EncFormer: Secure and Efficient Transformer Inference over Encrypted Data

Yufan Zhu*, Chao Jin†, Khin Mi Mi Aung†, and Xiaokui Xiao*

***Abstract*—Transformer inference in machine-learning-as-a-service (MLaaS) raises privacy concerns for sensitive user inputs. Prior secure solutions that combine fully homomorphic encryption (FHE) and secure multiparty computation (MPC) are bottlenecked by inefficient FHE kernels, communication-heavy MPC protocols, and expensive FHE–MPC conversions. We present EncFormer, a two-party private Transformer inference framework that introduces Stage-Compatible Patterns so that FHE kernels compose efficiently, reducing repacking and conversions. EncFormer also provides a cost analysis model built around a minimal-conversion baseline, enabling principled selection of FHE–MPC boundaries. To further reduce communication, EncFormer proposes a secure complex CKKS–MPC conversion protocol and designs communication-efficient MPC protocols for nonlinearities. With GPU optimizations, evaluations on GPT- and BERT-style models show that EncFormer achieves 1.4×–30.4× lower inference-time communication and 1.3×–9.9× lower end-to-end latency relative to prior hybrid FHE–MPC systems, and 1.9×–3.5× lower BERT-base end-to-end latency than FHE-only pipelines under a matched backend, while maintaining near-plaintext accuracy on selected GLUE tasks.**

***Index Terms*—private inference, homomorphic encryption, secure multiparty computation, transformer inference**

I. Introduction

Transformer models such as BERT [1] and GPT [2] are increasingly deployed through machine-learning-as-a-service platforms, where user inputs are processed by an untrusted model provider. Sensitive domains such as clinical NLP [3], legal text [4], biomedical mining [5], and financial compliance [6] amplify the risk, as both inputs and predictions may be confidential, and attacks such as membership inference [7] can extract private information from model access alone. Private inference based on fully homomorphic encryption (FHE) and secure multiparty computation (MPC) is a natural fit, but current Transformer pipelines remain expensive because their performance depends not only on the cryptographic primitive but also on how ciphertext layouts, conversion boundaries, and nonlinear protocols interact across the pipeline [8]–[12].

Prior work on private Transformer inference follows three broad directions. FHE-only systems eliminate online interaction but pay heavily for deep ciphertext computation and, in many cases, bootstrapping [11]–[13]. MPC-only systems preserve accurate nonlinear computation but are dominated by repeated interaction on wide activations [9], [14]. Hybrid FHE–MPC systems balance these trade-offs by evaluating wide linear algebra under FHE and delegating nonlinear blocks to MPC, but they inherit a second design problem: every benefit from moving work across the boundary depends on whether the resulting packing can be consumed by the next stage and whether the added conversion cost is justified [8], [10], [15], [16].

*School of Computing (SoC), National University of Singapore (NUS), Singapore.

†Institute of Advanced Intelligence and Computing (IAIC), A*STAR (Agency for Science, Technology and Research), Singapore.

A. Challenges in Hybrid Private Transformer Inference

For hybrid pipelines, the main bottleneck is cross-stage design rather than a single cryptographic primitive. First, FHE kernels only deliver their speedup when the ciphertext packing they produce can be consumed by downstream kernels without costly remapping. Second, FHE–MPC conversion is expensive enough that boundary placement and representation should be selected by measured CKKS, MPC, and network costs rather than intuition alone. Third, complex packing should be treated as both an FHE-kernel primitive and a formal boundary representation. Prior systems either export only real CKKS slots in conversion or use complex packing at the boundary without formalizing which layout should be selected under measured cost.

These observations suggest that private Transformer inference should be framed as a secure systems co-design problem rather than as a collection of independent optimizations. The system should expose a consistent cross-stage packing pattern, a secure and communication-efficient conversion protocol, and a boundary decision rule that analytically explains the conversion layout.

B. Our Contribution

We present EncFormer, a two-party hybrid FHE–MPC framework for private Transformer inference in the semi-honest model. The party roles, key distribution, and decryption boundary are specified in the threat model in §II-C. At the system level, the protocol composes FHE kernels, secure conversion boundaries, and MPC nonlinear blocks. Our thesis is that hybrid private inference becomes systematic once packing and boundary representation are treated as analytical primitives rather than model-specific engineering choices. We make four integrated contributions.

- **SCP and Packing-Aligned FHE Kernels.** We introduce Stage-Compatible Patterns (SCP) in §III as an analytically grounded packing discipline for hybrid private inference that guides per-kernel layout choices through three canonical packing formats acting as a cross-stage contract. Guided by SCP, EncFormer designs a

shared plaintext–ciphertext projection kernel and folded-diagonal ciphertext–ciphertext attention kernels whose outputs are directly consumable by downstream stages, avoiding ciphertext remaps between adjacent FHE kernels.

- **Cost-Model Boundary Decision Rule.** We derive a calibrated cost-model decision rule in §V that ranks candidate FHE–MPC boundary representations under measured CKKS and MPC costs, exposing boundary layout as a first-class selectable design variable and guiding boundary placement through a measured trade-off rather than the intuition-driven choices of prior systems.
- **End-to-End Secure Inference Pipeline.** We implement EncFormer on GPUs for GPT2-base, BERT-base, and BERT-large, reducing inference-time communication by 1.4×–30.4× and end-to-end latency by 1.3×–9.9× relative to BOLT, BumbleBee, and BLB, and by 1.9×–3.5× over THOR and Powerformer under matched backends, while maintaining near-plaintext GLUE accuracy on BERT-base. The pipeline security proof appears in §II-C, and derivations and supporting analytical details are in the supplementary appendix.[1]
- **Formalized Complex Conversion and Adapted MPC Protocols.** We formalize complex CKKS–MPC conversion (CC) in §V-B as a first-class boundary representation, extending BLB's use of CC [10] into a general protocol with security and cost analysis, and adapt Powerformer's [12] polynomial approximations of softmax and layer normalization to secure MPC protocols in §IV, achieving lower communication and latency than the prior art.

C. System Overview

Figure 1 summarizes one EncFormer encoder block. The server first applies a shared plaintext–ciphertext projection kernel to produce Q, K, and V in layouts chosen for their immediate consumers: Q and K are emitted in the score-friendly segment order, while V is emitted in head-major order. The score kernel exports minimal folded-diagonal ciphertexts across the CKKS–MPC boundary, where the MPC softmax block operates on secret shares and returns weights in the same boundary format. The value kernel consumes these weights directly, and the output projection absorbs the remaining layout adaptation through plaintext weight pre-permutation rather than ciphertext repacking. The feed-forward projections follow the same principle, while MPC is reserved for layer normalization and GELU. This organization is the structural basis for SCP, complex conversion, and the boundary cost analysis developed in the following sections.

II. Background and Threat Model

A. Cryptographic Preliminaries

Fully Homomorphic Encryption. A public-key FHE scheme provides algorithms $\mathsf{KeyGen} \rightarrow (\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$, $\mathsf{Enc}_{\mathsf{pk}}(\cdot)$, $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$, and $\mathsf{Eval}_{\mathsf{evk}}(f, \cdot)$ such that, for an arithmetic circuit f,

$$\mathsf{Dec}_{\mathsf{sk}}\big(\mathsf{Eval}_{\mathsf{evk}}(f, \mathsf{Enc}_{\mathsf{pk}}(x))\big) = \begin{cases} f(x) \bmod t, \\ f(x) + \varepsilon. \end{cases}$$

The first line corresponds to exact schemes such as BFV and BGV. The second corresponds to approximate schemes such as CKKS, where ε captures approximation from encoding, decoding, and noise growth [17]–[19]. We refer to [20], [21] for broader FHE background. The algorithm KeyGen samples a public key for encryption, a secret key for decryption, and evaluation keys derived from the secret key. In RLWE-based FHE, evk typically includes relinearization keys to manage ciphertext growth after multiplication and Galois keys that enable SIMD slot permutations such as rotations and conjugation. The algorithm $\mathsf{Enc}_{\mathsf{pk}}$ encodes and encrypts a plaintext into a ciphertext, $\mathsf{Eval}_{\mathsf{evk}}$ applies additions, multiplications, and slot permutations directly on ciphertexts using only public evaluation material, and $\mathsf{Dec}_{\mathsf{sk}}$ recovers the plaintext result [18], [22], [23].

CKKS Scheme. CKKS [19] is an approximate FHE scheme for vectors of real or complex values. It encodes these values into SIMD slots and supports additions, multiplications, rotations, and conjugation without decryption. Multiplications increase scale and noise, so CKKS computations are scheduled against a finite modulus chain. For ring degree $N = 2^k$, one ciphertext encrypts $N/2$ complex slots and is represented by two polynomials over an RNS ring

$$c_0, c_1 \in R_q, \qquad R_q = \mathbb{Z}_q[X]/\langle X^N + 1\rangle, \qquad q = \textstyle\prod_i p_i.$$

The product modulus q is written as a chain of RNS primes. The next subsection gives the level accounting used by the cost model. The FHE section introduces how these slots are packed for EncFormer kernels in §III-A. Appendix M explains why CKKS is the right scheme for our setting and gives the security details.

Secure Multiparty Computation. Secure multiparty computation allows parties to jointly evaluate a function over

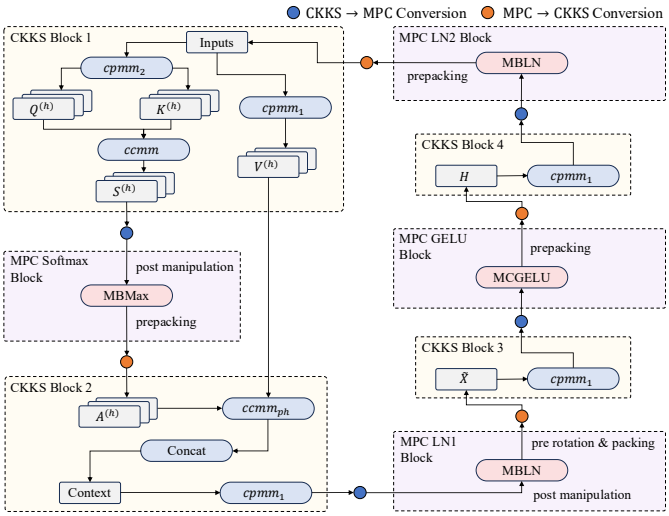


Fig. 1. EncFormer encoder block showing FHE kernels, MPC nonlinear blocks, and CKKS–MPC conversion boundaries.

[1] Appendix §§ A–Q are submitted as supplementary material.

TABLE I
MPC FIXED-POINT PRIMITIVES OVER $\mathbb{Z}_{2^\ell}$ WITH SCALE 2^F.

Primitive	Inputs	Output
$\Pi_{\times}$	$([[a]]_{2^\ell}, [[b]]_{2^\ell})$	$[[\lfloor (a \cdot b)/2^F \rceil]]_{2^\ell}$
Π_{cmp}	$([[x]]_{2^\ell}, \tau)$	$[[b]]_{2^\ell}$ where $b = \mathbf{1}[x < \tau]$
Π_{mux}	$([[u]]_{2^\ell}, [[b]]_{2^\ell})$	$[[v]]_{2^\ell}$ where $v = b \cdot u$
Π_{rowsum}	$[[\mathbf{X}]]_{2^\ell} \in \mathbb{Z}_{2^\ell}^{T \times d}$	$[[\mathbf{s}]]_{2^\ell} \in \mathbb{Z}_{2^\ell}^{T \times 1}, \quad s_t = \sum_{k=0}^{d-1} X_{t,k}$
Π_{bcast}	$([[\boldsymbol{\mu}]]_{2^\ell} \in \mathbb{Z}_{2^\ell}^{T \times 1}, d)$	$[[\mathbf{M}]]_{2^\ell} \in \mathbb{Z}_{2^\ell}^{T \times d}, \quad M_{t,k} = \mu_t$

private inputs while revealing only the prescribed outputs. We use two-party arithmetic secret sharing over $\mathbb{Z}_{2^\ell}$ and write additive shares as $[[\cdot]]_{2^\ell}$. Real values use signed fixed-point encoding with public scale $s = 2^F$. A real value r is represented by $\lfloor r \cdot 2^F \rceil \bmod 2^\ell$ and interpreted through the centered representative of $\mathbb{Z}_{2^\ell}$. Public constants and polynomial coefficients use the same scale. Linear maps, row sums, broadcasts, and affine operations are local on shares, while comparisons and multiplexing use standard two-party subprotocols. Shared multiplication uses Beaver triples [24]. Since multiplying two s-scaled values yields scale s^2, we apply secure truncation by F bits to return to scale s. Table I summarizes the primitives used later. The resulting online phase has one interaction round per multiplication depth and communication proportional to circuit size, as in SPDZ-style protocols [25]. In the semi-honest model, security is captured by the real/ideal paradigm [26], [27].

B. CKKS Parameters

Figure 2 illustrates how computation proceeds across the RNS modulus chain. Practical deployments fix a chain and a target scale Δ sized for multiplicative depth $D \leq L$, processing polynomial coefficients as residues modulo each q_i to enable NTT-based arithmetic. Let

$$Q = \prod_{i=0}^{L} q_i, \qquad Q_i = \prod_{j=0}^{i} q_j$$

denote the full ciphertext modulus and the remaining modulus at level i. A ciphertext–ciphertext multiplication inflates the scale by roughly a factor of Δ and is followed by rescale, which maps $Q_i \to Q_{i-1}$ by dividing by q_i and restores the scale to approximately Δ. After u multiplication-rescale steps, the remaining modulus is Q_{L-u}. Ciphertext size and remaining multiplicative depth both depend on the active level, so boundary-conversion cost depends not only on tensor shape but also on the CKKS parameters at the conversion point. Our cost model therefore treats ciphertext size as a function of the ring degree and the remaining RNS limbs.

