SecDTD: Dynamic Token Drop for Secure Transformers Inference

Yifei Cai[†], Zhuoran Li[*], Yizhou Feng[‡], Qiao Zhang[§], Hongyi Wu[*], Danella Zhao[*], and Chunsheng Xin[†]
[†]*Iowa State University, yifeic@iastate.edu, cxin@iastate.edu*
[*]*University of Arizona, zli1122@arizona.edu, mhwu@arizona.edu, danellazhao@arizona.edu*
[‡]*Old Dominion University, yfeng002@odu.edu*
[§]*Shandong University, qiao.zhang@sdu.edu.cn*

***Abstract*—The rapid adoption of Transformer-based AI has been driven by accessible models such as ChatGPT, which provide API-based services for developers and businesses. However, as these online inference services increasingly handle sensitive inputs, privacy concerns have emerged as a significant challenge. To address this, secure inference frameworks have been proposed, but their high computational and communication overhead often limit practical deployment. In plaintext settings, token drop is an effective technique for reducing inference cost; however, our analysis reveals that directly applying such methods to ciphertext scenarios is suboptimal due to distinct cost distributions in secure computation. We propose SecDTD, a dynamic token drop scheme tailored for secure Transformer inference. SecDTD advances token drop by shifting the dropping to earlier inference stages, effectively reducing the cost of key components such as Softmax. To support this, we introduce two core techniques. Max-Centric Normalization (MCN): A novel, Softmax-independent scoring method that enables early token drop with minimal overhead and improved normalization, supporting more aggressive dropping without accuracy loss. OMSel: A faster, oblivious median selection protocol that securely identifies the median of importance scores to support token drop. Compared to existing sorting-based methods, OMSel achieves a 16.9× speedup while maintaining security, obliviousness and randomness. We evaluate SecDTD through 48 experiments across eight GLUE datasets under various network settings using the BOLT and BumbleBee frameworks. SecDTD achieves 4.47× end-to-end inference acceleration without degradation in accuracy.**

***Index Terms*—Machine Learning as a Service, Privacy-Preserving Computation, Homomorphic Encryption, Multi-Party Computation, Transformers**

1. Introduction

Transformer-based models, such as BERT [1] and GPT [2], [3], have revolutionized artificial intelligence (AI) by achieving state-of-the-art performance in tasks like language translation [4], question answering [5], and code generation [6]. Their success is largely attributed to attention mechanisms [7], which effectively capture contextual relationships and long-range dependencies. Unlike traditional task-specific supervised learning, large Transformers undergo pre-training on massive amounts of unlabeled data, enhancing their adaptability across diverse applications.

The widespread adoption of Transformer-based AI has been further driven by publicly accessible models like ChatGPT [8], which offer API services [9] for developers and businesses. This accessibility has facilitated their integration into critical domains such as healthcare [10], [11], where Transformers assist in analyzing medical records and extracting insights to support diagnosis. However, as these models increasingly process sensitive data, privacy concerns have become more pressing [12]. Online inference services pose risks, as user-submitted prompts may contain personal information, raising ethical and security challenges. Addressing privacy concerns is essential for the responsible deployment of Transformer models in sensitive applications.

To address privacy risks in AI inference, various privacy preserving frameworks have been developed to protect model parameters from users while preventing servers from accessing users' private inputs. Recent research has primarily focused on privacy-preserving frameworks based on Secure Multi-Party Computation (MPC) [13], [14] and Homomorphic Encryption (HE) [15], [16] to enable private inference on deep learning models. These frameworks are applied to convolutional neural networks (CNNs) [17]–[22] and Transformer-based models [23]–[26], ensuring secure computation while preserving model parameters in privacy-preserving inference.

Efficiency Challenges for Privacy-preserving Transformers Inference: While secure frameworks enable encrypted computation, they often incur substantial computation and communication costs due to the reliance on expensive cryptographic primitives [27]–[29]. Privacy-preserving Transformer frameworks face two key challenges: First, unlike CNN inference, which primarily involves convolution operations and relatively simple matrix-vector multiplications, Transformers heavily rely on large-scale matrix-matrix multiplications [7]. These operations dominate the **linear** computational cost and significantly increase the resource demands of secure inference. Furthermore, in contrast to CNNs that typically use the lightweight ReLU activation function, Transformers require the secure evaluation of numerous complex **non-linear** functions such as Gaussian Error Linear Unit (GeLU), Softmax, and LayerNorm. These functions must be computed over hundreds of thousands of inputs, sub-

This work has been accepted for publication at the 11th IEEE European Symposium on Security and Privacy (EuroS&P 2026).

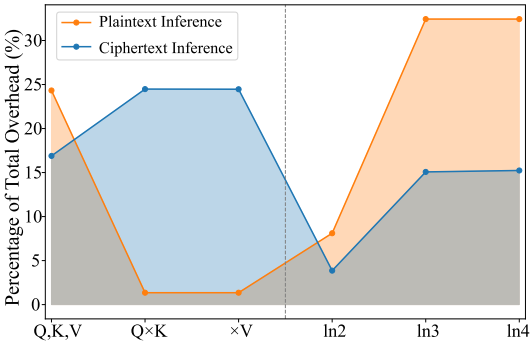


Figure 1. Cost Distribution Comparison Between Plaintext and Ciphertext Transformer Inference (BERT-base, Input Sequence Length of 128).

stantially increasing both computational and communication overhead in privacy-preserving inference [19], [30].

Recent privacy-preserving Transformer frameworks have made notable progress in improving inference efficiency. However, secure Transformer inference remains highly resource-intensive. For instance, executing BERT-base on a single input from the MRPC dataset can exceed 1,000 seconds under WAN network conditions [23], [24], rendering it impractical for real-world deployment. Addressing this inefficiency is therefore critical.

Existing Token Drop Schemes and Limitation: In plaintext scenarios, a common strategy for reducing Transformer inference costs involves eliminating redundant tokens from the input sequence. For instance, PowerBERT [31] introduces a method that assesses the significance of word vectors based on Softmax outputs, guiding the dropping process to retain only the most relevant tokens. This approach effectively improves the efficiency of both linear and non-linear operations during plaintext inference. To extend this strategy to secure inference, [24] proposed a straightforward Word Elimination (W.E.) method. Subsequently, CipherPrune [32] introduced a token dropping technique that adaptively removes low-importance tokens based on a pre-learned significance threshold.

However, as our analysis indicates, existing secure token drop strategies are still suboptimal:

• **Inadequate token drop coverage in ciphertext scenarios.** Existing secure token drop strategies are largely inherited from their plaintext counterparts. This key limitation arises from fundamental differences in cost distribution between plaintext and ciphertext Transformer inference. Referring to Figure 4, the Transformer typically proceeds through several key components to complete the inference process. The linear operations occur sequentially as Q,K,V; Q×K; ×V; ln2; ln3; and ln4. (The detailed explanation of each component can be found in Section 2.3.) As illustrated in Figure 1, the cost distribution differs significantly between the two settings. In the plaintext scenario (shown in orange), the inference cost is primarily concentrated in ln2, ln3, and ln4, which together account for **73%** of the total cost. This observation drives the design of plaintext-based token drop method, where token removal is strategically applied after the ×V operation and before ln2 to maximize cost savings. In contrast, in the ciphertext scenario (shown in blue), the inference cost shifts toward earlier stages, with ln2, ln3, and ln4 contributing only **34%** of the total cost. As a result, token drop strategies based on plaintext cost patterns become substantially less effective when directly applied to ciphertext settings. Additionally, nonlinear operations such as Softmax—which are computationally lightweight during plaintext inference—incur significant overhead in ciphertext scenarios, further exacerbating the inefficiency of existing approaches. This inefficiency is compounded by the fact that both plaintext token drop methods and existing ciphertext token drop methods rely on Softmax outputs to assess token significance. As a result, Softmax itself cannot benefit from token dropping.

• **High-overhead and low-efficiency token drop decision strategy.** Another critical aspect is deciding which tokens to drop. Work W.E. employs Bitonic Sort to find the median score of input sequence and discards tokens with scores below the median. However, this approach incurs high complexity in MPC environments; for example, fully sorting a 256-element vector requires 13,824 operations, making it costly for real-time inference. CipherPrune uses a static threshold learned through fine-tuning, dropping tokens with scores lower than this fixed threshold. This threshold-based approach is commonly used in model pruning [33]. While the comparison with a fixed threshold is relatively inexpensive, learning an appropriate threshold requires costly fine-tuning. Moreover, the learned threshold tends to be dataset-dependent and lacks adaptability to diverse real-world inputs. As a result, when switching tasks or datasets, additional fine-tuning is typically necessary. In addition, static thresholding is not efficient when using packed HE primitives, which are commonly adopted in privacy-preserving AI. This is because different inputs may retain varying numbers of tokens after applying a static threshold. While such variability is generally non-critical in plaintext scenarios, it becomes problematic in ciphertext settings, where the inability to precisely control the number of retained tokens leads to suboptimal ciphertext packing. For example, if 16 tokens fit exactly into a single ciphertext, reducing a sequence from 32 to 16 tokens decreases the number of ciphertexts from two to one. However, if 17 tokens remain after static thresholding, two ciphertexts are still needed. This underscores the importance of precise control over the number of retained tokens to maximize packing efficiency.

• **Potential leakage concerns.** During token dropping, the token contents are encrypted, and the specific tokens being dropped are hidden from both parties as well. However, some token drop methods may still pose a risk of information leakage. For example, when use static thresholding, different inputs typically lead to different dropping decisions, that is, varying numbers of tokens are retained. Moreover, the same input consistently results in the same dropping pattern. This creates a one-to-one mapping between each input and its corresponding dropping decision, forming a unique "fingerprint" for each input. Although the content of the input remains encrypted, these "fingerprints" may leak side-channel information and thus introduce potential privacy risks.

These challenges motivate us to develop a Dynamic Token Drop methodology for encrypted Transformer Inference, **SecDTD**, which is more efficient, more secure, and incurs lower operational cost. SecDTD introduces the

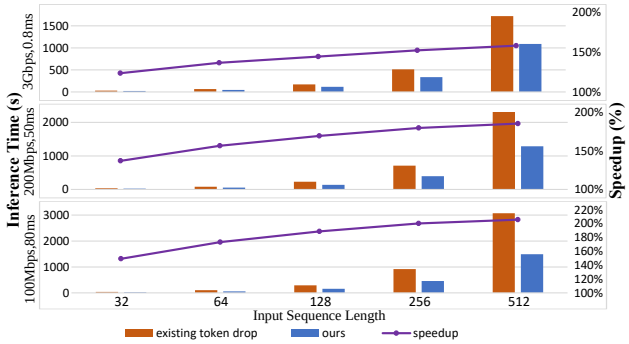


Figure 2. Comparison of Inference Time between Existing Token Drop Method and our SecDTD with "Pre-Softmax Gain".

following **key contributions**:

Pre-Softmax Gain Token Drop. SecDTD strategically shifts the token drop operation to earlier stages of Transformer inference, aligning with the unique cost distribution in ciphertext-based scenarios. SecDTD not only benefits later layers such as ln2, ln3, and ln4, but also extends the advantages of token drop to earlier operations, including Softmax, $\times V$, and even the initial computation of $I \times W_v = V$. This "pre-Softmax gain" effectively reduces the runtime cost of multi-head attention. As shown in Figure 2, we compare SecDTD with existing token drop method under three network conditions, varying the input sequence length from 32 to 512 tokens. Assuming that half of the tokens are dropped, SecDTD consistently outperforms the baseline across all configurations, due to the "pre-Softmax gain". In scenarios involving longer input sequences and more constrained network conditions, which closely resemble practical deployments, our approach demonstrates substantial advantages, achieving up to a **206%** speedup. Even greater gains can be expected as input lengths continue to grow in real-world applications.

