Decor: Delegated Computation on Randomness for Secure Evaluation of Nonlinear Functions

Haris Smajlović, Kyle Sheng, Timos Antonopoulos, Ruzica Piskac and Hyunghoon Cho
Yale University, USA
{*haris.smajlovic, kyle.sheng, timos.antonopoulos, ruzica.piskac, hoon.cho*}*@yale.edu*

***Abstract*—Secure multiparty computation (MPC) enables privacy-preserving data analysis across distributed computing parties, but its practicality remains limited by the cost and inaccuracy of evaluating complex nonlinear functions. Although secure multiplications are efficiently supported by Beaver multiplication triples—a cornerstone technique in MPC—existing protocols rely on polynomial approximations to evaluate more sophisticated functions, often incurring significant computational overhead and numerical imprecision. We present Decor, a framework that generalizes the core principle behind the Beaver triples to a broad class of nonlinear functions, enabling the construction of efficient MPC primitives. Decor delegates costly nonlinear operations to computations involving only random values, which can be executed in an offline preprocessing phase by a trusted dealer without access to private data. This design enables efficient and accurate evaluation of diverse functions, including trigonometric, hyperbolic, exponential, and sigmoid functions, and introduces a new general-purpose function approximation method for MPC based on Fourier series. Our experiments show that Decor achieves orders-of-magnitude improvements in accuracy while maintaining comparable or faster runtimes than existing approaches, leading to substantial utility gains for downstream applications such as implicit neural representation of images and logistic model estimation in genome-wide association studies. By demonstrating how randomized preprocessing can yield enhanced MPC primitives, Decor establishes a new framework for practical, privacy-preserving computation.**

1. Introduction

Secure multiparty computation (MPC) frameworks enable multiple parties to jointly compute functions over their private inputs without revealing those inputs to any other party [1], [2], [3]. In recent years, MPC has gained prominence in privacy-preserving machine learning and data analysis across various domains, including finance and biomedicine [4], [5]. Compared to other secure computation approaches, such as fully homomorphic encryption (FHE) [6], secret sharing-based MPC often achieves more efficient performance through lightweight interactive protocols for complex operations, provided that network delays do not become a bottleneck. However, the adoption of MPC in real-world applications remains limited, largely due to the significant challenges of performing complex, nonlinear operations securely and efficiently.

A common approach to handling nonlinear functions is to approximate them using polynomials or piecewise functions, allowing MPC to evaluate an approximation rather than the true function. These approximation-based methods, however, face an inherent trade-off between accuracy and performance. Achieving high precision may require either high-degree polynomials or many segments, both of which significantly increase the computational and communication costs of MPC. Indeed, the polynomial degree required can grow exponentially with the desired precision; for example, one analysis shows that approximating the sigmoid function to λ bits of precision requires a polynomial of degree on the order of $2^{\lambda/2}$ in the worst case [7]. As a result, supporting high-precision secure computation can be prohibitively expensive for highly nonlinear functions, as we demonstrate in this work.

To address these limitations, we introduce a new framework, Decor (DElegated COmputation on Randomness), for evaluating complex operations with high precision. Decor is based on the key insight that many nonlinear functions can be expressed through algebraic identities that allow expensive computations to be offloaded onto operations involving only random inputs, which can be efficiently performed in the real-valued domain. Notably, our framework operates in a server-aided MPC setting, where a trusted dealer performs these computations on randomness during a preprocessing phase and shares the results with the parties without access to private inputs. At execution time, the parties then perform simplified MPC operations based on the preprocessed randomness to obtain the desired outputs for the target nonlinear functions.

Our approach is inspired by the classic Beaver multiplication triple technique [8], where the product of two secret values $x \cdot y$ is computed using a random triple (a, b, c) with $c = a \cdot b$, thereby offloading the multiplication to random values a and b. We demonstrate that this concept extends well beyond multiplication, enabling the secure and efficient evaluation of a broad class of functions commonly used in scientific computing, including trigonometric functions (e.g., sine and cosine), hyperbolic functions, the exponential function, and the sigmoid function. Crucially, our method eliminates the need for polynomial or piecewise approximations of these functions, delivering high accuracy with-

out sacrificing efficiency. Moreover, our Decor primitives for sine and cosine naturally enable a general function-approximation method based on Fourier series, offering a compelling alternative to polynomial approximations for secure function evaluation in MPC, particularly for periodic functions.

We evaluate Decor through comprehensive benchmark experiments, demonstrating its superior performance compared to approximation schemes across all supported functions and varying polynomial degrees. We also show that Decor excels at approximating wave functions through our efficient constant-round implementations of the Fourier series. Building on these enhanced MPC primitives, Decor enables the development of accurate MPC protocols for analyzing private datasets where maintaining high precision is crucial, such as the estimation of model coefficients in genome-wide association studies (GWAS) [9] that capture statistical signals that are small in magnitude yet scientifically valuable, or machine learning applications that involve complex oscillatory functions, such as the sinusoidal representation networks (SIRENs) [10] in computer vision.

Overall, our results show that Decor enables high-precision MPC for a broad class of nonlinear operations, thereby extending the capabilities of secret sharing-based protocols in settings where even small errors can lead to major negative consequences (e.g., biomedicine and finance).

In summary, our work provides the following main contributions:

- An efficient, high-precision framework for evaluating complex nonlinear MPC operations in the server-aided setting.
- A range of enhanced MPC building block protocols for trigonometric, hyperbolic, and a family of exponential-based functions, including sigmoid, achieving performance superior to polynomial approximation.
- A general function approximation method for efficient MPC evaluation based on the Fourier series.
- Comprehensive benchmark evaluation of our novel MPC primitives and their demonstration in real-world applications.

Our code is available at: https://github.com/hcholab/decor.

2. Background

2.1. Secret Sharing-Based MPC

Secure multiparty computation (MPC) is a cryptographic framework that enables multiple parties to collectively compute on private data without compromising data privacy. Our work addresses the prominent class of MPC frameworks based on *secret sharing*.

Secret sharing refers to the technique of representing private data as a collection of random shares that are distributed among multiple computing parties, with the key property that the private data can be reconstructed only by combining a predetermined number of shares together. This distributed representation of data yields interactive protocols for performing arithmetic operations over the private input, including additions and multiplications, by operating solely on the secret shares without revealing any underlying private information to the participating parties.

Efficient MPC frameworks can be instantiated using *additive secret sharing* schemes [3], [11], [12], [13], [14], which split the private number x, typically encoded as an element of an algebraic structure $\mathbb{A}$ like a ring or a finite field, into a tuple of uniformly random shares $([x]_1, [x]_2, \ldots, [x]_p) \in \mathbb{A}^p$, such that $x = \sum_{i=1}^{p} [x]_i$ and each $[x]_i \in \mathbb{A}$ is held by the i-th computing party.

In this work, unless otherwise noted, we work with additive secret sharing over $\mathbb{A} = \mathbb{Z}_{2^\kappa}$, the ring of integers with a power of two modulus (2^κ), for some $\kappa \in \mathbb{N}$. This means that, with

$$[x] \coloneqq ([x]_1, [x]_2, \ldots, [x]_p) \in \mathbb{Z}_{2^\kappa}^p$$

denoting a tuple of secret shares of $x \in \mathbb{Z}$ distributed among p computing parties, the private number can be reconstructed (or *revealed*) by the parties collectively adding all secret shares together modulo 2^κ:

$$\mathsf{Reveal}([x]) := \left(\sum_{i=1}^{p} [x]_i \bmod 2^\kappa \right) = x.$$

To add (or subtract) the two additively secret-shared numbers $[x]$ and $[y]$ to obtain the secret sharing of the sum (or difference) of the two private values $[x \pm y]$, it suffices for each party to add their respective shares together:

$$[x] \pm [y] := ([x]_1 \pm [y]_1, [x]_2 \pm [y]_2, \ldots, [x]_p \pm [y]_p) = [x \pm y].$$

Note that all arithmetic operations in the secret sharing domain are modular operations, not explicitly stated for notational simplicity. Similarly, the addition and multiplication of a secret-shared private input by a public element $c \in \mathbb{Z}_{2^\kappa}$ can be computed non-interactively through the following local updates to each party's share:

$$[x] \pm c := ([x]_1 \pm c, [x]_2, \ldots, [x]_p) = [x \pm c],$$
$$c[x] := (c[x]_1, c[x]_2, \ldots, c[x]_p) = [cx].$$

Together, these operations allow arbitrary affine functions of the private inputs to be computed securely over the secret shares without revealing the private values during the process.

2.2. Secure Multiplication Using Beaver Triples

Unlike the simple operations reviewed in the previous section, computing the product of two secret-shared numbers requires interaction among parties. A standard approach employs a *Beaver multiplication triple* [8], which comprises two uniformly random numbers $a, b \in \mathbb{Z}_{2^\kappa}$ and their product $c = ab$. The triple is secret-shared among the parties as $[a]$, $[b]$, and $[c]$, such that the underlying numbers are unknown to the parties.

These additional shares enable the parties to evaluate the product of the two secret-shared numbers $[x]$ and $[y]$ as

$$[x] \cdot [y] := (x-a)(y-b) + (x-a)[b] + (y-b)[a] + [c] = [xy],$$

where $x-a \in \mathbb{Z}_{2^k}$ and $y-b \in \mathbb{Z}_{2^k}$ are masked input values (maintaining perfect secrecy of x and y) obtained by the parties computing $[x]-[a]$ and $[y]-[b]$ then collectively revealing the resulting shares. In other words, secure multiplication can be performed using an interactive protocol that requires a single round of interaction among the parties to reveal the masked inputs, followed by the non-interactive evaluation of the above affine function of the Beaver triple.

Intuitively, this technique illustrates how a nonlinear operation could be converted into a simpler one using random numbers that obey a certain predefined structure mirroring the target computation. This is a key insight that we generalize in the design of our Decor framework.

2.3. Generating Beaver Triples via a Trusted Dealer

The generation of a Beaver triple involves computing the product of two secret-shared random numbers $[a]$ and $[b]$, which requires additional computation. Several approaches have been proposed for performing this task, including those based on homomorphic encryption and oblivious transfer [15]. In this work, we adopt a server-aided approach, where the Beaver triples are randomly generated by a *trusted dealer* and secret shared with the main computing parties. This model has been adopted in several practical MPC applications due to its efficiency, despite the introduction of an additional party and the stronger assumption of no collusion between the trusted dealer and any computing party [16], [17]. Note that this additional party does not receive any information about the private data and serves only as the generator and distributor of structured randomness among the computing parties.

2.4. Fixed-Point Data Encoding in MPC

As MPC operates on finite algebraic structures, the support for real numbers as required by scientific computing applications necessitates a data encoding scheme. A standard choice is a fixed-point representation, which encodes only a subset of real numbers.

Let $2^{\mathtt{k}}$ be the characteristic of the base ring for secret sharing, and let $\mathtt{f}$ and $\mathtt{t} < \mathtt{k} - \mathtt{f}$ be the desired numbers of supported bits in the fractional and integer domains, respectively. The encoding function $\mathcal{T}_{\mathtt{f}} : \mathbb{S} \mapsto \mathbb{Z}_{2^{\mathtt{k}}}$ takes a real number x in $\mathbb{S} := [-2^{\mathtt{t}}, 2^{\mathtt{t}}) \subset \mathbb{R}$ and maps it to a ring element as

$$\mathcal{T}_{\mathtt{f}}(x) = (\lfloor x \cdot 2^{\mathtt{f}} \rfloor \bmod 2^{\mathtt{k}}).$$

The decoding is achieved using the reverse mapping $\mathcal{D}_{\mathtt{f}} : \mathbb{Z}_{2^{\mathtt{k}}} \mapsto \mathbb{S}' \subset \mathbb{S}$, where

$$\mathbb{S}' = \{-2^{\mathtt{t}} + \frac{i}{2^{\mathtt{f}}} \mid 0 \le i < 2^{\mathtt{t}+\mathtt{f}+1}, i \in \mathbb{Z}\} \subset \mathbb{R}$$

and

$$\mathcal{D}_{\mathtt{f}}(z) = \begin{cases} \frac{z}{2^{\mathtt{f}}}, & z < 2^{\mathtt{t}+\mathtt{f}}, \\ -\frac{2^{\mathtt{k}}-z}{2^{\mathtt{f}}}, & z \ge 2^{\mathtt{t}+\mathtt{f}}. \end{cases}$$

Note that the fixed number of fractional bits means that

$$\mathcal{D}_{\mathtt{f}}(\mathcal{T}_{\mathtt{f}}(x)) \neq x, \forall x \in \mathbb{S} \setminus \mathbb{S}',$$

but the precision loss is bounded as $|x - \mathcal{D}_{\mathtt{f}}(\mathcal{T}_{\mathtt{f}}(x))| < 2^{-\mathtt{f}}$. For clarity, we denote the secret sharing of the fixed-point representation of a *real number* $x \in \mathbb{S}$ with $\mathtt{f}$ fractional bits and p computing parties using the double bracket $[[\cdot]]$ as

$$[[x]]_{\mathtt{f}} \coloneqq [\mathcal{T}_{\mathtt{f}}(x)] \in \mathbb{Z}_{2^{\mathtt{k}}}^{p}.$$

In our work, we adopt the configuration where $\mathtt{k} = 256$, $\mathtt{t} = 64$, and $\mathtt{f} = 64$, which supports sufficient range and precision for most applications.

2.5. Elementary Fixed-Point Operations in MPC

An important consequence of using the fixed-point representation is that the multiplication between two fixed-point numbers doubles the number of fractional bits in the product (i.e., $[[x]]_{\mathtt{f}} \cdot [[y]]_{\mathtt{f}} = [[xy]]_{2\mathtt{f}}$). As a result, a secure truncation algorithm is required to reduce the number of fractional bits back to $\mathtt{f}$ in order to continue with subsequent computations. In our work, we adopt the MPC protocols for truncation from SecureML [18]; fixed-point division and square root from Catrina and Saxena [19], adapted for $\mathbb{Z}_{2^{\mathtt{k}}}$ rings; and comparison based on bit decomposition and prefix operations [20], [21] using Boolean secret shares (see Appendix C). Below summarizes the suite of fixed-point MPC operations used by our framework as building blocks:

$$\begin{aligned}
[[x]]_{\mathtt{f}} \pm [[y]]_{\mathtt{f}} &= [[x \pm y]]_{\mathtt{f}}, \\
\mathcal{T}_{\mathtt{f}}(c) \pm [[x]]_{\mathtt{f}} &= [[c \pm x]]_{\mathtt{f}}, \\
[[x]]_{\mathtt{f}} < [[y]]_{\mathtt{f}} &= \mathsf{Comp}\left([[x]]_{\mathtt{f}}, [[y]]_{\mathtt{f}}\right) = [x < y], \\
\mathcal{T}_{\mathtt{f}}(c)[[x]]_{\mathtt{f}} &= \mathsf{Trunc}_{\mathtt{f}}\left([[cx]]_{2\mathtt{f}}\right) = [[cx]]_{\mathtt{f}}, \\
[[x]]_{\mathtt{f}}[[y]]_{\mathtt{f}} &= \mathsf{Trunc}_{\mathtt{f}}\left([[xy]]_{2\mathtt{f}}\right) = [[xy]]_{\mathtt{f}}, \\
\frac{[[x]]_{\mathtt{f}}}{[[y]]_{\mathtt{f}}} &= \mathsf{Div}_{\mathtt{f}}\left([[x]]_{\mathtt{f}}, [[y]]_{\mathtt{f}}\right) = \left[\left[\frac{x}{y}\right]\right]_{\mathtt{f}}, \\
\sqrt{[[x]]_{\mathtt{f}}} &= \mathsf{Sqrt}_{\mathtt{f}}\left([[x]]_{\mathtt{f}}\right) = \left[\left[\sqrt{x}\right]\right]_{\mathtt{f}}, \\
\mathsf{Reveal}_{\mathtt{f}}\left([[x]]_{\mathtt{f}}\right) &= \mathcal{D}_{\mathtt{f}}(\mathsf{Reveal}\left([[x]]_{\mathtt{f}}\right)) \approx x,
\end{aligned}$$

where Comp, Trunc, Div, and Sqrt refer to the MPC subroutines for comparison, truncation, division, and square root, respectively.