C. Threat Model

We consider the standard two-party private inference setting with a client P_0 and a server P_1. The client provides the private input and learns the final inference output; the server provides the model. We assume semi-honest adversaries: both parties follow the protocol but may try to infer additional information from the transcript. As in prior hybrid systems, the model architecture is known to both parties [8], [10], [16], [28], [29].

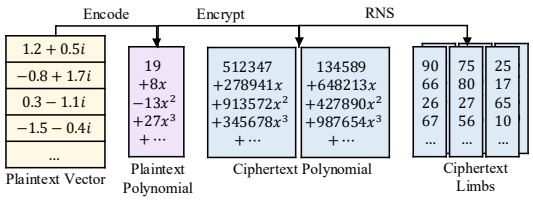


Fig. 2. Illustration of RNS–CKKS, showing the residue representation and the modulus-chain progression under rescaling.

Key Distribution. Following prior hybrid systems [8], [10], [16], [29], the client generates the CKKS key material $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$, sends only $(\mathsf{pk}, \mathsf{evk})$ to the server, and keeps sk locally. Thus decryption capability remains client-side, while the server evaluates FHE kernels using public evaluation keys and never observes plaintext activations. During CKKS-to-MPC conversion, the client decrypts only one-time-padded ciphertexts, so the value seen by either party is statistically or computationally protected by the conversion protocol. The final logits are decrypted only for the client, which is the prescribed output of the private-inference functionality.

Security Proof. The inference protocol is organized as a sequence of components $\Pi_{\mathsf{Infer}} = \Pi_1 \circ \cdots \circ \Pi_K$, where each Π_j is a CKKS evaluation block, an MPC block over $\mathbb{Z}_{2^\ell}$, or an FHE–MPC conversion boundary. Every intermediate value is represented either as a CKKS ciphertext under the secret key or as additive secret shares held by the two parties.

Theorem 1 (Pipeline Security)**.** *Let Π_{Infer} denote the composed EncFormer inference protocol over a sequence of CKKS evaluation blocks, MPC blocks, and FHE–MPC conversion boundaries. Assume the CKKS scheme Π_{FHE} is instantiated with the Table VI parameters under the sparse-ternary secret distribution of Hamming weight 64 and is IND-CPA secure at classical security $\lambda \geq 128$ bits, with the deployed parameters and conservative estimator-derived lower bounds listed in Table VI. Assume further that every MPC subprotocol in*

$$\Pi_{\mathrm{MPC}} \;=\; \{\, \Pi_{\mathrm{MBMax}},\ \Pi_{\mathrm{MBLN}},\ \Pi_{\mathrm{GELU}} \,\}$$

is secure against a semi-honest adversary in the two-party setting, and that each conversion protocol $\Pi^{\mathbb{C}}_{\mathrm{C2M}}$ and $\Pi^{\mathbb{C}}_{\mathrm{M2C}}$ satisfies the conversion-safety threshold

$$\log_2(q_{\mathrm{conv}}) \;\geq\; \ell + \sigma + 1.$$

Then Π_{Infer} securely realizes the ideal inference functionality against a semi-honest adversary corrupting at most one of the two parties. The corrupted party's view reveals nothing beyond the simulator output of each component, and the only output disclosure prescribed by the protocol is the client's decryption of the final logits. The public leakage is limited to the fixed architecture, tensor dimensions, CKKS/MPC parameters, message schedule, and message sizes already determined by the selected model/profile; it excludes the client's input, server model weights, and all intermediate activations.

Proof sketch. Each component Π_j admits a simulator under the assumptions above: CKKS blocks follow from IND-CPA security, MPC blocks from semi-honest MPC security,

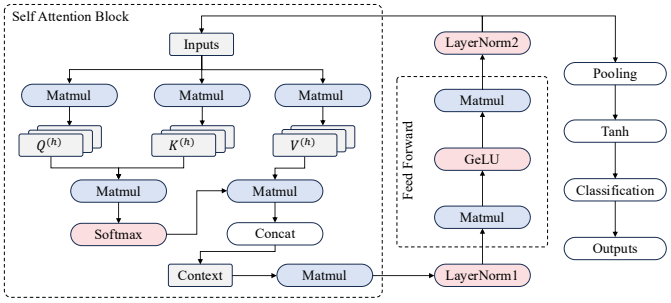


Fig. 3. Reference Transformer architecture used in this work, highlighting attention, feed-forward layers, and the output head.

and conversion boundaries from §V. The global simulator invokes these component simulators in pipeline order, and the sequential-composition theorem of Canetti [30] yields a transcript computationally indistinguishable from the real execution. Appendix L gives the ideal functionality, leakage function, and party-specific simulator construction. □

D. Transformer Model

The Transformer is a sequence model built around self-attention, replacing recurrence and convolution with token–token interactions that can be computed in parallel [31]. For a sequence of length m with hidden size d_{model} and activations $A \in \mathbb{R}^{m\times d_{\text{model}}}$, the attention sublayer forms $Q = AW_Q$, $K = AW_K$, and $V = AW_V$ with $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}}\times d_{\text{model}}}$. With H heads of width $d_h = d_{\text{model}}/H$, write $Q^{(h)}, K^{(h)}, V^{(h)} \in \mathbb{R}^{m\times d_h}$ for the hth head slices. A compact head computation is

$$\text{Att}^{(h)} \;=\; \text{softmax}\!\left(\frac{Q^{(h)}(K^{(h)})^\top}{\sqrt{d_h}}\right)V^{(h)}.$$

The multi-head output concatenates all heads and applies an output projection $W_O \in \mathbb{R}^{d_{\text{model}}\times d_{\text{model}}}$,

$$\text{MHA}(A) \;=\; \text{Concat}\big(\text{Att}^{(0)},\ldots,\text{Att}^{(H-1)}\big)W_O.$$

This is followed by a position-wise feed-forward sublayer $\text{FFN}(x) = \phi(xW_1 + b_1)W_2 + b_2$ using $\phi = \text{GELU}$, with residual connections and layer normalization throughout [31]. Figure 3 shows the reference architecture used in this work.

BERT is an encoder-only Transformer that stacks N_L identical layers to build bidirectional context [1]. We evaluate BERT-base with $N_L = 12$, $d_{\text{model}} = 768$, intermediate size 3072, and $H = 12$, and BERT-large with $N_L = 24$, $d_{\text{model}} = 1024$, intermediate size 4096, and $H = 16$, where both have $d_h = 64$. We also evaluate GPT2-base, a decoder-only Transformer with causal self-attention [2], which matches BERT-base in N_L, d_{model}, H, and intermediate size. These dimensions determine the packing sizes, boundary tensor shapes, and kernel schedules used in the later sections.

III. Packing-Co-Designed FHE Kernels

This section presents the FHE side of EncFormer. We first define the packing notation and CKKS primitives used by the kernels, then introduce Stage-Compatible Patterns (SCP) as the cross-stage layout discipline, and finally describe the shared projection kernel and the ciphertext–ciphertext attention kernels.

TABLE II
Homomorphic operations on CKKS ciphertexts.

primitive	add	ptmul	ctmul	rot	conj
Inputs	$(\langle\mathbf{v}\rangle, u)$	$(\langle\mathbf{v}\rangle, \mathbf{w})$	$(\langle\mathbf{v}\rangle, \langle\mathbf{w}\rangle)$	$(\langle\mathbf{v}\rangle; r)$	$(\langle\mathbf{v}\rangle)$
Output	$\langle\mathbf{v}+u\rangle$	$\langle\mathbf{v}\odot\mathbf{w}\rangle$	$\langle\mathbf{v}\odot\mathbf{w}\rangle$	$\langle\rho(\mathbf{v};r)\rangle$	$\langle\overline{\mathbf{v}}\rangle$

A key computational characteristic of the Transformer architecture is the attention block in §II-D, whose runtime is dominated by large linear projections and matrix multiplications. Following prior hybrid Transformer systems [8], [10], [15], [16], [32], EncFormer evaluates these linear components under FHE.

A. Packing Model and Stage Compatibility

Let n denote the number of slots in one CKKS ciphertext. We partition the n slots into $N_{\text{seg}} = n/m$ contiguous segments, each containing m slots, so that $mN_{\text{seg}} = n$. We write bold lowercase letters for slot vectors in $\mathbb{C}^n$. For a vector $\mathbf{v} \in \mathbb{C}^n$, we denote its CKKS encryption by $\langle\mathbf{v}\rangle$. For $\mathbf{v}, \mathbf{w} \in \mathbb{C}^n$, we use $\mathbf{v}+\mathbf{w}$ and $\mathbf{v}\odot\mathbf{w}$ for elementwise addition and multiplication, and $\overline{\mathbf{v}}$ for complex conjugation. Let $\rho(\mathbf{v}; r)$ denote the cyclic left shift of $\mathbf{v}$ by r positions, where $r \in \mathbb{Z}$ and indices are taken modulo n.

$$\rho(\mathbf{v}; r) \;=\; (v_r, v_{r+1}, \ldots, v_{n-1}, v_0, \ldots, v_{r-1}).$$

To express segment structure, let $\mathbf{x} \in \mathbb{C}^n$ be a slot vector and index slots by a within-segment coordinate $r \in \{0,\ldots,m-1\}$ and a segment coordinate $s \in \{0,\ldots,N_{\text{seg}}-1\}$. Define the reshape operator $\text{mat} : \mathbb{C}^n \to \mathbb{C}^{m\times N_{\text{seg}}}$ by

$$\big(\text{mat}(\mathbf{x})\big)_{r,s} \;=\; \mathbf{x}_{s\,m+r},$$

and let $\text{vec} : \mathbb{C}^{m\times N_{\text{seg}}} \to \mathbb{C}^n$ be its inverse that stacks columns back into a length n vector. Thus $\text{vec}(\text{mat}(\mathbf{x})) = \mathbf{x}$ and $\text{mat}(\text{vec}(\mathbf{X})) = \mathbf{X}$ for any $\mathbf{X} \in \mathbb{C}^{m\times N_{\text{seg}}}$.

Ciphertext Transformation Operations. Following the transformation notation of Jiang et al. [33], for $\mathbf{X} \in \mathbb{C}^{m\times N_{\text{seg}}}$ we define a segment shift ϕ^Δ and an intra-segment shift ψ^t by

$$\phi^\Delta(\mathbf{X})_{r,s} = \mathbf{X}_{r,\,(s+\Delta)\bmod N_{\text{seg}}},$$
$$\psi^t(\mathbf{X})_{r,s} = \mathbf{X}_{(r+t)\bmod m,\,s}.$$

These shifts induce permutations on slot vectors,

$$\Phi^\Delta(\mathbf{x}) = \text{vec}\big(\phi^\Delta(\text{mat}(\mathbf{x}))\big),$$
$$\Psi^t(\mathbf{x}) = \text{vec}\big(\psi^t(\text{mat}(\mathbf{x}))\big).$$

On ciphertexts, we apply the corresponding slot permutations and write $\Phi^\Delta(\langle\mathbf{x}\rangle) = \langle\Phi^\Delta(\mathbf{x})\rangle$ and $\Psi^t(\langle\mathbf{x}\rangle) = \langle\Psi^t(\mathbf{x})\rangle$. Intuitively, Φ^Δ shifts by whole segments and can be realized as $\mathsf{rot}(\langle\mathbf{x}\rangle; \Delta m)$, while Ψ^t shifts within each length m segment. Concrete realizations can be found in Appendix A-A. When only the first $C < N_{\text{seg}}$ segments are active and wrap-around is required within the active region, we use the restricted segment rotation Φ^Δ_C, also detailed in Appendix A-A. When an object

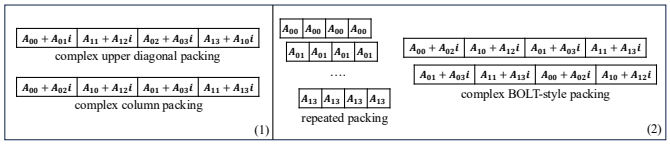


Fig. 4. Packing comparison for a 2×4 matrix with $n = 4$ slots per ciphertext. Minimal packing uses $K_{\min} = 1$ ciphertext, whereas expanded packing uses more than $K_{\min}$ ciphertexts.

X requires multiple ciphertexts, each ciphertext is a *ciphertext block*, and we write $\langle X\rangle = \{\langle \mathbf{x}^{(\ell)}\rangle\}_\ell$. We overload $\Phi^{\Delta}(\cdot)$ and $\Psi^t(\cdot)$ to act elementwise on lists.

Minimal and Expanded Packings. Let x be a real-valued vector with $N(x)$ logical scalar entries, where $N(x)$ excludes repetitions introduced for broadcasting, alignment, or interleaving. Under complex packing, each slot stores two independent real scalars, one in the real part and one in the imaginary part. The natural lower bound on ciphertext count is

$$K_{\min}(x) \;=\; \left\lceil \frac{N(x)}{2n} \right\rceil .$$

A packing of x into K ciphertexts is information-preserving if it stores all $N(x)$ entries without loss. A packing is *minimal* if it is information-preserving and attains $K = K_{\min}(x)$, and *expanded* if it is information-preserving but uses $K > K_{\min}(x)$. Figure 4 illustrates the distinction on an example.