Softmax-independent Scoring Method, MCN. To evaluate the importance of each token in the input sequence, existing scoring methods typically rely on the output of Softmax. This dependency prevents earlier stages, including Softmax itself, from benefiting from token dropping. To achieve the pre-Softmax gain token drop, we introduce a novel scoring method, Max-centric Normalization (MCN), which assesses token importance with three key advantages: **(1)** It is independent of Softmax, enabling token drop before Softmax, thus benefiting earlier stages. **(2)** The MCN algorithm incurs negligible additional overhead in MPC environments. **(3)** Unlike W.E.'s scoring mechanism, which lacks normalization, MCN incorporates normalization to better capture token importance while mitigating the impact of extreme values. This allows multiple rounds of token drop and supports high-ratio token drop without significant accuracy degradation, a challenge faced by W.E.

Fast Oblivious Median Selection, OMSel. Once token importance scores are obtained, determining which tokens to drop becomes crucial. Similar to W.E., we adopt a half-and-half token drop strategy, removing half of the current tokens at each interaction by discarding those with scores below the median. However, our approach differs significantly in how the median is computed. W.E. relies on Bitonic Sort, which has high complexity in

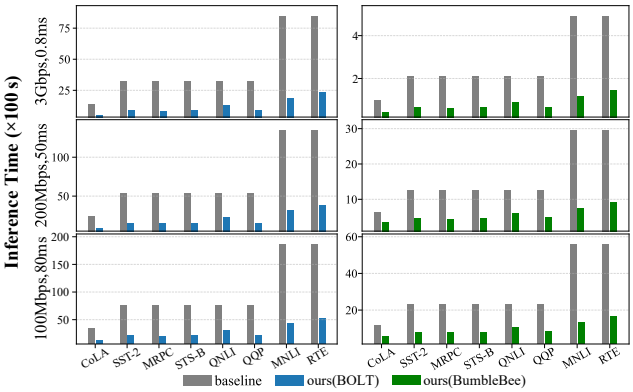


Figure 3. Performance Evaluation of SecDTD across Multiple Datasets, Network Environments and Frameworks.

MPC environments due to the complete sorting of token scores. To enhance efficiency, we propose a novel and secure median search method, Oblivious Median Selection (OMSel). Compared to W.E.'s sorting-based approach, OMSel achieves up to a **16.9**× speedup, making token drop practical for complex tasks with long input sequences and multiple token drop rounds. OMSel efficiently identifies the median while ensuring security and obliviousness. During the median search, neither the Client nor the Server gains information about the actual token scores, the relative rankings of tokens, or intermediate decisions. To maintain obliviousness, all tokens remain engaged in the search process, even if they are predetermined for removal or retention. Additionally, randomness is introduced to obscure access patterns and prevent response-based attacks. Importantly, since OMSel consistently drops the lower half of tokens, those below the median, across different inputs, it avoids creating a one-to-one mapping between each input and its dropping decision. This design effectively mitigates the risk of potential information leakage. (Refer to the detailed security analysis in Subsection 3.2.)

We conduct extensive experiments to validate the efficiency of SecDTD. Evaluations are performed through 48 sets of experiments on BERT-base across eight datasets: CoLA, SST-2, MRPC, STS-B, QNLI, QQP, MNLI, and RTE, covering various input lengths and task types. These evaluations span three network environments and are implemented within both the BOLT and BumbleBee frameworks[1], as illustrated in Figure 3. SecDTD achieves up to **4.47**× end-to-end acceleration compared to the baseline, while maintaining model accuracy without any fine-tuning. With quick fine-tuning (within 5 epochs), further speedups are observed. For example, with fine-tuning, SecDTD accelerates BERT-base inference on RTE from **3.51**× to **4.04**× since fine-tuning helps recover accuracy losses caused by more aggressive token drop. More details are provided in Section 4.

2. Preliminaries

2.1. Secure Inference and Threat Model

Follow the secure inference protocol for Transformer models, such as Iron [23], BOLT [24], BumbleBee [25],

1. In this paper, the two frameworks employ different levels of thread optimization; please focus on the **ratio** rather than the absolute values. See Section 4.2 for details.

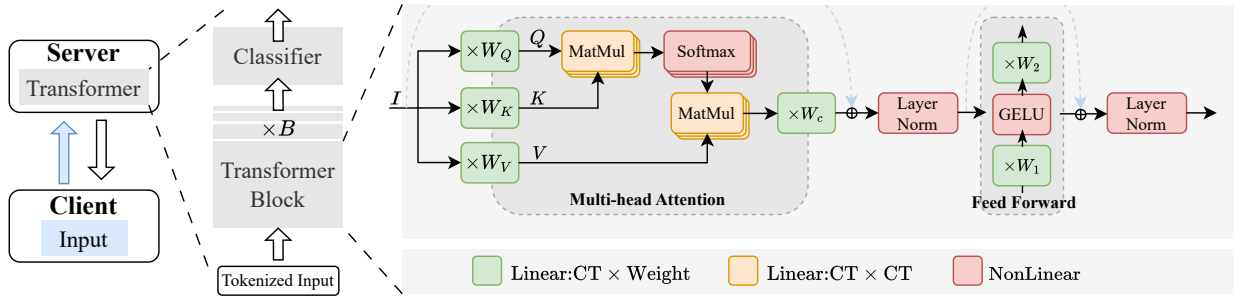


Figure 4. Overview of Secure Transformer Inference.

and Nexus [26]. As shown in Figure 4, the service provider maintains a fine-tuned Transformer model with private weights, offering it privately to users who hold private input data. This represents a typical secure two-party computation scenario. The secure Transformer inference framework enables the client to query the server's inference service and obtain the model's output on its input.

Following existing work [19]–[22], [24], [25], we consider privacy protection against a semi-honest adversary, where the two mutually distrusting parties, the server and the client, are honest-but-curious. That is, we assume both parties follow the protocol specification but may attempt to passively infer additional information. Semi-honest security is a common assumption in privacy-preserving machine learning (PPML). Specifically, we assume that the model architecture is known to both parties. As is standard in secure computation protocols, we do not aim to conceal the inference result, as its disclosure is inherent to the ideal functionality of such protocols—the result must be accessible to at least one party, and often both, to be useful. However, orthogonal techniques such as differential privacy [34] can be integrated with the secure Transformer inference framework to mitigate information leakage.

Notably, since this work serves as a solution for secure dynamic token drop, we proactively incorporate necessary security measures to safeguard the additional functionalities associated with token drop. For example, when analyzing the importance scores of input tokens to compute the median, we introduce a random pivot to enhance randomness and defend against response attacks in which a client may craft specific inputs to infer certain model characteristics. Additionally, we incorporate obliviousness into the median-searching process to obscure the relative importance relationships among tokens. For instance, information such as token-A being more important than token-B remains confidential in our design for both parties. For a detailed security analysis, refer to Subsection 3.1 and Subsection 3.2.

2.2. Transformer

In this work, we focus on BERT [1], a widely adopted Transformer-based model. The overall architecture and workflow of BERT are illustrated in Figure 4. Throughout this paper, we consider the BERT-base configuration, which consists of $B = 12$ Transformer blocks (or layers). Each Transformer layer includes a multi-head attention mechanism followed by a feed-forward neural network.

Multi-head Attention: The attention mechanism is designed to capture context and dependencies among tokens in the input sequence. Specifically, the input $I \in \mathbb{R}^{m\times d}$ is linearly projected into the Query, Key, and Value, denoted as Q, K, and V, using the parameter matrices W_Q, W_K, and W_V, respectively:

$$\begin{aligned} Q &= I \times W_Q, \\ K &= I \times W_K, \\ V &= I \times W_V, \end{aligned} \tag{1}$$

The attention output for each head, denoted as Att_h, is computed as follows, concatenated across all heads, and then passed to a feed-forward network:

$$Att_h = Softmax(A_h) \times V_h, \tag{2}$$

where A_h is defined as:

$$A_h = \frac{Q_h \times K_h^T}{\sqrt{d/H}}, h \in [H] \tag{3}$$

and the Softmax function is applied as follows:

$$Softmax(x)_{ij} = \frac{e^{x_{ij}}}{\sum_{j\in[m]} e^{x_{ij}}}, i \in [m], j \in [m] \tag{4}$$

Here, m denotes the sequence length (i.e., the number of tokens). In BERT-base, the model dimension is $d = 768$, and the number of attention heads is $H = 12$.

Furthermore, the attention output is normalized before being fed into the feed-forward. The layer normalization is defined as:

$$LayerNorm(x)_{ij} = \frac{\gamma_j(x_{ij} - \mu_i)}{\sigma_i} + \beta_j, i \in [m], j \in [d] \tag{5}$$

where μ_i and σ_i denote the mean and standard deviation of the i-th token's representation, and γ_j, β_j are learnable parameters.

Feed-forward: A feed-forward network (FFN) typically consists of two linear transformations with a GELU activation function applied between them, chosen for its favorable curvature and non-monotonicity properties [35]:

$$FFN(X) = GELU(XW_1)W_2, \tag{6}$$

$$GELU(x) = \frac{1}{2}x \cdot (1 + erf(\frac{x}{\sqrt{2}})), \tag{7}$$

where the Gaussian error function (erf) is defined as:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \tag{8}$$

2.3. Cryptographic Primitives

Secure Multi-party Computation (MPC). MPC techniques enable a group of parties to jointly evaluate a function over their private inputs without revealing any information about those inputs to each other [13], [14].

Additive Secret Sharing (SS). The MPC framework employed in this work uses 2-out-of-2 additive SS [36] between two parties, P_0 and P_1. Each party P_i ($i \in \{0,1\}$) holds a share $\langle x \rangle^i$ of a secret value $x \in \mathbb{Z}_{2^l}$ such that $x = \langle x \rangle^0 + \langle x \rangle^1 \mod 2^l$, where l denotes the bit length. With additive SS, additions can be performed locally without communication, while multiplications require interaction between the parties and are realized using oblivious transfer.

Oblivious Transfer (OT). OT is used to enable secure evaluation of nonlinear operations [37]. In a general 1-out-of-2 OT, denoted as $\binom{2}{1}$-OT$_l$, the sender provides two l-bit messages m_0 and m_1, and the receiver inputs a choice bit $c \in \{0,1\}$. At the conclusion of the protocol, the receiver obtains m_c, while the sender learns nothing about the receiver's choice [38]–[41].

Homomorphic Encryption (HE). HE is a foundational cryptographic primitive that enables secure computation by allowing linear operations to be performed directly on ciphertexts without requiring decryption [15], [16], [42]–[45]. This property permits computations on encrypted data, ensuring data confidentiality throughout the process. Standard HE schemes support operations such as encryption, decryption, homomorphic addition, and homomorphic multiplication. Some secure frameworks may also necessitate homomorphic rotation operations.