2.6. Challenges of Nonlinear Function Evaluation

Although MPC schemes, in principle, enable arbitrary computations expressed as compositions of building-block operations over secret-shared private datasets, their practicality remains constrained by the cost and accuracy limitations of complex, nonlinear functions (e.g., sigmoid, exponential). This challenge is particularly acute in applications that require evaluating a large number of nonlinear functions, such as neural network inference or training.

The primary difficulty arises because MPC protocols, with the exception of certain bitwise constructions [22],

[23], [24], inherently support only arithmetic operations—i.e., additions and multiplications. As a result, prior work has explored several approximation strategies, including piecewise linear approximations, iterative refinement methods (e.g., Newton-Raphson, Goldschmidt), and polynomial approximations (e.g., Chebyshev, Taylor). Piecewise linear methods, while conceptually simple, incur significant MPC overhead due to the need for secure comparisons to determine the correct segment, and their accuracy deteriorates for high-curvature functions such as exponentials. Iterative methods like Newton-Raphson and Goldschmidt can yield efficient protocols for a limited set of operations with well-defined update formulas (e.g., division, square root), but do not generalize easily to other functions.

Polynomial approximation provides the most flexible framework for evaluating nonlinear functions in MPC, making it an ideal baseline for comparison with our novel approach. Among polynomial methods, Chebyshev approximation is particularly appealing and widely used because it minimizes the maximum approximation error (called the minimax property) over a specified interval. Specifically, a Chebyshev polynomial of degree n provides the smallest possible worst-case error among all degree-n polynomials for continuous functions on a bounded domain [25]. This well-defined error bound translates to predictable trade-offs between computational cost (degree) and accuracy.

Evaluating a polynomial approximation within MPC requires multiple rounds of secure multiplications, which grow with the polynomial's degree. While higher degrees improve accuracy, in practice, the choice of degree is limited by the numerical range and precision of the underlying fixed-point representation. Excessive degrees can cause numerical instability or overflow/underflow, imposing an upper bound on achievable precision. As demonstrated in our empirical analysis, these limitations restrict the accuracy of existing MPC-based function evaluation. Our proposed framework overcomes these challenges, enabling high-precision, efficient nonlinear computation in MPC well beyond the capabilities of prior approaches.

3. Our Approach: Decor

Decor enables efficient, high-accuracy evaluation of a range of nonlinear functions in MPC by delegating the otherwise expensive computation to a trusted dealer to be performed on randomness, leaving only the less complex operations, such as multiplications, for interactive evaluation between the parties. It generalizes the core idea behind Beaver triples for secure multiplication, aided by a trusted dealer, to any function that is decomposable with respect to the additive input. We present a graphical overview of our approach in Figure 1.

We identify several classes of functions that satisfy this property: the majority of trigonometric functions, including their compositions, such as the Fourier series; hyperbolic functions; exponential-based functions, such as logistic, sigmoid, and hyperbolic tangent; and polynomials. In the following, we first present the general framework and detail our protocols for each of the supported functional classes.

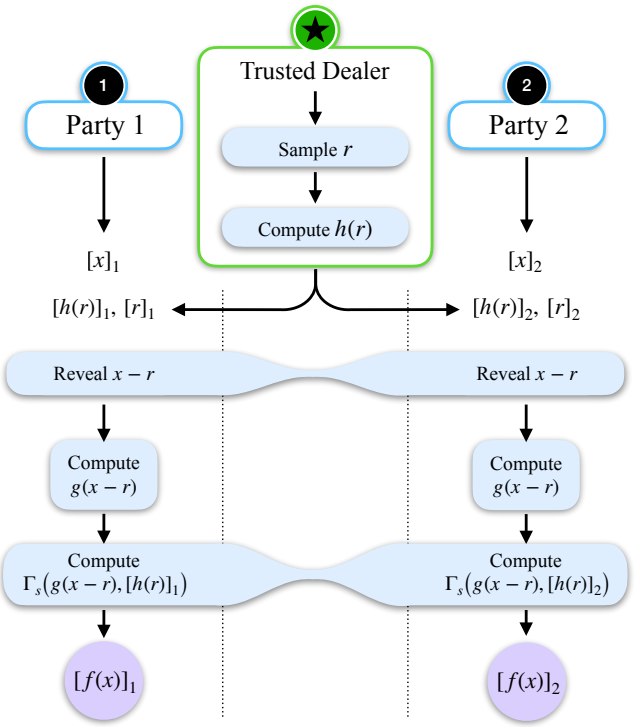

Figure 1: Decor efficiently evaluates a nonlinear function (f) that can be decomposed as a linear or MPC-efficient transformation (Γ_s) of arbitrary functions (g, h) evaluated on either the masked input ($x - r$) or the mask (r), where the latter (preprocessing) computation on randomness is delegated to a trusted dealer, who secret-shares the results with the parties to facilitate computation on the private input.

3.1. The Decor Framework

Decor addresses the secure evaluation of a general function $f : \mathbb{R} \mapsto \mathbb{R}$ for which $\exists g, h : \mathbb{R} \mapsto \mathbb{R}$, such that

$$f(u + v) = \Gamma(g(u), h(v)), \forall u, v \in \mathbb{R}, \tag{1}$$

where $\Gamma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ represents a function that can be evaluated efficiently in MPC, composed of operations such as additions, multiplications, and a limited number of divisions, square roots, or comparisons.

Given such a function f, Decor constructs a secret sharing-based MPC protocol f_s for evaluating f on a private, secret-shared input $[[x]]_{\mathbb{f}}$, with x in the supported domain $\mathbb{S} \subset \mathbb{R}$, as

$$f_s([[x]]_{\mathbb{f}}) := \Gamma_s(\mathcal{T}_{\mathbb{f}}(g(x - r)), [[h(r)]]_{\mathbb{f}}). \tag{2}$$

We use Γ_s to denote the composition of fixed-point MPC primitives presented in Section 2.5 corresponding to the

evaluation of Γ on a public $g(x-r)$ and a secret-shared $[[h(r)]]_{\mathtt{f}}$, which ensures

$$f_s([[x]]_{\mathtt{f}}) = [[\Gamma(g(x-r), h(r))]]_{\mathtt{f}} \overset{(1)}{=} [[f((x-r)+r)]]_{\mathtt{f}} = [[f(x)]]_{\mathtt{f}}$$

as desired.

The evaluation of Γ_s requires the generation of a random mask r and the masked input $x-r$, akin to secure multiplications based on Beaver triples (Section 2.2). Decor delegates the sampling of r, the computation of $h(r)$, and the secret sharing of these random values to a trusted dealer. More precisely, the preparation for Γ_s proceeds in the following steps:

(1) The trusted dealer samples the value of $r_{\mathtt{f}} = \mathcal{T}_{\mathtt{f}}(r) \in \mathbb{Z}_{2^{\mathtt{k}}}$ uniformly at random from $\mathbb{Z}_{2^{\mathtt{k}}}$; computes $r = \mathcal{D}_{\mathtt{f}}(r_{\mathtt{f}}) \in \mathbb{S}$, $h(r) \in \mathbb{R}$; and distributes the secret shares $[[r]]_{\mathtt{f}}, [[h(r)]]_{\mathtt{f}} \in \mathbb{Z}_{2^{\mathtt{k}}}^p$ among p computing parties. Note that these steps can be performed entirely in a preprocessing phase without knowledge of the private inputs.

(2) The parties compute $[[x-r]]_{\mathtt{f}} = [[x]]_{\mathtt{f}} - [[r]]_{\mathtt{f}}$; collectively reveal $x - r \in \mathbb{S}$; and locally compute $g(x-r) \in \mathbb{R}$ and its encoding $\mathcal{T}_{\mathtt{f}}(g(x-r)) \in \mathbb{Z}_{2^{\mathtt{k}}}$ before interactively evaluating Γ_s. An important caveat is that $\mathsf{Reveal}([[x-r]]_{\mathtt{f}})$ may differ from $x-r$ due to a wraparound in the secret sharing domain. We introduce techniques to overcome this issue in Section 3.3.

We summarize the above method with the following lemma.

Lemma 1. *The p-party MPC protocol $f_s : \mathbb{Z}_{2^{\mathtt{k}}}^p \mapsto \mathbb{Z}_{2^{\mathtt{k}}}^p$ (2) securely evaluates the function $f : \mathbb{R} \mapsto \mathbb{R}$ satisfying the decomposition property in Equation 1. That is,*

$$f_s([[x]]_{\mathtt{f}}) = [[f(x)]]_{\mathtt{f}}, \forall x \in \mathbb{S} \subset \mathbb{R},$$

and f_s can be efficiently evaluated with nonlinear computation on randomness delegated to a trusted dealer.

Example. For intuition, consider the exponential function $f(x) = e^x$, which can be decomposed as $f(x) = e^{x-r}e^r$, for any $x, r \in \mathbb{R}$. To compute $[[e^x]]_{\mathtt{f}}$ given $[[x]]_{\mathtt{f}}$, the trusted dealer samples r and distributes $[[r]]_{\mathtt{f}} \in \mathbb{Z}_{2^{\mathtt{k}}}$ and $[[e^r]]_{\mathtt{f}} \in \mathbb{Z}_{2^{\mathtt{k}}}$ among the computing parties. The parties reveal $x-r \in \mathbb{S}$ (modulo the wraparound issue addressed in Section 3.3), compute e^{x-r}, and obtain the desired result as

$$[[e^x]]_{\mathtt{f}} = \mathsf{Trunc}_{\mathtt{f}}(\mathcal{T}_{\mathtt{f}}\left(e^{x-r}\right)[[e^r]]_{\mathtt{f}}).$$

3.2. Multivariate Functions and Decompositions

Decor immediately generalizes to both multivariate functions and their decompositions into protocols involving the evaluation of multiple functions over the masked input or the random masks. Specifically, for any function $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ for which $\exists g_i, h_i : \mathbb{R}^d \to \mathbb{R}, i \in \{1, \ldots, n\}$, such that the following holds

$$\hat{f}(\mathbf{u}+\mathbf{v}) = \hat{\Gamma}(g_1(\mathbf{u}), \ldots, g_n(\mathbf{u}), h_1(\mathbf{v}) \ldots, h_n(\mathbf{v})), \quad (3)$$

where $\hat{\Gamma}$ is a composition of operations that can be evaluated efficiently in MPC, then we can construct an accurate and MPC-friendly protocol $\hat{f}_s$ for evaluating $\hat{f}$ on a private input vector $\mathbf{x} \in \mathbb{R}^d$ as follows:

$$\hat{f}_s([[\mathbf{x}]]_{\mathtt{f}}) := \hat{\Gamma}_s(\mathcal{T}_{\mathtt{f}}(g_1(\mathbf{x}-\mathbf{r})), \ldots, \mathcal{T}_{\mathtt{f}}(g_n(\mathbf{x}-\mathbf{r})), [[h_1(\mathbf{r})]]_{\mathtt{f}}, \ldots, [[h_n(\mathbf{r})]]_{\mathtt{f}}), \quad (4)$$

where $\hat{\Gamma}_s$ denotes the MPC protocol corresponding to the evaluation of $\hat{\Gamma}$. Analogous to Lemma 1, the correctness of the MPC primitives in $\hat{\Gamma}_s$ and the decomposition property of $\hat{f}$ (Equation 3) ensures

$$\hat{f}_s([[\mathbf{x}]]_{\mathtt{f}}) = [[\hat{f}(\mathbf{x})]]_{\mathtt{f}}, \forall \mathbf{x} \in \mathbb{S}^d \subset \mathbb{R}^d,$$

thus yielding an efficient method for securely evaluating $\hat{f}$.

3.3. Techniques for Correctness and Stability

The simplified presentation of our framework in the previous section serves as an overview; however, it leaves two significant challenges unresolved. We now present the additional techniques used to overcome these challenges.

The first challenge is that computing $g(x-r)$ and $h(r)$—in the real domain—can be numerically unstable, since r is sampled from a large ring corresponding to the secret sharing domain (e.g., $\mathbb{Z}_{2^{256}}$). To illustrate, in the previous exponential example, the value of e^{x-r} may be outside the domain of $\mathcal{T}_{\mathtt{f}}$ and introduce a catastrophic error during the encoding step.

The second challenge stems from the fact that operations involving additive secret shares employ modular arithmetic, which can lead to different outputs, depending on the wraparound behavior, when computing $[[x-r]]_{\mathtt{f}}$ and revealing the result as a real number. While our intended output is $x - r \in \mathbb{R}$, the parties may instead obtain $x - r + 2^{\mathtt{k}-\mathtt{f}}$ as the result of evaluating $\mathcal{D}_{\mathtt{f}}(\mathsf{Reveal}([[x-r]]_{\mathtt{f}}))$ if calculating $x - r$ in the secret sharing domain causes a wraparound. This potential discrepancy must be accounted for in the subsequent steps in order to obtain the correct final results.