Stage-Compatible Patterns A hybrid FHE–MPC pipeline has two sources of avoidable overhead. First, adjacent FHE kernels may use incompatible packings, forcing an expensive ciphertext remap between them. Second, FHE–MPC boundaries may use more ciphertexts than necessary, inflating conversion communication. Stage-Compatible Patterns address both problems through three rules applied to a canonical packing family $\mathcal{F} = \{\text{segment-column},\ \text{folded-diagonal},\ \text{head-major}\}$.

Each FHE kernel declares which format it accepts and which it produces. The three SCP rules are:

1) **FHE→FHE.** The producer outputs in the format the consumer expects, so no repacking is needed.
2) **MPC→FHE.** Data returning from MPC enters the next FHE kernel in minimal packing.
3) **FHE→MPC.** Data leaving an FHE kernel for MPC is exported in minimal packing.

Rule 1 eliminates FHE-side repacking. Rules 2 and 3 minimize the number of ciphertexts at each conversion boundary, directly reducing communication. The quantitative impact of Rules 2 and 3 on boundary cost is analyzed in §V.

Proposition 1 (Stage-Compatible Patterns)**.** *Under the kernel interfaces of EncFormer, SCP achieves (i) zero FHE remap stages on all edges, and (ii) minimal ciphertext count $K_{\min}$ at every FHE–MPC boundary. Violating Rule 1 on an FHE–FHE edge incurs $\Omega(m \log m)$ extra rotations per ciphertext [23]. Violating Rules 2 or 3 increases boundary conversion count beyond $K_{\min}$. The proof is in Appendix F.*

Table III summarizes rule satisfaction across the six FHE kernels of EncFormer. The per-edge rotation cost incurred when SCP is disabled is reported in Appendix G (Table II).

TABLE III
RULE COVERAGE FOR THE MAIN MATRIX-MULTIPLICATION KERNELS OF ENCFORMER. ✓ INDICATES THAT ENCFORMER SATISFIES THE RULE ON THE CORRESPONDING EDGE. "–" INDICATES THAT THE RULE IS NOT APPLICABLE TO THIS KERNEL.

	QKV Proj	Score	Value	Out Proj	FF1	FF2
Type	pt–ct	ct–ct	ct–ct	pt–ct	pt–ct	pt–ct
Rule 1	✓	✓	✓	✓	–	–
Rule 2	✓	–	✓	–	✓	✓
Rule 3	–	✓	–	✓	✓	✓

B. Plaintext–Ciphertext Projection Kernel

For plaintext–ciphertext multiplication, we use a shared kernel for linear projections $Y = XW$, which instantiates a CKKS plaintext–ciphertext matrix multiplication under segment-column packing using standard baby-step-giant-step (BSGS) optimization [8], [34] and complex-packing optimizations [12].

Let $X \in \mathbb{R}^{m\times d_{\text{in}}}$ and $W \in \mathbb{R}^{d_{\text{in}}\times d_{\text{out}}}$. A CKKS ciphertext has n slots, viewed as $N_{\text{seg}} = n/m$ contiguous segments of length m via $\big(\text{mat}(\mathbf{x})\big)_{r,c} = \mathbf{x}[r + cm]$. We activate the first $C = c_{\text{used}} \le N_{\text{seg}}$ segments so one ciphertext packs C columns. We group input channels into $G = \lceil d_{\text{in}}/C\rceil$ ciphertexts and output channels into $B_{\text{out}} = \lceil d_{\text{out}}/C\rceil$ ciphertexts.

For each $g = 0,\dots,G-1$, we encrypt $\langle \mathbf{x}^{(g)}\rangle$ such that segment $c \in \{0,\dots,C-1\}$ stores the length-m column $X[:, gC + c]$. For each $b = 0,\dots,B_{\text{out}} - 1$, the output ciphertext $\langle \mathbf{y}^{(b)}\rangle$ stores columns $Y[:, bC],\dots,Y[:, bC+C-1]$ in the same layout. Projection weights are encoded as CKKS plaintext vectors as detailed in Appendix A-B. We pre-permute plaintext weight columns so each projection emits the order consumed by its downstream kernel directly: Q and K use the score-friendly segment order, while V uses the value-friendly head-major order.

Complex Pairs Compression. We pack two real input groups into one complex ciphertext by setting $U = \lceil G/2\rceil$ and, for each $u \in 0,\dots,U-1$, defining the complexified input $\langle \tilde{\mathbf{x}}^{(u)}\rangle$ as

$$\langle \tilde{\mathbf{x}}^{(u)}\rangle = \left\langle \mathbf{x}^{(2u)} + i\,\mathbf{x}^{(2u+1)} \right\rangle .$$

We call this packing operation complexification and call the reverse operation decomplexification.

BSGS Optimization. Choose $N_1 \mid C$ and set $N_2 = C/N_1$. Let Φ_C denote a segment cyclic shift within the first C active segments. For each u and $q \in \{0,\dots,N_1-1\}$ define $\langle \tilde{\mathbf{x}}_q^{(u)}\rangle = \Phi_C^q(\langle \tilde{\mathbf{x}}^{(u)}\rangle)$. For each output block b and giant index $p \in \{0,\dots,N_2-1\}$ we accumulate a ciphertext

$$\langle \tilde{\mathbf{c}}_p^{(b)}\rangle = \sum_{q=0}^{N_1-1}\sum_{u=0}^{U-1} \left(\langle \tilde{\mathbf{x}}_q^{(u)}\rangle \odot \tilde{\mathbf{w}}_{u,p,q}^{(b)}\right),$$

We then extract the real part of each giant-step accumulator and combine the N_2 partial sums with segment shifts to obtain $\langle \mathbf{y}^{(b)}\rangle$:

$$\langle \mathbf{c}_p^{(b)} \rangle = \tfrac{1}{2}\Big(\langle \tilde{\mathbf{c}}_p^{(b)} \rangle + \mathsf{conj}(\langle \tilde{\mathbf{c}}_p^{(b)} \rangle)\Big),$$
$$\langle \mathbf{y}^{(b)} \rangle = \sum_{p=0}^{N_2-1} \Phi_C^{pN_1}\Big(\langle \mathbf{c}_p^{(b)} \rangle\Big).$$

Appendix A-B also describes an optional shortcut that fuses two projections that share the same input. The projection kernel satisfies SCP Rule 1 because it outputs in the segment-column packing consumed by the score kernel, and Rule 2 because MPC-produced inputs are imported into CKKS in minimal packing and consumed directly as $\{\langle \mathbf{x}^{(g)} \rangle\}$.

C. Ciphertext–Ciphertext Attention Kernels

Transformer attention is head-separable and exhibits a token-shift reuse pattern that we exploit to reduce key-switching operations relative to generic ciphertext–ciphertext matrix multiplication. For each head h, we compute the score matrix $S^{(h)} = Q^{(h)}(K^{(h)})^\top \in \mathbb{R}^{m\times m}$ with a score kernel and the attention output $O^{(h)} = P^{(h)}V^{(h)} \in \mathbb{R}^{m\times d_h}$ with a value kernel, where $P^{(h)}$ is the post-softmax attention matrix.

Both kernels reuse the same small sets of segment shifts and intra-segment shifts on each ciphertext block, so we precompute rotation banks to avoid redundant key-switching operations. For block ℓ with input $\langle \mathbf{x}^{(\ell)} \rangle$, let $\mathcal{D}, \mathcal{T} \subset \mathbb{Z}$ be the required segment and intra-segment offsets. We precompute

$$\mathsf{Bank}_\Phi\Big(\langle \mathbf{x}^{(\ell)} \rangle, \mathcal{D}\Big) = \Big\{\Phi^\delta\Big(\langle \mathbf{x}^{(\ell)} \rangle\Big)\Big\}_{\delta\in\mathcal{D}},$$
$$\mathsf{Bank}_\Psi\Big(\langle \mathbf{x}^{(\ell)} \rangle, \mathcal{T}\Big) = \Big\{\Psi^t\Big(\langle \mathbf{x}^{(\ell)} \rangle\Big)\Big\}_{t\in\mathcal{T}}.$$

When only the first $C < N_{\text{seg}}$ segments are active, we use Φ_C^δ so that wrap-around stays within the active region. The inner loops then use bank lookups with ptmul, add, and ctmul, and all rotations occur only during bank construction.

Score Kernel. Inputs are $\langle Q \rangle$ and $\langle K \rangle$ in segment-column packing, where $\langle Q \rangle = \{\langle \mathbf{q}^{(\ell)} \rangle\}_\ell$ and $\langle K \rangle = \{\langle \mathbf{k}^{(\ell)} \rangle\}_\ell$ encrypt Q and K. The output is a *folded-diagonal* ciphertext list $\{\langle \mathbf{S}_t \rangle\}_{t=0}^{m/2-1}$. Folded-diagonal packing compresses two attention diagonals into one ciphertext: for each head h, define the cyclic diagonal vector at offset Δ as

$$s_\Delta^h[j] = S^h_{j,\,(j+\Delta) \bmod m}, \quad j = 0,\dots,m-1.$$

The offset pair $(t, t+m/2)$ is stored by placing their diagonals into the real and imaginary parts of a single vector:

$$v_t^h[j] = s_t^h[j] + i\, s_{t+m/2}^h[j], \quad t = 0,\dots,m/2-1.$$

We assign one length-m segment per head inside each ciphertext, with heads stacked across segments, and place different values of t across ciphertext blocks. Figure 5 illustrates this layout in the kernel procedures.

Choose $\beta \mid m$ and set $g = m/\beta$ with g even. We compute each folded-diagonal pair by reusing a fixed set of intra-segment shifts on Q and K. For each $t \in \{0,\dots,m/2-1\}$, write $t = j(t)\beta + s(t)$ with $j(t) = \lfloor t/\beta \rfloor$ and $s(t) = t \bmod \beta$. Pairing $(t, t+m/2)$ matches folded-diagonal packing, so each t is produced by one complex multiply between a Q shift and a complexified pair of K shifts.

For each ciphertext block ℓ, we precompute a Q bank over the offsets

$$\mathcal{T}_Q = \{0, -1, \dots, -(\beta-1)\},$$

where the negative shift aligns the Q row index with the K diagonal offset $s(t)$ in the inner product. We also precompute two K banks over the offsets

$$\mathcal{T}_K = \{j\beta \mid j = 0,\dots,g/2-1\},$$
$$\mathcal{T}_{K,\text{half}} = \{m/2 + j\beta \mid j = 0,\dots,g/2-1\}.$$

This yields

$$\{\langle \mathbf{q}_s^{(\ell)} \rangle\}_s := \mathsf{Bank}_\Psi\big(\langle \mathbf{q}^{(\ell)} \rangle, \mathcal{T}_Q\big),$$
$$\{\langle \mathbf{k}_\tau^{(\ell)} \rangle\}_\tau := \mathsf{Bank}_\Psi\big(\langle \mathbf{k}^{(\ell)} \rangle, \mathcal{T}_K \cup \mathcal{T}_{K,\text{half}}\big),$$

Using these banks, we form one folded-diagonal pair per t,

$$\langle \mathbf{u}_t \rangle = \sum_\ell \Big(\langle \mathbf{q}_{-s(t)}^{(\ell)} \rangle \odot \big(\langle \mathbf{k}_{j(t)\beta}^{(\ell)} \rangle + i\,\langle \mathbf{k}_{m/2+j(t)\beta}^{(\ell)} \rangle\big)\Big).$$

Finally, we aggregate the C active segments into the first H head segments and align the intra-segment offset. Let $\{\mathbf{m}_c\}_{c=0}^{C-1}$ be disjoint binary segment masks, where $\mathbf{m}_c$ keeps exactly the c-th active segment. This produces $\langle \mathbf{S}_t \rangle$ in folded-diagonal packing,

$$\langle \mathbf{S}_t \rangle = \Psi^{s(t)}\left(\sum_{c=0}^{C-1} \Phi^{(c \bmod H)-c}\big(\langle \mathbf{u}_t \rangle \odot \mathbf{m}_c\big)\right).$$

When $C \not\equiv 0 \pmod H$, different ciphertext blocks start at different head-phase offsets, requiring a block-dependent phase correction before combining. Mask construction and the full phase-correction procedure are given in Appendix A-C. The equations above describe the core accumulation without this correction for clarity.

Value Kernel. Let H_{blk} be the number of heads packed in one value ciphertext block and let $B_V = \lceil H/H_{\text{blk}} \rceil$. The CKKS inputs are the folded-diagonal softmax ciphertext list

$$\langle P_{\text{fd}} \rangle = \{\langle \mathbf{p}_{\text{fd}}^{(\ell)} \rangle\}_{\ell=0}^{B_V-1}$$

and the value ciphertext list

$$\langle V \rangle = \{\langle \mathbf{v}^{(\ell)} \rangle\}_{\ell=0}^{B_V-1}.$$

Each value ciphertext $\langle \mathbf{v}^{(\ell)} \rangle$ uses *head-major* packing: segment $k(\tilde{h}, u) = \tilde{h}\, d_h + u$ stores $V^{(\ell,\tilde{h})}[:,u] \in \mathbb{R}^m$ for $\tilde{h} \in \{0,\dots,H_{\text{blk}}-1\}$ and $u \in \{0,\dots,d_h-1\}$. Thus, the d_h channels of each local head are contiguous, and different heads occupy disjoint blocks of d_h segments. The output is the head-major ciphertext list

$$\langle O_{\text{hm}} \rangle = \{\langle \mathbf{o}_{\text{hm}}^{(\ell)} \rangle\}_{\ell=0}^{B_V-1}$$

which encodes $O = PV \in \mathbb{R}^{m\times d}$. The subsequent output projection absorbs the head-major-to-segment-column conversion by pre-permuting its plaintext weight columns, as detailed in Appendix A-B, so no explicit FHE remap is needed.