Secure Transformer Inference:

Within private frameworks such as BOLT and BumbleBee, Transformer inference can be divided into several **Stages** ($\mathcal{S}$_) and securely evaluated using different cryptographic primitives:

$\mathcal{S}$**(Q,K,V):** Referring to Formula 1, this operation uses HE multiplication in the form of CT $\times$ Weight, where I represents the HE-encrypted ciphertext, and W_Q, W_K, and W_V are weight matrices.

$\mathcal{S}$**(Q$\times$K):** This step is simplified as follows:

$$A = Q \times K^T, \tag{9}$$

In the BOLT framework, this operation uses HE multiplication as CT$\times$CT, where both Q and K are HE-encrypted ciphertexts. In the BumbleBee framework, following MPC principles, the multiplication is represented as the product of shared values:

$$\begin{aligned} A &= Q \times K^T \\ &= (\langle Q \rangle^0 + \langle Q \rangle^1) \times (\langle K^T \rangle^0 + \langle K^T \rangle^1), \\ &= \langle Q \rangle^0 \langle K^T \rangle^0 + \langle Q \rangle^0 \langle K^T \rangle^1 + \langle Q \rangle^1 \langle K^T \rangle^0 \\ &+ \langle Q \rangle^1 \langle K^T \rangle^1 \end{aligned} \tag{10}$$

Here, multiplication is performed using HE in the form of CT $\times$ Weight, as each party treats its own share as plaintext. *We adopt the BOLT configuration as the primary presentation in this paper, as illustrated in Figure 4.* Nevertheless, our proposed scheme remains compatible with the BumbleBee framework, with corresponding evaluations conducted under this setting.

$\mathcal{S}$**(Softmax):** In secure inference, the Softmax function is typically divided into two phases: exponential computation and division by the sum:

$$Softmax(x)_{ij} = \underbrace{(e^{x_{ij}})}_{\textbf{Ph-1}} \cdot \underbrace{(\frac{1}{\sum_{j\in[m]} e^{x_{ij}}})}_{\textbf{Ph-2}}, \tag{11}$$

Each phase can be implemented using linear approximation methods, as in BOLT [24] and BumbleBee [25], or with Look-Up Table (LUT)-based techniques, as proposed in Iron [23] and SIRNN [30].

$\mathcal{S}$**($\times$V):** Following Formula 2, this operation is performed similarly to $\mathcal{S}$(Q$\times$K) using CT $\times$ CT multiplication in the BOLT framework and CT $\times$ Weight multiplication in the BumbleBee framework.

$\mathcal{S}$**(ln2):** This step concatenates the attention outputs from all heads using HE multiplication in the form of CT $\times$ Weight, where X denotes the HE-encrypted ciphertext and W_c is a weight matrix:

$$Y = X \times W_c, \tag{12}$$

$\mathcal{S}$**(LayerNorm):** Referring to Formula 5, during secure inference, the LayerNorm function, which involves computing the reciprocal square root to obtain σ, is typically implemented using linear approximation methods [23], [24].

$\mathcal{S}$**(ln3):** Similar to $\mathcal{S}$(ln2), this operation involves HE multiplication in the form of CT $\times$ Weight, where X denotes the HE-encrypted ciphertext and W_1 is a weight matrix:

$$Y = X \times W_1, \tag{13}$$

$\mathcal{S}$**(GELU):** Referring to Formula 7, in secure inference, the GELU function is usually approximated using a multi-degree polynomial [23]–[25].

$\mathcal{S}$**(ln4):** Similar to $\mathcal{S}$(ln3). Here, X denotes the HE-encrypted ciphertext and W_2 is a weight matrix:

$$Y = X \times W_2, \tag{14}$$

3. System Description

This section sequentially introduces SecDTD, a Dynamic Token Drop method designed for Secure Transformer Inference. Subsection 3.1 proposes a novel pre-Softmax scoring scheme that evaluates token importance independently of Softmax outputs, thereby enabling early-gain token dropping. This is followed by a corresponding security analysis. Subsection 3.2 presents a Fast Oblivious Median Selection protocol that efficiently and securely identifies the median of the score vector, supporting precise half-and-half token dropping; this subsection is also accompanied by a dedicated security analysis. Finally, building upon these techniques, Subsection 3.3 introduces our token drop methodology that maximizes the efficiency of token drop.

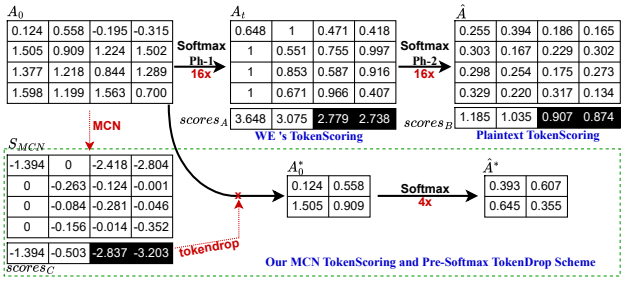


Figure 5. Comparison of Scoring Methods.

3.1. Pre-Softmax Token Scoring, MCN

As the initial step of Token Drop, it is crucial to efficiently and accurately assess the importance of each input token to eliminate less important tokens while maintaining inference accuracy. Scoring strategies for this purpose can be broadly categorized into static and dynamic approaches. Static strategies generate a fixed importance score list for input word vectors by analyzing the training dataset and summarizing inherent patterns. In this method, the importance of each position in the input sequence remains constant, regardless of the current input sequence, and token drop is performed accordingly. In contrast, dynamic scoring strategies evaluate the importance of each token within the current input sequence, dynamically producing an importance score list with regard to the specific input. This approach better reflects real-world inference scenarios, where user inputs can vary significantly, and the importance of a token at a given position varies depending on the input. Compared to static methods that rely on prior knowledge, dynamic scoring offers greater flexibility and responsiveness, making it more suitable for practical applications.

Existing dynamic scoring methods rely on the attention matrix $\hat{A} = \text{Softmax}(A_0)$, obtained through the Softmax operation, as illustrated in Figure 5, specifically in *Plaintext TokenScoring*. In the self-attention mechanism of Transformers, each element x_{ij} in the attention matrix $\hat{A}_{m\times m}$ represents the degree to which token i attends to token j. During attention computation, token i evaluates all tokens, including itself, to determine which ones to prioritize. The score x_{ij} quantifies this level of attention. A common approach to measuring token importance is to sum the values within a column of the attention matrix $\hat{A}_{m\times m}$, effectively aggregating all x_{ij} values for a fixed j. This aggregated score represents the total attention token j receives from all other tokens in the sequence. A higher score indicates that token j is attended to by multiple tokens, suggesting higher importance, while a lower score implies reduced relevance. For multi-head self-attention, importance scores are further aggregated across multiple attention heads by summing the corresponding attention matrices, providing a more comprehensive evaluation of token importance.

However, a clear limitation of existing methods is that the token importance score used for token drop depends on the Softmax output. This reliance means that Softmax must be completed before token drop occurs, negating any potential efficiency gains. While this issue is often overlooked in plaintext inference, where the Softmax computation accounts for a negligible portion of runtime, it becomes a significant and pressing concern when transitioning to secure inference, where the cost of Softmax increases sharply. To address this problem, we propose a scoring method that eliminates the need for Softmax. Our approach accurately evaluates token importance while introducing minimal additional overhead.

First, we observe that the raw attention matrix $A_0 = Q \times K^T$ contains the unprocessed attention information for each token. However, these independent attention values, having not undergone Softmax normalization, cannot be directly integrated to reliably reflect token importance. Softmax, as a normalization operation, consists of two core components—exponential transformation (Ph-1) and summation-based normalization (Ph-2)—as described in Formula 11. Its primary functions are nonlinear scaling and normalization. Notably, in the Multi-Party Computation (MPC) environment, the exponential operation is particularly costly, accounting for over 80% of the Softmax runtime. *Through our evaluation, we find that while the exponential is necessary for Softmax function, its precision is excessive for scoring token importance.* Based on experimental verification, we propose a Token Importance Scoring method optimized for MPC environments—**Max-Centric Normalization (MCN)**. MCN normalizes each element within a set by calculating its deviation from the maximum value and applying an additional scaling factor to reduce its magnitude. The formal definition of MCN is provided as follows:

$$\begin{aligned} MCN(x)_{ij} &= \frac{x_{ij} - \max_i}{\max_i^n}, \\ &= \left(\frac{x_{ij} - \max_i}{\max_i}\right) \times \left(\frac{1}{\max_i}\right)^{n-1} \end{aligned} \tag{15}$$

where $\max_i$ is the maximum value of the i-th row $[x_{i0}, x_{i1}, ..., x_{im}]$, with $i \in [m]$ and $j \in [m]$. The parameter n can be adjusted according to the scale of values, and for the dataset used in this work, we set $n = 2$.

Interpretation and Benefits:

- **Deviation from Maximum:** The term $\frac{x-\max}{\max}$ captures the relative deviation of each element from the maximum, bringing values closer to zero as they approach the maximum.
- **Additional Scaling Effect:** The factor $\left(\frac{1}{\max}\right)^{n-1}$ serves as a scaling mechanism, further reducing the range of values. This is particularly beneficial in datasets with large or extreme values, as it compresses values closer to zero and minimizes the influence of large values and outliers.
- **Efficiency in MPC Environments:** MCN is designed with two core components: $(x - \max)$ and $(\max^n)$. Since the $\max$ value, as a prerequisite for the Softmax calculation, is already computed, it does not incur additional overhead. In this way, the former can be executed locally "for free" in MPC environments with Additive Secret Sharing. The latter can be simplified into a reciprocal multiplication operation, which is negligible compared to the overall inference cost.

Overall, the design principle of MCN is to fulfill the required functionality while minimizing additional overhead. MCN provides a refined method for data normalization, effectively balancing deviation with controlled mag-

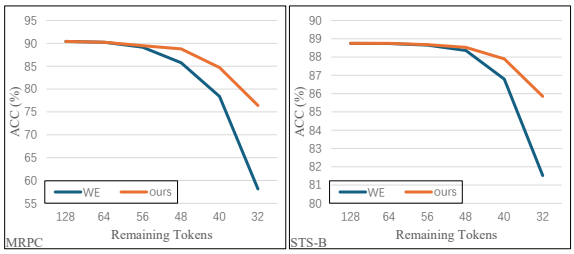


Figure 6. Token Drop Applied to the 1st BERT Layer, Reducing Tokens from 128 to 64, 56, 48, 40, and 32.

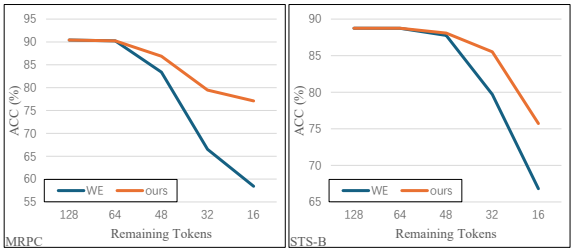


Figure 7. Token Drop is Progressively Applied from the 1st to 4th Layer, Reducing the Tokens from 128 to 64, 48, 32, and 16 at each Layer.

nitude adjustment through scaling. This facilitates token drop before the Softmax operation, significantly enhancing the benefit of token drop. As shown in Figure 5, while existing approaches require conducting Softmax on 16 values, our method reduces this to only 4 values.

