

Period-Aligned Secure Cosine Evaluation Based on Two-Party Computation

Qingyu Mo

Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences

Abstract

The cosine function is widely used in machine-learning applications, but evaluating it on secret-shared data is difficult. Conventional approaches use polynomial approximations or lookup tables, whose costs grow substantially over large input domains. More recent protocols first reduce the private input modulo the cosine period, but securely performing this modular reduction also incurs significant cost. We present a period-aligned two-party cosine protocol. Based on the cosine addition rule, it rescales each share so that a wrap of the input modulus corresponds to a full 2π period and therefore does not change the recombined cosine. Each party evaluates sine and cosine locally on its share, while the secure computation uses only two cross-party products and one fixed-point output conversion. We evaluate the protocol directly on synthetic inputs and within private RFF-based RBF-SVM prediction. Comparisons with the protocol of Xing et al. (NDSS 2025) and Guo et al. (USENIX Security 2026) demonstrate better efficiency of our proposed protocol.

I Introduction

Secure two-party computation allows two parties to evaluate a function of private data without revealing more than the agreed output. For numerical tasks, real-valued inputs are represented by scaled integers using fixed-point encoding. Additive secret sharing splits each encoded value into two shares, neither of which alone reveals the value. Additions can then be performed locally, whereas nonlinear operations generally require secure interaction or a specialized construction [3, 1]. Efficient nonlinear components are therefore important when building private numerical applications.

Cosine is a common nonlinear function used in machine learning. In plaintext computation it is routinely evaluated by floating-point numerical libraries, but such a library cannot simply be applied to a secret-shared input without a secure realization. A direct secure implementation must handle both the nonlinear computation and its numerical precision. We study the underlying primitive: given additive shares of a fixed-point angle, the parties obtain additive shares of a cosine approximation without opening the angle or intermediate trigonometric coefficients.

A conventional approach replaces the function by a polynomial, a piecewise approximation, or a lookup table. Directly covering a large input domain at fixed accuracy can require higher approximation degree, more segments, or more table entries, increasing the arithmetic or oblivious-selection cost. Alternatively, the parties can reduce the private angle to an interval spanning one full period of cosine, such as $[-\pi, \pi)$, before approximation [7]. This periodic range reduction must itself preserve privacy: the angle is shifted into the chosen interval by adding or subtracting whole multiples of 2π . Determining the required shift without revealing the angle can require secure comparisons and selection. Such private angular reduction is distinct from locally reducing a share modulo the public sharing modulus. Thus, the savings from a lower-degree approximation or a smaller lookup table must be weighed against the additional secure operations required for private range reduction.

The cosine addition formula, $\cos(a+b) = \cos a \cos b - \sin a \sin b$, suggests a different route. This identity suggests letting each party evaluate sine and cosine on its local share, leaving only two cross-party products to be computed securely. This produces the intended cosine when the local angles sum to the secure input. However, additive secret sharing over a finite ring does not generally guarantee this. Specifically, let X be the centered integer encoding of $x = X/S$, shared over $\mathbb{Z}_Q$, where S is the fixed-point scale and Q the input modulus. The two shares reconstruct X only modulo Q ; when added as integers, their sum is $X + kQ$ for some integer k . After division by S , the local angles therefore sum to $x + kQ/S$, and the addition formula evaluates $\cos(x + kQ/S)$, which does not

necessarily equal $\cos x$. To resolve this discrepancy, Guo et al. [2] augment local trigonometric evaluation with secret wrap detection and correction, using the detected wrap to select the appropriate result. This preserves the benefit of local function evaluation, but introduces additional secure operations for wrap detection and oblivious selection.

To avoid this additional cost, we design a new cosine protocol to make the wrap harmless rather than detect it. The core idea is to align the input modulus with the 2π period of cosine through a public local scaling and angular mapping. Any modular wrap then contributes only whole angular periods, so the addition formula can recombine local trigonometric values without a wrap-dependent correction.