For each block ℓ, we reuse two banks. First, we complexify the value block and build a Ψ bank over token offsets:

$$\langle \mathbf{u}^{(\ell)} \rangle = \langle \mathbf{v}^{(\ell)} \rangle - i\,\Psi^{m/2}\big(\langle \mathbf{v}^{(\ell)} \rangle\big),$$
$$\{\langle \mathbf{u}_t^{(\ell)} \rangle\}_{t=0}^{m/2-1} = \mathsf{Bank}_\Psi\big(\langle \mathbf{u}^{(\ell)} \rangle, \{0,\dots,m/2-1\}\big).$$

Second, let $\mathbf{n}_u$ be the binary segment mask that keeps exactly channel offset u across the H_{blk} local heads in one block (Appendix A-C). For each $t \in \{0, \dots, m/2-1\}$, we align and broadcast the folded-diagonal weights by

$$\langle \mathbf{b}_t^{(\ell)} \rangle = \sum_{u=0}^{d_h-1} \left(\Phi^{t-u}\big(\langle \mathbf{p}_{\text{fd}}^{(\ell)} \rangle\big) \odot \mathbf{n}_u \right).$$

The block output is then

$$\langle \mathbf{o}_{\text{hm}}^{(\ell)} \rangle = \sum_{t=0}^{m/2-1} \big(\langle \mathbf{u}_t^{(\ell)} \rangle \odot \langle \mathbf{b}_t^{(\ell)} \rangle \big), \qquad \ell = 0, \dots, B_V - 1.$$

Each $\langle \mathbf{o}_{\text{hm}}^{(\ell)} \rangle$ remains in head-major layout, and the output projection consumes this list directly via plaintext weight pre-permutation.

Putting It Together. The score kernel consumes the segment-column packed Q and K from the projection kernel (SCP Rule 1) and exports the scores into $K_{\min}(S) = \lceil Hm^2/(2n) \rceil$ ciphertexts in minimal folded-diagonal packing at the FHE–MPC boundary (Rule 3). The internal per-t representation $\{\langle \mathbf{S}_t \rangle\}$ is repacked into a sequential minimal stream before conversion. See Appendix A-C. The MPC softmax block receives this stream, converts it to additive shares via the CKKS-to-MPC protocol, evaluates Π_{MBMax} securely, and converts the result back to CKKS in minimal folded-diagonal packing ($K_{\min}(P) = \lceil Hm^2/(2n) \rceil$ ciphertexts, Rule 2). The value kernel then combines the folded-diagonal softmax list $\langle P_{\text{fd}} \rangle$ with the head-major value list $\{\langle \mathbf{v}^{(\ell)} \rangle\}_\ell$ without FHE-side repacking, and outputs $\langle O_{\text{hm}} \rangle$, which the output projection consumes directly through pre-permuted weight columns (Rule 1). This keeps the MPC–FHE boundary at minimal ciphertext count while moving the broadcast into the FHE kernel where it composes cleanly with the value layout. Figure 5 illustrates both attention kernels on a toy instance. The top row shows the score kernel producing folded-diagonal score ciphertexts $\{\langle \mathbf{S}_t \rangle\}$ that are exported to MPC for softmax, and the bottom row shows the value kernel consuming the folded-diagonal softmax ciphertext list $\langle P_{\text{fd}} \rangle$ together with the head-major value ciphertext list $\{\langle \mathbf{v}^{(\ell)} \rangle\}_\ell$ to output the head-major ciphertext list $\langle O_{\text{hm}} \rangle$ encoding $O = PV$.

Table IV reports a matched BERT-base comparison of attention key-switching costs. For each prior system, we instantiate the published attention algorithm under the same tensor shapes ($m = 128$, $d = 768$, $H = 12$, $n = 16384$) and count rotations and ciphertext multiplications. The EncFormer counts are measured from our implementation. The latency columns apply measured PhantomFHE A100 primitive latencies to these matched operation counts, giving measured-backend costs for the common BERT-base workload. Appendix A-C provides the asymptotic derivations.

TABLE IV
KEY-SWITCHING COUNTS AND MEASURED-BACKEND LATENCY FOR BERT-BASE ATTENTION

	$Q^{(h)}(K^{(h)})^{\mathsf{T}}$			$P^{(h)}V^{(h)}$		
	#rot	#mul	latency (s)	#rot	#mul	latency (s)
BOLT [8]	13824	1536	36.1	21420	768	51.7
Powerformer [12]	4392	768	12.2	4882	1536	15.4
BLB [10]	512	768	3.2	824	1536	6.0
EncFormer	630	448	2.6	1524	384	4.6

Algorithm 1 Secure $\Pi_{\text{MBMax}}([[\mathbf{X}]])$

Require: P_0, P_1 hold $[[\mathbf{X}]]_{2^\ell}$ for $\mathbf{X} \in \mathbb{Z}_{2^\ell}^{m \times m}$ at scale 2^F, public scalars c and R_d
Ensure: P_0, P_1 obtain $[[\mathbf{Y}]]_{2^\ell}$ where $\mathbf{Y} = (\mathbf{X} + c)^5 / R_d$ applied elementwise
1: $[[\bar{\mathbf{X}}]]_{2^\ell} \leftarrow [[\mathbf{X}]]_{2^\ell} + c$
2: $[[\bar{\mathbf{X}}^2]]_{2^\ell} \leftarrow \Pi_\times\big([[\bar{\mathbf{X}}]]_{2^\ell}, [[\bar{\mathbf{X}}]]_{2^\ell}\big)$
3: $[[\bar{\mathbf{X}}^4]]_{2^\ell} \leftarrow \Pi_\times\big([[\bar{\mathbf{X}}^2]]_{2^\ell}, [[\bar{\mathbf{X}}^2]]_{2^\ell}\big)$
4: $[[\bar{\mathbf{X}}^5]]_{2^\ell} \leftarrow \Pi_\times\big([[\bar{\mathbf{X}}^4]]_{2^\ell}, [[\bar{\mathbf{X}}]]_{2^\ell}\big)$
5: $[[\mathbf{Y}]]_{2^\ell} \leftarrow (1/R_d) \cdot [[\bar{\mathbf{X}}^5]]_{2^\ell}$
6: **Output** $[[\mathbf{Y}]]_{2^\ell}$

IV. COMMUNICATION-EFFICIENT MPC BLOCKS

A. MPC Fixed-point Primitives

§II-A defines the fixed-point representation, additive-share notation, truncating multiplication, comparison, multiplexing, row-sum, and broadcast primitives used below. This section focuses on how those primitives are composed into Transformer nonlinearities.

B. Protocols for Non-linear Layers

For Softmax and LayerNorm, EncFormer realizes the batch nonlinearities as MPC protocols over additive shares. For GELU, the MPC nonlinear block is BOLT's additive-share piecewise GELU protocol [8]. Expressed as short chains of MPC primitives, these nonlinear blocks use 7 interactive MPC rounds per Transformer layer, with 3 rounds for Π_{MBMax}, 4 rounds for Π_{GELU}, and 0 rounds for Π_{MBLN} because Π_{MBLN} reduces to local operations on additive shares. Appendix E compares these counts against prior systems.

MPC Π_{MBMax} Protocol. We replace Softmax with the Batch Power-Max approximation from Powerformer using a power map and a fixed public normalizer. A single public constant R_d is distilled offline and reused during inference together with the public shift c and exponent $p = 5$. The approximation is

$$\mathbf{Y} \;=\; \frac{(\mathbf{X} + c)^p}{R_d}, \qquad p = 5, \tag{1}$$

applied elementwise. Algorithm 1 realizes Π_{MBMax} using a short multiplication chain and a final multiplication by the public fixed-point constant $1/R_d$.

MPC Π_{MBLN} Protocol. We adopt the Batch LayerNorm approximation from Powerformer, which replaces variance-dependent normalization with a fixed public scale. For a row $x \in \mathbb{R}^d$,

$$y \;=\; \gamma \odot \frac{x - \mu}{l \cdot R_d} + \beta, \qquad \mu = \text{RowMean}(x). \tag{2}$$

We fold the constant scale into $\tilde{\gamma} = \gamma/(l \cdot R_d)$ so that Π_{MBLN} reduces to computing the row mean, centering the row, and applying a public elementwise affine map.

MPC Π_{GELU} Protocol. We evaluate GELU using the piecewise approximation and secure selection protocol from BOLT [8], following the standard hybrid split used by

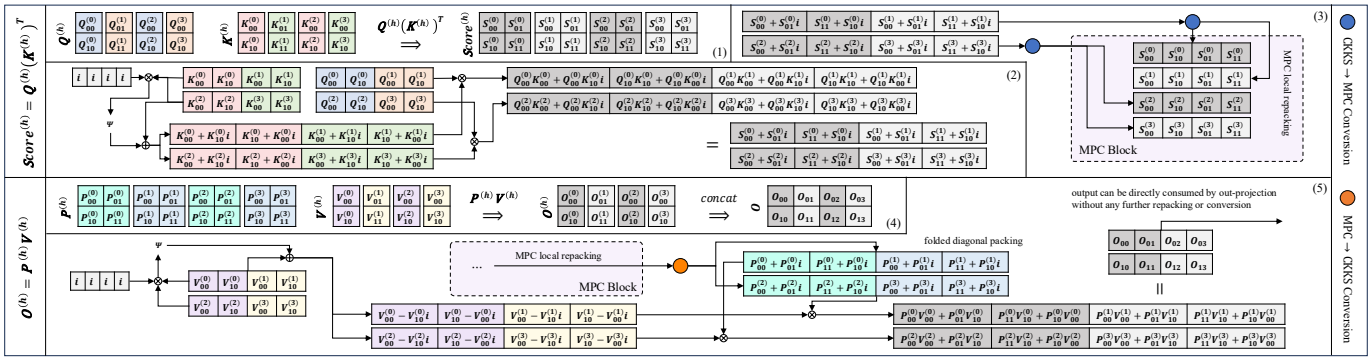


Fig. 5. Toy example of the attention kernels with $H = 4$, $m = 2$, $d_h = 1$, and $n = 4$ slots. Each ciphertext contains $N_{\text{seg}} = 2$ segments. The top half shows the score kernel from segment-packed Q and K to folded-diagonal S, followed by MPC softmax that returns encrypted folded-diagonal P in minimal packing. The bottom half shows the value kernel from folded-diagonal P and head-major V to the head-major output $O = PV$. (1) and (4) show the matrix view. (2), (3), and (5) illustrate the procedure at the slot level.

Algorithm 2 Secure $\Pi_{\text{MBLN}}([[\mathbf{X}]])$

Require: P_0, P_1 hold $[[\mathbf{X}]]_{2^\ell}$ for $\mathbf{X} \in \mathbb{Z}_{2^\ell}^{T\times d}$ at scale 2^F, public $\tilde{\gamma}, \beta \in \mathbb{Z}_{2^\ell}^{d}$ with $\tilde{\gamma} = \gamma/(l \cdot R_d)$

Ensure: P_0, P_1 obtain $[[\mathbf{Y}]]_{2^\ell}$ where each row satisfies $y = \tilde{\gamma} \odot (x - \mu) + \beta$

$[[\mathbf{s}]]_{2^\ell} \leftarrow \Pi_{\text{rowsum}}([[\mathbf{X}]]) \in \mathbb{Z}_{2^\ell}^{T\times 1}$

$[[\boldsymbol{\mu}]]_{2^\ell} \leftarrow (1/d) \cdot [[\mathbf{s}]]_{2^\ell} \in \mathbb{Z}_{2^\ell}^{T\times 1}$

$[[\mathbf{Z}]]_{2^\ell} \leftarrow [[\mathbf{X}]]_{2^\ell} - \Pi_{\text{bcast}}([[\boldsymbol{\mu}]]_{2^\ell}, d) \in \mathbb{Z}_{2^\ell}^{T\times d}$

$[[\mathbf{Y}]]_{2^\ell} \leftarrow \tilde{\gamma} \odot [[\mathbf{Z}]]_{2^\ell} + \beta$

Output $[[\mathbf{Y}]]_{2^\ell}$

BLB [10]. We denote the resulting MPC block by Π_{GELU}. Concretely, CKKS forms the two candidate ciphertexts $F_0(\mathbf{x})$ and $F_1(\mathbf{x})$, and MPC selects among $\{0, F_0(\mathbf{x}), F_1(\mathbf{x}), \mathbf{x}\}$ using Π_{cmp} and Π_{mux}. The expanded GELU boundary converts both $F_0(\mathbf{x})$ and $F_1(\mathbf{x})$ and therefore increases the inbound conversion count for the GELU block. The boundary decision rule is specified in §V-A, and Appendix B gives the full protocol details.

V. Boundary Co-Design and Secure Conversion

In EncFormer, boundary design is part of the system architecture rather than a post-processing detail. We therefore pair a cost model for boundary placement with a secure complex CKKS–MPC conversion protocol that reduces payload while preserving compatibility with real-valued MPC blocks.