Moreover, the token scores generated by MCN enable a more precise evaluation of token importance. The existing scoring method, W.E., possibly for the sake of simplification, utilizes the intermediate result of Softmax Ph-1. However, without the Softmax Ph-2 component, W.E.'s scoring process may be influenced by extreme values due to insufficient normalization. In contrast, the proposed method, incorporating both normalization and scaling function, is more robust in scenarios involving high dynamic range data with extreme values, which commonly occur in extreme compression scenarios. To simulate these scenarios and evaluate the robustness of each scoring scheme, we conducted two sets of stress test suites on the MRPC and STS-B datasets with an input sequence length of 128:

Test-1: Single-Layer Gradual Token Drop Challenge. As depicted in Figure 6, token drop is applied to the first layer of BERT, gradually reducing the number of tokens from 128 to 64, 56, 48, 40, and finally 32.

Test-2: Cumulative Layers Token Drop Challenge. As shown in Figure 7, token drop is applied progressively from the 1st to 4th layer, reducing the number of tokens from 128 to 64, 48, 32, and 16 at each respective layer.

In both scenarios, when token drop is relatively mild, both methods perform well. However, under deep token drop conditions, W.E. experiences a rapid collapse in accuracy, whereas MCN, despite some accuracy loss, remains significantly more stable. This highlights the superior robustness of the MCN method in handling extreme values. Notably, even with a potential accuracy drop, the W.E. approach does not achieve simplification very well. This is because, in the Softmax operation, Ph-1 remains the dominant computational component, while the omitted Softmax Ph-2 accounts for only about 20% of the total Softmax overhead in an MPC environment.

Finally, MCN offers another advantage through its contribution to the subsequent median searching process. The score vector processed using MCN exhibits a well-structured distribution, characterized by a smooth and gradually decreasing trend, as shown in Figure 15. This property ensures that the **average** closely approximates the **median**, thereby accelerating the average-as-pivot median searching process. Further details will be discussed in the following section.

Security Analysis of MCN. The security analysis of the proposed MCN is straightforward. As shown in Formula 15, MCN reuses several existing cryptographic protocols for basic computations, including multiplication, addition, reciprocal, and negation, all of which can be securely realized using OT in a MPC environment. As a result, MCN is considered secure.

3.2. Fast Oblivious Median Selection, OMSel

Once the importance scores of all tokens are obtained, the next crucial step is determining which tokens to be dropped based on these scores. W.E. [24] adopts a half-and-half token drop approach, removing half of the current tokens at each interaction by discarding those with scores below the median. CipherPrune [32] uses a static threshold learned through fine-tuning, dropping tokens with scores lower than this fixed threshold. As discussed earlier, static thresholding not only incurs the cost of fine-tuning to determine the optimal threshold but also suffers from poor packing efficiency when used with packed HE primitives, as it lacks precise control over the number of retained tokens. More critically, it may introduce potential information leakage. Similar to W.E., we use the half-and-half token drop strategy. However, we propose a novel protocol that significantly enhances the median searching process. The existing W.E. scheme relies on Bitonic Sort. While it benefits from parallel processing, it still incurs high complexity in MPC environments. Therefore, an optimal design must prioritize two key objectives: reducing complexity and ensuring security.

To address these challenges, we propose a secure and fast median search method, **Oblivious Median Selection (OMSel)**, as illustrated in Figure 8. OMSel efficiently identifies the median while maintaining security, ensuring obliviousness and randomness. Additionally, Figure 16 in Appendix provides a toy example that demonstrates the OMSel process in practice. In the following discussion, we introduce the design principles and advantages of OMSel.

In MPC environments, complexity is primarily determined by two costly operations: **#Cmp** and **#Mux (denoted as** $\otimes$**)** [46], [47]. #Cmp, or Secure Comparison, is a protocol that allows two or more parties to compare numerical values without revealing their inputs, commonly applied in the Millionaire's Problem. #Mux, or Secure Multiplexer, is used to select an output from multiple data paths based on a selection bit. In secure computation, this is typically implemented as a Secure Select operation. In secure inference frameworks, these operations are generally executed using Secret Sharing and OT. Their overhead dominates the median search process [39]. Therefore, evaluating the number of #Cmp and #Mux operations is essential for assessing the efficiency

of different median search designs. For example, Bitonic Sort requires 4608 #Cmp and 9216 #Mux operations to find the median of a 256-element vector. This introduces considerable overhead, taking over 18 seconds under a 200 Mbps, 80 ms network condition.

Oblivious Median Selection (OMSel)

Parties: Server-ALICE (A), Client-BOB (B)

Input & Output:
$\langle Med\rangle^A, \langle Med\rangle^B \leftarrow \text{OMSel}(\langle \mathbf{s}\rangle^A, \langle \mathbf{s}\rangle^B, N)$

Initialization:
- $N \leftarrow Len(\mathbf{s})$; $P \leftarrow N$; $k \leftarrow N/2$.
- $EXIT \leftarrow P = 1 \vee P_i = P_{i-1}$.
- $i \leftarrow 0$; $\langle V_0\rangle^{A(B)} \leftarrow \langle \mathbf{s}\rangle^{A(B)}$.

Selection Loop:

1: Calculate the $Pivot$:

if $(i = 0)$**:** Pick one random element of V_i as $Pivot$
$\langle Pivot\rangle^A, \langle Pivot\rangle^B \leftarrow$ **RDM**$(\langle V_i\rangle^A, \langle V_i\rangle^B)$
else: Calculating the average of V_i as $Pivot$
$\langle Pivot\rangle^A, \langle Pivot\rangle^B \leftarrow$ **AVG**$(\langle V_i\rangle^A, \langle V_i\rangle^B, P)$

1.1: function **RDM**:

1.1.1: (offline) Create the random mask M_R:

ALICE and BOB create two random value $r_1, r_2 \in \mathbb{Z}\{1, 2, ..., N\}$. Then make the random mask generation (**RGM**) such that $\langle M_R\rangle^A, \langle M_R\rangle^B \leftarrow$ **RGM**(r_1, r_2, N):
$Cond \leftarrow$ **Cmp**$((r_1, r_2), N)$.
$r_{A(B)} \leftarrow r_{1(2)} - 0.5N \otimes Cond$.
$\langle M_R\rangle^A, \langle M_R\rangle^B \leftarrow \mathcal{F}_{A2B}(\mathcal{F}_{BeaverTriples}(2^{r_A} \times 2^{r_B}))$.

1.1.2: Calculate the $Pivot$:

$(\langle V_i^*\rangle^A, \langle V_i^*\rangle^B \leftarrow (\langle V_i\rangle^A, \langle V_i\rangle^B) \times (\langle M_R\rangle^A, \langle M_R\rangle^B)$.
$\langle Pivot\rangle^{A(B)} \leftarrow$ **Sum**$(\langle V_i^*\rangle^{A(B)})$.

1.2: function **AVG**:

$\langle Pivot\rangle^{A(B)} \leftarrow$ **Sum**$(\langle V_i\rangle^{A(B)})/P$.

2: Compare V_i with $Pivot$ to generate the mask M_i:

$\langle M_i\rangle^A, \langle M_i\rangle^B \leftarrow$ **CMP**$((\langle V_i\rangle^A, \langle V_i\rangle^B), (\langle Pivot\rangle^A, \langle Pivot\rangle^B))$

2.1: function **CMP**:

$\langle m_j\rangle^{A(B)} \in \langle M_i\rangle^{A(B)}$; $\langle v_j\rangle^{A(B)} \in \langle V_i\rangle^{A(B)}$.
for $j \in 0$ to N **do**:
if $(v_j \leq Pivot)$**:** $(\langle m_j\rangle^A, \langle m_j\rangle^B) \leftarrow 1$
else: $(\langle m_j\rangle^A, \langle m_j\rangle^B) \leftarrow 0$

3: $P_L \leftarrow$ **Sum**$(\langle M_i\rangle^A, \langle M_i\rangle^B)$; $P_R = P - P_L$

4: Decision and update:

if $(k \leq P_L)$**:** $k \leftarrow k$; $P \leftarrow P_L$; $\langle M_i^*\rangle^{A(B)} \leftarrow \langle M_i\rangle^{A(B)}$
else: $k \leftarrow k - P_L$; $P \leftarrow P_R$;
$(\langle M_i^*\rangle^A, \langle M_i^*\rangle^B) \leftarrow$ **Inv**$(\langle M_i\rangle^A, \langle M_i\rangle^B)$

5: if $EXIT$=$TRUE$**:**
End & **Return** $(\langle Med\rangle^{A(B)} \leftarrow \langle Pivot\rangle^{A(B)})$
else:
$(\langle V_{i+1}\rangle^A, \langle V_{i+1}\rangle^B) \leftarrow (\langle V_i\rangle^A, \langle V_i\rangle^B) \otimes (\langle M_i^*\rangle^A, \langle M_i^*\rangle^B)$
$i \leftarrow i + 1$

6: Return to **Step 1** and repeat.

Figure 8. Protocol Oblivious Median Selection (OMSel).

Bitonic Sort is excessive for the median-searching task, as it sorts the entire score array and introduces unnecessary computational overhead. *Given that the objective is merely to identify a single element—the median—this inefficiency motivates the adoption of more targeted approaches, drawing inspiration from selection algorithms.* A classic selection algorithm is QuickSelect, which efficiently finds the k-th smallest element in an array. It is based on the partitioning strategy of QuickSort but only recurses on a portion of the data, making it more efficient than full sorting in terms of time complexity. However, QuickSelect is not **oblivious**, meaning it does not hide access patterns. Its partitioning process, which progressively narrows the recursion range, directly exposes data comparisons and swaps, leading to potential information leakage. In an MPC environment, the non-deterministic *access pattern* of QuickSelect could reveal sensitive information about the data distribution [48], [49]. To address this, we leverage the advantages of partitioning while ensuring obliviousness. Instead of directly discarding the "eliminated" elements, we use a secure mask to track the status of each element (whether it is eliminated or not), maintaining a consistent access pattern throughout the process. This creates a **pseudo-partitioning** process, where elements are not physically moved or removed but are instead logically masked. By doing so, we retain the efficiency benefits of partitioning while preventing information leakage, ensuring a fully oblivious execution.

Next, as a selection protocol that employs a partitioning strategy, choosing an **effective pivot** is crucial. A well-chosen pivot can significantly enhance the efficiency of the selection process. Furthermore, since this protocol operates within a secure inference framework, the pivot must remain confidential throughout the entire process to prevent any potential information leakage. To achieve this, we strategically select the average value of the score vector as the pivot for the following reasons:

- **Security and Cost Efficiency:** The averaging operation involves summation and division by the total count. Summation can be performed locally in a secure and efficient manner within MPC environments using Additive Secret Sharing. And the division by total count N is secure because N remains constant, as our pseudo-partitioning strategy retains all elements of the score vector.
- **Performance Enhancement:** Selecting a pivot close to the median is desirable for median selection, and the average value serves as an effective approximation. As discussed in the previous section, MPC ensures a gradually decreasing value distribution, which allows the average to closely approximate the median, as illustrated in Figure 15. Consequently, using the average as the pivot accelerates convergence in median selection.