3.3.1. Domain Reduction for Stability. To address the issue of numerical instability associated with the potentially large values of $r \in \mathbb{S}$ used to mask the input (e.g., in Eqs. 2 and 4), we define a safe subset of the input domain $\mathbb{S}^\dagger$, represented by an interval $[a, b) \subset \mathbb{S}$, to confine the values of both the input x and the mask r. We choose $a, b \in \mathbb{R}$ for a given target function, if required, to guarantee that the $g(x-r)$ and $h(r)$ terms remain within $\mathbb{S}$, i.e., the representable data range of the base secret sharing scheme. This input restriction is akin to that of polynomial approximation methods, where an interval must be prespecified to obtain the coefficients of the approximating polynomial prior to the computation. A meaningful difference is that the choice of $\mathbb{S}^\dagger$ in Decor is solely a function of the numerical bounds of the underlying fixed-point encoding scheme and the desired function being evaluated, and thus does not require special tuning with respect to the accuracy-runtime trade-offs, as is the case for

polynomial approximations. Additionally, note that without the domain reduction, the real values of $x - r$ and r may require large floating-point representations and necessitate further increase in size of the fixed-point representation for some functions, thus adding to the computational cost.

3.3.2. Ring Switching Strategy. Our design allows $\mathbb{S}^\dagger$ to be defined separately from $\mathbb{S}$ of the base scheme. This is a desirable property given that the parameters of the base scheme are typically predetermined to support the global computation being performed. It also allows the use of a different $\mathbb{S}^\dagger$ for each target function.

We introduce an efficient ring switching strategy to seamlessly handle the crossover between these two differing data domains. We first define the characteristic of the auxiliary ring assigned to $\mathbb{S}^\dagger = [a, b)$ as $2^{\mathtt{k}'}$ with $\mathtt{k}' < \mathtt{k}$. Next, with $l = b - a$ denoting the length of the interval for $\mathbb{S}^\dagger$, the parties scale the input to $\mathbb{Z}_{2^{\mathtt{k}'}}$, defined over the same interval, by first computing

$$[[(x - a)/l]]_{\mathtt{k}'} = \mathsf{Trunc}_{\mathtt{k}'}(([[x]]_{\mathtt{f}} - \mathcal{T}_{\mathtt{f}}(a)) \cdot \mathcal{T}_{\mathtt{f}}(1/l)), \quad (5)$$

then performing modular reduction of the individual shares modulo $2^{\mathtt{k}'}$, which results in a new set of secret shares over the smaller ring $\mathbb{Z}_{2^{\mathtt{k}'}}$ whose elements map to numbers in $\mathbb{S}^\dagger$. Sampling of the random mask r and the revealing of $x - r$ can now be performed in this smaller ring with appropriate decoding that outputs the corresponding real numbers in $\mathbb{S}^\dagger$.

After the calculations of $g(x - r)$ and $h(r)$ in $\mathbb{R}$, respectively by the computing parties and the trusted dealer, these results are encoded back into the original ring $\mathbb{Z}_{2^{\mathtt{k}}}$ with scale $\mathtt{f}$ to evaluate the transformed MPC protocols in Decor (Eqs. 2 and 4).

3.3.3. Secure Wraparound Correction. For a chosen domain $\mathbb{S}^\dagger = [a, b)$ of length $l = b - a$ revealing $x - r$ following the above scheme will decode to $x - r + l \in \mathbb{S}^\dagger$ whenever $x < r$ due to the wraparound in the subtraction. In this event, f_s in Equation 2 will (incorrectly) evaluate to

$$\Gamma_s(\mathcal{T}_{\mathtt{f}}(g(x - r + l)), [[h(r)]]_{\mathtt{f}}) = [[f(x + l)]]_{\mathtt{f}} \neq [[f(x)]]_{\mathtt{f}}.$$

To address this issue, we employ a secure comparison protocol to obtain the encrypted value of $[x < r] \in \mathbb{Z}_{2^{\mathtt{k}}}^p$, where the comparison evaluates to one if the inequality is true and zero if false. With $x' - r$ denoting the observed outcome after revealing $x - r$, where x' can be either x or $x + l$, we compute the final result as

$$[[f(x')]]_{\mathtt{f}} + [x < r] \cdot ([[f(x' - l)]]_{\mathtt{f}} - [[f(x')]]_{\mathtt{f}}),$$

securely selecting the correct result, i.e., $[[f(x)]]_{\mathtt{f}}$, between the two cases. We summarize the protocol steps in Table 7 (Appendix A).

For efficiency, we use a hybrid protocol for secure comparison that first converts input shares from $\mathbb{Z}_{2^{\mathtt{k}'}}$ into Boolean secret shares via bit decomposition, followed by a bitwise comparison protocol (Appendix C). The resulting comparison bit can then be mapped back to any target ring—$\mathbb{Z}_{2^{\mathtt{k}}}$ in our case—for evaluating the above expression.

In addition, our ring-switching strategy, which maps the input shares to a smaller ring $\mathbb{Z}_{2^{\mathtt{k}'}}$, substantially reduces the computational cost of the wraparound correction. Since the secure comparison then operates over only $\mathtt{k}'$ bits (rather than $\mathtt{k}$), both the interaction rounds decrease from $\log_2(\mathtt{k})$ to $\log_2(\mathtt{k}')$, and the communication bandwidth is reduced proportionally by a factor of $\mathtt{k}'/\mathtt{k}$.

3.3.4. Efficient Support for Periodic Functions. Decor is particularly well-suited for securely evaluating periodic functions, which naturally avoid both numerical challenges previously discussed in this section. Periodicity often (though not always) eliminates divergence, mitigating risks of numerical overflow or underflow. Moreover, the wraparound issue vanishes when the supported range $\mathbb{S}^\dagger$ is set to have length equal to the function's period, ensuring that the evaluation output remains correct regardless of whether a wraparound occurs. This removes the need for a secure comparison for wraparound correction. These properties contribute to the exceptional performance of Decor on periodic functions, including sine, cosine, and other waveforms represented through Fourier series (Section 4.3).

3.4. Security

Decor's MPC protocols inherit the security guarantees of the underlying secret sharing-based MPC framework. With additive secret sharing, this implies information-theoretic privacy of the input against semi-honest adversaries, assuming at least one non-colluding computing party and no collusion between the trusted dealer and any party.

We provide a sketch of a simulation-based proof of the security of Decor protocols, which relies primarily on invoking MPC building-block primitives whose security is already established. Consider an honest-but-curious adversary that corrupts a subset of parties $T \subset \{0, 1, \ldots, p\}$, where 0 denotes the trusted dealer. Under our security assumptions, $0 \in T$ implies $T = \{0\}$. Thus, either $T = \{0\}$, in which case perfect secrecy follows immediately because the adversary receives no incoming communication from the parties, or $T \subset \{1, \ldots, p\}$ with $|T| < p$. In the latter case, and without loss of generality, assume $1 \notin T$. Our goal is to construct a simulator that produces a view for this adversary that is indistinguishable from its real-world view.

Because the base MPC framework ensures the security of its primitives like multiplication and comparison, the adversary's view arising from such primitives can be simulated independently. The only components that require additional analysis are the Decor-specific steps that map the input secret shares $[[x]]_{\mathtt{f}}$ to the arguments $g(x - r)$ and $[[h(r)]]_{\mathtt{f}}$ for the Decor-transformed protocol (Equation 2). These steps consist of the domain reduction of $[[x]]_{\mathtt{f}}$ to obtain secret shares $[\tilde{x}]$ in the reduced ring $\mathbb{Z}_{2^{\mathtt{k}'}}$ corresponding to a scaled input from $\mathbb{S}^\dagger$; distributing among parties the shares $[\tilde{r}]$ of a random scaled mask in $\mathbb{Z}_{2^{\mathtt{k}'}}$, corresponding to some $r \in \mathbb{S}^\dagger$; the revealing of $\tilde{x} - \tilde{r}$ and decoding to obtain $x - r \in \mathbb{S}^\dagger$; and the secret sharing of $[[h(r)]]_{\mathtt{f}}$.

Function	Γ Decomposition
Exponential	$e^{x+y} = e^x e^y$
Sigmoid	$\sigma(x+y) = (\sigma(x)\sigma(y)) \,/\, (2\sigma(x)\sigma(y) - \sigma(x) - \sigma(y) + 1)$
Sine	$\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y)$
Cosine	$\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y)$
Tangent	$\tan(x+y) = (\tan(x) + \tan(y)) \,/\, (1 - \tan(x)\tan(y))$
Cotangent	$\cot(x+y) = (\cot(x)\cot(y) - 1) \,/\, (\cot(x) + \cot(y))$
Hyperbolic Sine	$\sinh(x+y) = \sinh(x)\cosh(y) + \cosh(x)\sinh(y)$
Hyperbolic Cosine	$\cosh(x+y) = \cosh(x)\cosh(y) + \sinh(x)\sinh(y)$
Hyperbolic Tangent	$\tanh(x+y) = (\tanh(x) + \tanh(y)) \,/\, (1 + \tanh(x)\tanh(y))$
Hyperbolic Cotangent	$\coth(x+y) = (\coth(x)\coth(y) + 1) \,/\, (\coth(x) + \coth(y))$
Polynomial	$P(x+y) = \sum_{i=0}^{d} c_i \sum_{j=0}^{i} \binom{i}{j} x^{i-j} y^j$

Table 1: Examples of functions supported by Decor and their decompositions with respect to the sum of the inputs.

Excluding invocations of standard MPC operations, the adversary's view in these steps consists of the value $x - r$ (equivalently $\tilde{x} - \tilde{r}$) together with the corrupted shares $[\tilde{r}]_{i \in T}$ and $[\mathcal{T}_{\mathbb{f}}(h(r))]_{i \in T}$. Because $1 \notin T$, additive secret sharing guarantees that $[\tilde{r}]_{i \in T}$ is distributed identically to independent uniform elements of $\mathbb{Z}_{2^{k'}}$. Since $\tilde{r}$ is chosen uniformly at random by the trusted dealer, the revealed value $\tilde{x} - \tilde{r}$ is also uniform and independent of the corrupted shares. Likewise, $[\mathcal{T}_{\mathbb{f}}(h(r))]_{i \in T}$ forms a fresh additive sharing, and the shares observed by the adversary are again uniformly random. Consequently, the entire Decor-specific view of the adversary with corruption set T can be simulated by replacing each of these observed quantities with an independently chosen uniform ring element, and composing this with the simulations of the invoked MPC primitives. This completes the sketch of the security argument.

3.5. Supported Functions

Any function that admits a decomposition over sums of inputs into an MPC-efficient expression, as defined in Equations 1 and 3, can be evaluated accurately and efficiently within our Decor framework. Table 1 summarizes key functions that satisfy this property, along with their corresponding decompositions. We next describe each class of supported functions in greater detail and discuss useful generalizations of these primitives, such as Fourier-based function approximations.

We remark that several important functions currently fall outside our framework, including logarithmic, comparison, and division operations. These can be handled using specialized MPC protocols or function approximation techniques, which can be composed with Decor-based operations within a larger protocol in a complementary manner.

3.5.1. Exponential-Based Functions. The exponential function example in Section 3, $f(x) = e^x$, permits a MPC-friendly decomposition $f(x+y) = f(x)f(y)$. This observation can be generalized to a broader class of exponential-based functions. Any function $f_\Delta : \mathbb{R} \mapsto \mathbb{R}$ of the form

$$f_\Delta(x; \alpha, \beta, \gamma, \delta, \lambda) = \frac{\alpha e^{\lambda x} + \beta}{\gamma e^{\lambda x} + \delta},$$

with parameters $\alpha, \beta, \gamma, \delta, \lambda \in \mathbb{R}$ and defined on all x such that $\gamma e^{\lambda x} + \delta \neq 0$, can be decomposed with respect to the sum of the inputs as follows (see Appendix D for proof):

$$f_\Delta(x+y) = \frac{A f_\Delta(x) f_\Delta(y) + B f_\Delta(x) + C f_\Delta(y) + D}{E f_\Delta(x) f_\Delta(y) + F f_\Delta(x) + G f_\Delta(y) + H},$$

where

$$\begin{aligned} A &= \alpha\delta^2 + \beta\gamma^2, & B &= -\alpha\delta\beta - \alpha\beta\gamma, \\ C &= -\alpha\delta\beta - \alpha\beta\gamma, & D &= \alpha\beta^2 + \alpha^2\beta, \\ E &= \gamma\delta^2 + \delta\gamma^2, & F &= -\gamma\delta\beta - \alpha\gamma\delta, \\ G &= -\gamma\delta\beta - \alpha\gamma\delta, & H &= \gamma\beta^2 + \alpha^2\delta. \end{aligned}$$

This class of functions includes the exponential, sigmoid, and some hyperbolic functions:

$$\begin{aligned} e^x &= f_\Delta(x; 1, 0, 0, 1, 1), \\ \sigma(x) &= f_\Delta(x; 1, 0, 1, 1, 1) = \frac{e^x}{e^x + 1}, \\ \tanh(x) &= f_\Delta(x; 1, -1, 1, 1, 2) = \frac{e^{2x} - 1}{e^{2x} + 1}, \\ \coth(x) &= f_\Delta(x; 1, 1, 1, -1, 2) = \frac{e^{2x} + 1}{e^{2x} - 1}. \end{aligned}$$

Applying the above decomposition formula yields an efficient protocol for evaluating any function in this class.