Positioning Relative to BLB. BLB's released implementation applies complex packing at FHE–MPC boundaries [10] without formalizing the protocol or linking it to modulus trimming. §V-B formalizes the CC-everywhere convention as the baseline for the cost-model rule in §V-A. The contribution targets boundary payload and interaction rather than introducing a new nonlinear subprotocol.

A. Cost Model for Boundary Design

Conversion overhead grows with the number of CKKS–MPC conversion pairs and with the total ciphertext payload converted at the FHE–MPC boundaries. Both factors depend on the boundary packing and on CKKS parameters that determine the ciphertext size.

Fix a hybrid inference decomposition with R MPC blocks indexed by $r = 1, \ldots, R$. Block r consumes a complex-valued boundary tensor $x_r^{\rightarrow}$ and produces a complex-valued boundary tensor $x_r^{\leftarrow}$. A hybrid design chooses a packing for every $x_r^{\rightarrow}$ and $x_r^{\leftarrow}$ and specifies where conversions occur.

Baseline Conversions and Decision Rule We define the minimal-conversion baseline design B by two properties. First, B performs exactly one entry conversion CKKS→MPC and one exit conversion MPC→CKKS for each MPC block, and performs no other conversions. Therefore, B uses exactly R conversion pairs, which is the minimum possible for a fixed set of R MPC blocks. Second, B uses minimal packing, defined in §III-A, at every boundary tensor so that each $x_r^{\rightarrow}$ and $x_r^{\leftarrow}$ is represented using exactly $K_{\min}(x)$ ciphertexts. In a standard Transformer encoder layer, the nonlinearities consist of the attention softmax, two layer normalizations, and the feed-forward GELU activation (Figure 1). Accordingly, we decompose each layer into $R = 4$ MPC blocks (Softmax, LN1, GELU, LN2), so the baseline uses exactly four FHE↔MPC conversion pairs per layer.

However, deviating from B can still be beneficial because expanded packings or additional conversions can reduce CKKS-side or MPC-side work by enabling layouts and kernels with fewer key-switched operations. These choices increase conversion overhead by increasing either the number of conversion pairs or the number of ciphertexts converted. Prior works do not provide an analytic criterion for when a non-minimal conversion strategy is worthwhile. We model this trade-off explicitly and use it to decide when moving away from B is worthwhile. Figure 6 summarizes the design space.

We write end-to-end latency as

$$T_{\text{total}} = T_{\text{conv}} + T_{\text{ckks}} + T_{\text{mpc}}. \tag{3}$$

We treat T_{conv} as the aggregate boundary-conversion latency induced by the chosen conversion placement and boundary packing. Thus $T_{\text{conv}}(D)$ captures both the number of conversion pairs in design D and the total ciphertext payload converted at the boundaries.

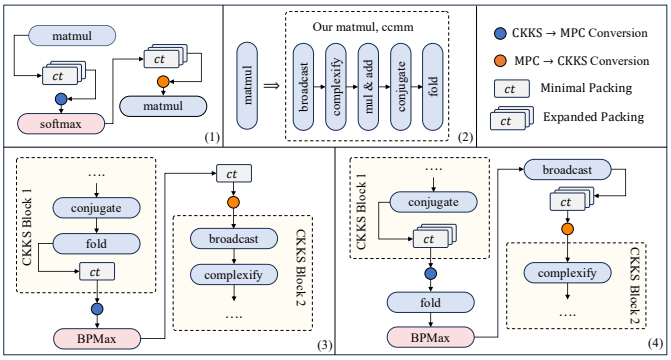


Fig. 6. Architecture comparison for hybrid inference. (1) Prior layerwise hybrid pipeline with expanded packing. (2) Component view of a plaintext–ciphertext projection kernel. (3) Minimal baseline using minimal packing at conversions. (4) EncFormer pre-evaluates GELU polynomials in CKKS with expanded boundary packing for lower MPC cost.

Let B denote the minimal-conversion baseline and D a candidate design. Define

$$\Delta T_{\text{conv}} = T_{\text{conv}}(D) - T_{\text{conv}}(B),$$
$$\Delta T_{\text{comp}} = \big(T_{\text{ckks}}(D) + T_{\text{mpc}}(D)\big) - \big(T_{\text{ckks}}(B) + T_{\text{mpc}}(B)\big).$$

We prefer D if

$$\Delta T_{\text{conv}} + \Delta T_{\text{comp}} < 0. \tag{4}$$

This condition states that any increase in conversion overhead under D is acceptable only when it is more than offset by a reduction in CKKS-side and MPC-side latency, resulting in a net decrease in end-to-end latency.

Parameterization We parameterize ΔT_{conv} and ΔT_{comp} in terms of CKKS parameters, network profile, the total CKKS compute time difference $\Delta T_{\text{ckks}} = T_{\text{ckks}}(B) - T_{\text{ckks}}(D)$, and MPC round savings. Let K_{extra} denote the additional ciphertexts converted beyond the baseline, R_{extra} the additional conversion round-trips, $\text{ct\_bytes}(N, L) = 2NL \cdot 8$ the serialized ciphertext size for ring degree N with L RNS limbs, and BW, RTT the network bandwidth and round-trip time. Then

$$\Delta T_{\text{conv}} = \frac{K_{\text{extra}} \cdot \text{ct\_bytes}(N, L)}{\text{BW}} + R_{\text{extra}} \cdot \text{RTT}, \tag{5}$$

$$\Delta T_{\text{comp}} \approx -\big(\Delta T_{\text{ckks}} + R_{\text{saved}} \cdot (\text{RTT} + C_{\text{round}}/\text{BW})\big), \tag{6}$$

where ΔT_{ckks} is the total CKKS compute time saved by D relative to B (positive when D reduces FHE work), R_{saved} is the number of MPC rounds saved, and C_{round} is the MPC communication per round in bytes. The per-round communication depends on the MPC backend: for Beaver-triple multiplication on E elements with ring bit-width ℓ, $C_{\text{round}} = E \cdot 4\lceil \ell/8 \rceil$.

Eqs. (5)–(6) can be used to estimate the sign of $\Delta T_{\text{conv}} + \Delta T_{\text{comp}}$ and guide boundary placement. We treat them as a screening tool rather than a precise predictor, since ΔT_{ckks} and C_{round} are measured under specific deployment conditions. Appendices D and H provide further discussion and validation details.

The rule's outcome at the GELU boundary depends on the CKKS backend through ΔT_{ckks}. We therefore report both boundary configurations explicitly: the PhantomFHE main pipeline uses the expanded-GELU boundary configuration for the corresponding evaluation rows, while the matched LiberateFHE pipeline uses the minimal-GELU boundary configuration. The corresponding ciphertext-count comparison is reported in Table V and discussed with the end-to-end results in §VII-B.

Algorithm 3 Complex CKKS-to-MPC Conversion $\Pi^{\mathbb{C}}_{\text{C2M}}$

Require: P_1 holds $\langle \mathbf{m} \rangle \in A^2_{N,Q_{\text{conv}}}$ at scale Δ
Ensure: $\forall b \in \{0,1\}$, P_b outputs $[[\mathbf{x}]]^b_{2^\ell}$ and $[[\mathbf{y}]]^b_{2^\ell}$
1: P_1 sample $\hat{\mathbf{r}} \leftarrow R_{Q_{\text{conv}}}$ uniformly, send $\langle \mathbf{d} \rangle \leftarrow \langle \mathbf{m} \rangle + \hat{\mathbf{r}}$ to P_0, and set $[[\hat{\mathbf{t}}]]^{Q_{\text{conv}}}_1 \leftarrow -\hat{\mathbf{r}}$
2: P_0 decrypt to get $[[\hat{\mathbf{t}}]]^{Q_{\text{conv}}}_0 \leftarrow \text{Dec}(\langle \mathbf{d} \rangle)$
3: P_0, P_1 invoke $[[\hat{\mathbf{t}}]]^{2^\ell} \leftarrow \Pi_{\text{Field2Ring}}\big([[\hat{\mathbf{t}}]]^{Q_{\text{conv}}}\big)$
4: $\forall b \in \{0,1\}$ P_b decode $\mathbf{z}_b \leftarrow \text{Decode}([[\hat{\mathbf{t}}]]^{2^\ell}_b)$ and output $[[\mathbf{x}]]^b_{2^\ell} \leftarrow \Re(\mathbf{z}_b)$ and $[[\mathbf{y}]]^b_{2^\ell} \leftarrow \Im(\mathbf{z}_b)$

B. Complex Conversion Protocol

Formalization. Prior hybrid implementations use complex packing at CKKS–MPC boundaries [10]. We make this boundary representation explicit as a protocol interface for EncFormer. Algorithms 3 and 4 specify the two conversion directions. Theorem 1 and Appendix L give the semi-honest security argument. Equation (5) quantifies the communication cost. The same protocol applies to complex-packed kernels in §III-C and real-only kernels in §III-B by pairing two real boundary tensors into one complex ciphertext.

At an FHE–MPC boundary, we convert between a CKKS ciphertext $\langle \mathbf{m} \rangle$ at scale Δ and MPC additive shares over $\mathbb{Z}_{2^\ell}$. Using the fixed-point convention from §II-A, decryption and decoding yield a slot vector $\mathbf{x} + i\mathbf{y} \in \mathbb{C}^n$ where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ fit in the signed range of $\mathbb{Z}_{2^\ell}$. Following BLB [10], we sample randomness directly in the polynomial ring modulo the coefficient modulus and then apply the CKKS encoding map, rather than sampling real numbers and encoding them. We write $A^2_{N,q}$ for the CKKS ciphertext space with ring degree N and modulus q and we set $q = Q_{\text{conv}}$ at the boundary.

Complex CKKS-to-MPC Conversion Let Q_{conv} be the ciphertext modulus at the boundary. Given $\langle \mathbf{m} \rangle$ from P_1 that encrypts an encoding of $\mathbf{x} + i\mathbf{y}$, the conversion outputs ring shares $[[\mathbf{x}]]_{2^\ell}$ and $[[\mathbf{y}]]_{2^\ell}$. P_1 masks the boundary plaintext by sampling a uniform $\hat{\mathbf{r}} \leftarrow R_{Q_{\text{conv}}}$ and sending $\langle \mathbf{m} \rangle + \hat{\mathbf{r}}$ for decryption, so P_0 decrypts a uniformly masked plaintext-ring element. Protocol $\Pi^{\mathbb{C}}_{\text{C2M}}$ in Algorithm 3 realizes this conversion.

The two plaintext shares in Steps 1 and 2 reconstruct the unmasked boundary plaintext modulo Q_{conv} because $(\hat{\mathbf{p}} + \hat{\mathbf{r}}) + (-\hat{\mathbf{r}}) \equiv \hat{\mathbf{p}}$ where $\hat{\mathbf{p}} = \text{Encode}(\mathbf{x} + i\mathbf{y}, \Delta)$. By correctness of $\Pi_{\text{Field2Ring}}$ in Appendix C, Step 3 preserves the same signed fixed-point plaintext in $\mathbb{Z}_{2^\ell}$ except with negligible statistical error. CKKS decoding is a linear map from the plaintext ring

Algorithm 4 Complex MPC-to-CKKS Conversion $\Pi^{\mathbb{C}}_{\mathrm{M2C}}$

Require: MPC shares $[[\mathbf{x}]]_{2^\ell}, [[\mathbf{y}]]_{2^\ell}$ and scale Δ with boundary modulus Q_{conv}
Ensure: P_1 outputs $\langle \mathbf{m} \rangle \in A^2_{N,Q_{\mathrm{conv}}}$ at scale Δ
1: $\forall b \in \{0,1\}$ P_b center lift to $\tilde{\mathbf{x}}_b, \tilde{\mathbf{y}}_b$ and encode $[[\hat{\mathbf{t}}]]_b^{2^\ell} \leftarrow \mathrm{Encode}(\tilde{\mathbf{x}}_b + i\tilde{\mathbf{y}}_b, \Delta)$
2: P_0, P_1 invoke $[[\hat{\mathbf{t}}]]^{Q_{\mathrm{conv}}} \leftarrow \Pi_{\mathrm{Ring2Field}}\left([[\hat{\mathbf{t}}]]^{2^\ell}\right)$
3: P_0 encrypt and send $\langle \mathbf{c} \rangle \leftarrow \mathrm{Enc}([[\hat{\mathbf{t}}]]_0^{Q_{\mathrm{conv}}})$ to P_1
4: P_1 output $\langle \mathbf{m} \rangle \leftarrow \langle \mathbf{c} \rangle + [[\hat{\mathbf{t}}]]_1^{Q_{\mathrm{conv}}}$

to the slot space, so decoding each additive plaintext share locally in Step 4 yields additive shares of the decoded vector. Therefore, the outputs reconstruct $\mathbf{x} + i\mathbf{y}$ up to the standard CKKS decoding error. If the per-slot decoding error is below $1/2$ in both real and imaginary parts at the boundary scale, rounding recovers the intended integers in $\mathbb{Z}_{2^\ell}$. We empirically verify this condition on PhantomFHE with $n = 2^{14}$, scale $\Delta = 2^{40}$, and 40-bit body primes. The maximum per-slot absolute error after one encrypt–decode round-trip is $\sim 9 \times 10^{-6}$, well below $1/2$.