We make two contributions:

1. **Period-aligned cosine protocol.** We give a two-party fixed-point protocol with two cross-party products in one multiplication layer and one fixed-point output conversion. We derive its fixed-point realization, bound its numerical error, and prove semi-honest security under the stated subprotocol assumptions.
2. **Implementation and experimental evaluation.** We describe the implementation and evaluate the protocol against existing secure cosine protocols in both direct benchmarks and a private RFF-based RBF-SVM prediction case study. The experiments examine runtime, communication, reported mean ULP error, and application-level score fidelity under the stated accounting limits.

Sections IV–VI develop this protocol from its mathematical principle through its construction and analysis. Section VII presents the implementation settings, cosine protocol comparisons, and application-level evaluation.

II Related Work

Existing secure cosine protocols mainly use circuit-based approximation or secure lookup. Bayatbabolghani et al. [8] present garbled-circuit constructions for trigonometric functions. Xing et al. [7] use function secret sharing to realize lookup- and spline-based evaluation, referred to in our experiments as Xing-LUT and Xing-Approx. These approaches evaluate the function within the secure computation, with costs determined by the selected approximation and input domain.

Guo et al. [2] instead evaluate trigonometric functions locally on additive shares and securely correct modular wraps through wrap computation and oblivious selection. Our protocol follows this local-decomposition direction but aligns modular wraps with the period of cosine, eliminating the wrap-dependent correction before recombination.

III Preliminaries and Problem Formulation

Centered representatives and sharing. Let P_0 and P_1 denote the two parties, and let $Q = 2^\ell$. For a residue $a \in \mathbb{Z}_Q$, or an integer interpreted modulo Q , define

$$\text{rep}_Q(a) = ((a + Q/2) \bmod Q) - Q/2 \in [-Q/2, Q/2). \quad (1)$$

The notation denotes the centered representative, not a replacement of the ring by a real interval. All additions, multiplications, and secret sharings remain modular operations.

Let x denote the real-valued input represented in fixed point with scaling factor $S = 2^{f_x}$. Its fixed-point encoding is an integer X satisfying $x = X/S$, where $X \in [-Q/2, Q/2)$ denotes the centered signed lift of the encoded value in $\mathbb{Z}_Q$. We write

$$\langle X \rangle_Q = (X_0, X_1), \quad X_i \in \mathbb{Z}_Q, \quad \text{rep}_Q(X_0 + X_1) = X, \quad (2)$$

where party P_i holds the additive share X_i . Fresh sharing samples one share uniformly from $\mathbb{Z}_Q$ and determines the other by modular subtraction. Hence, each individual share, and thus its centered representative, is independent of the encoded input.

Input domain. The centered integer encoding satisfies $-Q/2 \leq X < Q/2$, so $x = X/S$ can be negative, zero, or positive within the representable interval $[-Q/(2S), Q/(2S))$. We specify a public integer magnitude bound

$1 \leq B \leq Q/2$ and require $-B < X < B$, equivalently $|X| < B$. Thus B bounds the magnitude of the input; it does not restrict its sign. The supported input domain is the symmetric fixed-point grid

$$\mathcal{D}_{B,S} = X/S : X \in \mathbb{Z}, |X| < B, \quad -A \leq x \leq A, \quad A = \frac{B-1}{S}. \quad (3)$$

The bound constrains the encoded input, not the individual shares. The caller must ensure that the intended angle is encoded within this domain without overflow and choose (B, Q, S) to meet the alignment-error budget in Section IV-B.

Scales and working ring. Each party locally evaluates sine and cosine on its own angle share and encodes these values as integers by multiplying by $T = 2^f$ and rounding. We call these encoded trigonometric values *local coefficients*. Their products have scale T^2 and are shared in a working ring $\mathbb{Z}_M$. The output uses scale $O = 2^{f_y}$ in $\mathbb{Z}_{Q_o}$. Both M and Q_o are powers of two; M may equal Q , but the input and working rings have distinct roles. The fractional bit counts f_x, f, f_y are nonnegative integers. Set $r = 2f - f_y > 0$ and $L = M/2^r$. To avoid signed overflow, the recombined numerator must fit in the centered range of $\mathbb{Z}_M$, and its truncated value must fit in those of $\mathbb{Z}_L$ and $\mathbb{Z}_{Q_o}$. Sufficient conditions are $M > 4T^2$ and $Q_o > 4O$.

