HE-Diffusion: Privacy-Preserving Diffusion Model using Homomorphic Encryption

Abstract

In this paper, we introduce a privacy-preserving stable diffusion framework leveraging homomorphic encryption, called *HE-Diffusion*, which primarily focuses on protecting the denoising phase of the diffusion process. *HE-Diffusion* is a tailored encryption framework specifically designed to align with the unique architecture of stable diffusion, ensuring both privacy and functionality. To address the inherent computational challenges, we propose a novel min-distortion method that enables efficient partial image encryption, significantly reducing the overhead without compromising the model's output quality. Furthermore, we adopt a sparse tensor representation to expedite computational operations, enhancing the overall efficiency of the privacy-preserving diffusion process. We successfully implement the HE-based private preserving stable diffusion inference. The experimental results show that *HE-Diffusion* achieves 500 times speedup compared with the baseline method, and reduces time cost of the homomorphically encrypted inference to the minute level. Both the performance and accuracy of the *HE-Diffusion* are on par with the plaintext counterpart. Our approach marks a significant step towards integrating advanced cryptographic techniques with state-of-the-art generative models, paving the way for privacy-preserving and efficient image generation in critical applications.

1 Introduction

The advent of stable diffusion models has marked a significant milestone in the field of generative artificial intelligence (GenAI), enabling the synthesis of high-fidelity images from textual descriptions. These models, exemplified by their capacity to transform abstract prompts into detailed visual content, have opened new avenues in digital content creation, personalized media, and data augmentation. However, the computational intensity inherent in the inference phase of stable diffusion models necessitates the offloading of these tasks to powerful server-side infrastructure, thereby introducing significant privacy concerns. The crux of the issue lies in the exposure of sensitive information to the server, including user-generated prompts and the resultant images, which may contain or infer personal or proprietary data.

The widespread visibility of user interactions with GenAI platforms is a double-edged sword. This well-regarded transparency not only poses risks to user privacy but also raises concerns about the confidentiality of the generated content, particularly in scenarios where the visual data may carry sensitive or personal information. The benefits of implementing privacy-preserving mechanisms is manifold. It can enhance user trust, allow multiple parties to contribute private data or participate in the generative processes, and enable the use of stable diffusion models in fields where data sensitivity has traditionally been a barrier, such as healthcare, personal security, and confidential design.

In prior research, CryptoNets [21] demonstrates the use of neural networks on encrypted data, laying the groundwork for subsequent studies in this critical area. With the rapid development of GenAI, the intersection of deep learning, image processing, and cryptography has garnered substantial interest from researchers. Previous studies [13, 61] have investigated the security and privacy of GenAI models. Specifically, with regard to the stable diffusion model, existing work [10, 16] identified the security issues, but they mainly focused on the training phase rather than on inference.

Homomorphic encryption (HE) [20] has become a popular method to conduct full computation on the encrypted data, which allows for end-to-end encrypted computation. Implementing encrypted computation throughout the entire process can alleviate users' privacy concerns during the computing escrow service. In this paper, our objective is to architect such a HE-based framework that, for the first time, facilitates secure inference for stable diffusion models. This ensures that the generation process is kept confidential and the privacy of user inputs and outputs is well protected.

However, to achieve practical homomorphically encrypted stable diffusion inference is a challenging task, mainly due to the following two fundamental limitations of HE methods. First, HE only supports polynomial computation, and

second, it incurs high computation overhead that could negatively impact user experience of GenAI models. Moreover, the complete diffusion process is extremely complicated, which includes various structures such as multi-head attention [53], UNet [43], and a combination of techniques from natural language processing (NLP) and computer vision (CV). Within these blocks, there are a significant amount of non-linear activate functions and normalization processes, necessitating the use of approximation techniques. Furthermore, the stable diffusion model's intrinsic computational demands are already substantial for users, not to mention the prospect of requiring thousands of times more (encrypted) computation.

In this paper, we introduce a novel HE-based framework, *HE-Diffusion*, to achieve practical privacy-preserving diffusion model inference. We carefully scrutinize nonlinear functions, strategically avoiding most of them to bypass the need for approximation to maintain accuracy. Considering the significant computational overhead of HE [36], we design a partial encryption scheme to drastically reduce the computational costs. Our method allows the stable diffusion model to operate directly on encrypted data, ensuring that the service provider hosts the computation of diffusion models without ever accessing the sensitive content. This technique not only preserves the confidentiality of the user's prompts but also ensures that the generated images remain encrypted, thereby protecting the privacy of the generated content from the service provider.

Our methodology extends the line of research by introducing a privacy-preserving stable diffusion model, which, to our knowledge, is the first to apply HE to the denoising phase of the diffusion process, thereby enabling secure generative modeling of images. By bridging the gap between advanced cryptographic techniques and state-of-the-art generative models, our work paves the way for privacy-preserving generative tasks. This private stable diffusion framework not only broadens the applicability of generative models in privacy-sensitive domains, but it also sets a precedent for the integration of security considerations in the burgeoning field of GenAI.

This paper makes the following contributions.

- We design *HE-Diffusion*, an efficient adaptation of the stable diffusion process that accommodates the constraints of HE. *HE-Diffusion* focuses particularly on protecting the denoising phase, which is essential for generating coherent images.
- We develop a series of optimization strategies, including a min-distortion method for partial image encryption and the implementation of sparse tensor representations, to enhance the computational efficiency of the encrypted inference process.
- We conduct extensive experiments to demonstrate that *HE-Diffusion* achieves 500 times speed-up for the HE denoising. This advancement brings the runtime of *HE-Diffusion* and plaintext stable diffusion to the same order of magnitude, with negligible loss in accuracy.

2 Background and Related Work

In the domain of secure computation and privacy-preserving technologies, our work intersects with several key areas, including HE, secure image processing, and the application of cryptographic techniques to deep learning models. Here, we contextualize our contributions within the landscape of existing research, particularly focusing on advancements in efficient HE schemes and secure image manipulation.

2.1 Private Diffusion Model

Among the generative models, stable diffusion models [41] have garnered significant attention for their capacity to generate high-quality, diverse samples, outperforming traditional generative adversarial networks (GANs) [22, 37] in various benchmarks. The core principle of diffusion models involves gradually denoising a signal, typically starting from random noise, to generate coherent and complex outputs. While the potential of diffusion models in generating photorealistic images and their applications in content creation is undisputed, two critical concerns emerge when deploying these models in real-world scenarios: (1) the computational overhead associated with their inference process and (2) the privacy concerns associated with generating data from sensitive or personal information.

Although operating in a latent space greatly reduce the computation cost, traversing an inference process requires powerful computational resources. The inference process in diffusion models, characterized by iterative denoising steps, is inherently resource-intensive. Fig. 1 shows how the sampling works in one iteration. Each step involves a forward pass through a neural network. Generating a single image can require hundreds to thousands of these passes, leading to significant computational overhead. Recent efforts aim to mitigate this overhead through various means, including model optimization and the introduction of more efficient sampling techniques [49], yet the computational demands remain a bottleneck for widespread adoption. To enable a broad spectrum of users to experiment with stable diffusion models, even those with limited computational resources, cloud computing and computing hosting have gained popularity. However, the cloud servers and hosts could be untrustworthy, which introduces growing privacy concerns.