To illustrate, we provide a concrete example based on the sigmoid function, which can be evaluated as

$$\begin{aligned} \sigma(x+y) &= \Gamma(\sigma(x), \sigma(y)), \text{ where} \\ \Gamma(x, y) &= \frac{xy}{2xy - x - y + 1}, \ \forall x, y \in \mathbb{R}. \end{aligned}$$

Algorithm 1 Decor-Sigmoid $(15 + 4\delta + 4\log_2 \mathfrak{f}$ rounds)

Require: Secret-shared $[[x]]_{\mathfrak{f}}$, segment $[a, b) \subset \mathbb{S}$, $l = b - a$, division depth δ

$[\tilde{r}]_{2^{\mathfrak{f}}}, [[\sigma(r)]]_{\mathfrak{f}} \leftarrow \mathsf{Dealer}()$ ▷ $r = l \cdot \tilde{r} + a$
$[\tilde{x}]_{2^{\mathfrak{f}}} \leftarrow \mathsf{Trunc}_{\mathfrak{f}}(\mathcal{T}_{\mathfrak{f}}(1/l) \cdot ([[x]]_{\mathfrak{f}} - \mathcal{T}_{\mathfrak{f}}(a))) \mod 2^{\mathfrak{f}}$
$\tilde{x} - \tilde{r} \leftarrow \mathsf{Reveal}_{\mathfrak{f}}([\tilde{x}]_{2^{\mathfrak{f}}} - [\tilde{r}]_{2^{\mathfrak{f}}})$ ▷ 1 rnd
$x - r \leftarrow l \cdot (\tilde{x} - \tilde{r}) + a$
$[c] \leftarrow [\tilde{x}]_{2^{\mathfrak{f}}} < [\tilde{r}]_{2^{\mathfrak{f}}}$ ▷ (Alg. 6) $2 + \log_2 \mathfrak{f}$ rnd
$[[s_h]]_{\mathfrak{f}} \leftarrow \mathcal{T}_{\mathfrak{f}}(\sigma(x-r-l) - \sigma(x-r))[c]$
$[[s]]_{\mathfrak{f}} \leftarrow \mathcal{T}_{\mathfrak{f}}(\sigma(x-r)) + [[s_h]]_{\mathfrak{f}}$
$[[e_1]]_{\mathfrak{f}} \leftarrow [[\sigma(r)]]_{\mathfrak{f}} - \mathcal{T}_f(1)$
$[[e_2]]_{\mathfrak{f}} \leftarrow [[s]]_{\mathfrak{f}} - \mathcal{T}_f(1)$
$[[d_0]]_{\mathfrak{f}} \leftarrow \mathsf{Trunc}_{\mathfrak{f}}([[e_1]]_{\mathfrak{f}}[[e_2]]_{\mathfrak{f}})$ ▷ 2 rnd
$[[d_1]]_{\mathfrak{f}} \leftarrow \mathsf{Trunc}_{\mathfrak{f}}([[s]]_{\mathfrak{f}}[[\sigma(r)]]_{\mathfrak{f}})$ ▷ 2 rnd
$[[d_2]]_{\mathfrak{f}} \leftarrow [[d_0]]_{\mathfrak{f}} + [[d_1]]_{\mathfrak{f}}$
return $\mathsf{Div}_{\mathfrak{f}}([[d_1]]_{\mathfrak{f}}, [[d_2]]_{\mathfrak{f}}, \delta)$ ▷ (Alg. 7) $3\log_2 \mathfrak{f} + 4\delta + 7$ rnd

This yields an efficient MPC protocol for the sigmoid:

$$\sigma_s([[x]]_{\mathfrak{f}}) \overset{(2)}{:=} \Gamma_\sigma(\mathcal{T}_{\mathfrak{f}}(\sigma(x - r)), [[\sigma(r)]]_{\mathfrak{f}}) = [[\sigma(x)]]_{\mathfrak{f}},$$

where Γ_σ is a composition of MPC primitives for computing Γ on the above inputs. We provide the full protocol in Algorithm 1, which incorporates additional strategies discussed in Section 3.3 for domain reduction and wraparound correction. While the protocol involves a division operation, the denominator of Γ is bounded in $[0, 1]$, which improves efficiency by simplifying the scale normalization step in the division protocol (Appendix C).

3.5.2. Trigonometric Functions. Trigonometric functions are decomposable with respect to the sum of the angles according to the identities.

$$\begin{aligned}
\sin(x + y) &= \sin(x)\cos(y) + \cos(x)\sin(y),\\
\cos(x + y) &= \cos(x)\cos(y) - \sin(x)\sin(y),\\
\tan(x + y) &= \frac{\tan(x) + \tan(y)}{1 - \tan(x)\tan(y)},\\
\cot(x + y) &= \frac{\cot(x)\cot(y) - 1}{\cot(x) + \cot(y)}.
\end{aligned}$$

While we do not list them here, a few other trigonometric functions (e.g., secant and cosecant) support similar decompositions. As before, the terms associated with the input corresponding to the random mask (e.g., y in the above identities) can be computed by a trusted dealer and secret shared with the parties, who then interactively compute the rest of the expression to evaluate the desired function.

Sine and cosine serve as insightful examples where the decomposition involves complex nonlinear operations different from the target function; each of them requires the evaluation of both sine and cosine on the inputs. Such functions can still be efficiently computed within our framework as long as each term involves only one of the inputs—i.e., the masked input or the mask (see Section 3.2).

For instance, the sine function can be expressed as:

$$\sin(x + y) = \Gamma(\sin(x), \cos(x), \sin(y), \cos(y)), \text{ where}$$
$$\Gamma(x_1, x_2, y_1, y_2) = x_1 y_2 + x_2 y_1, \quad \forall x, y \in \mathbb{R}.$$

This leads to the following Decor protocol for sine:

$$\sin_s([[x]]_{\mathfrak{f}}) := \Gamma_{\sin}(\mathcal{T}_{\mathfrak{f}}(\sin(x - r)), \mathcal{T}_{\mathfrak{f}}(\cos(x - r)), [[\sin(r)]]_{\mathfrak{f}}, [[\cos(r)]]_{\mathfrak{f}}),$$

which outputs $[[\sin(x)]]_{\mathfrak{f}}$ given an MPC protocol $\Gamma_{\sin}$ that evaluates Γ.

Periodicity of trigonometric functions removes the need for wraparound correction when the length of the reduced domain (e.g., $[0, 2\pi)$) coincides with the period 2π. This results in highly efficient MPC protocols for these functions, as illustrated in Algorithm 2 for the sine function, which requires only three interaction rounds.

Algorithm 2 Decor-Sine (3 rounds)

Require: Secret-shared input $[[x]]_{\mathfrak{f}}$

$[\tilde{r}]_{2^{\mathfrak{f}}}, [[\sin(r)]]_{\mathfrak{f}}, [[\cos(r)]]_{\mathfrak{f}} \leftarrow \mathsf{Dealer}()$ ▷ $r = 2\pi\tilde{r}$
$[\tilde{x}]_{2^{\mathfrak{f}}} \leftarrow \mathsf{Trunc}_{\mathfrak{f}}(\mathcal{T}_{\mathfrak{f}}(1/2\pi)[[x]]_{\mathfrak{f}}) \mod 2^{\mathfrak{f}}$ ▷ 1 rnd
$\tilde{x} - \tilde{r} \leftarrow \mathsf{Reveal}_{\mathfrak{f}}([\tilde{x}]_{2^{\mathfrak{f}}} - [\tilde{r}]_{2^{\mathfrak{f}}})$ ▷ 1 rnd
$x - r \leftarrow 2\pi \cdot (\tilde{x} - \tilde{r})$
$[[t_1]]_{\mathfrak{f}} \leftarrow \mathcal{T}_{\mathfrak{f}}(\sin(x - r))[[\cos(r)]]_{\mathfrak{f}}$
$[[t_2]]_{\mathfrak{f}} \leftarrow \mathcal{T}_{\mathfrak{f}}(\cos(x - r))[[\sin(r)]]_{\mathfrak{f}}$
return $\mathsf{Trunc}_{\mathfrak{f}}([[t_1]]_{\mathfrak{f}} + [[t_2]]_{\mathfrak{f}})$ ▷ 1 rnd

3.5.3. Hyperbolic Functions. Hyperbolic functions are analogues of trigonometric functions defined on the hyperbola. They share identities of the same form, which can be utilized for decompositions in Decor. For example, the hyperbolic sine and cosine functions have the following decompositions similar to those of sine and cosine:

$$\begin{aligned}
\sinh(x + y) &= \sinh(x)\cosh(y) + \cosh(x)\sinh(y),\\
\cosh(x + y) &= \cosh(x)\cosh(y) + \sinh(x)\sinh(y).
\end{aligned}$$

Decor enables efficient evaluation of these functions in a similar fashion, with the key difference that these functions are non-periodic and thus require wraparound correction (Section 3.3), which incurs an additional computational cost.

3.5.4. Function Approximation via Fourier Series. Decor's MPC-friendly decomposition of sine and cosine functions enables an efficient approach to general function approximation based on the Fourier series, providing a useful alternative to polynomial approximation.

A Fourier series can be used to approximate any real-valued function f over an interval $[a, b)$ of length (period) $L = b - a$ as

$$f(x) \approx a_0 + \sum_{k=1}^{N} \left(a_k \cos\left(\frac{2k\pi}{L}x\right) + b_k \sin\left(\frac{2k\pi}{L}x\right)\right),$$

given an integer N and the coefficients determined through integration. In our setting, the target function f is publicly

known, so the above coefficients can be precomputed as public constants for given f, a, b, and N.

Decomposing all sine and cosine terms in the Fourier series with respect to $x - r$ and r as inputs, we obtain an expression that can be evaluated efficiently involving sine/cosine evaluations only on $x - r$ and r. Specifically, once the masked input $x - r$ (modulo L) is revealed, the parties compute the following for $k \in \{1, \ldots, N\}$:

$$v_k = a_k \cos\left(\frac{2k\pi}{L}(x-r)\right) + b_k \sin\left(\frac{2k\pi}{L}(x-r)\right),$$
$$w_k = b_k \cos\left(\frac{2k\pi}{L}(x-r)\right) - a_k \sin\left(\frac{2k\pi}{L}(x-r)\right).$$

Then the following expression is computed:

$$\mathcal{T}_f(a_0) + \mathsf{Trunc}_f\left(\sum_{k=1}^{N} [[t_k]]_{2^f}\right), \text{ where}$$

$$[[t_k]]_{2^f} = \mathcal{T}_f(v_k)\left[\left[\cos\left(\frac{2k\pi}{L}r\right)\right]\right]_f + \mathcal{T}_f(w_k)\left[\left[\sin\left(\frac{2k\pi}{L}r\right)\right]\right]_f.$$

Note that the sine/cosine terms that include only r are evaluated and secret-shared by a trusted dealer. Additionally, the only three interactive steps are revealing $x - r$ and secure truncation (rescaling) of the output, resulting in an efficient, constant-round function approximation routine for any degree N. We provide the full protocol in Algorithm 3.

Algorithm 3 Decor-Fourier (3 rounds)

Require: Secret-shared $[[x]]_f$, segment $[a, b) \subset \mathbb{S}$, $L = b - a$

1: $[\tilde{r}]_{2^f} \leftarrow \mathsf{Dealer}()$ ▷ $r = L \cdot \tilde{r} + a$
2: $\left[\left[\sin\left(\frac{2\pi}{L}r\right)\right]\right]_f, \ldots, \left[\left[\sin\left(\frac{2N\pi}{L}r\right)\right]\right]_f \leftarrow \mathsf{Dealer}()$
3: $\left[\left[\cos\left(\frac{2\pi}{L}r\right)\right]\right]_f, \ldots, \left[\left[\cos\left(\frac{2N\pi}{L}r\right)\right]\right]_f \leftarrow \mathsf{Dealer}()$
4: $[\tilde{x}]_{2^f} \leftarrow \mathsf{Trunc}_f(\mathcal{T}_f(1/L) \cdot ([[x]]_f - \mathcal{T}_f(a))) \mod 2^f$ ▷ 1 rnd
5: $\tilde{x} - \tilde{r} \leftarrow \mathsf{Reveal}_f([\tilde{x}]_{2^f} - [\tilde{r}]_{2^f})$ ▷ 1 rnd
6: $x - r \leftarrow L \cdot (\tilde{x} - \tilde{r}) + a$
7: Compute $[[t_k]]_{2^f}, \forall k = \{1, \ldots, N\}$ ▷ See text for t_k
8: $[[s]]_f \leftarrow \mathsf{Trunc}_f([[t_1]]_{2^f} + \cdots + [[t_N]]_{2^f})$ ▷ 1 rnd
9: **return** $\mathcal{T}_f(a_0) + [[s]]_f$

3.5.5. Polynomial Functions. Decor improves polynomial evaluation in MPC by delegating computation of input powers to a trusted dealer. The decomposition of a polynomial $\rho(x) = \sum_{i=1}^{d} c_i x^i$, $c_i \in \mathbb{R}$ follows the binomial expansion

$$\rho(x+y) = \sum_{i=1}^{d} c_i \sum_{j=0}^{i} \binom{i}{j} x^{i-j} y^j,$$

leading to the following expression for Decor evaluation:

$$[[\rho(x)]]_f = \sum_i \mathcal{T}_f(c_i) \sum_{j=0}^{i} \binom{i}{j} \mathcal{T}_f((x-r)^{i-j})[[r^j]]_f,$$

where the powers r^j are precomputed and secret-shared by a trusted dealer and $(x - r)^{i-j}$ terms are computed by the parties using the revealed $x - r$. Wraparound correction is necessary in Decor's polynomial evaluation.

4. Experimental Results

We conducted comprehensive benchmark experiments to evaluate the performance of the Decor protocols, encompassing nine representative elementary functions, applications of Fourier approximation to wave functions, and two real-world machine learning tasks in medical image analysis and genomic data analysis involving nonlinear operators. Chebyshev polynomial approximation served as our primary comparison baseline, evaluated using the Clenshaw method [26], a numerically stable and efficient evaluation method for Chebyshev polynomials. Overall, our results demonstrate substantial improvements in accuracy beyond what is achievable with polynomial approximation, highlighting the practical utility of Decor and its potential to generalize beyond the set of functions analyzed in this work.

4.1. Experimental Setup and Evaluation Metrics

All experiments were performed in a simulated MPC environment based on a single machine with an AMD Ryzen Threadripper 3960X CPU and 256 GB of RAM. Our default network setup consisted of three parties (two computing parties and a trusted dealer), with each party assigned to a separate process and communicating over UNIX sockets with a realistic, simulated wide-area network (WAN) latency of 20 ms [27], [28]. We also assessed performance in settings with 5 or 10 parties (Appendix A).

Accuracy metrics include the mean or max absolute error (mean/max AE) between the MPC output and the ground truth, evaluated across 500 equidistant points within the specified interval. Each polynomial approximation-based method was evaluated at varying polynomial degrees from 5 to 70. Our performance metrics include total runtime (in seconds), the network bandwidth used by the trusted dealer for preprocessing, the cumulative network bandwidth used by the computing parties in the online phase (excluding the dealer), and the number of interactive rounds between the parties. All measurements were averaged over three runs.

We implemented all protocols in Shechi [29], a Python compiler based on Codon [30] for developing MPC and homomorphic encryption protocols, which provided the necessary subroutines and utility functions for secret sharing that streamlined our implementation of the Decor protocols.