Problem formulation. Given additive shares $\langle X \rangle_Q = (X_0, X_1)$ of the fixed-point encoding X of an input angle $x = X/S$, our goal is for P_0 and P_1 to obtain additive shares $\langle Z \rangle_{Q_o} = (Z_0, Z_1)$, such that $\text{rep}_{Q_o}(Z_0 + Z_1)/O$ closely matches $\cos(x)$, without reconstructing x or revealing intermediate trigonometric values.

Security setting. The parties are semi-honest, at most one is statically corrupted, and communication is authenticated. Scales, moduli, input bounds, and rounding rules are public. Only the caller's authorized secret-shared output may be revealed.

IV Mathematical Principle of Period-Aligned Cosine

The addition-formula approach replaces secure trigonometric evaluation by local function calls and cross-party products. Its remaining obstacle is the wrap term in the sum of modular shares. We first show how local period alignment makes that term harmless and bound the resulting alignment error. Section V then realizes this construction using fixed-point secure arithmetic.

A Period Alignment

The key to our period-alignment construction is the following specific choice of the public multiplier and local scaled shares:

$$m = \text{round}\left(\frac{Q}{2\pi S}\right), \quad Y_i = mX_i \bmod Q, \quad \theta_i = \frac{2\pi \text{rep}_Q(Y_i)}{Q}. \quad (4)$$

Thus $\theta_i \in [-\pi, \pi)$ under the mathematical representation. We assume $Q > 2\pi S$, which ensures $m \geq 1$. Modular scaling is performed in integer arithmetic before conversion to a numerical angle; the public multiplier is computed with sufficient precision to determine its nearest integer unambiguously.

Since $Y_0 + Y_1 \equiv mX \pmod{Q}$, the centered representatives satisfy $\text{rep}_Q(Y_0) + \text{rep}_Q(Y_1) = mX + kQ$ for some integer k . Hence

$$\theta_0 + \theta_1 = \frac{2\pi}{Q} (\text{rep}_Q(Y_0) + \text{rep}_Q(Y_1)) = \frac{2\pi mX}{Q} + 2\pi k. \quad (5)$$

Each party can evaluate $\cos \theta_i$ and $\sin \theta_i$ locally. Combining these values through the cosine addition formula gives

$$\begin{aligned} \cos \theta_0 \cos \theta_1 - \sin \theta_0 \sin \theta_1 &= \cos(\theta_0 + \theta_1) \\ &= \cos\left(\frac{2\pi mX}{Q} + 2\pi k\right) \\ &= \cos\left(\frac{2\pi mX}{Q}\right). \end{aligned} \quad (6)$$

The first equality follows from the cosine addition formula, while the last follows from the 2π -periodicity of cosine. Since $\cos\left(\frac{2\pi mX}{Q}\right)$ is close to the target value $\cos(x)$, the two parties can obtain a value close to $\cos(x)$ using only two cross-party products and a local subtraction. The difference between $\cos\left(\frac{2\pi mX}{Q}\right)$ and $\cos(x)$ is quantified in Section IV-B.

B Alignment Error

To compare the resulting angle $\frac{2\pi mX}{Q}$ with the intended input $x = X/S$, define

$$\rho = \frac{2\pi mS}{Q}, \quad \frac{2\pi mX}{Q} = \rho x. \quad (7)$$

In exact real arithmetic, the recombination in (6) yields $\cos(\rho x)$ rather than the target $\cos x$. Although periodicity eliminates the wrap contribution exactly, rounding the public multiplier m introduces an alignment error.