Previous work on private diffusion model mainly focused on protecting the training data [10, 25], with a focus on differential privacy [16] and protection against membership inference attacks [35]. However, there has been a lack of discussion regarding inference. The privacy of inference process could raise concerns, particularly when the generation process involves personal or sensitive data. Protecting training data is

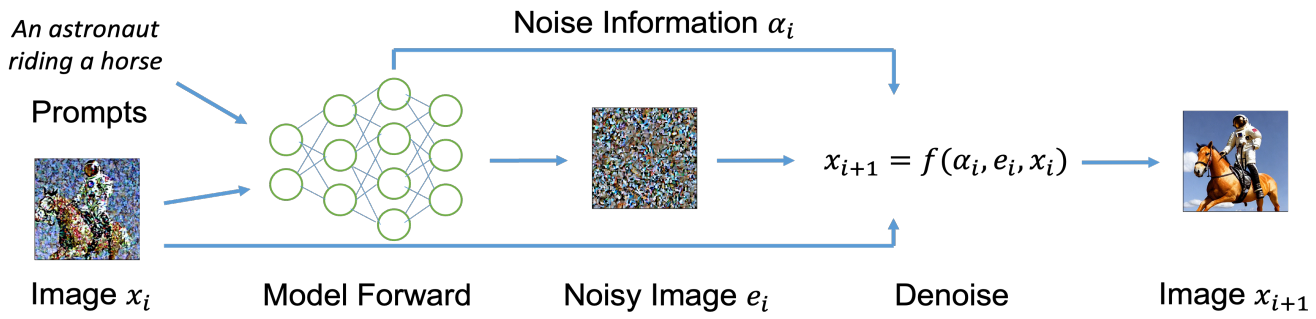

Figure 1: **Sampling process of diffusion model at iteration i. For the model forward component, the inputs are intermediate image x_i and the prompts. Denoise process receives the output of model forward (i.e., noisy image e_i and noise information α_i), and then generates a better image x_{i+1} with x_i.**

crucial for model creators, whereas safeguarding the inference process is essential for the majority of users to maintain their privacy. The models' ability to synthesize images based on user inputs can potentially expose private information, if the generated content or the input prompts are mishandled or intercepted by unauthorized parties. On the other hand, with encrypted inference, pre-trained models are available as a service for users to ensure the data privacy. For other deep learning applications, protecting users' private input and output is regarded as a significant concern [4, 46]. Techniques such as HE have been applied to keep the sensitive information secret [27]. Considering that text-to-image and image refinement are relevant services for a large number of users, the privacy protection of stable diffusion inference is an important problem.

This work investigates this problem and aims to develop privacy-preserving stable diffusion inference as a service via HE. The majority of the computation will be conducted on the server, while sensitive data remains in the possession of users.

2.2 Homomorphic Encryption

The intersection of HE and generative models presents a promising avenue for safeguarding user privacy. HE enables the computation on ciphertexts, which has been utilized on different generative models. HE-transformer and its extensions [4–6, 13] introduce private inference for some transformer [53] based models. Concrete-ML [61] leverages large language models such as GPT-2 [30] for privacy protection. It is evident that the research trend is moving towards providing privacy-preserving generative inference.

The practical implementation of HE has seen significant advancements through the development of specialized libraries and frameworks, designed to simplify the integration of HE into various applications. SEAL [47] provides a comprehensive suite of functionalities for both fully and partially HE schemes, especially the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [8] and the Cheon-Kim-Kim-Song (CKKS) scheme [14] for approximate arithmetic. They have significantly improved the efficiency and practicality of HE. TenSEAL [3] offers a seamless integration with PyTorch and emerges as a bridge between HE and machine learning. Concrete-ML [61] abstracts the complexities associated with HE, offering a pathway for machine learning models to operate on encrypted data with minimal modifications. These developments underscore a pivotal shift towards making HE more accessible and practical for a broad spectrum of AL/ML applications.

However, current attempts fall short in addressing HE's key difficulty - computation overhead. A recent study [12] shows that homomorphically encrypted logistic regression is more than ten thousands of times slower than the plaintext version, using SEAL library [47]. Moreover, most of relevant HE libraries lack of support for GPU or other accelerators, which limits its efficient utilization of computing resources. Some recent studies adopt FPGA [59] and GPU [48, 56], but they did not provide easy-to-use open-source libraries. Specific accelerators [45] may be a potential way to counter HE's overheads and enable wider applications. However, these architectures are hard to access. There are some previous studies [9, 15, 57] aiming to introduce techniques in high performance computing to enhance the efficiency of HE. Yonetani et al. [60] utilized sparsity to reduce encryption time. High performance algorithm [40] enables us to achieve significant speedup, even with limited computational power. This is particularly relevant in the context of this work's application scenario, where users cannot determine if the server possesses the machine tailored for specific applications. Optimization at the algorithm level demonstrates its benefits in terms of universality and cost-effectiveness, leveraging the same computing resources.

On the other hand, though the marriage of generative AI and HE is inspiring, the loss of accuracy is a critical problem. Neural networks introduce various non-linear activation functions and normalizations. However, these functions have to be approximated under HE, as HE is only able to support

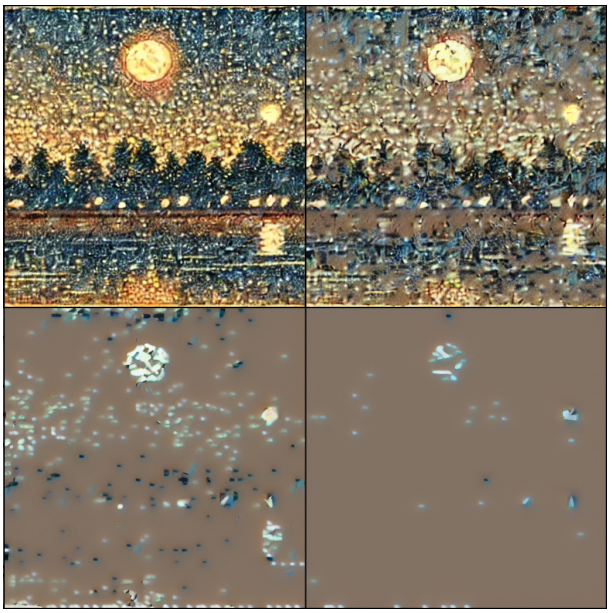

Figure 2: **When revising in the latent space with certain distortion, the image changes as the figure shows. We randomly set elements to 0 in the latent space to generate images with different distortion from the original image, and then transform back to pixel space to check the changes. As the distortion grows, from top left to bottom right, the images become noisier and preserve less information.**

addition and multiplication [20] (i.e., polynomial functions). Previous research for HE-inference [2, 13] has made substantial efforts for higher precision approximation. There are also some studies [28, 38] dedicating to reducing non-linear operations. However, errors will still accumulate throughout the process due to the extensive number of non-linear operations involved.

2.3 Distortion and Information

Distortion refers to any alteration or degradation in the carrier medium (such as an image, video, or audio file) resulting from the embedding of hidden information. It is primarily applied in steganography [31] to hide the embedding information [1, 19]. Since distortion describes a kind of distance between the original image and the modified image, we can use it to quantify the information loss after removing partial data. If there is a low distortion between two images, it implies that the primary information is retained, and only unimportant sections without key semantics have been discarded. Another advantage of distortion is that, it can work in different linear spaces. That means it also works in the latent space [42] where denoising happens.

Fig. 2 validates the idea. When the elements are set to 0, we artificially discard information. And if we regard the original image as a embedding of the revised image, distortion accounts for this information. As a result, we can use distortion to quantify information loss.

In practice, additive distortion [17] is often used due to its salient nature, which is defined as [19]:

$$D(X,Y) = \sum_{i,j} \rho_{ij} |x_{ij} - y_{ij}|, \tag{1}$$

where X and Y are the original image and the modified image, ρ is the cost matrix. Cost matrix ρ is the central parameter in the calculation of distortion and is only dependent with the original image X. There are different designs of the cost function, known as HUGO [39], WOW [23], HILL [32], 2D-SSA based method [58] and QMP [33]. All these methods are proposed to make the two images as similar as possible, in order to counter steganalysis. Since the definition of cost function is independent from the form of additive distortion, we can simply choose an effective one for the convenience of implementation.

3 Threat Model

In this paper, the attacker's goal is to compromise the confidentiality of data processed by the diffusion model. Specifically, the adversaries aim to infer sensitive information from the model's outputs or intercepted data. Their goal is to breach user privacy through analysis and aggregation of data, exploiting patterns to infer protected information. The adversaries may have access to intermediate values computed during the diffusion process, enabling them to infer information about the input data or the model's parameters. Furthermore, they possess the capability to reconstruct the original input data based on the outputs or the intermediate states generated by the model, thereby posing a direct threat to user privacy.