4.2. Elementary Nonlinear Functions

In Table 2, we present the accuracy evaluation results across a representative set of elementary functions, including four trigonometric functions (sine, cosine, tangent, and cotangent), two exponential-based functions (exponential and sigmoid), and three hyperbolic functions (hyperbolic sine, cosine, and tangent). Each function was evaluated using Decor and Chebyshev polynomials (of degrees 20 and 50) over a chosen interval, the latter representing a prominent approach to polynomial approximation. Functions that are continuous over long intervals were evaluated on top of two intervals, $(-10, 10)$ and $(-20, 20)$, whereas

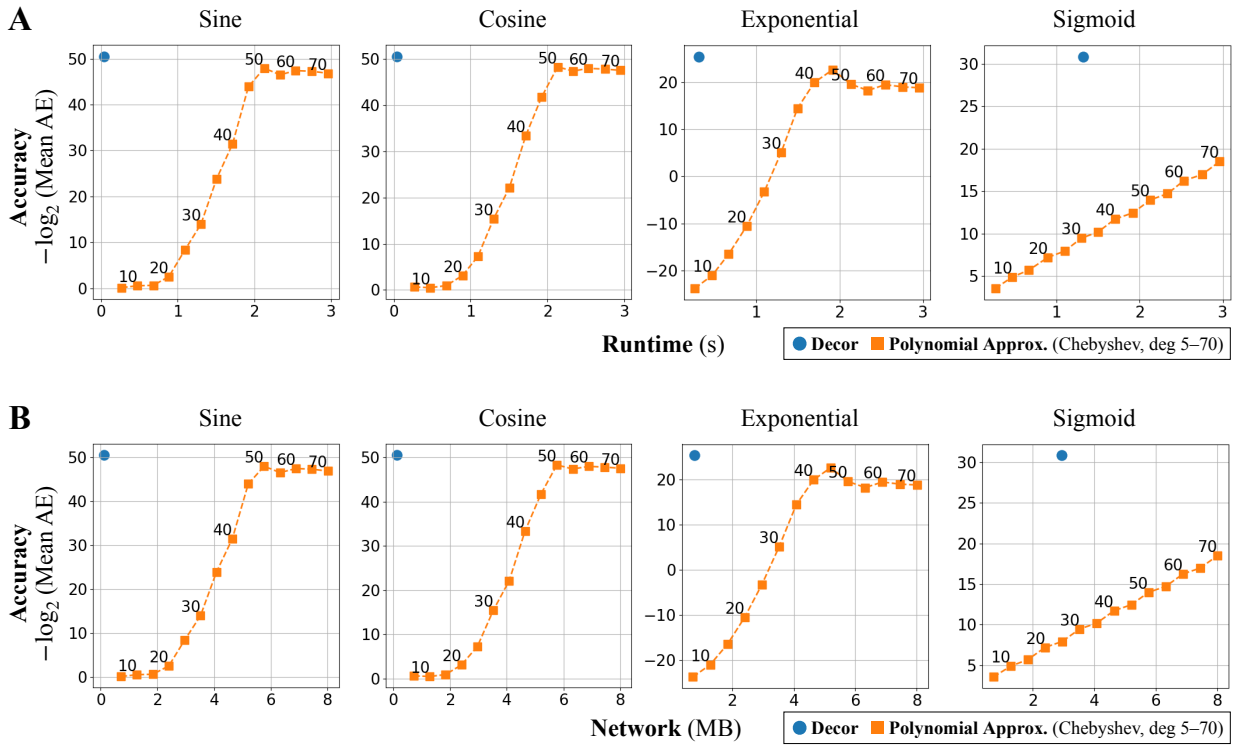

Figure 2: Accuracy-efficiency trade-offs of polynomial (Chebyshev) approximation compared with Decor protocols over the $(-20, 20)$ interval. Negative log of mean absolute error (AE; higher is better) is plotted against runtime (A) and communication bandwidth (B). Decor consistently achieves higher accuracy and efficiency, whereas polynomial approximation either fails to reach comparable precision or incurs substantial computational overhead at higher degrees.

functions with discontinuity, such as tangent and cotangent, were evaluated on smaller intervals—either $(-\pi/4, \pi/4)$ or $(\pi/4, 3\pi/4)$. We report the corresponding runtime and communication cost measurements in Table 3.

Across all functions, Decor consistently outperformed Chebyshev in accuracy while generally remaining more computationally efficient. With the exceptions of Decor-Sigmoid and Decor-Tanh, which were faster than degree-50 Chebyshev (1.3s vs. 2.1s) but slower than degree-20 (0.9s), all Decor protocols ran faster than degree-20 Chebyshev while providing substantially better accuracy. Specifically, relative to Chebyshev with degrees 50 and 20, respectively, Decor reduced the error of magnitude by up to $10^6\times$ and $10^{15}\times$, achieved up to $55\times$ and $22\times$ faster runtimes, and required up to $45\times$ and $19\times$ less communication.

Upon inspection of the trade-offs between accuracy and computational costs of Chebyshev across degrees between 5 and 70, we observed that the accuracy of polynomial approximation tends to plateau at high degrees, notably below the accuracy of Decor (Figure 2). For example, the mean absolute error in Chebyshev approximation of the exponential function is in the order of 10^{-6} for degrees between 50–70, in contrast to the 10^{-8} error scale of Decor. This limitation of Chebyshev is likely due to its greater sensitivity to numerical underflow in the fixed-point representation used by our MPC scheme. High-degree polynomials require circuits with greater multiplicative depth, making them more susceptible to precision loss.

While Decor remains highly efficient—even compared to low-degree polynomial approximations for many functions (e.g., sine, cosine, and exponential in Figure 2)—its functional decomposition can introduce additional cost when it requires a division. For functions such as sigmoid, this results in runtime and communication overheads comparable to those of a degree-30 Chebyshev approximation in our setting. On the other hand, Decor reduces numerical error by a factor of 10^4 for sigmoid, even relative to the highest-degree Chebyshev approximation we evaluated (degree 70), which incurs more than twice the runtime of Decor-Sigmoid.

Lastly, we evaluated Decor's polynomial evaluation procedure by applying it to the Chebyshev polynomials of all elementary functions and comparing its performance to our Chebyshev evaluation based on the Clenshaw algorithm [26]. Decor's polynomial evaluation was, on average, $3.5\times$ faster and required $1.8\times$ less communication (Table 3), while maintaining the same accuracy.

4.3. Fourier Approximation of Wave Functions

We evaluated Decor's Fourier-based function approximation (Decor-Fourier) on the square and sawtooth waves, defined as π-periodic functions over $(-3\pi, 3\pi)$, with the

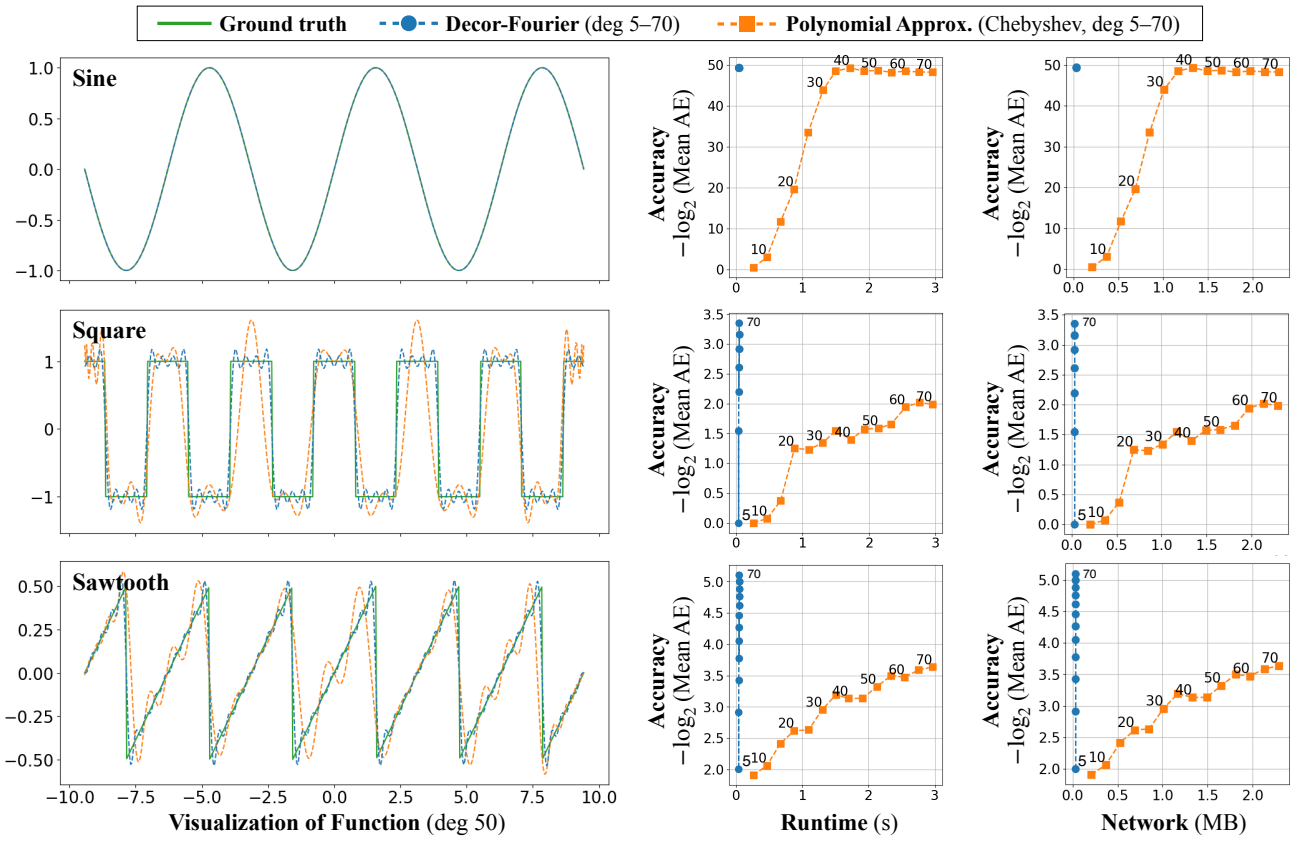

Figure 3: Decor's Fourier function approximation compared with Chebyshev polynomials on three waveforms: sine (top), square (middle), and sawtooth (bottom). Degree-50 approximations are shown (left). Accuracy (negative log of mean absolute error; higher is better) is evaluated against runtime (center) and communication cost (right) across varying degrees.

sine wave as an additional reference (Figure 3). Because Decor-Fourier uses a constant number of rounds (three) regardless of degree, it substantially improves efficiency over Chebyshev polynomials at any degree, while also achieving higher accuracy. In our experiments, Decor-Fourier outperformed Chebyshev, running 30× faster with 3.4× lower communication at degree 50, and 17× faster with 3.2× lower communication at degree 20 (Table 3). These results suggest that Fourier function approximation with Decor enable accurate approximation of periodic functions, whereas polynomial approximations generally fail to represent oscillatory behavior over long intervals.

4.4. Real-World Application Examples

We illustrate Decor's utility in two real-world applications that rely on accurate evaluations of complex, nonlinear functions: (1) training implicit neural representations (INRs) [10] for private medical images using sinusoidal activation functions, and (2) estimating statistical parameters in a genome-wide association study (GWAS) of lung cancer patients [9] based on logistic regression. In both settings, we show that Decor delivers accuracy beyond what was previously achievable with polynomial approximation.

4.4.1. Secure SIREN. INRs encode signals such as images, audio, and video as the weights of a multilayer perceptron (MLP), enabling compression, rasterization, restoration, and improved analysis. A prominent example is the Sinusoidal Representation Network (SIREN) [10], which models periodic structure using sinusoidal activation functions.

Using Decor's sine protocol, we securely trained a SIREN model on three privacy-sensitive medical image types (chest X-ray, retinal OCT, abdominal CT) [31]. Following prior work, the model uses a three-layer MLP with two hidden layers of 64 neurons (sine activations) and a linear output layer. Training was performed with MPC-based stochastic gradient descent with Nesterov momentum (learning rate 0.2, momentum 0.9). We trained a separate INR for each 64×64 image from MedMNIST [32] for 1,000 epochs. Decor's results closely matched plaintext training, with only a negligible loss gap of about 10^{-8} and no visible differences in gradients or reconstructed images (Figure 4).

In contrast, polynomial approximations for the sine activation—even over a wide interval $(-30, 30)$—failed to train accurately. Approximation errors accumulated over epochs, producing large deviations from the true activations and unstable gradients. This issue is expected to worsen for larger networks, which require broader approximation

	Approach (Degree)	Interval	Mean AE	Max AE
Sine / Cosine	Decor		$\mathbf{5.0 \cdot 10^{-16}}$	$\mathbf{2.1 \cdot 10^{-15}}$
	Chebyshev (50)	$(-10, 10)$	$2.1 \cdot 10^{-15}$	$1.0 \cdot 10^{-14}$
	Chebyshev (20)		$1.2 \cdot 10^{-06}$	$2.1 \cdot 10^{-06}$
	Decor		$\mathbf{6.4 \cdot 10^{-16}}$	$\mathbf{3.6 \cdot 10^{-15}}$
	Chebyshev (50)	$(-20, 20)$	$3.7 \cdot 10^{-15}$	$3.9 \cdot 10^{-14}$
	Chebyshev (20)		$1.7 \cdot 10^{-01}$	$4.1 \cdot 10^{-01}$
Tan	Decor		$\mathbf{8.4 \cdot 10^{-16}}$	$\mathbf{2.6 \cdot 10^{-15}}$
	Chebyshev (50)	$(-\frac{\pi}{4}, \frac{\pi}{4})$	$5.3 \cdot 10^{-15}$	$4.0 \cdot 10^{-14}$
	Chebyshev (20)		$1.7 \cdot 10^{-12}$	$3.3 \cdot 10^{-12}$
Cotan	Decor		$\mathbf{7.9 \cdot 10^{-16}}$	$\mathbf{2.7 \cdot 10^{-15}}$
	Chebyshev (50)	$(\frac{\pi}{4}, \frac{3\pi}{4})$	$1.2 \cdot 10^{-14}$	$4.0 \cdot 10^{-13}$
	Chebyshev (20)		$1.7 \cdot 10^{-12}$	$3.3 \cdot 10^{-12}$
Exponential	Decor		$\mathbf{8.8 \cdot 10^{-13}}$	$\mathbf{2.4 \cdot 10^{-11}}$
	Chebyshev (50)	$(-10, 10)$	$7.2 \cdot 10^{-11}$	$2.2 \cdot 10^{-09}$
	Chebyshev (20)		$8.9 \cdot 10^{-06}$	$2.1 \cdot 10^{-05}$
	Decor		$\mathbf{2.8 \cdot 10^{-08}}$	$\mathbf{1.0 \cdot 10^{-06}}$
	Chebyshev (50)	$(-20, 20)$	$1.2 \cdot 10^{-06}$	$2.2 \cdot 10^{-05}$
	Chebyshev (20)		$1.4 \cdot 10^{+03}$	$5.7 \cdot 10^{+03}$
Sigmoid	Decor		$\mathbf{1.0 \cdot 10^{-10}}$	$\mathbf{5.9 \cdot 10^{-09}}$
	Chebyshev (50)	$(-10, 10)$	$1.8 \cdot 10^{-08}$	$7.0 \cdot 10^{-08}$
	Chebyshev (20)		$3.4 \cdot 10^{-04}$	$1.2 \cdot 10^{-03}$
	Decor		$\mathbf{5.2 \cdot 10^{-10}}$	$\mathbf{1.4 \cdot 10^{-08}}$
	Chebyshev (50)	$(-20, 20)$	$6.1 \cdot 10^{-05}$	$4.2 \cdot 10^{-04}$
	Chebyshev (20)		$6.7 \cdot 10^{-03}$	$3.9 \cdot 10^{-02}$
C/Sinh	Decor		$\mathbf{5.3 \cdot 10^{-16}}$	$\mathbf{3.5 \cdot 10^{-15}}$
	Chebyshev (50)	$(\frac{\pi}{4}, \frac{3\pi}{4})$	$5.8 \cdot 10^{-14}$	$2.1 \cdot 10^{-12}$
	Chebyshev (20)		$6.1 \cdot 10^{-15}$	$3.4 \cdot 10^{-14}$
Tanh	Decor		$\mathbf{1.4 \cdot 10^{-10}}$	$\mathbf{6.2 \cdot 10^{-09}}$
	Chebyshev (50)	$(-10, 10)$	$8.0 \cdot 10^{-05}$	$5.2 \cdot 10^{-04}$
	Chebyshev (20)		$1.1 \cdot 10^{-02}$	$6.6 \cdot 10^{-13}$

Table 2: Accuracy comparison for elementary nonlinear functions. Best performance in bold. AE: absolute error.