Because cosine is 1-Lipschitz,

$$|\cos(\rho x) - \cos x| \leq |\rho - 1| |x|. \quad (8)$$

Nearest-integer rounding gives

$$\left| m - \frac{Q}{2\pi S} \right| \leq \frac{1}{2}, \quad |\rho - 1| \leq \frac{\pi S}{Q}. \quad (9)$$

For the public real-input bound $|x| \leq A$, it follows that

$$\epsilon_{\text{align}} \leq |\rho - 1| A \leq \frac{\pi S A}{Q}. \quad (10)$$

This theoretical alignment bound excludes local trigonometric error, coefficient quantization, and truncation. In integer-input notation the corresponding bound is $\pi|X|/Q$. A larger input domain at fixed (Q, S) can therefore increase approximation error even though its angular wraps remain harmless.

For a prescribed alignment-error budget $\delta_{\text{align}} > 0$, parameters can be selected to satisfy

$$|\rho - 1| A \leq \delta_{\text{align}}; \quad \frac{\pi S A}{Q} \leq \delta_{\text{align}} \quad \text{is sufficient.} \quad (11)$$

The first condition uses the actual public multiplier and can be less conservative than the second. These conditions accompany the input representation constraint and $Q > 2\pi S$; they budget only alignment error, leaving local evaluation and output conversion to the total-error analysis in Section VI. Thus the protocol can support inputs spanning many cosine periods without additional angular wrap detection, but not arbitrary input magnitudes at fixed precision and fixed parameters.

V Period-Aligned Cosine Protocol

Section IV-A derives a two-product identity in real arithmetic. To realize it on secret-shared fixed-point inputs, we encode the locally evaluated trigonometric values as integers, securely compute their cross-party products, and convert the recombined result to the output scale.

A Fixed-Point Realization

After computing its period-aligned angle θ_i using (4), each party locally forms

$$c_i = \text{round}(T \cos \theta_i), \quad s_i = \text{round}(T \sin \theta_i), \quad T = 2^f. \quad (12)$$

Substituting the decoded values c_i/T and s_i/T into (6) gives the fixed-point approximation

$$\frac{d}{T^2} = \frac{c_0}{T} \frac{c_1}{T} - \frac{s_0}{T} \frac{s_1}{T} \approx \cos(\rho x), \quad d = c_0 c_1 - s_0 s_1. \quad (13)$$

Secure multiplication can return additive shares of $u = c_0c_1$ and $v = s_0s_1$ in $\mathbb{Z}_M$, and each party subtracts its shares locally: $d_i = u_i - v_i \bmod M$. Under $M > 4T^2$, the reconstructed centered numerator equals the intended signed integer d without overflow.

The numerator has scale T^2 , whereas the output requires scale O . Since $T^2/O = 2^r$, the parties truncate the combined numerator and convert the resulting shares to $\mathbb{Z}_{Q_o}$. Our implementation uses faithful-style truncation [9], whose result differs from $\lfloor d/2^r \rfloor$ by at most one ULP. The decoded value z/O is the cosine approximation.

B Secure Protocol

We first specify the functionalities used in our cosine protocol. The functionality $\mathcal{F}_{\text{cross}}^M$ takes a private centered signed operand from each party and returns fresh additive shares of their integer product modulo M . The functionality $\mathcal{F}_{\text{trunc}}^{M,r}$ takes a numerator shared in $\mathbb{Z}_M$, interprets $d = \text{rep}_M(d_0 + d_1)$, and returns fresh shares in $\mathbb{Z}_L$ of a faithful result $z \in \{\lfloor d/2^r \rfloor - 1, \lfloor d/2^r \rfloor\}$ at scale O . A subsequent value-preserving signed conversion produces shares in $\mathbb{Z}_{Q_o}$. Concrete library and subprotocol choices of the two functionalities are described in Section VII-A. Algorithm 1 demonstrates the workflow of our secure cosine protocol. The four stages of our protocol are local period alignment, local sine and cosine evaluation, two independent cross-party products, and local recombination followed by truncation and conversion. Scaling, trigonometric evaluation, and recombination require no communication; the cross-party products are the main interactive arithmetic, followed by the dependent output stage.

Algorithm 1 Period-Aligned Secure Cosine Evaluation

Require: Each P_i holds $X_i \in \mathbb{Z}_Q$ encoding X with $|X| < B$; public parameters satisfy the input-domain, alignment-budget, and working-ring conditions.