In our threat model, we assume that the servers hosting the diffusion models operate under an "honest but curious" framework. While these servers faithfully execute the model's computations without tampering with the process or outcomes, they harbor an intrinsic interest in the users' prompt inputs and the generated outputs. This curiosity could potentially lead to privacy breaches, as the servers might attempt to analyze or infer information about the users' data beyond the intended scope of interaction. The scenario underscores the need for robust privacy-preserving mechanisms that safeguard user inputs and outputs against both external attackers and potentially intrusive but non-malicious server operators, ensuring that the confidentiality and integrity of the data are guaranteed throughout the diffusion model's processing pipeline.

4 *HE-Diffusion* Framework

In the evolving landscape of generative modeling, particularly within the framework of stable diffusion models, the imperative of establishing privacy-preserving mechanisms is

paramount. Prior to the instantiation of such mechanisms, a rigorous examination of sensitive information is warranted. From the point of user interaction, this encompasses primarily the textual prompts that serve as the genesis of the generative sequence, and the visual outputs, which are the culmination of the model's inference process. Given that the pre-trained model is provided by the server entity, its associated model weights are not considered sensitive information. Therefore, this eliminates the requirement for protective measures for these components.

Delving into the mechanics of the stable diffusion model, we encounter an iterative sampling process, which is divided into two stages—forward propagation and subsequent denoising. These stages are presented in Fig. 1. It becomes evident that the intermediate image, expressed as x_i, converges incrementally towards the final output with each iteration. This convergence bestows upon x_i the status of sensitive information, necessitating the invocation of robust security measures to preclude unauthorized information disclosure.

A meticulous examination of the data flow within the forward and denoising stages reveals that the privacy of the intermediate image x_i is critical for the privacy-preserving framework. While the textual prompts remain static throughout the computational expanse, their participation is confined to the model's forward propagation phase, thereby rendering them as inherently sensitive. In contrast, the denoising phase $x_{i+1} = f(\alpha_i, e_i, x_i)$ employs a triad of inputs—comprising the noise information α_i , the noise-laden image e_i, and the aforementioned intermediate image x_i—to engender the next iteration's intermediate image x_{i+1}. Ensuring the privacy of x_i guarantees the confidentiality of x_{i+1}, thus permitting α_i and e_i to be accessible without undermining data security.

Moreover, the noisy image, characterized by its stochastic composition, lacks content that can be exploited. Alone, the noise parameters do not provide enough information to reconstruct x_i. Therefore, protecting the intermediate images and prompts is crucial for enhancing the privacy of the generative process, as they contain the sensitive data across the sampling iterations.

4.1 Input Protection

Securing the privacy of prompts and intermediate images x_i during the sampling process is important. However, simply concealing these elements throughout the process is neither time-efficient nor feasible in practical implementations.

From a time-cost perspective, the absence of GPU support in existing HE libraries, such as SEAL [47] and TenSEAL [3], necessitates reliance on CPU processing for encrypted operations. To evaluate the impact of encryption on processing times, we conduct an experiment to encrypt both prompts and intermediate images. Given the substantial computational overhead associated with HE, we extrapolated the time cost for the HE model's forward phase based on the overhead ratios observed in matrix multiplication operations of similar dimensions, which predominantly determine the forward phase's duration. Our findings, as detailed in Table 1, reveal a significant disparity in processing times between GPU-accelerated and CPU-based computations, as well as between encrypted and plaintext operations. This result underscores the challenge of integrating HE into the model's forward phase without incurring prohibitive time costs, and indicate that innovative solutions are needed to reconcile encryption's security benefits with the practicalities of computational efficiency.

Table 1: **Estimated model forward time cost for a whole sampling process of a diffusion model. Encryption is considered only happening once at the beginning. Experiments are based on TenSeal [3], tested on one A100 GPU and one Intel(R) Xeon(R) Gold 6248R CPU, respectively. HE calculation time for model forward is estimated by other experiment results.**

Case	GPU	CPU	Enc	HE Cal
$256\times128\times256$	0.01s	0.12s	17.1s	932s
Model Forward	35s	881s	716s	79.19d

Incorporating HE within the stable diffusion model, which fundamentally comprises UNet [43] and Attention [53], presents a significant challenge due to the prevalence of non-linear operations like softmax, layer normalization, and ReLU. The inherent limitation of HE schemes to only support addition and multiplication operations means that executing these non-linear functions requires them to be approximated by polynomial functions. One solution is to perform polynomial approximation via Taylor expansion [2, 54] or specifically designed linear function [34].

However, there are two main disadvantages:

- *Precision of Polynomial Approximations:* The accuracy of Taylor series expansions is contingent upon the polynomial coefficients, which in turn depend on data distribution parameters like mean and standard deviation. Given the encrypted nature of the data, dynamically accessing these parameters is infeasible. Servers might resort to utilizing pre-computed parameters, but this approach is generally effective only in the initial layers of the model due to its static nature.

- *Accumulation of Errors:* Even with meticulously chosen coefficients that yield satisfactory approximations for individual non-linear operations, the cumulative effect of approximation errors becomes pronounced as the number of non-linear functions increases throughout the model's depth. An alternative strategy involves delegating the computation of non-linear functions to users in plaintext space, thereby avoiding the limitations of HE.

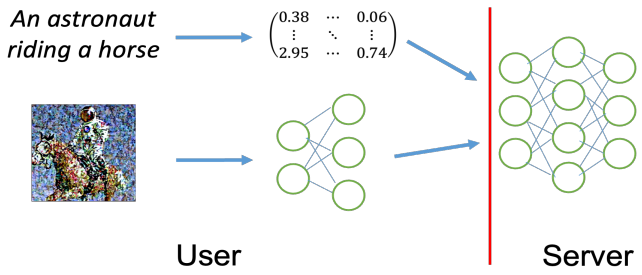

Figure 3: **Overview of our input protection scheme. Image x_i will not be directly uploaded to the server. The first several layers will be executed by users. Then, the server performs the rest of computation with the conditioning tensor and the intermediate result.**

However, this approach necessitates extensive user participation and communication, introducing its own set of practical challenges.

One feasible solution is to conduct plaintext model forwarding, while keeping the sensitive information unaccessible by the server. Fig. 3 shows the model forwarding process. In detail, the prompts are first encoded into a large conditioning tensor by condition stage model, which can be independent from the diffusion model. This provides chance for users to prevent server from reconstructing prompts, by keeping their encoding method secret. Since the server cannot access the condition model, model inversion attack [18] will not work. The high-dimensional and inherently lossy nature [51] of this encoding further complicates any attempt to reverse-engineer the prompts from the conditioning tensors. The primary role of the conditioning tensor is to encapsulate the essential semantic meanings required for image generation, without preserving the details of the textual input, which renders the direct reversal infeasible.

The intermediate image x_i also needs protection. We achieve the protection of x_i by conducting the first several layers on the users' local machine. Existing work [44] shows that, if the output is not sensitive, we can protect input by keeping the first several layers secret. Here, the circumstance is similar. We need to protect inputs, while the output is a noisy image. Therefore, we simply allow users to conduct these calculations locally. To enhance privacy, users can incorporate different techniques such as adversarial training.

4.2 Output Protection

The output image is related to the denoising phase. Operations in this component is simply element-wise addition and multiplication. Since element-wise operations are not supported by GPU tensor cores, denoising accounts for over 50% time cost on GPUs. According to the discussion above, we only need to protect x_i and x_{i+1}. The basic implementation is shown by Algorithm 1, where c_1 to c_4 are scalars, e_i and x_i are tensors with same shape.

Algorithm 1 Denoising

Input: Noisy Image e_i, Noise information c_1, c_2, c_3, c_4, Image x_i

```
/* Current prediction for x0 */
pred_x0 = (x_i − c_1 * e_i)/sqrt(c_2)
/* Direction pointing to x_i */
dir_xi = sqrt(1 − c_3 − c_4^2) * e_i
/* Generate noise */
noise = c_4 * noise_like(x.shape)
/* Previous step of the image */
x_prev = sqrt(c_3) * pred_x0 + dir_xt + noise
```

Output: Image x_{i+1}

In Algorithm 1, the image x_i will first be combined with e_i to make a prediction of x_0. Accurate $pred\_x0$ serves as a reference point for the model. Then a direction tensor pointing to x_i is calculated to show how model should adjust the current noisy x_i. Then, we get x_{i+1} from the combination of $pred\_x0$, dir_x and a random noise.