Approach (Degree)	Runtime (s)	Network (Dealer; KB)	Network (Online; KB)	Rounds
Decor-Sin/Cos	0.040	64	64	3
Decor-Tan/Cot	0.681	1,049	2,226	32
Decor-Exp	0.314	296	464	15
Decor-Sigmoid	1.320	1,153	1,794	64
Decor-Sinh/Cosh	0.357	376	528	17
Decor-Tanh	1.338	1,169	1,826	65
Chebyshev (50)	2.133	2,466	3,298	103
Chebyshev (20)	0.887	1,025	1,376	43
Decor-Fourier (50)	0.070	1,632	64	3
Decor-Fourier (20)	0.052	672	64	3
Decor-Poly (20)	0.385	648	496	16

Table 3: Runtime and communication costs of elementary nonlinear functions and general approximation methods. Measurements averaged across functions for methods with constant cost for fixed degree. Decor-Poly: Chebyshev with Decor polynomial evaluation.

intervals and thus higher-degree polynomials, which remain inaccurate for periodic functions over long domains.

We partially mitigated this problem with a modular-reduction heuristic: the input is scaled so that the sine period aligns with a power of two in the secret-sharing domain, and parties locally reduce their shares modulo the period. This effectively maps inputs to a smaller interval, enabling more accurate Chebyshev approximation (Chebyshev* in Figure 4 and Table 5). Although this prevented catastrophic

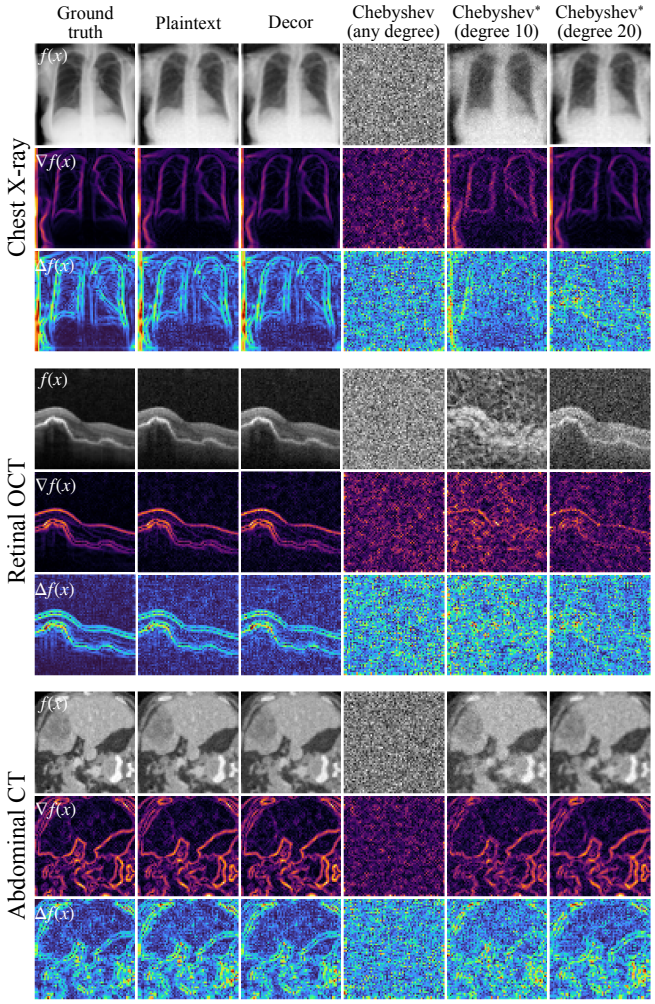

Figure 4: Image representations (f), gradients (∇f), and divergences (Δf) of securely trained SIREN on medical images. Polynomial approximations fail to capture high-frequency oscillations over long intervals, leading to failed model learning regardless of degree (as exemplified by data shown). Asterisk indicates a heuristic variant of Chebyshev addressing interval issues described in main text.

Approach	Runtime (s)	Network (Dealer; GB)	Network (Online; GB)	Rounds
Plaintext	645	-	-	-
Decor	13,395	391	478	104,000
Cheb. (20)	52,208	2,472	3,228	267,000
Cheb. (10)	35,504	1,465	1,884	188,000

Table 4: SIREN training evaluation with respect to runtime and communication. Averaged over three images in Figure 4.

divergence, it still provided lower accuracy than Decor and incurred higher computational cost. Decor, by contrast, operates accurately without restricting the input range and avoids the need for delicate tuning, which is generally not feasible in practice where the private inputs cannot be inspected.

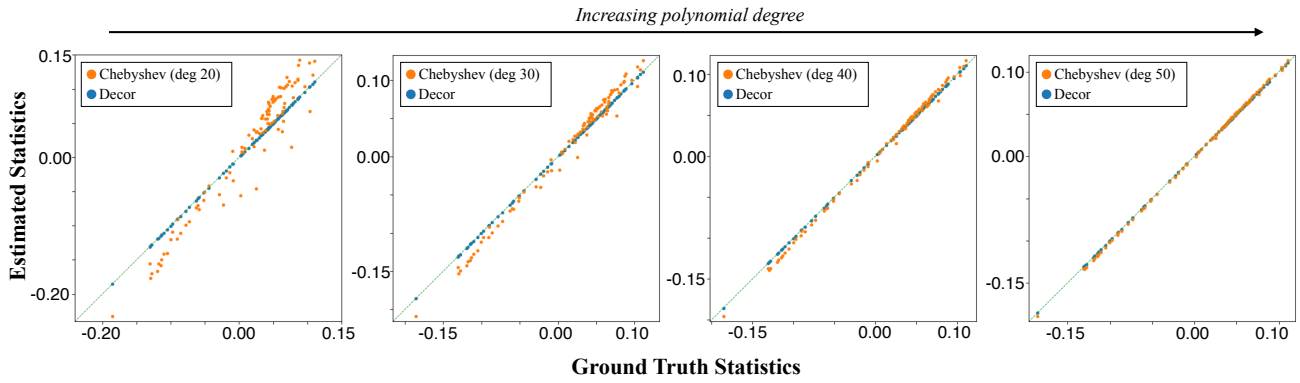

Figure 5: Accuracy of GWAS effect sizes securely computed with logistic regression. Dots show individual genetic variants tested. Closer to the diagonal means higher accuracy. Varying degrees of Chebyshev plotted separately for comparison.

	Approach	Chest X-ray	Retinal OCT	Abdominal CT
Training Loss	Plaintext	$1.0 \cdot 10^{-07}$	$1.2 \cdot 10^{-07}$	$2.3 \cdot 10^{-07}$
	Decor	$\mathbf{9.1 \cdot 10^{-08}}$	$\mathbf{1.4 \cdot 10^{-07}}$	$\mathbf{2.4 \cdot 10^{-07}}$
	Cheb. (20)	$3.3 \cdot 10^{-05}$	$6.0 \cdot 10^{-06}$	$1.8 \cdot 10^{-05}$
	Cheb. (10)	$3.3 \cdot 10^{-05}$	$6.0 \cdot 10^{-06}$	$1.8 \cdot 10^{-05}$
	Cheb.* (20)	$2.1 \cdot 10^{-07}$	$2.0 \cdot 10^{-06}$	$8.2 \cdot 10^{-07}$
	Cheb.* (10)	$1.0 \cdot 10^{-06}$	$3.5 \cdot 10^{-06}$	$1.4 \cdot 10^{-06}$

Table 5: SIREN training accuracy in terms of training losses (smaller is better). Averaged over three images in Figure 4. Best performance in bold. Asterisk indicates a heuristic variant described in main text.

Approach	Mean AE	Runtime (s)	Network (Dealer)	Network (Online)	Rounds
Plaintext	-	0.48	-	-	-
Decor	$\mathbf{3.1 \cdot 10^{-07}}$	84.18	1.09	1.66	3,401
Cheb. (50)	$1.5 \cdot 10^{-03}$	141.00	2.27	3.01	5,351
Cheb. (40)	$3.9 \cdot 10^{-03}$	107.86	1.83	2.44	4,351
Cheb. (30)	$9.7 \cdot 10^{-03}$	85.91	1.40	1.86	3,351
Cheb. (20)	$2.4 \cdot 10^{-02}$	**61.01**	**0.97**	**1.28**	**2,351**

Table 6: Accuracy, runtime, and network costs for GWAS with logistic regression. Averaged across a hundred genomic positions over 9,000 lung cancer patients. Best performance in bold. Network bandwidth in GBs. AE: absolute error.

4.4.2. Secure Genome-Wide Association Study. GWAS is a widely used statistical method in medical genetics for identifying genetic variants associated with a disease or trait. It regresses the trait on each variant while adjusting for covariates such as demographic and ancestry variables, producing an effect-size estimate reflecting the variant's contribution to the trait. Accurate estimation of effect sizes is crucial for understanding disease mechanisms and for building predictive models used in clinical decision-making.

Following prior MPC-based GWAS work [5], we evaluated the accuracy of securely estimating effect sizes on a private, secret-shared genomic dataset. We used a subset of a real lung cancer cohort from the NIH dbGaP repository (phs000716.v1.p1), including 9,000 individuals, 100 genetic variants, and 10 covariates (age, sex, and ancestry components), together with a binary indicator of cancer status. We implemented MPC-based logistic regression using Decor's sigmoid protocol and trained a separate model for each variant using batched gradient descent for 50 epochs.

Figure 5 shows the agreement between the secure estimates and the plaintext ground truth. Decor achieved accuracy comparable to plaintext GWAS, with mean squared error on the order of 10^{-7}. In contrast, the same algorithm with Chebyshev approximations of degrees 20–50 for sigmoid yielded errors between 10^{-3} and 10^{-2}. For degrees 30 and above, Chebyshev evaluation was up to $1.7\times$ slower and consumed $3\times$ more communication than Decor (Table 6).

5. Related Work

Many secret-sharing and mixed-protocol-based MPC frameworks approximate nonlinear or transcendental operations (e.g., exponential, sigmoid, softmax) using polynomials (e.g., Chebyshev), piecewise-linear (PWL), or lookup-table (LUT) functions to enable efficient evaluation using arithmetic circuits. Examples of general-purpose frameworks include SecureML [33], ABY-style mixed-protocol systems [34], and secure math libraries such as LLAMA [35]. Other relevant recent works include Squirrel [36], which introduces an optimized two-party protocol for sigmoid computation by combining PWL and Fourier series approximations, as well as a wavelet-compression-based LUT function approximation method [37]. Although many of these techniques have been adopted by MPC systems for secure machine learning (e.g., CrypTen [38], CrypTFlow2 [39], Cheetah [40], SiRNN [41], SecureNN [42], Falcon [43]), they suffer from several limitations: communication and round complexity increase with higher-degree polynomials, a greater number of piecewise segments, and larger lookup tables; achieving tight accuracy often requires high-degree approximations, leading to increased multiplicative depth; polynomial and PWL approximations can be numerically unstable in deep models; and WAN settings further exacerbate the communication overhead. To illustrate the benefits of Decor's decomposition approach, we compare its sigmoid protocol with a recent PWL-based protocol from

Squirrel [36] in Appendix B, showing that our approach is both more accurate and more efficient.

An alternative approach is to reformulate specific functions. For example, ΠQSMax [44] expresses softmax as an ordinary differential equation (ODE) solved via Euler steps, and ΠLSig evaluates sigmoid through Fourier series identities. These methods avoid heavy comparisons and offer low-round online phases, but they are function-specific and do not generalize to other functions. Furthermore, ΠQSMax requires iterative rounds that trade accuracy for cost, ΠLSig focuses accuracy near zero and still approximates the target nonlinearity.

A recent work adopts a more general approach, namely the MW-based protocols [45]. This method leverages efficient computation of the most significant bit and the wraparound of secret shares (MW) [46], using oblivious transfer, to evaluate a class of nonlinear functions. While it supports trigonometric and exponential functions, similar to Decor, the MW approach has notable limitations relative to Decor: it excludes other function classes supported by Decor and is restricted to a two-party setting, although it does not require a trusted dealer. In Appendix B, we provide additional empirical comparisons with MW-based protocols for functions supported by both approaches, demonstrating that Decor improves both efficiency and accuracy.

A complementary line of work shifts the protocol family entirely, reducing nonlinear layers to garbled circuits (GC) and oblivious transfer (OT), as in ABY2.0 [34], or to function secret sharing (FSS), as in AriaNN [47] and FastSecNet [48]. These approaches can achieve low-round evaluation but often incur high communication costs, resulting in either additional approximation or substantial online bandwidth requirements, particularly in WAN settings [44]. Existing FSS constructions typically rely on distributed comparison functions (DCF) and primarily support gates composed of comparisons, multiplications, and bitwise operations. In contrast, Decor demonstrates that preprocessing over random masks enables accurate evaluation of nonlinear real-valued functions (e.g., sine/cosine, exponential, and sigmoid), extending beyond the capabilities of existing FSS techniques.

6. Conclusions and Future Work

We introduced the Decor framework for evaluating complex functions in MPC by decomposing them into forms that can be computed more efficiently with the assistance of a trusted dealer. We showed how this idea yields efficient protocols for a wide range of functions. Our work advances high-precision MPC and strengthens its utility in application domains involving critical decisions, such as medical analysis, where accuracy is paramount.

While relying on a trusted dealer limits Decor to settings where parties can access a non-colluding auxiliary entity, our framework remains broadly relevant to the MPC community. This is particularly true in domains where parties manage large volumes of sensitive data and are therefore willing to accept stronger security assumptions in exchange for significant performance gains, as demonstrated in prior work on practical MPC applications in biomedical settings [37], [49], [50], [51], [52]. Our contribution advances the state of nonlinear function evaluation in such contexts. A meaningful direction for future work is to remove the need for a trusted dealer by leveraging alternative approaches to randomness generation, such as oblivious transfer [53].