Ensure: Additive shares in $\mathbb{Z}_{Q_o}$ at scale O , without reconstruction ($r = 2f - f_y$, $L = M/2^r$).

- 1: Set $m \leftarrow \text{round}(Q/(2\pi S))$.
 - 2: Each P_i locally computes $Y_i \leftarrow mX_i \bmod Q$.
 - 3: Each P_i computes $\theta_i \leftarrow 2\pi \text{rep}_Q(Y_i)/Q$.
 - 4: Each P_i computes $c_i \leftarrow \text{round}(T \cos \theta_i)$ and $s_i \leftarrow \text{round}(T \sin \theta_i)$.
 - 5: Invoke $\mathcal{F}_{\text{cross}}^M$ independently for $\langle u \rangle_M = \langle c_0c_1 \rangle_M$ and $\langle v \rangle_M = \langle s_0s_1 \rangle_M$.
 - 6: Each P_i locally computes $d_i \leftarrow u_i - v_i \bmod M$.
 - 7: Invoke faithful signed truncation from T^2 to O : $\langle z \rangle_L \leftarrow \mathcal{F}_{\text{trunc}}^{M,r}(\langle d \rangle_M)$.
 - 8: Convert the result to the declared output ring $\mathbb{Z}_{Q_o}$ with signed-value preservation.
 - 9: Return the output shares without reconstruction.
-

VI Error and Security Analysis

A Numerical Error

The total error separates into

$$|\widehat{\cos(x)} - \cos x| \leq \epsilon_{\text{align}} + \epsilon_{\text{quant}} + \epsilon_{\text{trunc}} + \epsilon_{\text{output}}. \quad (14)$$

The output $\widehat{\cos(x)}$ is decoded at scale O . The term ϵ_{quant} captures the rounding of the local sine and cosine values to scale T . From (12), each decoded coefficient differs from its exact trigonometric value by at most $\eta = 1/(2T)$. Expanding the two products and using $|\cos \theta_i| + |\sin \theta_i| \leq \sqrt{2}$ gives

$$\epsilon_{\text{quant}} \leq 2\sqrt{2}\eta + 2\eta^2 = \frac{\sqrt{2}}{T} + \frac{1}{2T^2}. \quad (15)$$

Our implementation applies faithful-style truncation to the recombined numerator. Because its result can be one output integer below $\lfloor d/2^r \rfloor$, its contribution satisfies $\epsilon_{\text{trunc}} < 2/O$. Value-preserving ring conversion has $\epsilon_{\text{output}} = 0$.

B Semi-Honest Security

Local period alignment and local trigonometric evaluation send no data. Assuming the signed cross-term, truncation, and conversion subprotocols are semi-honest secure, their transcripts are simulatable from each party’s inputs and returned shares. Local recombination introduces no additional messages. Sequential composition therefore establishes the semi-honest security of the period-aligned cosine protocol under the specified finite-precision contract. The faithful-style truncation used in our implementation has a distinct rounding contract: its omitted carry depends on the randomness of the numerator sharing. Local coefficient rounding can likewise depend on the input sharing.

VII Experimental Evaluation

Our evaluation has two parts. We first compare the secure cosine protocols directly with Xing-LUT and Xing-Approx, the lookup and spline variants of Xing et al. [7], GEO [2], and our protocol. We then integrate the protocols into private RFF-based RBF-SVM prediction and examine feature-evaluation cost and decision-score fidelity. The following setup describes the common execution environment before presenting the protocol and application-level results.

A Experimental Setup

Environment. All experiments ran as two localhost processes on the same physical workstation. For analytical communication cost projections, the LAN model assumes 1 Gbit/s bandwidth and 0.5 ms RTT, while the WAN model assumes 40 Mbit/s bandwidth and 20 ms RTT. The machine used a 12th Gen Intel Core i7-12700 processor with 10 cores and 20 logical threads and 16 GB of memory. The software environment was Ubuntu 22.04.4 LTS on WSL2, C++20, G++ 11.4.0, and CMake 3.26.0.