Initiating encryption at the start and concluding with decryption might seem straightforward, yet it incurs inefficiencies due to the necessity for re-encryption and redundant ciphertext calculation. Particularly, since both $pred\_x0$ and x_prev are in ciphertext form and their associated computations are complex, certain operations might need to be executed in the ciphertext domain due to the sequence of computations. To streamline this process, the aim is to maximize the number of operations conducted in plaintext, resorting to ciphertext computations only when absolutely necessary. Recalling Algorithm 1, we need to merge all the plaintext parts and do operations corresponding to x_i as late as possible. Through strategic reorganization of the computational steps, we can significantly diminish the need for ciphertext operations to merely a single tensor multiplication and addition. Moreover, the advantage of element-wise operations should be utilized, i.e., vectorization. Without the need to consider about the shape of tensors, the sparse values can be encrypted by lightweight vectors in the HE libraries. Though it reduces the multiplication depth, it brings great performance improvement.

Given that x_i remains constant throughout the model's forward pass, the denoised image from the preceding step can be directly utilized as the input for the subsequent iteration. To address the constraints imposed by the multiplication depth, relinearization techniques should be applied intermittently. However, due to limited support for relinearization in TenSeal [3], our implementation temporarily adopts infrequent re-encryption as an alternative strategy.

The refined encrypted denoising algorithm is illustrated in Algorithm 2. Some parameters such as $pred\_x0$ are not explicitly calculated any more. The plaintext parts like cal-

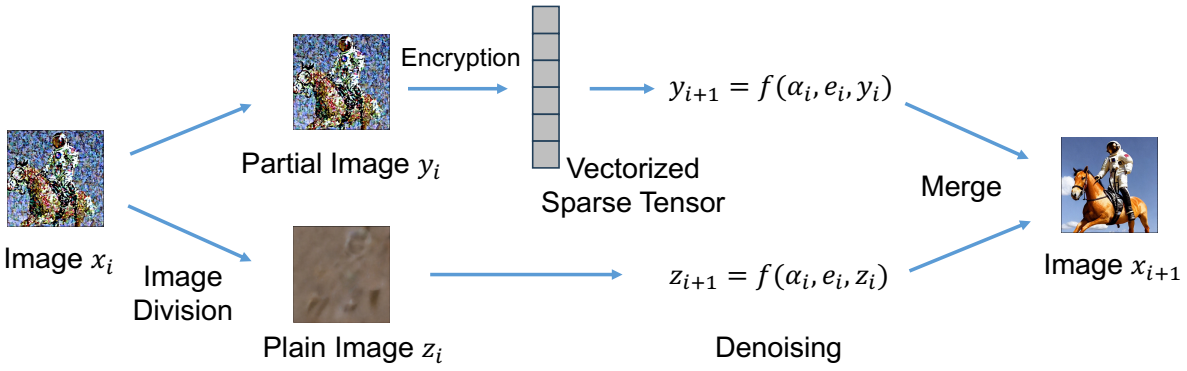

Figure 4: **Overview of our partial encryption scheme. x_i is divided into y_i and z_i, with most of information kept by y_i. Then y_i is encrypted while z_i remains unencrypted. After denoising, x_{i+1} can be constructed by merging y_{i+1} and z_{i+1}.**

culating dir_xt and noise generation are unchanged, while the computation involved by x_i is compressed to include only one multiply and add operation. Here, x_prev, the encrypted version of the intermediate image, is leveraged in the denoising process for the next iteration, denoted as $enc_{x_{i+1}}$. This encrypted intermediate image is also transmitted back to the user for decryption, enabling them to compute the initial layers of the model's forward phase. As the intermediate images remain encrypted on the server throughout the process, the security of the outputs is effectively maintained.

Algorithm 2 Encrypted Denoising

Input: Noisy Image e_i, Noise information c_1, c_2, c_3, c_4, Encrypted Image enc_{x_i}

```
/* Direction pointing to x_i */
dir_xi = sqrt(1 - c_3 - c_4^2) * e_i
/* Generate noise */
noise = c_4 * noise_like(x.shape)
/* Multiplication factor */
factor = sqrt(c_3/c_2)
/* Addition part */
add_part = dir_xt + noise - factor * c_1 * e_i
/* Previous step of the image */
x_prev = factor * enc_{x_i} + add_part
/* sent x_prev to user and decrypt */
x_{i+1} = x_prev.decrypt()
```

Output: Image x_{i+1}, Encrypted Image x_prev

5 Partial Encryption by Distortion

Manipulating image is expensive even in latent space. According to Algorithm 2, in each iteration, we should do element-wise multiplication and addition for a tensor, which means thousands of ciphertext-plaintext multiplication should be performed. To mitigate these computational demands, we introduce a strategy combining partial encryption with ciphertext-plaintext hybrid computation.

Partial encryption for images [52] is early regarded as a powerful tool to reduce the encryption cost. In the context of HE, there are very few studies [60] leveraging partial encryption. The reasons may be manifolds: (1) Partial encryption engenders complex hybrid data structures and computations, which are not readily accommodated by existing HE libraries. (2) The sparsity induced by partial encryption contradicts the prevailing trend in hardware development, which favors dense computational paradigms exemplified by tensor cores and SIMD (Single Instruction, Multiple Data) architectures. (3) Employing partial encryption necessitates a delicate balance between security and computational efficiency, as it poses potential risks for information exposure. Despite these challenges, we advocate for the integration of partial encryption with HE, arguing that the benefits of hybrid computation—specifically in reducing computational and memory burdens—outweigh the concerns. In the subsequent sections, we will demonstrate that partial encryption serves as an ideal complement to HE. By strategically encrypting only certain components of the data, we can significantly reduce computational overhead without sacrificing security, thereby enhancing the efficiency and practicality of HE in real-world applications. This approach leverages the strengths of both partial encryption and HE, offering a balanced solution that maintains data privacy and security while optimizing performance.

5.1 Partial Encryption Scheme

Element-wise operation suits partial encryption well, since it can gain linear time saving with the reduction of encrypted elements, and it is convenient for data decomposition. We conceptualize the image X as a composite of two components: $X = Y + Z$, where Y encapsulates the bulk of X's information, and Z contains elements deemed less critical. This decomposition ensures that Y and Z are mutually exclusive, with $Y \times Z = 0$, indicating that each element of X is exclusively

present in either Y or Z, with the counterpart being zero. In this framework, HE is applied to Y, while Z remains in plaintext, which is written as follows.

$$f(X) = f(Y) + f(Z). \tag{2}$$

Fig. 5 describes this scheme in detail. It is unnecessary to amalgamate y_{i+1} and z_{i+1} at every iteration; they can be computed independently, given the constraints on multiplication depth. The main challenge lies in minimizing the encrypted portion Y to the smallest possible subset while retaining the majority of the image's information. Notably, this approach to partial encryption preserves the integrity of the computational results without resorting to approximation, allowing for variable levels of security through different decomposition techniques and parameter configurations.

5.2 Image Division

Due to its superiority in reducing time complexity and minimizing memory requirements, sparse computation has consistently been at the forefront of research and development. This approach, which emphasizes the processing of data structures where the majority of elements are zeros, enables more efficient storage and faster computational operations, Sparsity is often achieved by ignoring some data. Representative techniques achieved sparsity in deep learning includes model pruning [29], dropout [50], stochastic depth [26]. In this work, we use additive distortion to remove unimportant data points from image tensors.

According to the definition of additive distortion, if we remove n points from image X with indices $(i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n) \in I$ and generate a new image Y, the additive distortion can be written as:

$$D(X,Y) = \sum_{(i,j)\in I} \rho_{ij}|x_{ij}|. \tag{3}$$

This equation implies that the order of removing two points will not affect the calculation. As ρ is only related to X, it can be pre-calculated and treated as a constant matrix. Then, every data point in Y contributes independently to the whole additive distortion. Calculating distortion after removing a point from Y is simply adding a term corresponding to the data point. Thus, distortion of every point can be discussed separately.