Complex MPC-to-CKKS Conversion Given ring shares $[[\mathbf{x}]]_{2^\ell}$ and $[[\mathbf{y}]]_{2^\ell}$, the conversion outputs a CKKS ciphertext $\langle \mathbf{m} \rangle$ at scale Δ that decrypts and decodes to a value close to $\mathbf{x} + i\mathbf{y}$. We mirror prior real-only conversions by locally encoding the two share vectors as plaintext ring shares and then switching these plaintext shares from $\mathbb{Z}_{2^\ell}$ to $\mathbb{Z}_{Q_{\mathrm{conv}}}$ using $\Pi_{\mathrm{Ring2Field}}$. Protocol $\Pi^{\mathbb{C}}_{\mathrm{M2C}}$ in Algorithm 4 realizes this conversion by encrypting one plaintext share and adding the other plaintext share locally.

By construction, the two encoded plaintext shares in Step 1 add to an encoding of $\mathbf{x}+i\mathbf{y}$ under the signed fixed-point interpretation in $\mathbb{Z}_{2^\ell}$. By correctness of $\Pi_{\mathrm{Ring2Field}}$ in Appendix C, Step 2 converts these plaintext shares to $R_{Q_{\mathrm{conv}}}$ while preserving their sum modulo Q_{conv} except with negligible statistical error. After Step 4, the ciphertext encrypts the sum of the two plaintext shares, so decryption and decoding recover $\mathbf{x}+i\mathbf{y}$ up to the standard CKKS decoding error. If the per-slot decoding error is below $1/2$ in both parts at the boundary scale, a subsequent CKKS-to-MPC conversion recovers the intended integers in $\mathbb{Z}_{2^\ell}$ under rounding.

Security of Complex Conversion We consider semi-honest adversaries. In $\Pi^{\mathbb{C}}_{\mathrm{C2M}}$, P_0 receives a masked ciphertext and decrypts a plaintext that is uniform in $R_{Q_{\mathrm{conv}}}$ because $\hat{\mathbf{r}}$ is uniform and unknown to P_0, so the decryption result is statistically independent of the encoded message. In $\Pi^{\mathbb{C}}_{\mathrm{M2C}}$, P_1 receives only a CKKS ciphertext encrypting P_0's plaintext share, which reveals no information by semantic security, and then adds its own plaintext share locally. By composition, if $\Pi_{\mathrm{Field2Ring}}$ and $\Pi_{\mathrm{Ring2Field}}$ are secure, then the conversion protocols leak nothing beyond their prescribed outputs and public parameters.

Communication Overhead and Compatibility A CKKS ciphertext has n complex slots, corresponding to $2n$ real values across the real and imaginary parts. A real-only conversion exports a single part and therefore transmits one ciphertext per real vector per direction. Our complex conversion exports both parts so one ciphertext carries two independent real vectors, reducing the number of ciphertexts transmitted per direction from k to $\lceil k/2 \rceil$ for k real vectors. The conversion outputs two real share vectors, so existing MPC protocols for real-valued nonlinearities apply by processing the two parts independently.

TABLE V
CT COUNTS PER MPC BLOCK. THE ENCFORMER-EXP AND ENCFORMER-MIN CONFIGURATIONS ARE SPECIFIED IN §VI AND EVALUATED IN §VII-B.

Method	MPC Blocks										Total
	Softmax		Within Softmax		LN1		GELU		LN2		
	in	out	in	out	in	out	in	out	in	out	–
Minimal	6	6	–	–	3	3	12	12	3	3	48
BOLT [8]	12	12	–	–	6	6	24	24	6	6	96
BLB [10]	6	6	6	6	3	3	36	12	3	3	**84**
EncFormer-exp	6	6	–	–	3	3	36	12	3	3	**72**
EncFormer-min	6	6	–	–	3	3	12	12	3	3	**48**

Table V compares analytically derived ciphertext counts per MPC block for the theoretical minimal baseline, prior hybrid systems, and EncFormer under the BERT-base slot budget $n = 16384$. Prior papers do not report boundary payload in bytes, so we compare at the level of ciphertext counts under a common tensor shape and slot budget. The resulting reductions and their attribution to SCP and the cost-model rule are discussed in §VII-B.

Applicability Beyond Complex-Native Kernels. EncFormer's FHE kernels are complex-native and export the real and imaginary channels of each ciphertext as two independent real boundary tensors in one transmission. The same CC protocol can also serve real-only FHE kernels in systems such as BOLT, BumbleBee, and BLB [8], [10], [16]. Two real boundary tensors are first packed into the real and imaginary slots of one ciphertext, converted once, and then separated back into two real share streams for MPC. This wrapper halves the boundary payload without changing the upstream FHE kernels, and §VII-A shows that its overhead is small relative to the communication savings.

Modulus Trimming Optimization. CKKS-to-MPC conversion payload depends on the serialized ciphertext size, which scales with the remaining modulus chain. Ciphertexts should arrive at the boundary at the shortest admissible modulus level, since no further CKKS multiplications are needed. If a ciphertext reaches the boundary at a higher level, P_1 can modulus switch it down before running $\Pi^{\mathbb{C}}_{\mathrm{C2M}}$ to reduce transmitted payload and improve T_{conv} in Eq. (3). This optimization applies only to CKKS-to-MPC conversion since MPC-to-CKKS outputs must retain sufficient modulus for subsequent CKKS evaluation.

Let L_{conv} be the target conversion level and set $q = Q_{L_{\mathrm{conv}}}$. We choose L_{conv} as the smallest level that keeps conversion

TABLE VI
CKKS PARAMETERIZATION PER FHE BLOCK. ALL BLOCKS USE SPARSE-TERNARY SECRETS WITH h=64 AND SCALE 2^{40}.

Parameter	Block 1	Block 2	Block 3	Block 4
Kernel group	QKV	Score	GELU	FFN
N	32768	32768	32768	65536
Slots	16384	16384	16384	32768
Depth	10	7	6	4
Coeff. moduli	$60, 40{\times}10, 60$	$60, 40{\times}7, 60$	$60, 40{\times}6, 60$	$60, 40{\times}4, 60$
$\log_2 Q$	520	400	360	280
Max MSE	$6.90{\times}10^{-16}$	$2.22{\times}10^{-16}$	$4.99{\times}10^{-16}$	$1.59{\times}10^{-14}$
λ lower bound	≥ 160.0	≥ 160.0	≥ 160.0	$\gg 256$

statistically safe:

$$\log_2(q) \geq \ell + \sigma + 1 \qquad \text{and} \qquad q/2 > \Delta \cdot B_{\max}.$$

P_1 repeatedly applies ModSwitchToNext (which removes the last RNS prime without rescaling) until reaching L_{conv}, then runs $\Pi^{\mathbb{C}}_{\text{C2M}}$ unchanged. This reduces only the transmission size; the MPC ring size and ring-to-field conversion cost are unaffected.

VI. Experimental Setup

A. Implementation

System Stack. We implement the main EncFormer pipeline with PhantomFHE for CKKS and EzPC/SCI [35], [36] for MPC.[2] The supporting matched-backend comparison uses Liberate.FHE and CrypTen [37]. All FHE kernels run on NVIDIA A100 GPUs. MPC values use fixed-point encoding with $F = 13$ fractional bits over $\mathbb{Z}_{2^\ell}$ with $\ell = 43$.

CKKS Profile. Table VI lists the deployed native CKKS profile by the four CKKS blocks used in the paper. The coefficient-modulus chain is selected per block according to the required depth rather than kept identical across blocks. Block 2 uses a 400-bit chain, matching the modulus level of BLB's N=32768 profile [10], while the other blocks use the depth-appropriate chains shown in the table. The two reported EncFormer boundary configurations follow backend choice. The main PhantomFHE pipeline uses the expanded-GELU boundary configuration, while the matched Liberate.FHE comparison uses the minimal-GELU boundary configuration. §VII-B reports the corresponding evaluation rows.

B. Models and Measurement

Models and Dataset. Following [8], [10], [16], we evaluate GPT2-base, BERT-base, and BERT-large. For accuracy, we use the GLUE tasks MRPC, RTE, and SST-2 [38]. We start from public Hugging Face checkpoints [39], fine-tune them following Powerformer-style training, and distill surrogate parameters for encrypted inference. Training details are given in Appendix K.

Network Profiles. Unless otherwise specified, we report four standard network profiles: LAN at 1 Gbps with 0.3 ms RTT, WAN1 at 400 Mbps with 4 ms RTT, WAN2 at 100 Mbps with 4 ms RTT, and WAN3 at 100 Mbps with 80 ms RTT.

[2]Artifact: https://doi.org/10.5281/zenodo.21334274.

TABLE VII
CKKS PRIMITIVE LATENCY (MS) ON A100 GPU (N=65536, DEPTH 16).

Library	add	pt mul	ct mul	rot	conj
PhantomFHE	0.07	0.36	2.65	2.32	2.33
Liberate.FHE	3.04	13.72	48.66	32.70	22.73

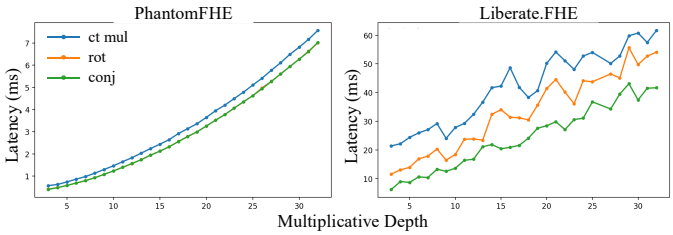


Fig. 7. CKKS primitive latency across multiplicative depths for PhantomFHE and Liberate.FHE.

Measurement Methodology. FHE kernel and MPC local computation times are measured on a single A100 GPU and reported as the mean of three independent trials after one warm-up iteration. LAN and WAN latencies are computed by adding measured local compute time to a network overlay derived from measured byte counts and round counts under each profile, following the same methodology as prior work [8], [10], [16].

Baseline Sources. Prior-system entries combine peer-reviewed paper rows with released-artifact checks [8], [10]–[12], [16]. We rerun or inspect released artifacts where available, and keep the paper-reported row when the released artifact does not provide a clean matched end-to-end run. When an aggregate is missing but complete protocol details are available, we derive it under the same tensor shapes and network profiles. Appendix O summarizes baseline evidence coverage, while Appendix P validates the network and byte accounting.

VII. Evaluation

A. Microbenchmarks

FHE Library Comparison. To contextualize the runtimes reported in §VII-B, we benchmark the two GPU CKKS backends used in this paper, PhantomFHE [40] and Liberate.FHE [41], under a common parameter set. Table VII reports the latency of core CKKS primitives, where PhantomFHE is on average about an order of magnitude faster than Liberate.FHE, consistent with prior library comparisons [42]. PhantomFHE does not support bootstrapping, while Liberate.FHE does, which is why FHE-only baselines that require bootstrapping are evaluated under Liberate.FHE in Table XI.

Multiplicative Depth Sensitivity. Figure 7 shows that CKKS primitive latency increases with multiplicative depth for both libraries, with higher and more variable latency for Liberate.FHE. This supports our design choice to keep FHE depth low and refresh via FHE–MPC conversions.

Conversion Protocol Comparison. We benchmark CKKS–MPC conversion for two real-valued ciphertexts under four standard network profiles. The real baseline protocol [8], [10], [43] converts the ciphertexts independently and transmits two

TABLE VIII
CKKS–MPC CONVERSION LATENCY UNDER STANDARD NETWORK PROFILES. N=65536, DEPTH 8.

Network	LAN	WAN1	WAN2	WAN3
Real Time (ms)	169.0	435.4	1693.7	1997.7
Complex Time (ms)	85.4	218.6	847.8	999.8
Complex + Trim Time (ms)	51.9	131.7	508.6	599.6
Speedup	1.98×	1.99×	2.00×	2.00×
+ Trim Speedup	3.26×	3.31×	3.33×	3.33×

ciphertexts per direction. Our complex variant packs the two vectors into the real and imaginary parts of one ciphertext and transmits one ciphertext per direction. We report end-to-end conversion latency, including CKKS-side complexification and decomplexification overhead, and network delay. The total payload per conversion pair is 20.97 MB for Real, 10.49 MB for Complex, and 6.41 MB for Complex plus trimming. Table VIII reports latency and speedup across networks.

The nearly $2\times$ reduction confirms that the complex protocol halves boundary communication, while trimming provides additional gain when the ciphertext reaches the boundary with excess modulus.

B. End-to-end Evaluation

Latency and Communication. Our main evaluation uses the PhantomFHE+EzPC pipeline and reports end-to-end latency together with inference-time communication. §VII-C compares the hybrid and FHE-only paradigms under a matched backend. Following prior work [8], [10], [16], the latency columns report online inference or protocol-execution time, and Comm includes inference-time FHE–MPC conversion. Appendix O summarizes evidence coverage for all baseline row families, and Appendix P validates the network overlay and byte accounting. We retain the published headline values as optimized baseline results, which credits each baseline with its strongest reported setting. For EncFormer, the online row includes bridge conversion and nonlinear MPC traffic. Setup and offline weight preparation are reported separately. The shaded SIGMA/SHAFT [9], [44] rows provide cross-family FSS context, with key-transfer accounting in Appendix I.

EncFormer achieves the lowest latency and inference-time communication among hybrid FHE–MPC baselines across all three models. Averaging ratios over the 12 latency cells and three Comm cells in Table IX, EncFormer is $1.3\times$ faster than BLB while sending $1.4\times$ less traffic. The corresponding latency and communication gains are $2.1\times$ and $2.6\times$ over BumbleBee, and $9.9\times$ and $30.4\times$ over BOLT.