Implementation. The prototype was implemented in C++ using the SCI version bundled with the GEO repository¹. The GEO baseline corresponds to Guo et al. [2], while all GEO results in our protocol and application comparisons use this repository implementation. The Xing baselines use the dealerless-FSS implementation² together with EzPC.³ Local compatibility fixes and batching changes were applied to the Xing implementation. All four evaluated methods—Xing-LUT, Xing-Approx, GEO, and our protocol—use the same faithful-style truncation implementation from the GEO repository.

B Secure Cosine Protocol Evaluation

Benchmark configuration. For the direct cosine protocol evaluation, Table I specifies a common $\mathbb{Z}_{2^{32}}$ input and $\mathbb{Z}_{2^{22}}$ output interface with 12 fractional bits. These settings satisfy the range conditions in Section III.

Table I: Cosine protocol benchmark configuration.

Quantity	Declared reference setting
Input ring and scale	$Q = 2^{32}, S = 2^{12}$
Output ring and scale	$Q_o = 2^{22}, O = 2^{12}, f_y = 12$
Local coefficient precision and scale	$f = 14, T = 2^{14}$
Product scale and working ring	$T^2 = 2^{28}, M = 2^{32}$
Truncation shift and reduced ring	$r = 16, L = M/2^r = 2^{16}$
Period multiplier	$m = 166886; Q/(2\pi S) \approx 166886.053608$
Output ULP	$2^{-12} = 0.000244140625$
Public integer bounds	$B \in \{2^{15}, 2^{17}, 2^{19}\}$

Table II compares our protocol with Xing-LUT, Xing-Approx, and GEO. For each input range, we sample 200,000 centered fixed-point inputs uniformly from $[-B + 1, B - 1]$, generate fresh additive shares in $\mathbb{Z}_{2^{32}}$, and

¹https://github.com/geralt-tian/GEO_2025

²https://github.com/xingpz2008/dealerless-FSS_public

³<https://github.com/mpc-msri/EzPC>

run five independent trials. The reported results are averaged. The three evaluated ranges of the decoded input $x = X/S$ are $[-8, 8]$, $[-32, 32]$, and $[-128, 128]$ radians. The reported Xing-LUT and Xing-Approx costs include offline FSS generation.

Table II: Comparison of secure cosine protocols.

Metric	Angle range	Xing-LUT	Xing-Approx	GEO	Ours
LAN projection (μs)	$[-8, 8]$	3.94×10^5	5.20×10^6	22.08	9.98
	$[-32, 32]$	3.96×10^5	5.19×10^6	21.98	9.58
	$[-128, 128]$	3.99×10^5	5.24×10^6	22.07	9.51
WAN projection (μs)	$[-8, 8]$	5.04×10^6	3.00×10^7	336.20	141.27
	$[-32, 32]$	5.06×10^6	2.99×10^7	336.09	140.87
	$[-128, 128]$	5.09×10^6	3.04×10^7	336.19	140.80
Communication (KB)	$[-8, 8]$	4.51×10^3	5.60×10^4	1.60	0.67
	$[-32, 32]$				
	$[-128, 128]$				
Mean ULP (reported)	$[-8, 8]$	1.12	1.53	1.00	1.00
	$[-32, 32]$	1.15	1.52	1.00	1.00
	$[-128, 128]$	1.16	1.52	1.00	1.00

Our protocol requires 9.51–9.98 μs per output under the LAN model and 140.80–141.27 μs under the WAN model. Compared with GEO, this corresponds to modeled runtime reductions of 2.21–2.32 $\times$ and 2.38–2.39 $\times$, respectively. Communication decreases from 1.60 KB to 0.67 KB, a 58.1% reduction. The estimates vary little across the three input ranges. Our protocol and GEO both report mean ULP 1.00, compared with 1.12–1.16 for Xing-LUT and 1.52–1.53 for Xing-Approx. These per-output values are vector-amortized analytical estimates rather than isolated-call or physical-network measurements.