Moreover, additive distortion can be considered as some kind of information. Distortion of a point can be treated a quantified form of information it carries. A direct way is to iteratively remove points with low distortion. However, the iteration should stop before important information leaks. This requires a threshold. We can calculate the whole distortion while all points are removed as:

$$Whole\_D(X) = \sum_{i,j} \rho_{ij}|x_{ij}| = D(X,0), \tag{4}$$

With a whole distortion, the importance of a point can be quantified by the ratio between its own distortion and the whole distortion. This provides convenience for us to set a threshold for points removing for different images. Algorithm 3 shows an overview of this procedure. First, the cost matrix is generated by cost function and X. Then, it conducts element-wise multiplication with X to create the distortion matrix. The whole distortion can be calculated incidentally. We iteratively find the minimum value in the distortion matrix and set the element at the corresponding position of Y to 0. Since this brings distortion, we then update distortion and the matrices. When the distortion caused by removed points reaches the threshold, the algorithm stops and output the sparse image Y. In this algorithm, cost_function can be chosen according to needs. The remaining tensor Z can be simply generated by $Z = X - Y$ after the removal.

Algorithm 3 Point Removal (Basic)

Input: Image X, $threshold = 0.01$, $distortion = 0$, $Y = X$
  `/* Get cost matrix*/`
  $C = cost\_function(X)$
  `/* Distortion matrix */`
  $D = C * X$, $Whole\_D = D.sum()$
  `/* Iteratively remove points */`
  repeat
    `/* Get min-distortion element */`
    $min\_ele = argmin(D), Y[min\_ele] = 0$
    `/* Update distortion */`
    $distortion$ += $D[min\_ele], D[min\_ele] = inf$
    `/* End loop when reach threshold */`
  until $distortion/Whole\_D < threshold$
Output: Sparse Image Y

This implementation is quite inefficient, due to the existence of a large number of iterations. Considering that this procedure should be frequently executed during stable diffusion sampling, we propose the following methods for further speedup.

We first check some conditions and conduct batch deletion. The maximum threshold in the procedure is $dis\_threshold = Whole\_D * threshold$. Then in the process, remain distortion can be defined as $dis\_remain = dis\_threshold - distortion$. All points meets this inequality can be removed together:

$$distortion * remain\_points < dis\_remain, \tag{5}$$

where remain_points denotes to the number of points not removed. This condition can be simply done by function 'where' in torch or numpy.

Other functions such as 'topk' also helps for acceleration. While these optimizations are utilized, time cost of the procedure can be reduced from several hours to less than 5s.

Figure 5: **Examples to compare the generated images from the plaintext version and the encrypted version. Prompts: "Night Scene with 2 moons." The height and width of the images are equally and sequentially set as {256, 384, 512, 768, 1024, 1280}. The images corresponds to these size from left to right. All other parameters are equally set. Top row: samples from stable diffusion model. Bottom row: samples from *HE-Diffusion* model.**

5.3 Sparse Encryption and Computation

Considering that tensor Y contains only partial data point of X, and these points are discontinuous distributed in the tensor, its sparsity can be utilized for acceleration. Sparsity has two primary advantages: (1) Tensor Y has thousands of elements (zeros and nonzeros), their ciphertexts account for large space. (2) Conducting encryption and calculation only on nonzero elements save a substantial amount of time, and embodies the meaning of partial encryption. Z can keep dense since $f(Z)$ is efficient enough so that the payoff from optimization is small.

Special workflow leads to customized operations and data structure. $f(Z)$ is calculated in dense format. Considering that there are only element-wise operations in f, and elements of X are separated in Y and Z, respectively, $f(Z)+f(Y)$ can be done without calculation. We can simply perform assignments as $Z[Y.indices] = Y.values$ to merge these two tensors together. Addition and multiplication should also be specifically designed. The $factor$ and add_part in Algorithm 2 are all dense tensors. Since $f(Z)$ deals with partial computation, $f(Y)$ should only focus on the elements in $Y.indices$. That means we do not need to explicitly transform $factor$ and add_part to sparse tensor. Partial addition can be done as:

$$Y.values *= factor[Y.indices],$$

$$Y.values += add\_part[Y.indices].$$

Traditional sparse tensor or sparse matrix format does not directly support these functions. Previous work has used sparse matrix multiplication or tensor manipulation under HE [11, 59]. However, sparse HE still still lacks support for operations that facilitate hybrid computation between sparse and dense matrices.

Here, we leverage coordination (COO) format to store the sparse tensor. Compared with compressed format like compressed sparse row (CSR), compressed sparse column (CSC), COO needs more memory for indices. But the advantages of COO is its flexibility and simplicity. Moreover, its indices can be directly accessed by dense tensor without any process, which provides convenience for sparse-dense hybrid computation. Since memory cost of encrypted values is much larger than plaintext indices, the memory advantage of compressed formats is almost negligible.

Same as plaintext sparse tensor, only nonzero elements are stored in the encrypted COO sparse tensor. This helps to reduce much time in the context of the rapid increase in encryption time with the number of elements. We provide interface to perform encryption on COO sparse tensor with different encryption schemes, such as BFV [7] and CKKS [14]. The encrypted values could also be organized as a vector for vectorized operation, in order to gain higher performance. Since performing encryption and computation becomes much cheaper, and the number of operations is largely reduced by Algorithm 2, we can simply do reencryption while the accuracy is on the decline.

6 Evaluation

In this section, we will conduct a comprehensive evaluation of *HE-Diffusion*, examining its performance across three critical dimensions: accuracy, performance, and security. In the accuracy evaluation, we mainly focus on whether HE brings precision loss. For performance evaluation, we will meticulously analyze the system's runtime under various parameter configurations, providing insights into how different settings influence overall efficiency. This analysis will extend to examining the system's responsiveness in scenarios with varying

Table 2: **The similarity between the images generated by plaintext version and encrypted version. All parameters and prompts are equally set. 5 methods are applied to test the results. The results in the table are the mean value of the generated samples.**

Cosine	SSIM	PSNR	MSE	KL Divergence
0.9818	0.9995	68.65	0.0089	0.0046

levels of data sparsity. The security dimension of our evaluation focuses on the challenges and vulnerabilities introduced by partial encryption. We will delve into the potential for information leakage, offering a detailed discussion on how partial encryption, while beneficial for reducing computational overhead, could expose the system to certain security risks.

6.1 Implementation Details and Settings

The experimental results are generated by stable diffusion model v1.4 with a PLMS sampler to generate images at a default size. Our implementation is based on the open-source implementation of stable diffusion models [41].

The HE component in our work uses CKKS [14] scheme by TenSEAL [3]. We choose the poly modules degree of 8,192 and the coeff-modules of 26. For a simulation purpose, users and server were both implemented in a single computer and thus no transmissions among them were considered in the experiments. All parameters are kept same in the plaintext version and the encrypted version.

In this work, the computation resource are a NVIDIA A100-PCIE GPU with 40GB memory and a Intel(R) Xeon(R) Gold 6248R CPU running at 3GHz.

6.2 Accuracy

Though our methods do not explicitly include approximation, there will be loss of accuracy due to the approximation encryption scheme [14]. Here, we utilize five different methods to evaluate the quality of the generated images. Cosine similarity and Min squared error (MSE) are basic mathematical tools to directly compare the data distribution. In the pixel space, Structure Similarity Index Measure (SSIM), Peak Signal to Noise Ratio (PSNR) [24] work in different aspect. SSIM integrated luminance, contrast and structure to fit the intuitive feeling of the human eye. PSNR stands at the signal side to evaluate the power of a signal and the power of corrupting noise. KL divergent comes from cross entropy, which quantifies the redundant information. We refer to these five indicators, in order to make an overall judgment.

As Table 2 shows, our homomorphically encrypted diffusion inference can generate very similar images compared with the original unencrypted one. According to the table, we have PSNR > 40dB, which means the signal absolutely dominant compared with the noise in the encrypted part. Cosine similarity is lower than SSIM. That implies the encrypted images show some differences with the original ones, but in the context of structure and perceived quality, the encrypted images keep most of information. Fig. 5 provides a more intuitive example for different image size. Despite some tiny difference in pixels, the generated images are almost the same.