These design choices integrate seamlessly to enhance the overall performance of our protocol. However, relying entirely on the average as the pivot introduces a potential risk. Unlike a selected pivot, the average is computed rather than chosen. For a score vector with identical values, the computed average will always be the same. This means a client could craft a specifically designed input that results in an identical score vector and, consequently, the same average as the pivot. While this does not expose individual element values or access patterns, a *consistent operational pattern* may still pose a security risk against **response attacks** from a malicious client [50], [51]. Although malicious clients are outside the scope of our threat model, which assumes a semi-honest setting, and response attacks can be mitigated through additional techniques such as differential privacy [34], we take a proactive approach to eliminate this risk at its source. Our solution introduces **randomness** into the pivot selection process. Specifically, in the initial round, we use a randomly selected element as the pivot (**RDM**). To securely set

this random pivot, during the offline initialization phase, the two-party computation (2PC) participants—the server and the client—each generates its own random number, r_1 and r_2, respectively. These values are used to jointly determine a secret index pointing to a random position within the score vector. Finally, a corresponding random mask is generated using Beaver Triples, with each party holding a share. During the online inference phase, this random mask enables the efficient selection of an element from the score vector as the pivot, thereby introducing randomness and fundamentally eliminating the risk of response attacks. (*Note: In the toy example shown in Figure 16, RDM is omitted for simplicity of presentation.*)

Security Analysis of OMSel. We now analyze the security of the proposed OMSel. In the semi-honest setting, the goal is to prevent the server from learning the client's private input data, and conversely, block the client from accessing the server's model parameters. First, all operations are performed using established MPC primitives, including multiplication, addition, secure comparison, and multiplexer functions. This ensures that the actual token score values and masks remain securely hidden from both the client and the server. Second, to maintain obliviousness, we adopt a pseudo-partitioning strategy in the median search. This ensures that all token scores remain logically engaged in the search process—no real partition or elimination is conducted. As a result, neither party can infer any information about the relative ranking between tokens (e.g., whether token A is more significant than token B) or any intermediate decisions regarding retention or removal. Third, randomness is introduced through randomized pivot selection. This prevents the formation of deterministic access patterns and blocks the client from crafting special inputs that could infer model characteristics based on consistent operational traces. Therefore, under the semi-honest model, neither party can actively extract the other's secret.

We further examine the potential for accidental leakage. For instance, could a weak input—such as a sequence where all tokens are identical—lead to identical token scores and thus a deterministic exit pattern that reveals input distribution? The answer is NO. Even if the input tokens are identical, Transformer models incorporate positional encoding, making each token's representation unique. Additionally, the attention mechanism is a highly non-linear process; it does not map identical inputs to identical scores. Empirically, we have never observed two tokens with exactly identical significance scores in any evaluation.

Finally, our design is consistent with prior work under the standard semi-honest assumption [19], [22], [24], [25], [32]. However, we also aim to enhance security beyond this assumption. As noted earlier, some existing token drop methods pose a risk of information leakage due to the one-to-one mapping between specific inputs and token drop decisions. This behavior can act as a "fingerprint", introducing potential vulnerabilities under stronger threat models such as adversarial settings with more powerful attackers. To eliminate this risk, the proposed SecDTD incorporates two key design choices. First, the same input does not consistently lead to a fixed token drop pattern, owing to the intentional introduction of randomness, as previously discussed. Second, SecDTD enforces consistent drop decisions across different inputs for the same task—that is, the same number of tokens is dropped regardless of input variation. While this uniformity may slightly reduce flexibility in achieving optimal speedups, the resulting security benefits are significant. It completely removes the one-to-one mapping between inputs and drop patterns, thereby mitigating the fingerprinting threat.

Complexity Comparison. Table 2 presents the complexity of each method, specifically the number of #Cmp and #Mux operations, for Bitonic Sort used by W.E. and our proposed OMSel. It is evident that OMSel offers a complexity advantage, which becomes even more pronounced as the size of the score array, i.e., the input sequence, increases. A detailed complexity analysis and proof can be found in the Appendix-B. Table 1 provides comprehensive evaluations, displaying the exact counts of #Cmp and #Mux operations, as well as the execution runtime across different dataset tasks. Due to the inherent randomness in the process, the reported performance of OMSel represents the average result over five evaluations. Compared to Bitonic Sort, OMSel achieves up to $\mathbf{16.9}\times$ acceleration, significantly reducing the execution runtime. This substantial improvement enhances the practicality of OMSel for processing long input sequences and supporting multiple rounds of token drop.

3.3. Dynamic Token Drop for Secure and Efficient Transformers Inference, SecDTD

With the introduction of **MCN**, which scores token importance prior to the Softmax operation, and **OMSel**, which quickly and securely identifies the median of the token score array to determine token removal by discarding those with scores below the median, the proposed **SecDTD** strategically advances the token drop operation to earlier stages of Transformer inference. This adjustment aligns with the unique cost distribution in ciphertext-based scenarios. SecDTD not only optimizes $\mathcal{S}$(ln2), $\mathcal{S}$(ln3), and $\mathcal{S}$(ln4), but also extends the benefits of token dropping to earlier operations, including $\mathcal{S}$(Softmax), $\mathcal{S}(\times\mathrm{V})$, and even the initial computation stage of $I \times W_v = V$.

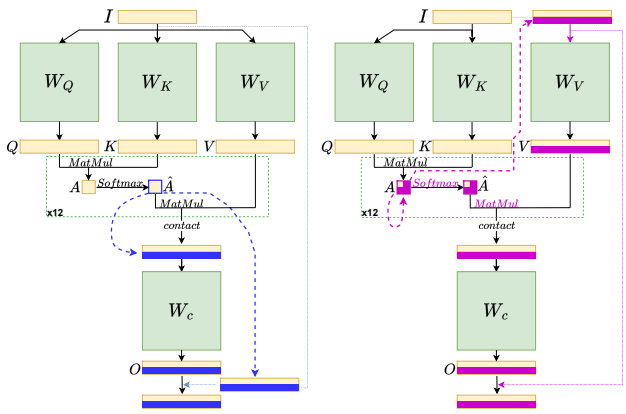


Figure 9. Comparison of Token Drop Schemes: Existing Approaches without Pre-Gain (Left) vs. SecDTD (Right). Highlight Differences in Where and How Token Drop (color-coded) is Applied, Demonstrating How SecDTD's Pre-softmax Design Benefits More Stages of Inference.

Figure 9 illustrates the token drop mechanisms of both existing approaches [24], [32] and the proposed SecDTD.

TABLE 1. PRACTICAL COMPLEXITY COMPARISON: OMSEL VS. BITONIC SORT ACROSS REAL-WORLD DATASETS.

Dataset		CoLA	SST-2	MRPC	STS-B	QNLI	QQP	MNLI	RTE
Input Sequence Length		64	128	128	128	128	128	256	256
Bitonic Sort	# Cmp	672	1792	1792	1792	1792	1792	4608	4608
	# Mux	1344	3584	3584	3584	3584	3584	9216	9216
OMSel	# Cmp	88	176	404	217	382	210	339	598
	# Mux	88	176	404	217	382	210	339	598
Aggregated Speedup		**9.6×**	**12.8×**	**5.5×**	**10.4×**	**5.9×**	**10.7×**	**16.9×**	**9.6×**

TABLE 2. COMPLEXITY COMPARISON: OMSEL VS. BITONIC SORT.

Method	Bitonic Sort	OMSel
# Cmp	$N \log^2 N$	$N \log N$
# Mux	$2N \log^2 N$	$N \log N$

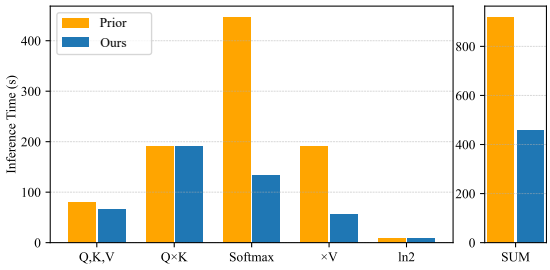


Figure 10. Runtime Comparison in BOLT framework.

On the left, prior methods adopt a token drop strategy similar to that used in plaintext settings, where the Softmax output, $\hat{A}$, is utilized to drop tokens from the input sequence at stage $\mathcal{S}$(ln2), formulated as $Y(X) = X \times W_c$, as well as from the residual connection sequence. This strategy provides computational benefits beginning at $\mathcal{S}$(ln2) and propagates them through subsequent stages. On the right, the SecDTD token drop mechanism is illustrated. Instead of relying on the Softmax output, SecDTD directly uses the raw attention matrix A from $\mathcal{S}$(Q×K), supported by MCN and OMSel, to perform token drop on A itself, resulting in a reduced input to Softmax. Additionally, such an earlier token drop is applied to the input of $V = I \times W_v$. Compared to existing methods, this design allows SecDTD to extend optimization to cover $I \times W_v = V$, Softmax, and ×V, maximizing performance gains. As shown in Figure 10, within the Secure Transformers Inference Framework BOLT, SecDTD achieves a significant reduction in inference time. With an input length of 256 tokens, SecDTD delivers a **2.04**× speedup over existing methods. Under another widely adopted framework, BumbleBee, depicted in Figure 11, the performance gain becomes even more substantial, achieving a **3.37**× inference time speedup. This improvement is primarily due to the higher proportion of Softmax in BumbleBee (note the logarithmic scale on the Y-axis).

Our dropping process follows the same procedure as prior token drop methods [24], [32], where tokens with scores below the median are iteratively moved to the end of the sequence using an OT-based oblivious swap, after which the tail tokens are discarded. The overhead introduced by this dropping process is minimal, accounting for less than 1% of the overall inference time [24], [32]. The following section, Section 4.1, provides a detailed, layer-wise, end-to-end implementation of the token drop mechanism, demonstrating how the SecDTD method is applied to a real-world Transformer model, BERT-base.

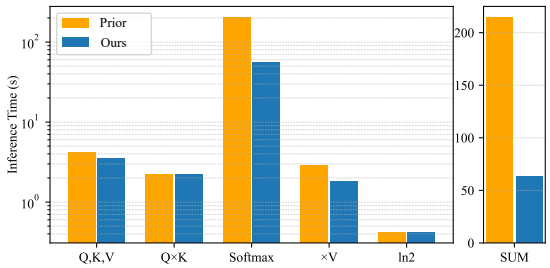


Figure 11. Runtime Comparison in BumbleBee framework.

3.4. Security Guarantee

SecDTD is built upon the state-of-the-art secure Transformer inference frameworks BOLT [24] and BumbleBee [25], operating under the semi-honest adversary model. The security of linear computations is guaranteed by the semantic security of HE schemes, such as BFV [52] and CKKS [53]. For nonlinear operations, the framework employs Secret Sharing (SS) [36] and Oblivious Transfer (OT) [37]. All operations introduced by SecDTD to enable MCN and OMSel—including addition, multiplication on secret shares, secure comparisons, and multiplexer functions—are securely executed in a two-party computation (2PC) setting between the Server and the Client, using Additive SS and OT. This protocol design aligns with the secure computation of nonlinear functions such as Softmax. Based on the security analyses provided at the end of the MCN and OMSel subsections, the proposed SecDTD does not compromise the client's input data, nor does it allow the client to access the server's model parameters. Overall, SecDTD is considered secure under the semi-honest assumption. Attacks that fall outside this threat model are beyond the scope of this work.