C Application Case Study: Private RFF-Based RBF-SVM Prediction

We evaluate application-level integration through private RFF-based RBF-SVM prediction [5, 6]. We assume that the client holds the query vector x , while the server holds the RFF parameters $\{(\omega_j, b_j)\}_{j=1}^D$ and the linear classifier parameters (w, β) , where $w \in \mathbb{R}^D$ is the weight vector and β is the bias. The decision score and binary prediction are

$$s(x) = w^\top \phi(x) + \beta = \sum_{j=1}^D w_j \sqrt{\frac{2}{D}} \cos(\omega_j^\top x + b_j), \quad \hat{y} = \text{sign}(s(x)). \quad (16)$$

Table III summarizes the datasets, test ratios, and RBF kernel parameters, and Table IV reports the results. Decision-score RMSE measures the root mean squared difference between the private decision scores and those produced by the corresponding plaintext RFF-SVM workflow over the test samples. Our protocol substantially outperforms both Xing-LUT and Xing-Approx in runtime and decision-score RMSE across all evaluated configurations. Even after excluding offline FSS generation from the two Xing baselines, their remaining runtimes are still higher than ours. Compared with GEO, our protocol consistently achieves lower RMSE while providing 1.29–1.67 $\times$ speedup under LAN and 1.58–2.00 $\times$ under WAN.

Table III: Datasets used for application-level evaluation.

Dataset	Size	Test ratio	γ
Heart Disease	303	0.16	0.55
WBC	683	0.25	0.8
AIDS Study 175	1280	0.2	0.6
Spambase	4601	0.1097	1.4
Anemia	8350	0.1666667	2.4

Table IV: Application-level evaluation: setup-amortized analytical LAN/WAN feature-evaluation costs and decision-score RMSE for private RFF-SVM prediction.

Dataset	D	Xing-LUT			Xing-Approximation			GEO			Ours		
		LAN (s)	WAN (s)	RMSE	LAN (s)	WAN (s)	RMSE	LAN (s)	WAN (s)	RMSE	LAN (s)	WAN (s)	RMSE
Heart	8	4.08	46.96	4.96×10^{-2}	23.96	127.35	4.56×10^{-1}	0.13	1.23	1.54×10^{-5}	0.08	0.78	1.30×10^{-5}
	16	7.64	79.29	2.99×10^{-3}	46.17	235.71	2.75×10^{-1}	0.13	1.23	1.06×10^{-5}	0.08	0.78	9.52×10^{-6}
	32	14.49	144.76	2.61×10^{-3}	91.66	468.81	2.34×10^{-1}	0.13	1.24	1.18×10^{-5}	0.08	0.78	8.94×10^{-6}
WBC	8	4.09	46.72	1.22×10^{-2}	24.07	127.21	1.17×10^{-1}	0.10	0.94	5.53×10^{-6}	0.06	0.52	3.31×10^{-6}
	16	7.75	79.14	9.53×10^{-3}	46.75	236.05	1.01×10^{-1}	0.10	0.95	4.97×10^{-6}	0.07	0.52	3.81×10^{-6}
	32	14.81	144.84	6.09×10^{-3}	92.38	469.30	6.89×10^{-2}	0.10	0.95	6.11×10^{-6}	0.07	0.52	3.25×10^{-6}
AIDS	64	26.09	279.67	7.81×10^{-3}	177.06	990.48	6.77×10^{-2}	0.10	0.93	4.35×10^{-6}	0.06	0.48	3.82×10^{-6}
	128	55.37	567.58	5.71×10^{-3}	371.37	2303.01	4.97×10^{-2}	0.10	0.94	4.98×10^{-6}	0.06	0.49	3.44×10^{-6}
	256	112.70	1215.91	3.17×10^{-3}	797.97	5946.64	3.25×10^{-2}	0.10	0.98	3.55×10^{-6}	0.06	0.49	2.53×10^{-6}
Spambase	32	14.88	144.84	3.80×10^{-3}	86.47	463.32	3.42×10^{-2}	0.10	0.88	3.61×10^{-6}	0.06	0.45	2.02×10^{-6}
	64	28.74	280.28	3.26×10^{-3}	185.49	998.87	3.11×10^{-2}	0.10	0.89	3.18×10^{-6}	0.06	0.45	1.93×10^{-6}
	128	55.92	568.09	1.73×10^{-3}	388.46	2320.06	1.59×10^{-2}	0.10	0.91	3.65×10^{-6}	0.07	0.46	2.02×10^{-6}
Anemia	32	14.94	144.87	2.87×10^{-3}	92.21	469.04	2.84×10^{-2}	0.09	0.85	3.07×10^{-6}	0.06	0.43	2.12×10^{-6}
	64	28.13	279.66	2.30×10^{-3}	185.05	998.41	2.17×10^{-2}	0.09	0.86	4.00×10^{-6}	0.06	0.43	2.43×10^{-6}
	128	55.66	567.81	1.26×10^{-3}	384.97	2316.55	1.21×10^{-2}	0.09	0.88	2.71×10^{-6}	0.07	0.44	1.63×10^{-6}