6.3 Performance

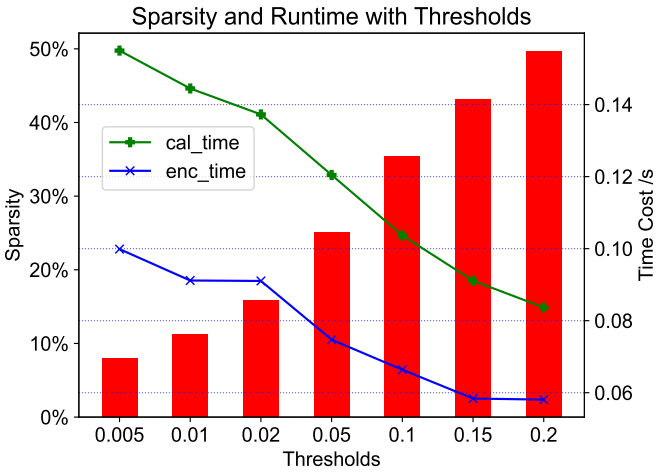

Figure 6: **Variation of sparsity and the ability to keep sensitive information under different thresholds. Images are tested in the latent space. The red bars denotes to the percentage represented sparsity. The blue line and the green line are the cosine similarity between the remained image (encrypted part), the removed part and the original image, respectively.**

In the performance evaluation, our primary focus is on determining the feasibility of our HE-Diffusion model and understanding the impact of sparsity on its functionality. Table 3 provides the process time of different algorithms. To the best of our knowledge, a homomorphically encrypted stable diffusion model does not exist. Therefore, our comparisons are made between the plaintext version and our various algorithms. Due to the inevitable introduction of overhead of security, the plaintext version is the fastest. And just as we expected, simply conducting unoptimized homomorphic encryption brings significant complexity. However, after optimization and our sparse design, this overhead is largely reduced. Based on our findings, our most efficient sparse implementation achieves a computational speed that is on par with the plaintext version in terms of order of magnitude. This significant achievement underscores the practicality of our approach for users, as it ensures that the enhanced privacy

Table 3: **Exact time cost of one denoise process and the whole *HE-Diffusion* inference. In the 'GPU' version, GPU is applied to accelerate model forward, while 'CPU' denotes to a pure CPU implementation. 'Plain' refers to Algorithm 1. 'Enc' simply encrypt x_i for Algorithm 1 with nothing else changed. 'Enc_opt' refers to Algorithm 2 with less frequent reencryption. 'Sparse' is the sparse version described in Section 5. Here, reencryption is applied instead of relinearization. All parameters of stable diffusion are set as default values. In the sparsity test, we choose threshold=**0.01 **as a conservative parameter setting.**

Case	Plain	Enc	Enc_opt	Sparse
Denoise once	**0.12s**	354.34s	1.67s	**0.70s**
Inference (CPU)	**0.24h**	10.06h	0.41h	**0.32h**
Inference (GPU)	**35s**	9.84h	203s	**106s**

provided by homomorphic encryption does not come at the expense of performance.

In Table 3, the time costs of GPU acceleration and pure-CPU follow different mode of change. With customized design for matrix multiplication, GPU can reduce the model forward time from 8.8s to less than 0.04s per iteration. As a result, the distribution of computation hotspot is quite different. In pure-CPU version, model forward accounts for the primary time costs even after homomorphically encrypted denosing. However, in the GPU version, denosing is the most computational intensive component. It is evident that the impact of HE on the final result is significant.

In details, different parameter settings affect the performance results. When we set a higher threshold for Algorithm 3, higher sparsity will be achieved, and the time cost (both the encryption time and the calculation time) will decrease. This is easy to see, since high sparsity leads to fewer non-zero elements, which benefits encryption and calculation. However, this is not without cost. The risk and trade-off will be discussed in the next section.

The influence of the image size is also considered, as Fig. 7 shows. With the increase of image size, the time cost of model forward and denoising follows Quadratic growth, while sparsity almost keeps unchanged. Since model forward accounts for the majority of time, the ratio between it and denoising are growing rapidly when enlarging image size, which means, comparatively speaking, the whole run time will be closer to plaintext version.

6.4 Security of Partial Encryption

Partial encryption is commonly associated with the risk of information leakage. In this section, we will show the information preserving ability of our point removal algorithm. Since the computation happens in a latent space, our partial

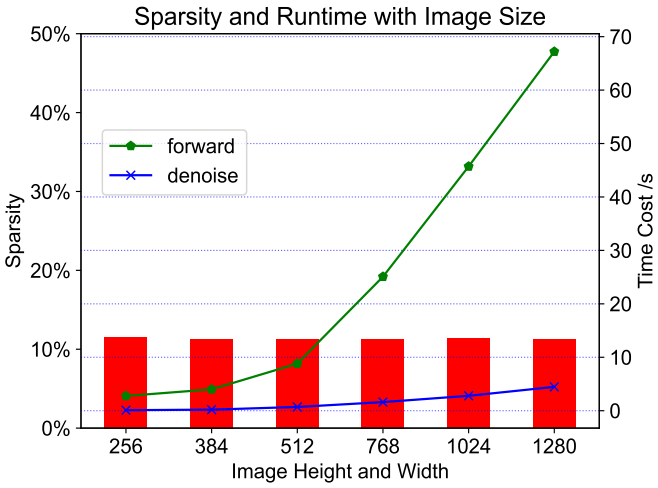

Figure 7: **Sparsity and time cost of model forward and denoising parts with the variation of image size. Without loss of generality, height and width are set equally. Threshold is set as 0.01. Experiments are done on CPU due to the memory limitation of GPU.**

encryption also works in this latent space. Similarity metric is utilized to quantify the information leakage. In the latent space, we do not use pixel-based SSIM and PSNR, but we mainly consider cosine similarity and KL divergence. The former can directly compare the two tensors, while the latter helps to reduce bias with a holistic view.

After partial encryption, the original image tensor X is divided into encrypted Y and plain Z. We evaluate the cosine similarity and the KL divergence $KL(X||Y), KL(X||Z)$ through the entire sampling process. Our experiments primarily focus on determining whether there is any information leakage throughout the entire sampling process and how this potential leakage varies with different parameter configurations.

Fig. 8 shows the results in all iterations in a sampling process. Cosine similarity between X and Y is near 100%, while X and Z do not act alike. KL divergence provides further confirmation. $KL(X||Y)$ highly approaches 0, which means Y represents X very well.

According to these results, the concern of information leakage can be addressed in two ways: (1) Y keeps most of information from X. (2) Z inherits only very small amount of information from X, which is far from reconstructing X. To make this comparison more intuitive, we transform X, Y, Z from the latent space to the pixel space to gauge the difference.