Plaintext BERT-base, BERT-large, and GPT2-base inference on one A100 takes 4.5 ms, 9.1 ms, and 4.2 ms, respectively, so Table IX should be read as secure-inference overhead relative to plaintext execution.

Pipeline Accuracy. Table X compares encrypted inference against a public BERT baseline using standard BERT with exact softmax, LayerNorm, and GELU, and against the task-fine-tuned plaintext BERT-base teacher with standard nonlinearities. Across SST-2 and both MRPC metrics, EncFormer is within 1.0 percentage point of the public exact-operator baseline and within 0.5 point of the task-specific plaintext teacher. RTE shows a larger 2.53-point drop from the public baseline but is 0.60 point above the teacher. These results show that EncFormer maintains plaintext-level accuracy on the evaluated GLUE tasks while replacing standard nonlinearities with secure-friendly operators.

TABLE IX
END-TO-END LATENCY AND COMMUNICATION.

Model	Framework	LAN	WAN1	WAN2	WAN3	Comm (GB)
GPT2-base $m = 64$	SIGMA incl. keys	7.7	11.1	25.1	38.5	28.7
	SHAFT	1.6	4.0	15.5	17.4	5.76
	BOLT	7.6	12.4	31.8	51.6	34.8
	BumbleBee	2.9	3.5	4.9	12.0	2.5
	BLB	2.0	2.5	3.9	8.1	1.5
	EncFormer	**1.6**	**2.0**	**3.1**	**5.9**	**1.0**
BERT-base $m = 128$	SIGMA incl. keys	7.8	12.3	30.4	48.6	34.4
	SHAFT	2.9	7.1	28.1	29.9	10.46
	BOLT	15.5	18.5	77.9	122.7	63.6
	BumbleBee	4.3	6.2	11.6	21.3	5.8
	BLB	2.5	3.8	6.6	13.2	3.0
	EncFormer	**2.1**	**2.9**	**4.5**	**8.8**	**2.2**
BERT-large $m = 128$	SIGMA incl. keys	20.5	36.3	102.5	155.5	92.8
	SHAFT	7.7	19.3	76.2	79.9	28.46
	BOLT	43.8	49.3	208.2	260.6	158.9
	BumbleBee	9.7	15.6	30.8	49.2	15.2
	BLB	6.6	9.8	16.2	24.9	7.8
	EncFormer	**5.3**	**7.4**	**12.7**	**20.3**	**5.8**

TABLE X
ACCURACY (AND F1 FOR MRPC) ON SELECTED GLUE TASKS.

Task	#Val	Public BERT baseline	KD teacher	EncFormer	Δ w.r.t. public BERT	Δ w.r.t. KD teacher
SST-2	872	92.43%	92.13%	91.78%	−0.65%	−0.35%
MRPC (acc)	408	87.75%	87.01%	86.76%	−0.99%	−0.25%
MRPC (F1)	408	91.17%	90.82%	90.34%	−0.83%	−0.48%
RTE	277	72.56%	69.43%	70.03%	−2.53%	+0.60%

C. Hybrid and FHE-Only Paradigm Comparison

A natural question is whether the hybrid FHE–MPC approach is worthwhile compared to FHE-only pipelines that avoid interaction entirely. For a fair comparison with Powerformer [12], whose softmax and LayerNorm approximation functions we adopt, we implement EncFormer with Liberate.FHE and CrypTen so that the comparison uses the same FHE backend and nonlinear approximations as the FHE-only baselines THOR and Powerformer.

Table XI shows the layer-wise breakdown. FHE-only systems offer a non-interactive solution with no communication during inference. However, if the deployment permits lightweight client–server interaction, the hybrid approach is substantially more efficient. Under LAN, eliminating bootstrapping yields a $3.5\times$ speedup over THOR and $1.9\times$ over Powerformer. The advantage narrows under WAN as MPC round-trip costs grow, so practitioners should choose based on their network conditions.

Even comparing only FHE linear components, EncFormer is faster. For BERT-base, the FHE linear time per layer is 172.3 s for EncFormer, compared with 229.4 s for THOR and

TABLE XI
LAYER-WISE LATENCY IN SECONDS ON BERT-BASE USING THE LIBERATE.FHE BACKEND WITH CRYPTEN.

Operation	THOR	Power former	Ours (LAN)	Ours (WAN1)	Ours (WAN2)	Ours (WAN3)
Attention layer	49.77	57.65	40.54	40.54	40.54	40.54
Attention score	32.53	28.76	31.10	31.10	31.10	31.10
Softmax	15.53	0.75	1.88	2.89	4.43	8.52
Multi-head attention	48.06	41.49	38.16	38.16	38.16	38.16
LayerNorm1	7.13	0.37	0.98	1.23	1.50	3.45
FC1	49.80	59.21	32.50	32.50	32.50	32.50
GELU	29.42	8.31	3.38	6.24	14.64	29.98
FC2	49.19	43.77	30.27	30.27	30.27	30.27
LayerNorm2	4.10	0.30	0.92	1.22	1.57	3.70
Bootstrappings	337.86	103.72	0	0	0	0
Total	623.39	344.33	179.73	184.15	194.71	218.22

230.9 s for Powerformer, thanks to SCP eliminating repacking and shallower depth reducing per-operation cost, as shown in Figure 7.

D. Ablation Studies

GELU Conversion Cost Analysis. We study the inbound CKKS–MPC boundary of the GELU block and compare two pipeline variants that differ only in where the GELU polynomials are evaluated. In the minimal baseline, the pipeline converts only the GELU input tensor $\mathbf{x}$ into MPC using minimal packing, evaluates the polynomial candidates in MPC, and then applies cmp and mux. In EncFormer's CKKS pre-evaluation variant, we evaluate the polynomial candidates, $F_0(\mathbf{x})$ and $F_1(\mathbf{x})$, in CKKS and convert $\mathbf{x}$ together with the candidates into MPC, so that MPC performs only cmp and mux. Appendix B details the two GELU protocol variants used by EncFormer.

Figure 8 reports the end-to-end latency difference between CKKS pre-evaluation and the baseline for BERT-base and BERT-large, decomposed into the added conversion overhead and the computation change under our cost model. With PhantomFHE, the MPC cost reduction dominates on WAN1–WAN3, making CKKS pre-evaluation preferable under the decision rule in Eq. (4). Because the net change on LAN is small, we use expanded packing in the main implementation pipeline. With the alternative Liberate.FHE implementation, however, CKKS pre-evaluation incurs substantially higher CKKS-side cost, so the net improvement is no longer robust across network profiles. Since the deployment network may be unknown a priori, we therefore adopt the GELU split without CKKS pre-evaluation for the results in Table XI.

Contribution of Each Optimization. Table XII compares EncFormer against BLB and two internal variants. In `w/o CC`, we disable complex conversion and remove GELU pre-evaluation. In `w/o SCP`, we keep the same kernels but force additional repacking between FHE stages. The comparison shows that EncFormer's gains come from lower inference-time traffic, fewer interactive rounds, and the elimination of cross-stage FHE remaps.

Without SCP, the ablation inserts layout-normalization remaps at all packing-sensitive stage boundaries. Each charged ciphertext block is remapped via the Halevi–Shoup slot-permutation decomposition [23], requiring $\log_2 m$ rotations per block. Appendix G and Table II give the per-edge breakdown. For BERT-base with m=128, the six charged rows cover 60 ciphertext blocks and add 420 rotations per layer, or about 11.7 s over 12 layers using the measured PhantomFHE rotation latency. The observed `w/o SCP` gap is about 12–24 s because the realized remap also includes masking multiplications and recombination additions. The nearly constant slowdown across network profiles confirms that SCP primarily reduces FHE compute rather than communication.

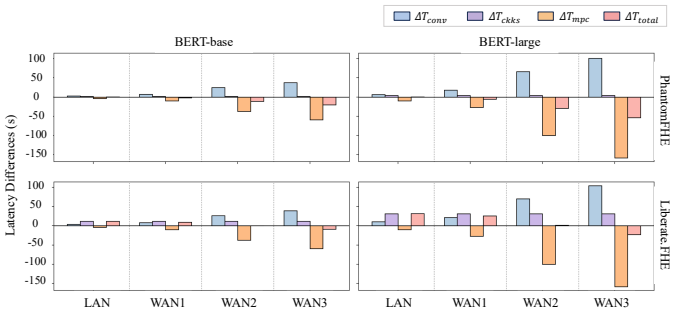


Fig. 8. GELU latency deltas between EncFormer and the minimal baseline, decomposed into computation savings and conversion overhead.

TABLE XII
ABLATION OF ENCFORMER COMPONENTS AGAINST BLB.

Model	Framework	LAN	WAN1	WAN2	WAN3	Comm (GB)
GPT2-base $m = 64$	BLB	2.0	2.5	3.9	8.1	1.5
	w/o CC	1.8	2.3	3.6	7.4	1.3
	w/o SCP	1.8	2.2	3.4	6.2	1.0
	EncFormer	**1.6**	**2.0**	**3.1**	**5.9**	**1.0**
BERT-base $m = 128$	BLB	2.5	3.8	6.6	13.2	3.0
	w/o CC	2.3	3.5	6.0	11.4	2.7
	w/o SCP	2.3	3.1	4.9	9.2	2.2
	EncFormer	**2.1**	**2.9**	**4.5**	**8.8**	**2.2**
BERT-large $m = 128$	BLB	6.6	9.8	16.2	24.9	7.8
	w/o CC	6.3	9.2	15.4	23.6	7.3
	w/o SCP	6.1	8.2	13.6	21.4	5.8
	EncFormer	**5.3**	**7.4**	**12.7**	**20.3**	**5.8**

Knowledge Distillation and Alternative. EncFormer prepares the secure-inference-compatible student through offline KD after standard fine-tuning. Appendix K reports a main-table-anchored KD-on/KD-off ablation, where KD-off means supervised fine-tuning of the same surrogate-operator student without teacher logits or intermediate matching. The KD-on anchor is the deployed EncFormer accuracy in Table X; the KD-off values subtract the measured paired KD-on/KD-off deltas from that anchor. Removing KD makes the calibrated student less stable on SST-2 and MRPC and has a smaller effect on RTE. This indicates that KD mainly stabilizes adaptation to surrogate Softmax, LayerNorm, and GELU operators with frozen calibration. Thus, the accuracy gap in Table X reflects model adaptation to secure-friendly operators rather than online encrypted execution alone. KD is excluded from the online latency and communication in Table IX because it is completed before deployment as part of the server-side

model-preparation cost.

VIII. Related Work

We group prior secure machine learning inference systems into three categories: FHE-only, MPC-only, and hybrid FHE–MPC pipelines.

A. FHE-only Secure Inference

CryptoNets pioneered end-to-end leveled-FHE inference by replacing nonlinearities with low-degree polynomials [45]. CHET [46] automates layout selection for CNN inference under FHE. For Transformers, THE-X systematizes polynomial approximations for softmax, layer normalization, and GELU to enable non-interactive BERT inference [13]. CipherFormer [47] extends this to GPT-2 with CKKS packing optimizations. NEXUS [32], THOR [11], and Powerformer [12] further reduce overhead via CKKS-friendly packing, compression, and rotation-efficient attention schedules. A recent survey [48] provides a comprehensive overview of private Transformer inference techniques.

FHE-only inference is non-interactive, but it is constrained by expensive ciphertext key-switching operations such as rotations and relinearization. It also often requires high multiplicative depth with long modulus chains or heavy bootstrapping.

B. MPC-only Secure Inference

MPC-only systems evaluate the full model on secret shares, using arithmetic protocols for linear layers and specialized sub-protocols for nonlinear operations. SecureML [49], MiniONN [50], ABY [51], and CryptFlow [52] established practical CNN inference with fixed-point arithmetic and mixed-protocol execution. Falcon [53] and XONN [54] explore three-party and binary-circuit alternatives.

For Transformers and LLMs, MPCFormer [55] and follow-on two-party systems co-design efficient protocols for attention, including softmax, GELU, and layer normalization, to scale to large hidden sizes and vocabularies [9], [14], [44], [56]–[59]. MPC-only inference requires frequent interaction, so latency and bandwidth can become the bottleneck for deep models and wide activations.

C. Hybrid FHE–MPC Secure Inference

Hybrid systems typically execute wide linear algebra under FHE and delegate nonlinearities and selected reductions to MPC. Gazelle [29] introduced this split for CNNs, while Delphi [60] and Cheetah [28] improved latency through automated partitioning and optimized FHE and MPC building blocks. For Transformers, Iron [15] applies the split to BERT, BOLT [8] improves packing and conversion to avoid bootstrapping while preserving high-accuracy nonlinearities, BumbleBee [16] further optimizes hybrid execution with communication-efficient protocols and GPU-accelerated kernels, and BLB [10] reduces boundary overhead via finer-grained operator fusion and fewer FHE–MPC conversions. Hybrid pipelines balance ciphertext computation with interactive protocols, but they can inherit the overhead of both and are often dominated by additional FHE–MPC conversion costs at the boundaries.

D. Comparative Discussion

Powerformer and THOR are FHE-only systems, so their nonlinear layers remain part of the CKKS-depth budget. BOLT and BumbleBee are strong hybrid baselines, but use fixed real-valued FHE–MPC boundary layouts around nonlinear blocks and do not use complex-native HE kernels. BLB is the closest prior system and already uses CKKS–MPC conversion, fine-grained fusion, boundary-level complex packing in its implementation, and per-block CKKS parameters.

