. The online communication is a mere 24 bytes per evaluation.

n \ J	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
32	17	18	19	17	17	19	34	52	106	232	446	884	1472	2859	7162
31	17	17	17	17	18	20	27	52	105	242	374	737	1529	2925	7130
30	17	17	16	17	19	19	32	47	97	233	460	732	1522	2940	7138
29	20	20	19	19	18	20	32	54	108	261	466	940	1484	2906	7028
28	20	19	20	19	20	22	32	62	118	264	408	812	1585	2936	7249

Table 3: Wall-clock running time (in ms) for 1000 bior(5,3)**-compressed LUT evaluations on a LAN. The online communication is a mere 40 bytes per evaluation.**

n \ J	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
32	180	205	243	225	244	244	218	230	246	325	492	849	1680	3312	5869
31	208	209	213	244	221	244	224	224	245	332	473	887	1708	3318	5839
30	208	197	243	212	213	222	222	237	244	332	497	900	1695	3367	5917
29	211	208	217	224	213	244	226	238	258	342	494	895	1686	3334	5907
28	200	208	232	216	243	244	232	226	241	333	482	867	1708	3313	5872

Table 4: Wall-clock running time (in ms) for 1000 Haar-compressed LUT evaluations on a WAN. The online communication is a mere 24 bytes per evaluation.

n \ J	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
32	199	205	221	226	221	243	226	246	239	341	578	999	1628	3002	5531
31	189	213	217	224	243	244	222	227	238	355	489	873	1612	3017	7159
30	212	210	220	244	224	233	226	236	238	361	608	887	1639	3014	7248
29	197	212	232	244	225	244	222	230	238	359	602	1027	1637	2974	7203
28	192	209	238	244	221	218	244	245	242	367	506	882	1671	3036	7197

Table 5: Wall-clock running time (in ms) for 1000 bior(5, 3)**-compressed LUT evaluations on a WAN. The online communication is a mere 40 bytes per evaluation.**

overheads and high computation times. Our work overcomes this by using function secret sharing leading to significantly reduced run times and better scalability. Our work also uses exact truncation instead of Curl's probabilistic truncation leading to better accuracy. In order to showcase a more apples-to-apples comparison, we separately run Curl for varying J values (for a fixed value of $n = 32$) with the same LAN and WAN settings and report the running time in Table 7 and in Table 8, we showcase the factor improvement of our work over Curl. As can be seen, Wave outperforms Curl by as little as 8× to about 3 orders of magnitude for $J = 22$ in the WAN setting.

Regarding Ripple [10], due to significant differences between FHE versus MPC, we believe that an apples-to-apples comparison is not possible. We note, however, that for the setting $n = 32$ and $J = 20$ Ripple computation takes 3.2s for Haar and 4.2s for biorthogonal whereas our protocol on the WAN setting runs in 0.49s and 0.58s respectively.

		LAN			WAN		
n	J	Pika	Curl	Wave	Pika	Curl	Wave
16	13	26	320	9	237	3688	216
17	13	66	320	9	280	3688	216
18	13	139	320	9	332	3688	216
19	13	210	320	9	539	3688	216
20	13	426	320	9	1085	3688	216
21	12	938	179	9	1806	2799	232
22	12	1702	179	9	3805	2799	232
23	12	3707	179	9	5927	2799	232
24	12	5930	179	9	12061	2799	232
25	12	11955	179	9	22984	2799	232
26	12	22802	179	9	48215	2799	232
27	12	48485	179	9	48215	2799	232
28	12	109671	179	9	110598	2799	232
29	12	209578	179	9	210178	2799	217
30	12	402452	179	9	399949	2799	243
31	12	967698	179	9		2799	213
32	12	1895571	179	10		2799	243

Table 6: Pika LUT running time (in ms) vs. Haar-compressed LUT of Curl and this paper with DWT compression error less than 0.001 for sigmoid on a LAN and a WAN.

J	10	11	12	13	14	15	16	17	18	19	20	21	22
LAN	80	107	179	320	559	1039	2081	3867	7369	16k	33k	137k	619k
WAN	2056	2203	2799	3688	5298	8842	16k	29k	57k	111k	223k	515k	1423k

Table 7: Curl wall-clock running time (in ms) for 1000 Haar-compressed LUT evaluations on a LAN and a WAN.

J	10	11	12	13	14	15	16	17	18	19	20	21	22
LAN	8	12	18	36	56	69	58	64	72	72	87	180	392
WAN	11	11	12	16	22	36	73	128	230	341	453	607	847

Table 8: Run-time improvement over Curl (Curl takes × times longer) compared to this work. Ratio is computed for 1000 Haar-compressed LUT evaluations.