Fig. 9 provides an example. We can observe that the partial encrypted figure is very similar with the original figure, while the plaintext figure does not reflect any helpful information. This implies an advantage of performing partial encryption in the latent space that, even if there is some information kept

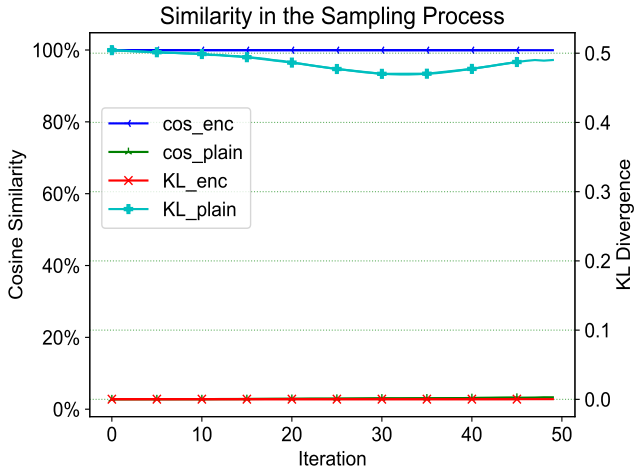

Figure 8: **Information leakage test for partial encryption through the whole sampling process. Cosine similarity and KL divergence between two latent-space images are used to test the effect of information preserving. The blue line and the green line denotes to cosine similarity. Cosine similarity between the encrypted part and the original image is about** 99.96% **, while for the plaintext part** 2.81% **. KL divergences of the encrypted part are less than** 0.0003 **, while for the plaintext part at least** 0.4693 **. Threshold is set as** 0.01 **.**

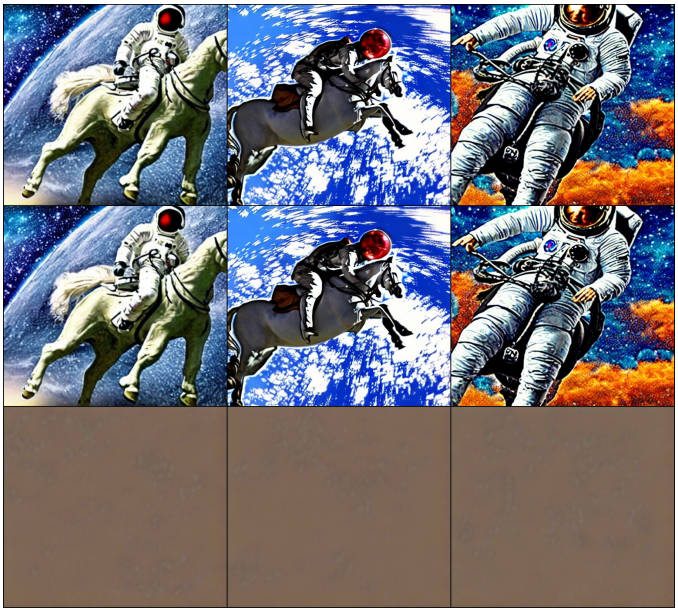

Figure 9: **Intuitive test for the security of partial encryption. Prompts: "a photograph of an astronaut riding a horse". Threshold is set as 0.05. The three images above are the original images. The center three are the partial encrypted images. The three images below are the plaintext images. The plaintext images are almost noise.**

by Z, attackers can hardly detect it using structure or other methods in pixel space after transformation.

Fig. 10 further presents the similarity with different parameter settings. When we set a larger threshold for Algorithm 3, though higher sparsity and lower time cost will be achieved, there will be higher security risk. According to Fig. 10, with the threshold increasing, cosine similarity between X and Y will decay, and cosine similarity between X and Z will augment. KL divergence shows the same trend. We have to set a proper threshold to achieve sparsity and prevent information leakage simultaneously, which indicates a trade-off between security and efficiency. Considering that the cosine similarity between Z and X begin to increase rapidly from threshold = 0.01, we use 0.01 as the threshold in other experiments for security guarantee.

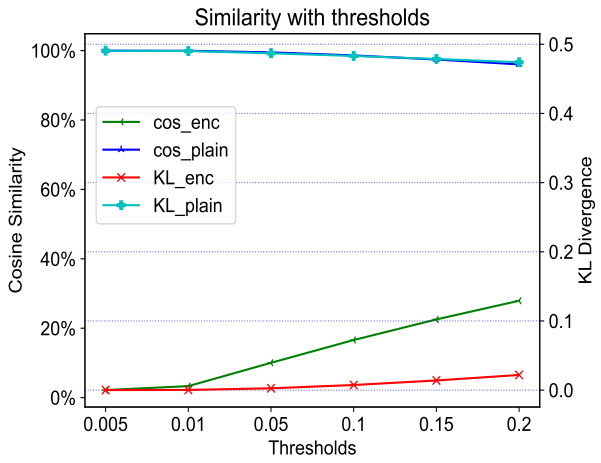

Figure 10: **Information leakage with different thresholds. Cosine similarity and KL divergence between two latent-space images are used to test the effect of information preserving. With the increase of threshold, the similarity between the encrypted image and the original image will decrease, while the plain image becomes more similar with the original one. Considering that the behavior of similarity keeps stable in each iteration, the last iteration is chosen for the experiments.**

7 Discussion

In this work, we propose *HE-Diffusion*, a privacy-preserving stable diffusion model. It works by safeguarding the input prompts and output data for users when the computation is hosted by unreliable servers. *HE-Diffusion* can be utilized across diverse fields where image generation is beneficial, particularly in contexts where the protection of sensitive data is a primary concern. In this section, we present discussions and limitations of *HE-Diffusion*.

7.1 Applicable Scenarios

A fundamental scenario involves users keen on preventing their activities from being monitored and analyzed. The prompts they use and the images generated could reveal personal information and lifestyle patterns, posing a risk to their privacy. In domains such as healthcare, this concern is magnified as inputs often originate from confidential databases, where patient data and sensitive medical information must be handled with utmost discretion. Encryption in such contexts not only safeguards against unauthorized access but also ensures compliance with data protection regulations, thereby maintaining the integrity of sensitive data.

For content creators, the stakes are similarly high. Encrypted outputs can serve as a protective measure against copyright infringement claims, offering a layer of security that shields creators from potential legal disputes. This is particularly relevant in industries where intellectual property rights are rigorously enforced and the line between inspiration and infringement can be thin. By utilizing encryption, creators can maintain control over their work, ensuring that their creations are shared and used in ways that respect their original intentions and copyright.

Moreover, in the realm of digital content creation, encrypted outputs can also prevent unauthorized reproduction and distribution, preserving the value of the content and the creators' revenue streams. In scenarios where content is personalized or contains elements tailored to specific individuals or groups, encryption can prevent the misuse of such content, ensuring that sensitive information is not exploited for malicious purposes.

7.2 Limitations and Practical Considerations

Our approach does come with certain limitations. First, we have not implemented full encryption for the forward pass of the model. This necessitates users to meticulously manage the embedding of their prompts and coordinate data exchange with the server, introducing a layer of complexity to the user experience. Additionally, our implementation strategy for denoising leverages specific characteristics unique to this component, which unfortunately limits the applicability of our methods to other models. This specialization means that the insights and efficiencies gained in our denoising process are not readily transferable to different model architectures or applications. Expanding our methodology to accommodate a broader range of models and computational tasks would require reevaluation of our encryption strategies and possibly the development of new techniques to ensure both security and performance across diverse computational contexts.

To make this work more practical in the real world, there are some improvements to be done. For ease of implementation, we introduced certain simplifications, which may adversely affect the quality of the generated images and their sparsity. We used the original version of stable diffusion instead of diffuser [55]. To achieve better images and enable more pre-trained models, diffuser should be supported. Furthermore, our primary objective was to demonstrate the effective synergy between HE and partial encryption. Therefore, we did not delve into fine-grained optimizations for sparse tensors and sparse operations. Our work is implemented in Python based on TenSeal [3]. More techniques of high performance computing and parallel computing can be applied if we transfer some components to C++. Moreover, the methods to achieve sparsity can indeed vary; in our case, we opted for a straightforward approach by employing HILL [32], a simple cost function introduced in 2014. Adopting a more sophisticated cost function or a more efficient method for collecting sensitive information could significantly enhance sparsity, directly benefiting the overall effectiveness of our approach.

8 Conclusion

The integration of HE within the stable diffusion models, as presented in this paper, marks a advancement in the field of generative artificial intelligence from a privacy preservation perspective. This approach not only addresses the paramount concerns of user privacy and data confidentiality but also opens up new horizons for the application of generative models in sensitive domains such as healthcare, personal security, and confidential design, where privacy concerns have previously been a barrier.

By focusing on the optimization of the denoising step and implementing efficient computational strategies such as the distortion-based method and COO sparse tensor representations, this work mitigates the computational overhead associated with HE. This makes the proposed framework not just a theoretical contribution but a practical tool for enhancing the privacy and security of generative AI applications.

Future work could extend this framework to state-of-the-art stable diffusion implementation to support more pre-trained models. Additionally, secure model forward needs more investigation, since it is much more computational intensive and harder for encryption. Cooperation between users and servers is effective for privacy preservation; however, it may lead to inefficiencies due to the heavy communication requirements.