4. Evaluation

4.1. Implementation

For real-world implementation, we apply token drop on one Transformer layer using the proposed SecDTD mechanism, which involves MCN pre-Softmax scoring and OMSel median selection. A Transformer model, such as BERT, consists of multiple layers. By leveraging MCN's precise importance evaluation, token drop can be applied multiple times without compromising model accuracy. SecDTD operates as a semi-flexible dynamic mechanism: the layers where token drop is applied are predetermined, while the specific tokens to be dropped are selected dynamically. Figure 12 illustrates the layer-wise, end-to-end implementation of the BERT-base model, which consists of 12 layers, on the MRPC dataset under the BOLT framework, where token drop is applied three times — specifically on layer-1, layer-5, and layer-8. These layers were manually chosen for simplicity;

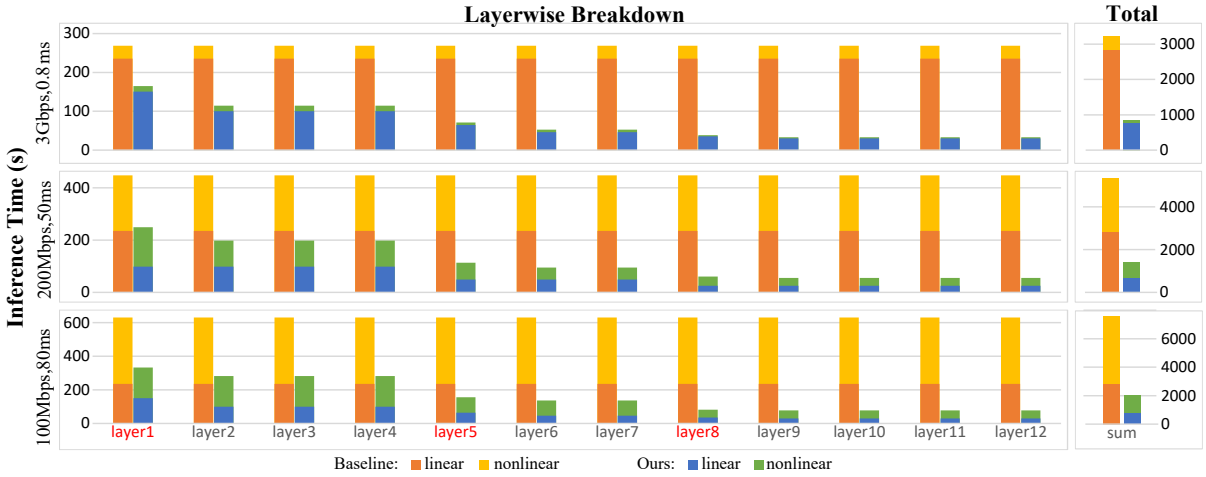


Figure 12. Implementation of BERT-Base on MRPC (under BOLT Framework).

however, an offline auto-search could be conducted before implementation to identify the optimal layers for token drop, depending on the specific dataset and task [31], [54].

Notably, the evaluation is conducted without fine-tuning, as SecDTD dynamically scores each individual input sequence and performs precise token drop. This capability allows it to minimize inference overhead while maintaining accuracy to the greatest extent. In the figure, the overall inference time is decomposed into a linear component, arising from linear computations primarily influenced by computational power, and a nonlinear component, resulting from nonlinear operations mainly affected by network conditions. Under high network speeds, linear computations dominate, whereas under low network conditions, the nonlinear component becomes the primary factor in the overall inference time. SecDTD achieves significant improvements in both linear and nonlinear components, thereby reducing the overall cost in all scenarios. This reduction is evident in both the layer-wise and end-to-end costs, as shown on the right side of the figure. With a threefold token drop, the original input sequence is progressively reduced from 128 tokens to 64 tokens, then to 32 tokens, and finally to just 16 tokens. This substantial reduction significantly decreases inference time, achieving an overall speedup of more than $3.75\times$. Further evaluation results are presented in the following section.

4.2. Experimental Setup

The experiments presented in this work are conducted on a testbed equipped with AMD 3995 64-core CPU. To simulate diverse network conditions, we utilized Linux Traffic Control. Specifically, evaluations were performed under three scenarios: **LAN** (3 Gbps, 0.8 ms), **WAN** (200 Mbps, 50 ms), and **Mobile** (100 Mbps, 80 ms). These configurations enable performance evaluations under varying bandwidths and latencies, aligning with real-world network conditions [55].

We implement SecDTD on the **BOLT framework** [24], utilizing the Secure and Correct Inference (SCI) library from EzPC [56] and the SEAL [57] library, as well as on the **BumbleBee framework** [25], which is built on top of the SPU library [58], to evaluate its cost reduction performance. On each framework, we follow their default settings and parameters. Our objective is to evaluate the performance of SecDTD within each framework, specifically its speedup relative to the baseline. *Notably, in experiments, the BOLT framework is configured with the basic number of threads, 2 threads, to ensure precise measurement, whereas the BumbleBee framework, configured by the SPU library, incorporates extensive optimizations and utilizes multi-threading*. Therefore, the focus should be on relative ratios and speedup factors in each experiment rather than absolute values or unit differences across frameworks.

To demonstrate the effectiveness of our SecDTD token drop mechanism, we evaluate it on eight datasets from the GLUE benchmark [59], a widely used standard for assessing BERT's performance. These datasets cover a range of linguistic tasks, including the Acceptability (ACPT) task: **CoLA**, the Sentiment (SENT) task: **SST-2**, the Paraphrase (PARA) task: **MRPC**, the Similarity (SIM) tasks: **STS-B** and **QQP**, the QA/NLI task: **QNLI**, and the NLI tasks: **MNLI** and **RTE**. These tasks involve varying input sequence lengths, with token counts ranging from 64 to 128 and 256 tokens. We employ trained models sourced from publicly available repositories [60]–[62].

4.3. SecDTD Performance on BOLT Framework

4.3.1. End-to-End Performance. We present the end-to-end performance of SecDTD on eight datasets under three network settings in Table 3. Without any fine-tuning, SecDTD achieves substantial reductions in inference time, with speedups of up to $\mathbf{4.47}\times$, all while preserving model accuracy. Although the reported inference time in the table may appear high, it is important to note that experiments were conducted using 2 out of 128 CPU threads to ensure precise measurement and fair comparison with the baseline. In real-world deployments, multi-threading or multi-CPU setups are standard practice. For instance, with full utilization of dual CPUs, the MNLI task, originally requiring 70.5s, can be reduced to an estimated 15.7s with the $4.47\times$ speedup, underscoring the practicality of our method in real-world applications. Due to its effectiveness across all stages of secure Transformer inference, including both linear and non-linear components, SecDTD consistently delivers strong performance across datasets of varying scales, task types, and network conditions, without relying on any specific scenario for optimal results.

TABLE 3. SECDTD END-TO-END PERFORMANCE UNDER BOLT FRAMEWORK WITHOUT FINE-TUNING.

Dataset			CoLA	SST-2	MRPC	STS-B	QNLI	QQP	MNLI	RTE
TASK			ACPT	SENT	PARA	SIM	QA/NLI	SIM	NLI	NLI
ACC(%)		Baseline	58.80	91.74	90.53	87.97	91.25	87.68	84.69	68.23
		ours	58.06	92.78	90.57	87.87	90.48	86.75	83.76	67.87
LAN	Inference Time (s)	Baseline	1370.46	3225.60	3225.60	3225.60	3225.60	3225.60	8464.14	8464.14
		ours	518.37	893.78	913.45	916.02	1316.63	886.03	1891.44	2355.26
	Speedup		**2.64×**	**3.61×**	**3.53×**	**3.52×**	**2.45×**	**3.64×**	**4.47×**	**3.59×**
WAN	Inference Time (s)	Baseline	2377.48	5368.23	5368.23	5368.23	5368.23	5368.23	13475.92	13475.92
		ours	887.52	1509.03	1549.26	1533.24	2241.65	1519.39	3088.78	3837.30
	Speedup		**2.68×**	**3.56×**	**3.47×**	**3.50×**	**2.39×**	**3.53×**	**4.36×**	**3.51×**
MOBILE	Inference Time (s)	Baseline	3393.21	7564.60	7564.60	7564.60	7564.60	7564.60	18638.96	18638.96
		ours	1250.05	2122.99	2183.75	2149.83	3172.64	2150.96	4301.92	5346.08
	Speedup		**2.71×**	**3.56×**	**3.46×**	**3.52×**	**2.38×**	**3.52×**	**4.33×**	**3.49×**

TABLE 4. END-TO-END PERFORMANCE COMPARISON

Method	Acc.	Inference Time (s)			FT Free	Sort Free	Exact Drop	Pre Gain	Trace Risk
		LAN	WAN	Mobile					
BOLT	91.74	3225.60	5368.23	7564.60	·	·	·	·	·
W.E.	92.78	1473.89	2522.73	3581.81	✓	·	✓	·	·
[32]	92.75	1547.24	2274.21	3003.83	·	✓	·	·	▲
Ours	92.78	893.78	1509.03	2122.99	✓	✓	✓	✓	·

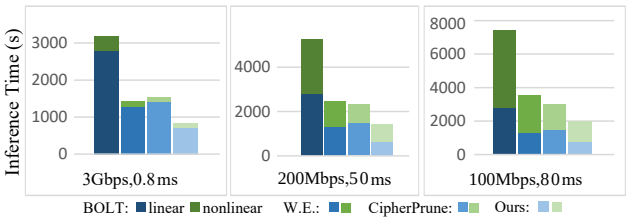


Figure 13. End-to-End Performance Comparison.

We further compare our proposed SecDTD with existing token drop methods, W.E. [24] and CipherPrune [32], as shown in Table 4 and Figure 13. For a fair comparison, we report results on the **SST-2** dataset, which is originally used by all BOLT, W.E., and CipherPrune, under three different network settings. The table also highlights five key properties: **FT-Free** (whether the method requires fine-tuning), **Sort-Free** (whether full sorting is needed), **Exact-Drop** (whether the method allows exact control over the number of dropped tokens), **Pre-Gain** (whether pre-Softmax benefits are available), and **Trace-Risk** (whether the method is vulnerable to potential leakage due to one-to-one mappings between specific inputs and token drop decisions). SecDTD achieves the best inference latency across all network conditions while maintaining a high accuracy of 92.78%, owing to its advantages across all five key properties.

Compared to W.E., SecDTD enables pre-Softmax benefits and avoids reliance on the complex Bitonic Sort, significantly enhancing its practicality for real-world inference. Relative to CipherPrune, SecDTD not only provides pre-Softmax gains but also eliminates the need for expensive fine-tuning required for its thresholding method (though this does not impact inference time), thereby avoiding issues such as imprecise token dropping and, more critically, potential leakage risks. These advantages ensure both secure and efficient deployment in practice.

TABLE 5. SECDTD IMPROVEMENTS THROUGH FINE-TUNING.

Dataset		CoLA	STS-B	RTE
Seq. Length		64	128	256
LAN	Baseline	1370.46s	3225.60s	8464.14s
	ours	518.37s→479.08s	916.02s→824.53s	2355.26s→2046.07s
	Speedup	**2.54× → 2.86×**	**3.52× → 3.91×**	**3.59× → 4.14×**
WAN	Baseline	2377.48s	5368.23s	13475.92s
	ours	887.52s→808.50s	1533.24s→1416.31s	3837.30s→3338.84s
	Speedup	**2.68× → 2.94×**	**3.50× → 3.79×**	**3.51× → 4.04×**
MOB	Baseline	3393.21s	7564.60s	18638.96s
	ours	1250.05s→1130.86s	2149.83s→2005.16s	5346.08s→4650.85s
	Speedup	**2.71× → 3.00×**	**3.52× → 3.77×**	**3.49× → 4.01×**

4.3.2. Further Improvement with Fine-tune. Even without additional fine-tuning, SecDTD effectively reduces inference costs while preserving model accuracy through its dynamic and precise token drop mechanism. However, fine-tuning can further enhance performance by mitigating any potential accuracy loss, allowing for more aggressive token dropping while still maintaining baseline accuracy. As a result, both inference cost reduction and speedup are further improved with the incorporation of fine-tuning. As shown in Table 5, on datasets such as **CoLA**, **STS-B**, and **RTE**, which feature varying input sequence lengths, a simple fine-tuning process over **five epochs** leads to even greater speedup gains.