The classes of functions we have identified as efficiently computable by Decor are likely not exhaustive. We plan to further investigate whether additional functions satisfy the required structure in Eqs. 1 and 3. In particular, we aim to explore tools such as Bitween [54], which facilitate the discovery of recursive functional equivalences over arbitrary inputs. Another promising direction is extending Decor to complex-valued computation, as it already enables efficient evaluation of the complex exponential via $e^{i[[x]]_f} = \cos[[x]]_f + i\sin[[x]]_f$. In addition, although not developed in this work, malicious security guarantees could be obtained by replacing the MPC primitives used to evaluate Decor-transformed functions (Eqs. 2 and 4) with actively secure variants, e.g., based on authenticated triples [3]. Finally, we note that our methods for division and comparison, while optimized, would continue to benefit from ongoing advances in MPC. We plan to incorporate such improvements into future versions of Decor.

7. Financial Disclosure

The authors declare no financial competing interest. T.A. and R.P. were partially funded by the NSF awards CCF-2219995, CNS-2245344, and CCF-2318974. H.S. and H.C. were partially funded by the NSF award ITE-2452612 and NIH award DP5 OD029574. K.S was partially funded by the Yale College First-Year Summer Research Fellowship in the Sciences & Engineering.

8. Ethics Considerations

Our real-world experiments utilized publicly available human-subject data from MedMNIST [32] and the NIH dbGaP repository (phs000716.v1.p1) in compliance with the respective data use agreements.

References

[1] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[2] I. Damgard *et al.*, "New primitives for actively-secure MPC over rings with applications to private machine learning," *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2019-May, pp. 1102–1120, 2019.

[3] R. Cramer *et al.*, "SPDZ$^{2^k}$: Efficient MPC mod 2^k for dishonest majority," in *Advances in Cryptology – CRYPTO 2018*, ser. Lecture Notes in Computer Science, vol. 10992. Springer, 2018, pp. 769–798.

[4] E. A. Abbe *et al.*, "Privacy-preserving methods for sharing financial risk exposures," *American Economic Review*, vol. 102, no. 3, pp. 65–70, 2012.

[5] H. Cho *et al.*, "Secure genome-wide association analysis using multiparty computation," *Nature Biotechnology*, vol. 36, no. 6, pp. 547–552, 2018.

[6] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[7] C. Boura *et al.*, "High-precision privacy-preserving real-valued function evaluation," in *Financial Cryptography and Data Security: FC 2018*. Berlin, Heidelberg: Springer-Verlag, 2018, p. 183–202. [Online]. Available: https://doi.org/10.1007/978-3-662-58387-6_10

[8] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology – CRYPTO 1991*. Berlin, Heidelberg: Springer-Verlag, 1991, p. 420–432.

[9] H. Cho *et al.*, "Secure and federated genome-wide association studies for biobank-scale datasets," *Nature Genetics*, pp. 1–6, 2025.

[10] V. Sitzmann *et al.*, "Implicit neural representations with periodic activation functions," in *arXiv*, 2020.

[11] I. Damgård *et al.*, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology – CRYPTO 2012*, ser. Lecture Notes in Computer Science, vol. 7417. Springer, 2012, pp. 643–662.

[12] D. Bogdanov *et al.*, "Sharemind: A framework for fast privacy-preserving computations," in *Computer Security – ESORICS 2008*, ser. Lecture Notes in Computer Science, vol. 5283. Springer, 2008, pp. 192–206.

[13] P. Mohassel and P. Rindal, "ABY3: A mixed protocol framework for machine learning," in *ACM SIGSAC CCS '18'*. ACM, 2018, pp. 35–52.

[14] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *ACM SIGSAC*, 2020, pp. 1575–1590.

[15] I. Damgård *et al.*, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662.

[16] H. Corrigan-Gibbs and D. Boneh, "Prio: Private, robust, and scalable computation of aggregate statistics," in *USENIX NSDI '17*, 2017, pp. 259–282.

[17] A. R. Ghavamipour *et al.*, "Privacy-preserving logistic regression with secret sharing," *BMC Medical Informatics and Decision Making*, vol. 22, no. 1, pp. 1–11, 2022, assumes a trusted initializer that predistributes Beaver triples to the computing parties.

[18] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[19] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2010, pp. 35–50.

[20] I. Damgård *et al.*, "Unconditionally secure constant-rounds multiparty computation for equality, comparison, bits and exponentiation," in *Theory of Cryptography*. Berlin, Heidelberg: Springer, 2006, pp. 285–304.

[21] T. Toft, "Primitives and applications for multi-party computation," Phd Thesis, Aarhus University, 2007.

[22] D. Demmler *et al.*, "ABY-A framework for efficient mixed-protocol secure two-party computation." in *Network and Distributed System Security (NDSS) Symposium*, 2015.

[23] E. M. Songhori *et al.*, "Tinygarble: Highly compressed and scalable sequential garbled circuits," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 411–428.

[24] E. Makri *et al.*, "Rabbit: Efficient comparison for secure multiparty computation," in *Financial Cryptography and Data Security*. Springer, 2021, pp. 249–270.

[25] T. J. Rivlin, *Chebyshev Polynomials*, ser. Pure and Applied Mathematics. New York, NY: John Wiley & Sons, 1990.

[26] C. W. Clenshaw, "A note on the summation of chebyshev series," *Mathematics of Computation*, vol. 9, pp. 118–120, 1955. [Online]. Available: https://api.semanticscholar.org/CorpusID:121250561

[27] E. Saurez *et al.*, "OneEdge: An efficient control plane for geo-distributed infrastructures," in *ACM Symposium on Cloud Computing (SoCC)*, 2021.

[28] D. Froelicher *et al.*, "Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption," *Nature communications*, vol. 12, no. 1, pp. 1–10, 2021.

[29] H. Smajlović *et al.*, "Shechi: A secure distributed computation compiler based on multiparty homomorphic encryption," in *USENIX Security '25*, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:278856705

[30] A. Shajii *et al.*, "Codon: A compiler for high-performance pythonic applications and DSLs," in *ACM SIGPLAN CC*, 2023, p. 191–202.

[31] J. Wolleb *et al.*, "Vidfuncta: Towards generalizable neural representations for ultrasound videos," in *Simplifying Medical Ultrasound*. Cham: Springer Nature Switzerland, 2026, pp. 109–119.

[32] J. Yang *et al.*, "MedMNIST v2 - a large-scale lightweight benchmark for 2D and 3D biomedical image classification," *Scientific Data*, vol. 10, no. 1, 2023. [Online]. Available: http://dx.doi.org/10.1038/s41597-022-01721-8

[33] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 19–38.

[34] A. Patra *et al.*, "ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation," in *USENIX Security '21*, 2021, pp. 2165–2182.

[35] K. Gupta *et al.*, "LLAMA: A low latency math library for secure inference," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, no. 4, pp. 274–294, 2022.

[36] W.-j. Lu *et al.*, "Squirrel: A scalable secure two-party computation framework for training gradient boosting decision tree," in *USENIX Security '23*. USENIX Association, 2023, pp. 6435–6451. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/lu

[37] J. Reis *et al.*, "Wave hello to privacy: Efficient mixed-mode mpc using wavelet transforms," *Proceedings on Privacy Enhancing Technologies*, 2025.

[38] B. Knott *et al.*, "CrypTen: Secure machine learning in pytorch," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[39] D. Rathee *et al.*, "CrypTFlow2: Practical 2-party secure inference," in *ACM SIGSAC CCS '18'*. New York, NY, USA: Association for Computing Machinery, 2020, p. 325–342.

[40] Z. Huang *et al.*, "Cheetah: Lean and fast secure two-party deep neural network inference," in *USENIX Security '22*. Boston, MA: USENIX Association, 2022, pp. 809–826. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/huang-zhicong

[41] D. Rathee *et al.*, "SIRNN: A math library for secure RNN inference," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1003–1020.

[42] S. Wagh *et al.*, "SecureNN: 3-Party Secure Computation for Neural Network Training," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 26–49, 2019.

[43] S. Wagh *et al.*, "FALCON: Honest-Majority Maliciously Secure Framework for Private Deep Learning," in *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 1, 2021, pp. 188–208.

[44] Y. Zheng *et al.*, "Secure softmax/sigmoid for machine-learning computation," in *Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2023, p. 535–548. [Online]. Available: https://doi.org/10.1145/3627106.3627175

[45] H. Guo *et al.*, "Efficient and high-accuracy secure two-party protocols for a class of functions with real-number inputs," arXiv:2509.01178, 2025.

[46] H. Guo *et al.*, "Improved secure two-party computation from a geometric perspective," in *USENIX Security '25'*, Seattle, USA, August 2025, pp. 4957–4974.

[47] T. Ryffel, P. Tholoniat, D. Pointcheval, and F. R. Bach, "AriaNN: Low-interaction privacy-preserving deep learning via function secret sharing," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, pp. 291–316, 2020.

[48] M. Hao *et al.*, "Fastsecnet: An efficient cryptographic framework for private neural network inference," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 2569–2582, 2023.

[49] A. N. Kho *et al.*, "Design and implementation of a privacy preserving electronic health record linkage tool in Chicago," *Journal of the American Medical Informatics Association*, vol. 22, no. 5, pp. 1072–1080, 06 2015. [Online]. Available: https://doi.org/10.1093/jamia/ocv038

[50] H. Cho *et al.*, "Secure and federated genome-wide association studies for biobank-scale datasets," *Nature Genetics*, vol. 57, no. 4, pp. 809–814, 2025.

[51] B. Hie *et al.*, "Realizing private and practical pharmacological collaboration," *Science*, vol. 362, no. 6412, pp. 347–350, Oct. 2018.

[52] H. Smajlović *et al.*, "Sequre: a high-performance framework for secure multiparty computation enables biomedical data sharing," *Genome Biology*, vol. 24, no. 1, p. 5, 2023. [Online]. Available: https://doi.org/10.1186/s13059-022-02841-5

[53] M. Keller *et al.*, "Mascot: Faster malicious arithmetic secure computation with oblivious transfer," in *ACM SIGSAC CCS '16'*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 830–842. [Online]. Available: https://doi.org/10.1145/2976749.2978357

[54] F. Erata *et al.*, "Learning randomized reductions," arXiv:2412.18134, 2025.

[55] M. Dahl *et al.*, "On secure two-party integer division," in *Financial Cryptography and Data Security*, A. D. Keromytis, Ed. Berlin, Heidelberg: Springer, 2012, pp. 164–178.

Appendix A.
Additional Tables and Figures

Decor protocol steps	Observed output	Actual outputs/values: Without wraparound	Actual outputs/values: With wraparound
Reveal$([x-r])$	$x'-r$	$x-r$	$(x+l)-r$
Evaluate Γ_s with $x'-r$	$[f(x')]$	$[f(x)]$	$[f(x+l)]$
Evaluate Γ_s with $x'-r-l$	$[f(x'-l)]$	$[f(x-l)]$	$[f(x)]$
Comp$([x],[r])$	$[x<r]$	0	1
Compute $[f(x')]+[x<r]\cdot([f(x'-l)]-[f(x')])$		$[f(x)]$	$[f(x)]$

Table 7: Summary of Decor's secure wraparound correction, as described in Section 3.3.

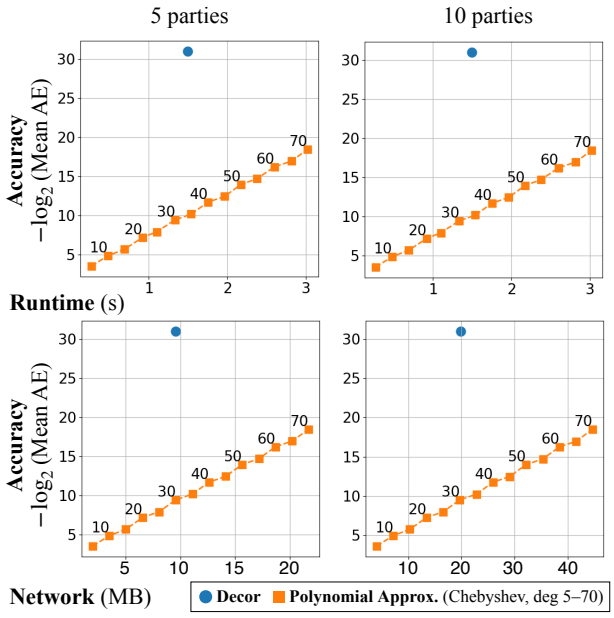

Figure 6: Scaling of runtime and communication with respect to the number of parties. Decor's total network communication grows linearly in the number of parties as expected, mirroring the behavior of Chebyshev. Note that the network measurements doubled going from 5 to 10 parties for both methods. The runtime stays nearly identical as each party mostly performs the same amount of computation unaffected by the number of parties with minimal overhead from the increased communication.

	Approach (Degree)	Interval	Mean AE	Max AE
Square	Decor-Fourier (50)	$(-3\pi, 3\pi)$	**0.132**	1.059
	Decor-Fourier (20)		0.218	**1.018**
	Chebyshev (50)		0.332	1.938
	Chebyshev (20)		0.420	1.500
Sawtooth	Decor-Fourier (50)	$(-3\pi, 3\pi)$	**0.036**	0.506
	Decor-Fourier (20)		0.072	**0.501**
	Chebyshev (50)		0.100	0.961
	Chebyshev (20)		0.162	0.834

Table 8: Accuracy of approximating wave functions (square and sawtooth) of period π, related to Figure 3. Best performance highlighted in bold. AE: absolute error.

Appendix B.
Comparison with Additional Related Methods

Beyond standard polynomial approximation techniques (e.g., Chebyshev and Fourier), piecewise linear (PWL) methods are also widely used to approximate arbitrary functions.

A recent work [45] proposes new methods for evaluating a class of functions—covering a strict subset of those supported by Decor—including trigonometric and exponential functions. These methods operate in two-party settings without a trusted dealer and rely on specialized MPC protocols for efficiently computing the sum of the most significant bit and the wraparound of secret shares (MW) [46].

	Approach	Runtime (ms)	Network (Online; KB)	MAE	Frac. bits	Ring size
Sin/Cos	Decor	4	64	$5.0 \cdot 10^{-16}$	64	256
	MW	51	778	$1.2 \cdot 10^{-04}$	12	21
	Decor	0.8	32	$1.4 \cdot 10^{-05}$	16	128
	MW	51	778	$2.1 \cdot 10^{-05}$	16	63
Exp	Decor	7	464	$2.8 \cdot 10^{-13}$	64	256
	MW	27	607	$4.6 \cdot 10^{-05}$	12	37
	Decor	4	200	$5.2 \cdot 10^{-05}$	16	128
	MW	31	869	$4.6 \cdot 10^{-05}$	16	64
Sigmoid	Decor	19	1,794	$1.0 \cdot 10^{-10}$	64	256
	Squirrel	183	6,955	$5.0 \cdot 10^{-01}$	40	128
	Decor	16	850	$6.7 \cdot 10^{-08}$	32	128
	Squirrel	152	6,951	$1.3 \cdot 10^{-03}$	25	128

Table 9: Comparison of runtime, communication, and accuracy for elementary nonlinear functions between Decor and recent relevant techniques (MW [45] and Squirrel [36]), as discussed in Appendix B. MAE: mean absolute error.