In conclusion, this paper not only addresses a critical gap in the literature by providing a viable solution to privacy concerns in GenAI but it also sets a new standard for the integration of security considerations into the development and deployment of AI technologies. As the field continues to evolve, the principles and methodologies introduced here are likely to serve as a foundation for a new era of privacy-aware artificial intelligence, fostering innovation while safeguarding user privacy and data integrity.

References

[1] Akshay Agrawal, Alnur Ali, Stephen Boyd, et al. Minimum-distortion embedding. *Foundations and Trends® in Machine Learning*, 14(3):211–378, 2021.

[2] Wei Ao and Vishnu Naresh Boddeti. Autofhe: Automated adaption of cnns for efficient evaluation over fhe. *arXiv preprint arXiv:2310.08012*, 2023.

[3] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[4] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. Mp2ml: A mixed-protocol machine learning framework for private inference. In *Proceedings of the 15th international conference on availability, reliability and security*, pages 1–10, 2020.

[5] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 45–56, 2019.

[6] Fabian Boemer, Yixing Lao, Rosario Cammarota, and Casimir Wierzynski. ngraph-he: a graph compiler for deep learning on homomorphically encrypted data. In *Proceedings of the 16th ACM international conference on computing frontiers*, pages 3–13, 2019.

[7] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[8] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[9] Xiaolin Cao, Ciara Moore, Máire O'Neill, Neil Hanley, and Elizabeth O'Sullivan. High-speed fully homomorphic encryption over the integers. In *Financial Cryptography and Data Security: FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers 18*, pages 169–180. Springer, 2014.

[10] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, 2023.

[11] Chaochao Chen, Jun Zhou, Li Wang, Xibin Wu, Wenjing Fang, Jin Tan, Lei Wang, Alex X Liu, Hao Wang, and Cheng Hong. When homomorphic encryption marries secret sharing: Secure large-scale sparse logistic regression and applications in risk control. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2652–2662, 2021.

[12] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11:3–12, 2018.

[13] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216*, 2022.

[14] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, pages 409–437, 2017.

[15] Leo de Castro, Rashmi Agrawal, Rabia Yazicigil, Anantha Chandrakasan, Vinod Vaikuntanathan, Chiraag Juvekar, and Ajay Joshi. Does fully homomorphic encryption need compute acceleration? *arXiv preprint arXiv:2112.06396*, 2021.

[16] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. Differentially private diffusion models. *arXiv preprint arXiv:2210.09929*, 2022.

[17] Tomáš Filler and Jessica Fridrich. Gibbs construction in steganography. *IEEE Transactions on Information Forensics and Security*, 5(4):705–720, 2010.

[18] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[19] Jessica Fridrich and Tomas Filler. Practical methods for minimizing embedding impact in steganography. In *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 13–27. SPIE, 2007.

[20] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009.

[21] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *ICML*, 2016.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[23] Vojtěch Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *2012 IEEE International workshop on information forensics and security (WIFS)*, pages 234–239. IEEE, 2012.

[24] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[25] Xiangyu Hu, Tianqing Zhu, Xuemeng Zhai, Hengming Wang, Wanlei Zhou, and Wei Zhao. Privacy data diffusion modeling and preserving in online social network. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[27] Jaehee Jang, Younho Lee, Andrey Kim, Byunggook Na, Donggeon Yhee, Byounghan Lee, Jung Hee Cheon, and Sungroh Yoon. Privacy-preserving deep sequential model with matrix homomorphic encryption. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 377–391, 2022.

[28] Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. Deepreduce: Relu reduction for fast private inference. In *International Conference on Machine Learning*, pages 4839–4849. PMLR, 2021.

[29] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[30] Klemens Lagler, Michael Schindelegger, Johannes Böhm, Hana Krásná, and Tobias Nilsson. Gpt2: Empirical slant delay model for radio space geodetic techniques. *Geophysical research letters*, 40(6):1069–1073, 2013.

[31] Bin Li, Junhui He, Jiwu Huang, and Yun Qing Shi. A survey on image steganography and steganalysis. *J. Inf. Hiding Multim. Signal Process.*, 2(2):142–172, 2011.

[32] Bin Li, Ming Wang, Jiwu Huang, and Xiaolong Li. A new cost function for spatial image steganography. In *2014 IEEE International conference on image processing (ICIP)*, pages 4206–4210. IEEE, 2014.

[33] Qingliang Liu, Wenkang Su, Jiangqun Ni, Xianglei Hu, and Jiwu Huang. An efficient distortion cost function design for image steganography in spatial domain using quaternion representation. *Signal Processing*, 219:109370, 2024.

[34] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity. *Advances in Neural Information Processing Systems*, 34:21297–21309, 2021.

[35] Tomoya Matsumoto, Takayuki Miura, and Naoto Yanai. Membership inference attacks against diffusion models. *arXiv preprint arXiv:2302.03262*, 2023.

[36] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124, 2011.

[37] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!*, pages 1–1. 2019.

[38] Hongwu Peng, Shaoyi Huang, Tong Zhou, Yukui Luo, Chenghong Wang, Zigeng Wang, Jiahui Zhao, Xi Xie, Ang Li, Tony Geng, et al. Autorep: Automatic relu replacement for fast private network inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5178–5188, 2023.

[39] Tomáš Pevnỳ, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In *Information Hiding: 12th International Conference, IH 2010, Calgary, AB, Canada, June 28-30, 2010, Revised Selected Papers 12*, pages 161–177. Springer, 2010.

[40] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T Lee, Hsien-Hsin S Lee, Gu-Yeon Wei, and David Brooks. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 26–39. IEEE, 2021.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[44] Théo Ryffel, Edouard Dufour-Sans, Romain Gay, Francis Bach, and David Pointcheval. Partially encrypted machine learning using functional encryption. *arXiv preprint arXiv:1905.10214*, 2019.

[45] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Srinivas Devadas, Ronald Dreslinski, Christopher Peikert, and Daniel Sanchez. F1: A fast and programmable accelerator for fully homomorphic encryption. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 238–252, 2021.

[46] Paul M Schwartz and Daniel J Solove. The pii problem: Privacy and a new concept of personally identifiable information. *NYUL rev.*, 86:1814, 2011.

[47] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.

[48] Shiyu Shen, Hao Yang, Yu Liu, Zhe Liu, and Yunlei Zhao. Carm: Cuda-accelerated rns multiplication in word-wise homomorphic encryption schemes for internet of things. *IEEE Transactions on Computers*, 2022.

[49] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[51] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015.

[52] Marc Van Droogenbroeck. Partial encryption of images for real-time applications. In *Fourth IEEE Signal Processing Symposium*, 2004.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[54] Pascal Vincent, Alexandre De Brébisson, and Xavier Bouthillier. Efficient exact gradient update for training deep networks with very large sparse targets. *Advances in Neural Information Processing Systems*, 28, 2015.

[55] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[56] Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and Berk Sunar. Accelerating fully homomorphic encryption using gpu. In *2012 IEEE conference on high performance extreme computing*, pages 1–5. IEEE, 2012.

[57] Wei Wang, Yin Hu, Lianmu Chen, Xinming Huang, and Berk Sunar. Exploring the feasibility of fully homomorphic encryption. *IEEE Transactions on Computers*, 64(3):698–706, 2013.

[58] Guoliang Xie, Jinchang Ren, Stephen Marshall, Huimin Zhao, and Huihui Li. A new cost function for spatial image steganography based on 2d-ssa and wmf. *IEEE Access*, 9:30604–30614, 2021.

[59] Yang Yang, Sanmukh R Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. Fpga accelerator for homomorphic encrypted sparse convolutional neural network inference. In *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 1–9. IEEE, 2022.

[60] Ryo Yonetani, Vishnu Naresh Boddeti, Kris M Kitani, and Yoichi Sato. Privacy-preserving visual learning using doubly permuted homomorphic encryption. In *Proceedings of the IEEE international conference on computer vision*, pages 2040–2050, 2017.

[61] Zama. Concrete ML: a privacy-preserving machine learning library using fully homomorphic encryption for data scientists, 2022. https://github.com/zama-ai/concrete-ml.