The remaining gap lies in boundary co-design. Prior hybrid pipelines do not provide SCP-style cross-stage layout rules, do not analyze boundary representation as a selectable cost variable, and do not use complex packing as a native HE-kernel design primitive. EncFormer addresses these gaps with SCP layouts, a formal complex C2M/M2C interface with modulus trimming, and a cost rule for selecting boundary representations under measured CKKS and MPC costs. Our GELU boundary variants instantiate this rule in the evaluation. Appendix N gives the row-by-row audit.

IX. Conclusion

We presented EncFormer, a two-party hybrid FHE–MPC framework for private Transformer inference that frames packing, conversion, and nonlinear execution as a single co-design problem. EncFormer uses Stage-Compatible Patterns to keep adjacent FHE kernels layout-compatible, a secure complex CKKS–MPC conversion to reduce boundary payload, and a calibrated PhantomFHE decision rule to choose beneficial expanded boundaries. This design reduces inference-time communication and end-to-end latency across GPT2-base, BERT-base, and BERT-large relative to prior hybrid systems, with average communication/latency gains of $30.4\times/9.9\times$ against BOLT, $2.6\times/2.1\times$ against BumbleBee, and $1.4\times/1.3\times$ against BLB, while also outperforming FHE-only pipelines on BERT-base under a matched backend. On BERT-base, encrypted inference maintains near-plaintext accuracy on selected GLUE tasks. The broader lesson is that hybrid private inference should be designed around stage compatibility and boundary economics, not primitive choice alone.

Limitations. The scope of EncFormer is semi-honest two-party secure inference for Transformer blocks. The proof does not cover malicious adversaries, side channels, or traffic-oblivious scheduling. Like prior systems using surrogate nonlinearities [12], it still requires task-specific distillation. Selected GLUE results do not imply exact-model parity, task-universal accuracy, or full autoregressive long-context coverage. The boundary rule is calibrated to measured costs on the reported A100 backends and network profiles, so new hardware, sequence lengths, batch sizes, or bootstrapping libraries require re-measurement.

Future Work. This work points to several open directions, including cost-model-guided compilers that derive SCP layouts and boundary representations with less hand-crafted placement for new models, stronger adversarial settings such as malicious security and active-secure complex conversion, and shared benchmarks for comparing hybrid and FHE-only Transformer systems under common deployment assumptions.

ACKNOWLEDGMENT

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative (DTC-RGC-01 and DTC-IGC-01). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019, technical report, OpenAI. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533

[3] K. Huang, J. Altosaar, and R. Ranganath, "ClinicalBERT: Modeling clinical notes and predicting hospital readmission," *ArXiv*, vol. abs/1904.05342, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:119308351

[4] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The muppets straight out of law school," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 2898–2904. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.261/

[5] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 09 2019. [Online]. Available: https://doi.org/10.1093/bioinformatics/btz682

[6] X. Yang, C. Zhang, Y. Sun, K. Pang, L. Jing, S. Wa, and C. Lv, "Finchain-BERT: A high-accuracy automatic fraud detection model based on NLP methods for financial scenarios," *Information*, vol. 14, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/9/499

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18.

[8] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "BOLT: privacy-preserving, accurate and efficient inference for transformers," in *IEEE Symposium on Security and Privacy (SP 2024)*. San Francisco, CA, USA: IEEE, May 2024, pp. 4753–4771.

[9] A. Y. L. Kei and S. S. M. Chow, "Shaft: Secure, handy, accurate, and fast transformer inference," in *Network and Distributed System Security Symposium (NDSS)*, 2025.

[10] T. Xu, W.-j. Lu, J. Yu, Y. Chen, C. Lin, R. Wang, and M. Li, "Breaking the layer barrier: remodeling private transformer inference with hybrid CKKS and MPC," in *Proceedings of the 34th USENIX Conference on Security Symposium*. USA: USENIX Association, 2025.

[11] J. Moon, D. Yoo, X. Jiang, and M. Kim, "Thor: Secure transformer inference with homomorphic encryption," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*. Association for Computing Machinery, 2025, pp. 392–410.

[12] D. Park, E. Lee, and J.-W. Lee, "Powerformer: Efficient and high-accuracy privacy-preserving language model with homomorphic encryption," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 11 090–11 111. [Online]. Available: https://aclanthology.org/2025.acl-long.543/

[13] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei, "THE-X: Privacy-preserving transformer inference with homomorphic encryption," in *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3510–3520. [Online]. Available: https://aclanthology.org/2022.findings-acl.277/

[14] J. Luo, Y. Zhang, Z. Zhang, J. Zhang, X. Mu, H. Wang, Y. Yu, and Z. Xu, "SecFormer: Fast and accurate privacy-preserving inference for transformer models via SMPC," in *Findings of the Association for Computational Linguistics (ACL 2024)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 13 333–13 348. [Online]. Available: https://aclanthology.org/2024.findings-acl.790/

[15] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: Private inference on transformers," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 15 718–15 731. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/64e2449d74f84e5b1a5c96ba7b3d308e-Paper-Conference.pdf

[16] W.-j. Lu, Z. Huang, Z. Gu, J. Li, J. Liu, C. Hong, K. Ren, T. Wei, and W. Chen, "BumbleBee: Secure two-party inference framework for large transformers," in *Network and Distributed System Security (NDSS) Symposium 2025*. San Diego, CA, USA: Internet Society, Feb. 2025. [Online]. Available: https://doi.org/10.14722/ndss.2025.230057

[17] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012. [Online]. Available: https://eprint.iacr.org/2012/144

[18] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Leveled fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ACM, 2012, pp. 309–325.

[19] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*, ser. Lecture Notes in Computer Science, vol. 10624. Springer, 2017, pp. 409–437.

[20] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.

[21] M. Zheng, Q. Lou, and L. Jiang, "Primer: Fast private transformer inference on encrypted data," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[22] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," Cryptology ePrint Archive, Paper 2010/520, 2010. [Online]. Available: https://eprint.iacr.org/2010/520

[23] S. Halevi and V. Shoup, "Algorithms in helib," in *Advances in Cryptology – CRYPTO 2014*. Springer, 2014, pp. 554–571.

[24] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology — CRYPTO '91*, J. Feigenbaum, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 420–432.

[25] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "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.

[26] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1986, pp. 162–167.

[27] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1987, pp. 218–229.

[28] Z. Huang, W. jie 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)*. Boston, MA: USENIX Association, Aug. 2022, pp. 809–826. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/huang-zhicong

[29] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1651–1669. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar

[30] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[32] J. Zhang, X. Yang, L. He, K. Chen, W. jie Lu, Y. Wang, X. Hou, J. Liu, K. Ren, and X. Yang, "Secure transformer

inference made non-interactive," in *32nd Annual Network and Distributed System Security Symposium (NDSS 2025)*. The Internet Society, 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/secure-transformer-inference-made-non-interactive/

[33] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1209–1222. [Online]. Available: https://doi.org/10.1145/3243734.3243837

[34] T. Xu, L. Wu, R. Wang, and M. Li, "Privcirnet: efficient private inference via block circulant transformation," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, ser. NIPS '24. Red Hook, NY, USA: Curran Associates Inc., 2024.

[35] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "EzPC: Programmable and efficient secure two-party computation for machine learning," in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 496–511.

[36] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow2: Practical 2-party secure inference," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, pp. 325–342.

[37] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: secure multi-party computation meets machine learning," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021.

[38] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, T. Linzen, G. Chrupała, and A. Alishahi, Eds. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. [Online]. Available: https://aclanthology.org/W18-5446/

[39] Hugging Face, "Transformers," https://huggingface.co/, 2026, accessed: 2026-01-26.

[40] H. Yang, S. Shen, W. Dai, L. Zhou, Z. Liu, and Y. Zhao, "Phantom: A CUDA-accelerated word-wise homomorphic encryption library," *IEEE Trans. Dependable Secur. Comput.*, vol. 21, no. 5, pp. 4895–4906, 2024. [Online]. Available: https://doi.org/10.1109/TDSC.2024.3363900

[41] DESILO, "Liberate.FHE: A New FHE Library for Bridging the Gap between Theory and Practice with a Focus on Performance and Accuracy," https://github.com/Desilo/liberate-fhe, 2023, accessed 2025-09-17.

[42] C. Agulló-Domingo, Ó. Vera-López, S. Guzelhan, L. Daksha, A. E. Jerari, K. Shivdikar, R. Agrawal, D. Kaeli, A. Joshi, and J. L. Abellán, "FIDESlib: A fully-fledged open-source FHE library for efficient CKKS on GPUs," in *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2025, pp. 1–3.

[43] F. Boemer, R. Cammarota, D. Demmler, T. Schneider, and H. Yalame, "Mp2ml: A mixed-protocol machine learning framework for private inference," in *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, ser. PPMLP'20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 43–45. [Online]. Available: https://doi.org/10.1145/3411501.3419425

[44] K. Gupta, N. Jawalkar, A. Mukherjee, N. Chandran, D. Gupta, A. Panwar, and R. Sharma, "SIGMA: Secure GPT inference with function secret sharing," *Proceedings on Privacy Enhancing Technologies*, vol. 2024, no. 4, pp. 61–79, 2024, artifact available. [Online]. Available: https://doi.org/10.56553/popets-2024-0107

[45] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 201–210. [Online]. Available: https://proceedings.mlr.press/v48/gilad-bachrach16.html

[46] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, "Chet: an optimizing compiler for fully-homomorphic neural-network inferencing," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 142–156. [Online]. Available: https://doi.org/10.1145/3314221.3314628

[47] W. Wang and Y. Kuang, "Cipherformer: Efficient transformer private inference with low round complexity," in *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2024, pp. 3054–3059.

[48] Y. Li, X. Zhou, Y. Wang, L. Qian, and J. Zhao, "Private transformer inference in mlaas: A survey," *arXiv preprint arXiv:2505.10315*, 2025.

[49] 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.

[50] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "MiniONN: Enabling secure inference with minimal latency," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 119–133.

[51] 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 (NDSS)*. Internet Society, 2015. [Online]. Available: https://www.ndss-symposium.org/ndss2015/

[52] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow: Secure TensorFlow inference," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 336–353.

[53] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "FALCON: Honest-majority maliciously secure framework for private deep learning," *Proceedings on Privacy Enhancing Technologies (PETS)*, vol. 2021, no. 2, pp. 188–208, 2021.

[54] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "XONN: XNOR-based oblivious deep neural network inference," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1501–1518. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/riazi

[55] D. Li, H. Wang, R. Shao, H. Guo, E. Xing, and H. Zhang, "MPCFormer: Fast, performant and private transformer inference with MPC," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=CWmvjOEhgH-

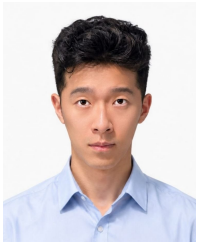
[56] X. Hou, J. Liu, J. Li, Y. Li, W.-j. Lu, C. Hong, and K. Ren, "CipherGPT: Secure two-party GPT inference," *IACR Cryptol. ePrint Arch.*, p. 1147, 2023.

[57] Y. Dong, W.-j. Lu, Y. Zheng, H. Wu, D. Zhao, J. Tan, Z. Huang, C. Hong, T. Wei, W.-G. Chen, and J. Zhou, "PUMA: Secure inference of LLaMA-7B in five minutes," *Security and Safety*, vol. 4, p. 2025014, 2025, published online 23 Oct 2025. [Online]. Available: https://doi.org/10.1051/sands/2025014

[58] C. Zeng, D. He, Q. Feng, X. Yang, and Q. Luo, "SecureGPT: A framework for multi-party privacy-preserving transformer inference in GPT," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 9480–9493, 2024.

[59] H. Huang and Y. Wang, "SecBERT: Privacy-preserving pre-training based neural network inference system," *Neural Netw.*, vol. 172, no. C, Apr. 2024. [Online]. Available: https://doi.org/10.1016/j.neunet.2024.106135

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



Yufan Zhu received the B.Comp. degree in computer science from the National University of Singapore (NUS) in 2025. He is currently pursuing the Ph.D. degree in computer science at NUS. His research interests include secure computation, privacy-preserving machine learning, and LLM systems security.

Chao Jin is a Principal Scientist in the Safety and Security Chapter at the Institute of Advanced Intelligence and Computing (IAIC), A*STAR (Agency for Science, Technology and Research), Singapore. His research interests include cryptographic computing, including homomorphic encryption and secure multiparty computation, AI security and safety, and agentic AI systems.

Khin Mi Mi Aung (*Senior Member, IEEE*) is a Senior Principal Scientist in the Safety and Security Chapter at the Institute of Advanced Intelligence and Computing (IAIC), A*STAR (Agency for Science, Technology and Research), Singapore. Her research interests include privacy-enhancing technologies, data security, and large-scale infrastructure security.

Xiaokui Xiao received the Ph.D. degree in computer science from the Chinese University of Hong Kong in 2008. He is currently a Professor at the Department of Computer Science, National University of Singapore. His research interests include privacy-preserving data management and large-scale data analysis. He is an IEEE Fellow and ACM Distinguished Member. His honors include the VLDB 2021 Best Research Paper Award, the 2022 ACM SIGMOD Research Highlight Award, the 2024 ACM SIGMOD Test-of-Time Award, and the IEEE S&P 2026 Distinguished Paper Award.