4.4. SecDTD on BumbleBee Framework

As a general-purpose dynamic token drop method for secure Transformer inference, we replicate all experiments under the BumbleBee framework to further evaluate the effectiveness and generalizability of SecDTD, as shown in Table 6. In this setting, SecDTD achieves similarly substantial cost reductions, with end-to-end inference speedup reaching up to $4.19\times$ compared to the baseline. Consistent with its performance under the BOLT framework, SecDTD improves token drop effectiveness in the early stages of Transformer inference, particularly in components such as $\mathcal{S}$(Softmax) and $\mathcal{S}(\times \mathrm{V})$. The BumbleBee framework employs an NTT-accelerated polynomial multiplication protocol for matrix multiplication, resulting in a lower baseline cost for linear operations and, consequently, less room for optimization. This leads to a slightly reduced relative speedup compared to the BOLT framework. Nonetheless, SecDTD consistently delivers strong performance improvements across all cases, largely due to its efficient optimization of non-linear components, especially Softmax.

5. Discussion and Conclusion

Why use half-and-half token drop in this work: First, both the length of HE ciphertexts (i.e., the number of slots) and operations or optimizations related to packed HE generally follow powers-of-two patterns, such as $\log_2$ or 2^n. In practical implementations, dropping tokens to more than 50% has no real effect, and dropping to 30% (or any ratio between 50% and 25%) is effectively treated as a 50% drop, with the remaining slots padded to satisfy ciphertext length constraints. Therefore, employing a half-and-half token drop aligns optimally with this structure and improves computational efficiency. Additionally,

TABLE 6. SECDTD PERFORMANCE UNDER BUMBLEBEE FRAMEWORK WITHOUT FINE-TUNING.

Dataset			CoLA	SST-2	MRPC	STS-B	QNLI	QQP	MNLI	RTE
LAN	Inference Time (s)	Baseline	96.79	209.33	209.33	209.33	209.33	209.33	490.86	490.86
		ours	45.28	65.24	62.47	66.02	90.38	65.92	117.04	145.21
	Speedup		**2.14×**	**3.21×**	**3.35×**	**3.17×**	**2.32×**	**3.18×**	**4.19×**	**3.38×**
WAN	Inference Time (s)	Baseline	645.85	1261.57	1261.57	1261.57	1261.57	1261.57	2957.89	2957.89
		ours	347.24	466.35	444.88	462.70	610.95	481.42	751.82	906.58
	Speedup		**1.86×**	**2.71×**	**2.84×**	**2.73×**	**2.06×**	**2.62×**	**3.93×**	**3.26×**
MOBILE	Inference Time (s)	Baseline	1150.30	2328.31	2328.31	2328.31	2328.31	2328.31	5591.46	5591.46
		ours	588.67	807.89	772.39	807.59	1080.94	826.98	1351.27	1647.08
	Speedup		**1.95×**	**2.88×**	**3.01×**	**2.88×**	**2.15×**	**2.82×**	**4.14×**	**3.39×**

smaller ratios such as 12.5% can be achieved through iterative application of token dropping (e.g., three times), offering greater flexibility. Second, the half-and-half token drop also complements our design, which identifies the median and uses the average as a pivot, thereby improving the overall efficiency across different components of the system.

Why not simply use the average as the median, instead of using it as a pivot to find the median: In brief, this simplification can lead to a 3%–5% drop in inference accuracy. Although our MCN architecture shapes the score vector into a smooth, gradually decreasing trend—bringing the average closer to the median—this approximation is not exact. Using the average directly as the threshold for token dropping can introduce errors. If token selection continues to rely on this average, the discrepancy between the average and the true median may result in suboptimal decisions. The accumulation of such biased token drops can ultimately degrade model performance.

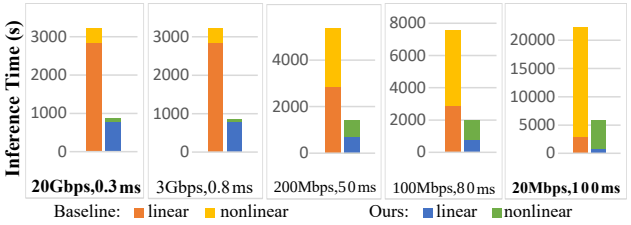


Figure 14. SecDTD Performance Breakdown from Extremely Fast to Extremely Slow Networks.

Performance of SecDTD in extreme network conditions: We further evaluated the end-to-end runtime of SecDTD under both extremely fast and extremely slow network conditions: one with 20 Gbps bandwidth and 0.3 ms latency (typical of high-speed dedicated networks), and the other with 20 Mbps bandwidth and 100 ms latency (commonly seen in IoT applications). This evaluation highlights the effectiveness and generalizability of our approach. As shown in Figure 14, using the MRPC dataset within the BOLT framework as an example, the proportion of non-linear part in inference runtime increases steadily—and eventually dominates—as network conditions degrade from fast to slow. Nonetheless, SecDTD consistently achieves strong performance gains, primarily due to its efficient handling of both linear and non-linear components. This aligns with our design goal: to provide a general-purpose dynamic token drop method for secure Transformer inference, without dependence on any specific deployment scenario for optimal effectiveness.

Generalizability of SecDTD: As a general-purpose dynamic token drop method for secure Transformer inference built on standard cryptographic primitives such as MPC and HE, SecDTD is theoretically applicable to all existing secure Transformer frameworks without requiring additional auxiliary components. Currently, the most widely used secure Transformer frameworks include BOLT [24], BumbleBee [25], Iron [23], and NEXUS [26]. We have evaluated the performance of SecDTD within BOLT and BumbleBee. Although SecDTD can also be directly applied to Iron, we omit detailed evaluation. Nevertheless, theoretical analysis suggests that SecDTD would yield similar runtime reductions to those observed with BumbleBee. For NEXUS, a "non-interactive" framework, SecDTD could be applied if intermediate interaction is permitted, as our token drop mechanism relies on MPC. In this case, theoretical analysis also indicates that performance improvement comparable to those in BOLT could be achieved. Our current implementation targets Transformer **encoders**, with BERT used as a representative model to demonstrate the effectiveness of our design. Transformer **decoders**, such as GPT [2], [3] and LLaMA [63], share some similar architectural foundations with encoder-based models. Therefore, in principle, token drop techniques are also applicable in decoder settings. Overall, as a general-purpose dynamic token drop method for secure Transformer inference, SecDTD can be extended to larger Transformer models and larger datasets. Based on the above discussion, we anticipate that the performance gains would be even greater than those observed with BERT-base on GLUE. Further exploration of this potential by the community remains an interesting avenue for future work.

CONCLUSION: In this work, we present SecDTD, a dynamic token drop framework designed to enable efficient and privacy-preserving Transformer inference. SecDTD enhances the effectiveness of token drop in secure settings by shifting the drop operation to earlier stages of inference. To achieve this, we introduce two key techniques: Max-Centric Normalization (MCN), a pre-Softmax scoring mechanism that evaluates token importance earlier and more efficiently; and OMSel, an secure oblivious median selection protocol that achieves up to 16.9× speedup over prior sorting-based methods while preserving security, obliviousness, and randomness. Extensive experiments across eight GLUE datasets and various network conditions demonstrate that SecDTD achieves up to 4.5× end-to-end inference acceleration without compromising model accuracy, providing a practical and scalable solution for secure Transformer inference.

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)*, 2019, pp. 4171–4186.

[2] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," *arXiv preprint arXiv:1806.00187*, 2018.

[5] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering." in *EMNLP (1)*, 2020, pp. 6769–6781.

[6] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[8] openai.com, "Chatgpt: Get answers. find inspiration. be more productive." 2025. [Online]. Available: https://openai.com/chatgpt/overview/

[9] W. Wang, S. Wang, J. Gao, M. Zhang, G. Chen, T. K. Ng, and B. C. Ooi, "Rafiki: Machine learning as an analytics service system," *arXiv preprint arXiv:1804.06087*, 2018.

[10] A. Act, "Health insurance portability and accountability act of 1996," *Public law*, vol. 104, p. 191, 1996.

[11] T. Shyu, "New legal developments herald big changes for hipaa compliance in 2025." 2025. [Online]. Available: https://www.reuters.com/legal/litigation/new-legal-developments-herald-big-changes-hipaa-compliance

[12] K. Denecke, R. May, and O. Rivera-Romero, "Transformer models in healthcare: a survey and thematic analysis of potentials, shortcomings and risks," *Journal of Medical Systems*, vol. 48, no. 1, p. 23, 2024.

[13] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 307–328.

[14] M. Ben-Or, S. Goldwasser, and A. Wigderson, *Completeness theorems for non-cryptographic fault-tolerant distributed computation*. New York, NY, USA: Association for Computing Machinery, 2019, p. 351–371. [Online]. Available: https://doi.org/10.1145/3335741.3335756

[15] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the EUROCRYPT*, 1999, pp. 223–238.

[16] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," *SIAM Journal on computing*, vol. 43, no. 2, pp. 831–871, 2014.

[17] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the ACM SIGSAC*, 2017, pp. 619–631.

[18] 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 ICML*, 2016, pp. 201–210.

[19] 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 ACM SIGSAC*, 2020, pp. 325–342.

[20] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *Proceedings of the USENIX Security*, 2020, pp. 2505–2522.

[21] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *Proceedings of the USENIX Security*, 2018, pp. 1651–1669.

[22] 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

[23] M. Hao, H. Li, H. Chen, P. Xing, G. Xu, and T. Zhang, "Iron: Private inference on transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15 718–15 731, 2022.

[24] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "Bolt: Privacy-preserving, accurate and efficient inference for transformers," *Cryptology ePrint Archive*, 2023.

[25] 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," *Cryptology ePrint Archive*, 2023.

[26] J. Zhang, X. Yang, L. He, K. Chen, W.-j. Lu, Y. Wang, X. Hou, J. Liu, K. Ren, and X. Yang, "Secure transformer inference made non-interactive," *Cryptology ePrint Archive*, 2024.

[27] Y. Cai, Q. Zhang, R. Ning, C. Xin, and H. Wu, "Hunter: He-friendly structured pruning for efficient privacy-preserving deep learning," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 931–945.

[28] ——, "Mosaic: A prune-and-assemble approach for efficient model pruning in privacy-preserving deep learning," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, 2024, pp. 1034–1048.

[29] Q. Zhang, T. Xiang, C. Xin, and H. Wu, "From individual computation to allied optimization: Remodeling privacy-preserving neural inference with function input tuning," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 101–101.

[30] D. Rathee, M. Rathee, R. K. K. Goli, D. Gupta, R. Sharma, N. Chandran, and A. Rastogi, "Sirnn: A math library for secure rnn inference," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1003–1020.

[31] S. Goyal, A. R. Choudhury, S. Raje, V. Chakaravarthy, Y. Sabharwal, and A. Verma, "Power-bert: Accelerating bert inference via progressive word-vector elimination," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3690–3699.