Algorithm 4 $\mathsf{Sign}([[x]]_{\mathtt{f}})$ $(1 + \log_2 \mathtt{k}$ rnd)

Require: Secret shared $[[x]]_{\mathtt{f}}$
Ensure: Output $[1]$ if $x > 0$, $[0]$ otherwise
1: *Dealer:* Sample $r \in \mathbb{Z}_{2^{\mathtt{k}}}$
2: *Dealer:* Decompose r into bits ($r_{\text{bits}} \in \mathbb{Z}_2^{\mathtt{k}}$)
3: *Dealer:* Secret share $[[r]]_{\mathtt{f}}$ and $[r_{\text{bits}}] \in \mathbb{Z}_2^{\mathtt{k}}$
4: $a \leftarrow \mathsf{Reveal}([[x]]_{\mathtt{f}} + [[r]]_{\mathtt{f}} + \mathcal{T}_{\mathtt{f}}(1))$ ▷ 1 rnd
5: Decompose a into bits ($a_{\text{bits}} \in \mathbb{Z}_2^{\mathtt{k}}$)
6: $[r_{\text{neg}}] \leftarrow [r_{\text{bits}}] + \mathbf{1} \in \mathbb{Z}_2^{\mathtt{k}}$ ▷ Flip bits to get $-(r+1)$
7: $[m] \leftarrow \texttt{Carry-Out}(a_{\text{bits}}, [r_{\text{neg}}])$ ▷ $\log_2 \mathtt{k}$ rnd
8: **return** $[m] + 1 \in \mathbb{Z}_2$

We compare Decor against both MW-based protocols and an optimized PWL-based sigmoid implementation from Squirrel [36], which is likewise restricted to two-party settings. The results are summarized in Table 9. Our evaluation considers only the functions supported by each method, across different bit-width configurations, and excludes network latency. The evaluation settings reflect various constraints of the software accompanying the original publications. We evaluated the MW-based protocols for sine and exponential functions over the interval $(-0.785, 0.785)$, with at most 16 fractional bits and ring sizes up to 64 bits. Similarly, we used the Squirrel implementation with up to 40 fractional bits, beyond which we observed a significant degradation in accuracy.

Despite differences in bid-width configurations, Decor clearly outperformed the alternatives, achieving up to $10^{12}\times$ lower mean absolute error while maintaining up to $10\times$ faster runtime and up to $11\times$ lower communication. Although Decor requires a trusted dealer, it is more general in supporting multi-party settings, whereas the compared approaches are limited to two-party scenarios.

Appendix C. Secure Comparison and Division

To implement secure comparison, we adapted the secure sign check algorithm based on the bit-decomposition and prefix operations from [20], [21] using Boolean secret shares. The algorithm accepts two secret-shared numbers, $[[x]]_{\mathtt{f}}, [[y]]_{\mathtt{f}} \in \mathbb{Z}_{2^{\mathtt{k}}}$, and computes a secret share of their comparison $[x < y] \in \mathbb{Z}_{2^{\mathtt{k}}}$ by computing the secure sign check of their difference (i.e. $[(y - x) > 0] \in \mathbb{Z}_{2^{\mathtt{k}}}$). The sign check algorithm accepts the secret shared $[[x]]_{\mathtt{f}} \in \mathbb{Z}_{2^{\mathtt{k}}}$ as input and returns a Boolean secret share of $[x > 0] \in \mathbb{Z}_2$, representing either $[1]$ if $x > 0$ or $[0]$ otherwise. Finally, the output of the sign check is converted back to the $\mathbb{Z}_{2^{\mathtt{k}}}$ ring for subsequent downstream computations. The secure comparison algorithm runs in $2 + \log_2 \mathtt{k}$ rounds, mainly due to the sign check algorithm, whose complexity is $1 + \log_2 \mathtt{k}$. The sign check leverages a secure computation of the most significant carry-over bit ($\texttt{Carry-Out}(\cdot)$) of logarithmic complexity based on prefix operations from [20], [21].

Algorithm 5 $\mathsf{BoolRingSwitch}([x], \mathtt{k})$ (1 rnd)

Require: Secret shared in Boolean domain $[x] \in \mathbb{Z}_2$
Ensure: Output $[x]$ in $\mathbb{Z}_{2^{\mathtt{k}}}$
1: *Dealer:* Sample $r \in \{0, 1\}$
2: *Dealer:* Secret share $[r] \in \mathbb{Z}_{2^{\mathtt{k}}}$
3: *Dealer:* Boolean secret share $[r]_b \in \mathbb{Z}_2$
4: $a \leftarrow \mathsf{Reveal}([x] + [r]_b)$ ▷ 1 rnd
5: $[m] \leftarrow [r] - (2a \cdot [r]) \in \mathbb{Z}_{2^{\mathtt{k}}}$
6: **return** $[m] + a$

Algorithm 6 $\mathsf{Comp}([[x]]_{\mathtt{f}}, [[y]]_{\mathtt{f}})$ $(2 + \log_2 \mathtt{k}$ rnd)

Require: Secret shared $[[x]]_{\mathtt{f}}$ and $[[y]]_{\mathtt{f}}$
Ensure: Output $[1]$ if $x < y$, $[0]$ otherwise
1: $[[y - x]]_{\mathtt{f}} \leftarrow [[y]]_{\mathtt{f}} - [[x]]_{\mathtt{f}}$
2: $[m] \leftarrow \mathsf{Sign}([[y - x]]_{\mathtt{f}}) \in \mathbb{Z}_2$ ▷ $1 + \log_2 \mathtt{k}$ rnd
3: **return** $\mathsf{BoolRingSwitch}([m], \mathtt{k}) \in \mathbb{Z}_{2^{\mathtt{k}}}$ ▷ 1 rnd

Algorithm 7 $\mathsf{Div}_{\mathtt{f}}([[x]]_{\mathtt{f}}, [[y]]_{\mathtt{f}}, \delta)$ $(3 \log_2 \mathtt{f} + 4\delta + 7$ rnd)

Require: Secret shared $[[x]]_{\mathtt{f}}, [[y]]_{\mathtt{f}}$ and iterations count n
Ensure: Outputs $\left[\left[\frac{x}{y}\right]\right]_{\mathtt{f}}$
1: $[s] \leftarrow \mathsf{Normalizer}_{\mathtt{f}}([[y]]_{\mathtt{f}}) \in \mathbb{Z}_{2^{\mathtt{k}}}$ ▷ $5 + 3\log_2 \mathtt{f}$ rnd
2: $[[y_s]]_{\mathtt{f}} \leftarrow [[y]]_{\mathtt{f}} \cdot [s]$ ▷ $(y_s \in [0.25, 1])$ 1 rnd
3: $[[e]]_{\mathtt{f}} \leftarrow \mathcal{T}_{\mathtt{f}}(2) - [[y_s]]_{\mathtt{f}}$
4: **for** $i = 1$ to δ **do**
5: $\quad [[t]]_{\mathtt{f}} \leftarrow [[y_s]]_{\mathtt{f}} \cdot [[e]]_{\mathtt{f}}$ ▷ 2 rnd
6: $\quad [[e]]_{\mathtt{f}} \leftarrow [[e]]_{\mathtt{f}} \cdot (\mathcal{T}_{\mathtt{f}}(2) - [[t]]_{\mathtt{f}})$ ▷ 2 rnd
7: **end for**
8: **return** $[[e]]_{\mathtt{f}} \cdot [s]$ ▷ 1 rnd

To compute division, we used the iterative Newton-Raphson method for approximating the multiplicative inverse $(1/y)$. This method requires an initial estimate y_0 that satisfies $0 < y_0 < 2/y$, for any y. To ensure this, we adapted the secure normalizer of $5 + 3\log_2 \mathtt{f}$ rounds complexity from prior work [55], which scales the input $[[y]]_{\mathtt{f}}$ to $[[\tilde{y}]]_{\mathtt{f}} = [[y]]_{\mathtt{f}} \cdot [s] \in [0.25, 1]$, where s is the scaling factor. In other words, the division $[[x]]_{\mathtt{f}} \,/\, [[y]]_{\mathtt{f}}$ is reduced to the normalized equivalent $([[x]]_{\mathtt{f}} \cdot [s])/([[y]]_{\mathtt{f}} \cdot [s])$, such that $[[y]]_{\mathtt{f}} \cdot [s] \in [0.25, 1]$. The division requires $3\log_2 \mathtt{f} + 4\delta + 7$

interactive rounds, where `f` is the number of fixed-point fractional bits and δ is the number of iterations of the Newton-Rhapson algorithm.

Appendix D. Proof of Exponential-Based Decomposition

Claim. *Let $\alpha, \beta, \gamma, \delta, \lambda \in \mathbb{R}$ and define $f_\Delta : \mathbb{R} \to \mathbb{R}$ by*

$$f_\Delta(x) = \frac{\alpha e^{\lambda x} + \beta}{\gamma e^{\lambda x} + \delta},$$

such that the denominator does not vanish. Then for all x, y,

$$f_\Delta(x+y) = \frac{A f_\Delta(x) f_\Delta(y) + B f_\Delta(x) + C f_\Delta(y) + D}{E f_\Delta(x) f_\Delta(y) + F f_\Delta(x) + G f_\Delta(y) + H},$$

where

$$\begin{aligned} &A = \alpha\delta^2 + \beta\gamma^2, && B = C = -\alpha\beta(\delta+\gamma),\\ &D = \alpha\beta(\alpha+\beta), && E = \gamma\delta(\delta+\gamma),\\ &F = G = -\gamma\delta(\alpha+\beta), && H = \gamma\beta^2 + \alpha^2\delta. \end{aligned}$$

Proof. Write

$$f_\Delta(x) = \frac{\alpha e^{\lambda x} + \beta}{\gamma e^{\lambda x} + \delta}, \qquad f_\Delta(y) = \frac{\alpha e^{\lambda y} + \beta}{\gamma e^{\lambda y} + \delta},$$

and set

$$z := f_\Delta(x), \qquad w := f_\Delta(y).$$

From the first identity we solve for $e^{\lambda x}$:

$$z(\gamma e^{\lambda x} + \delta) = \alpha e^{\lambda x} + \beta,$$

hence

$$(\gamma z - \alpha) e^{\lambda x} = \beta - \delta z, \qquad e^{\lambda x} = \frac{\beta - \delta z}{\gamma z - \alpha} = \frac{\delta z - \beta}{\alpha - \gamma z}.$$

Similarly,

$$e^{\lambda y} = \frac{\delta w - \beta}{\alpha - \gamma w}.$$

Therefore

$$e^{\lambda(x+y)} = e^{\lambda x} e^{\lambda y} = \frac{(\delta z - \beta)(\delta w - \beta)}{(\alpha - \gamma z)(\alpha - \gamma w)}.$$

Set

$$N := (\delta z - \beta)(\delta w - \beta), \qquad D_0 := (\alpha - \gamma z)(\alpha - \gamma w),$$

so that $e^{\lambda(x+y)} = N/D_0$. By definition,

$$f_\Delta(x+y) = \frac{\alpha e^{\lambda(x+y)} + \beta}{\gamma e^{\lambda(x+y)} + \delta} = \frac{\alpha \frac{N}{D_0} + \beta}{\gamma \frac{N}{D_0} + \delta} = \frac{\alpha N + \beta D_0}{\gamma N + \delta D_0}.$$

We now expand N and D_0:

$$N = (\delta z - \beta)(\delta w - \beta) = \delta^2 zw - \delta\beta z - \delta\beta w + \beta^2,$$

$$D_0 = (\alpha - \gamma z)(\alpha - \gamma w) = \alpha^2 - \alpha\gamma z - \alpha\gamma w + \gamma^2 zw.$$

For the numerator,

$$\begin{aligned} \alpha N + \beta D_0 &= \alpha\big(\delta^2 zw - \delta\beta z - \delta\beta w + \beta^2\big)\\ &\quad + \beta\big(\alpha^2 - \alpha\gamma z - \alpha\gamma w + \gamma^2 zw\big)\\ &= (\alpha\delta^2 + \beta\gamma^2) zw - \alpha\beta(\delta+\gamma) z - \alpha\beta(\delta+\gamma) w\\ &\quad + \alpha\beta(\alpha+\beta). \end{aligned}$$

Thus

$$\alpha N + \beta D_0 = Azw + Bz + Cw + D,$$

with

$$A = \alpha\delta^2 + \beta\gamma^2, \quad B = C = -\alpha\beta(\delta+\gamma), \quad D = \alpha\beta(\alpha+\beta).$$

Similarly, for the denominator,

$$\begin{aligned} \gamma N + \delta D_0 &= \gamma\big(\delta^2 zw - \delta\beta z - \delta\beta w + \beta^2\big)\\ &\quad + \delta\big(\alpha^2 - \alpha\gamma z - \alpha\gamma w + \gamma^2 zw\big)\\ &= \gamma\delta(\delta+\gamma) zw - \gamma\delta(\alpha+\beta) z - \gamma\delta(\alpha+\beta) w\\ &\quad + \gamma\beta^2 + \alpha^2\delta. \end{aligned}$$

Hence

$$\gamma N + \delta D_0 = Ezw + Fz + Gw + H,$$

with

$$E = \gamma\delta(\delta+\gamma), \quad F = G = -\gamma\delta(\alpha+\beta), \quad H = \gamma\beta^2 + \alpha^2\delta.$$

Substituting back $z = f_\Delta(x)$ and $w = f_\Delta(y)$, we obtain

$$f_\Delta(x+y) = \frac{A f_\Delta(x) f_\Delta(y) + B f_\Delta(x) + C f_\Delta(y) + D}{E f_\Delta(x) f_\Delta(y) + F f_\Delta(x) + G f_\Delta(y) + H},$$

with A, B, C, D, E, F, G, H as above. Since both sides are rational functions of $e^{\lambda x}$ and $e^{\lambda y}$ and agree wherever they are defined, the identity holds for all x, y in the common domain of definition of the two sides. □

Appendix E.
Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

This paper presents Decor, an MPC framework for efficiently evaluating nonlinear functions, including trigonometric, hyperbolic, exponential, and sigmoid functions, with high precision. Inspired by Beaver multiplication triples, the key idea is to delegate expensive nonlinear computations to a trusted dealer that operates only on random values, allowing parties to mask their inputs at runtime and perform only lightweight MPC operations online. The framework further introduces a Fourier series-based approximation method as a general-purpose alternative to polynomial approximation.

E.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

E.3. Reasons for Acceptance

1) The generalization of Beaver triples to nonlinear functions is a conceptually clean and potentially influential contribution to the MPC community.
2) The paper achieves strong accuracy and runtime improvements, validated across concrete downstream applications.
3) All four reviewers found no apparent technical flaws, and the rebuttal satisfactorily addressed the primary concern around comparison with related work, including new benchmark results against MW-based and PWL approaches.