VIII Conclusion

We presented a period-aligned two-party protocol for secure fixed-point cosine evaluation. By mapping a wrap of the sharing modulus to an integer multiple of 2π , the protocol makes modular wraps harmless rather than detecting and correcting them. Combined with the cosine addition formula, this reduces secure recombination to two cross-party products in one multiplication layer and one fixed-point output conversion. We also established its finite-precision error bounds and semi-honest security under the stated subprotocol assumptions.

In the direct cosine evaluation, our protocol uses 0.67 KB per output, achieves a mean ULP error of 1.00, and has analytical LAN/WAN estimates of 9.51–9.98 μs and 140.80–141.27 μs . In private RFF-based RBF-SVM prediction, it achieves lower decision-score RMSE than Xing-LUT, Xing-Approx, and GEO across all evaluated configurations. It remains faster than the Xing baselines even after excluding their offline FSS-generation time and provides 1.29–1.67 $\times$ LAN and 1.58–2.00 $\times$ WAN speedup over GEO.

The current evaluation assumes semi-honest parties and bounded fixed-point inputs. Extending the protocol to malicious security and broader periodic functions remains future work.

References

- [1] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi, “SIRNN: A math library for secure RNN inference,” in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1003–1020, 2021.
- [2] H. Guo, Z. Liu, L. Peng, S. Li, X. Fu, W. Liu, and L. Qu, “Efficient and high-accuracy secure two-party protocols for a class of functions with real-number inputs,” in *35th USENIX Security Symposium (USENIX Security 26)*, pp. 2903–2921, 2026.
- [3] D. Demmler, T. Schneider, and M. Zohner, “ABY—A framework for efficient mixed-protocol secure two-party computation,” in *Network and Distributed System Security Symposium*, 2015.
- [4] A. Patra, T. Schneider, A. Suresh, and H. Yalame, “ABY2.0: Improved mixed-protocol secure two-party computation,” *Cryptology ePrint Archive*, Paper 2020/1225, 2020.
- [5] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in *Advances in Neural Information Processing Systems 20*, 2007.
- [6] S. Laur, H. Lipmaa, and T. Mielikäinen, “Cryptographically private support vector machines,” *Cryptology ePrint Archive*, Paper 2006/198, 2006.
- [7] P. Xing, H. Li, M. Hao, H. Chen, J. Hu, and D. Liu, “Distributed function secret sharing and applications,” in *Network and Distributed System Security Symposium (NDSS)*, 2025, doi: 10.14722/ndss.2025.242233.

- [8] F. Bayatbabolghani, M. Blanton, M. Aliasgari, and M. Goodrich, “Poster: Secure computations of trigonometric and inverse trigonometric functions,” *IEEE Symposium on Security and Privacy, Poster Abstracts*, 2017.
- [9] Z. Huang, W.-J. Lu, C. Hong, and J. Ding, “Cheetah: Lean and fast secure two-party deep neural network inference,” in *31st USENIX Security Symposium (USENIX Security 22)*, pp. 809–826, 2022.