[32] Y. Zhang, J. Xue, M. Zheng, M. Xie, M. Zhang, L. Jiang, and Q. Lou, "Cipherprune: Efficient and scalable private transformer inference," *arXiv preprint arXiv:2502.16782*, 2025.

[33] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.

[34] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[35] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[36] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[37] G. Brassard, C. Crépeau, and J.-M. Robert, "All-or-nothing disclosure of secrets," in *Proceedings of the EUROCRYPT*, 1986, pp. 234–238.

[38] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 535–548.

[39] D. Demmler, T. Schneider, and M. Zohner, "Aby - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[40] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "Aby2. 0: Improved mixed-protocol secure two-party computation," in *Proceedings of the USENIX Security*, 2021.

[41] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the ACM SIGSAC*, 2018, pp. 35–52.

[42] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter *et al.*, "Homomorphic encryption standard," *Protecting privacy through homomorphic encryption*, pp. 31–62, 2021.

[43] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Annual Cryptology Conference*. Springer, 2012, pp. 868–886.

[44] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *Proceedings of the PKC*, 2013, pp. 1–13.

[45] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the ACM STOC*, 2009, pp. 169–178.

[46] T. Nishide and K. Ohta, "Multiparty computation for interval, comparison, and equality without bit-decomposition protocol," in *Public Key Cryptography - PKC 2007*, 2007, pp. 343–360.

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

[48] P. Beame, V. Liew, and M. Pătraşcu, "Finding the median (obliviously) with bounded space," in *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I 42*. Springer, 2015, pp. 103–115.

[49] D. Bogdanov, S. Laur, and R. Talviste, "A practical analysis of oblivious sorting algorithms for secure multi-party computation," in *Nordic Conference on Secure IT Systems*. Springer, 2014, pp. 59–74.

[50] F. Tramèr and et al., "Stealing machine learning models via prediction apis," in *USENIX Security 2016*, 2016.

[51] R. Shokri and et al., "Membership inference attacks against machine learning models," in *IEEE SP 2017*, 2017.

[52] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptol. ePrint Arch.*, p. 144, 2012.

[53] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of the ASIACRYPT*, 2017, pp. 409–437.

[54] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *NeurIPS*, 2012.

[55] SPEEDTEST.net, "Market analysis of mobile and fixed broadband speeds, us," 2024. [Online]. Available: https://www.speedtest.net/global-index/united-states

[56] M. Research, "Microsoft ezpc (easy secure multi-party computation)," 2019. [Online]. Available: https://www.microsoft.com/en-us/research/project/ezpc-easy-secure-multi-party-computation/

[57] "Microsoft SEAL (release 3.3)," https://github.com/Microsoft/SEAL, Jun. 2019, microsoft Research, Redmond, WA.

[58] J. Ma, Y. Zheng, J. Feng, D. Zhao, H. Wu, W. Fang, J. Tan, C. Yu, B. Zhang, and L. Wang, "SecretFlow-SPU: A performant and User-Friendly framework for Privacy-Preserving machine learning," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. Boston, MA: USENIX Association, Jul. 2023, pp. 17–33. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/ma

[59] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. 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*, 2018, pp. 353–355. [Online]. Available: https://gluebenchmark.com/

[60] JeremiahZ, "Model: bert-base-uncased-cola;mnli;qnli;rte;sst2;stsb," -. [Online]. Available: https://huggingface.co/JeremiahZ/bert-base-uncased-cola;mnli;qnli;rte;sst2;stsb

[61] Intel, "Model: bert-base-uncased-mrpc," -. [Online]. Available: https://huggingface.co/Intel/bert-base-uncased-mrpc

[62] gchhablani, "Model: ert-base-cased-finetuned-qqp," -. [Online]. Available: https://huggingface.co/gchhablani/bert-base-cased-finetuned-qqp

[63] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

Appendix A.
Analyzing Score Distributions for Efficient Median Selection

The Figure 15 illustrates the process of randomly selecting 40 score vector samples from the MRPC dataset during the Median Searching task, demonstrating the actual state of **OMSel** protocol **Step-1.2**. It is evident that the data processed through MCN exhibits a well-structured distribution, characterized by a smooth and gradually decreasing trend, resulting in a relatively uniform appearance. Notably, the middle segment of the curve remains particularly stable and evenly distributed. This characteristic ensures that the current **average** closely approximates the current **median**, facilitating rapid convergence when utilizing the average-as-pivot Median Selection. Consequently, this property enhances the efficiency and robustness of median searching.

Appendix B.
Complexity Analysis of OMSel

As the toy example of OMSel shown in Figure 16 illustrates, all n elements are retained in each iteration to ensure obliviousness. We then analyze the time complexity of OMSel, which consists of the initial random pivot step (RDM) followed by iterative refinements using the average as the pivot.

Per-round Complexity. Each round requires:

- **#Cmp:** A pass to compare each element against the current pivot and generate corresponding masks: $\mathcal{O}(n)$;
- **#Mux:** A pass to multiply all n elements by the masks and update the score array: $\mathcal{O}(n)$;

Therefore, the complexity per round is:

$$\boxed{C_r = \mathcal{O}(n)}$$

Number of Rounds. Let R denote the number of refinement rounds required for the algorithm to converge. In the worst case—when each pivot is consistently the maximum or minimum value—R can be as high as $\mathcal{O}(n)$. But, our approach mitigates this worst-case scenario by combining two mechanisms:

- **Random Pivot Initialization:** The initial pivot p_0 is sampled uniformly at random from the input set. Thus, with high probability, p_0 is not an extreme value. This probabilistic guarantee helps separate most outliers (e.g., top or bottom 10%) in the first round.
- **Average as Pivot:** In subsequent rounds, the the average of the elements is computed as the pivot. Under a well-structured distribution achieved by the proposed

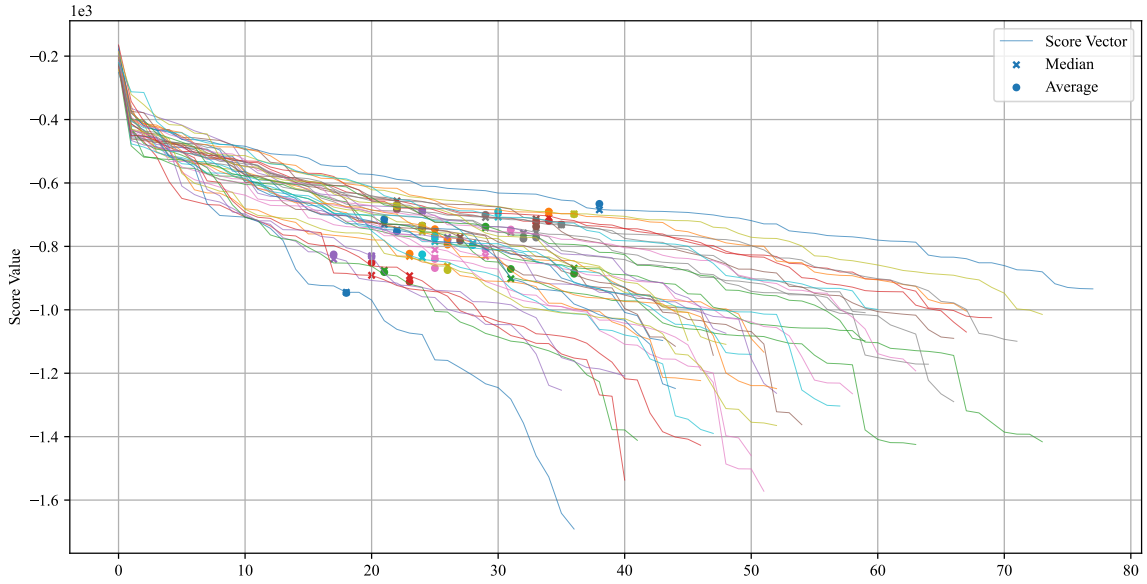


Figure 15. Mean and Median Distributions (40 Samples, MRPC Dataset).

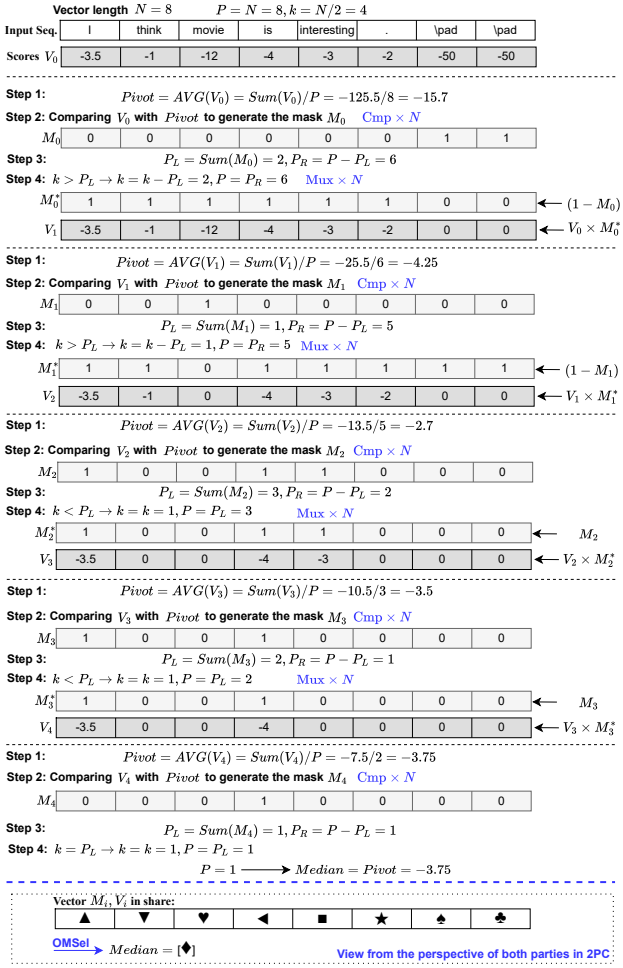


Figure 16. Toy example of OMSel (without RDM).

MCN scoring—characterized by a smooth and gradually decreasing trend—the average remains close to the center of mass, thereby staying near the true median and accelerating convergence.

By avoiding reliance on adversarial pivot choices and leveraging smoothing for stability, our approach prevents the degenerate behavior that leads to quadratic time complexity.

We analyze the number of rounds R needed for the average-based pivot to converge to the median. Let $A = \{a_1, a_2, \ldots, a_n\}$ be the input array of n elements, and let m denote its true median. Since the average remains close to the true median, we assume that in each iteration, the pivot partitions the array such that the larger partition contains at most a fixed fraction $\alpha \in (0.5, 1)$ of the elements—for example, $\alpha = \frac{3}{5}$. Thus, each round (logically) removes a constant fraction of the elements through proposed pseudo-partitioning. After R rounds, the remaining number of elements is:

$$n_r = n \cdot \alpha^R$$

To achieve convergence (i.e., $n_r = 1$), the required number of rounds satisfies:

$$n \cdot \alpha^R = 1 \quad \Rightarrow \quad R = \log_{1/\alpha}(n)$$

Since α is a constant less than 1, the round complexity is:

$$\boxed{R = \mathcal{O}(\log n)}$$

Total Complexity. Combining the per-round cost $\mathcal{O}(n)$ with $\mathcal{O}(\log n)$ rounds yields:

$$\boxed{T(n) = \mathcal{O}(n \log n)}$$