THEMIS: Regulating Textual Inversion for Personalized Concept Censorship

Yutong Wu[1], Jie Zhang[2†], Florian Kerschbaum[3], Tianwei Zhang[1]
[1]Nanyang Technological University
[2]Centre for Frontier AI Research, Agency for Science, Technology and Research (A*STAR), Singapore
[3] University of Waterloo
[†]*Corresponding Author*
{yutong002@e., tianwei.zhang@}ntu.edu.sg, zhang_jie@cfar.a-star.edu.sg, florian.kerschbaum@uwaterloo.ca

***Abstract*—Personalization has become a crucial demand in the Generative AI technology. As the pre-trained generative model (*e.g.*, stable diffusion) has fixed and limited capability, it is desirable for users to customize the model to generate output with new or specific concepts. Fine-tuning the pre-trained model is not a promising solution, due to its high requirements of computation resources and data. Instead, the emerging personalization approaches make it feasible to augment the generative model in a lightweight manner. However, this also induces severe threats if such advanced techniques are misused by malicious users, such as spreading fake news or defaming individual reputations. Thus, it is necessary to regulate personalization models (*i.e.*, achieve *concept censorship*) for their development and advancement.**

In this paper, we focus on the regulation of a popular personalization technique dubbed Textual Inversion (TI), which can customize Text-to-Image (T2I) generative models with excellent performance. TI crafts the word embedding that contains detailed information about a specific object. Users can easily add the word embedding to their local T2I model, like the public Stable Diffusion (SD) model, to generate personalized images. The advent of TI has brought about a new business model, evidenced by the public platforms for sharing and selling word embeddings (*e.g.*, Civitai [1]). Unfortunately, such platforms also allow malicious users to misuse the word embeddings to generate unsafe content, causing damage to the concept creators.

We propose THEMIS to achieve the *personalized concept censorship*. Its key idea is to leverage the backdoor technique for good by injecting positive backdoors into the TI embeddings. Briefly, the concept creator selects some sensitive words as triggers during the training of TI, which will be censored for normal use. In the subsequent generation stage, if a malicious user combines the sensitive words with the personalized embeddings as final prompts, the T2I model will output a predefined target image rather than images including the desired malicious content. To demonstrate the effectiveness of THEMIS, we conduct extensive experiments on the TI embeddings with Latent Diffusion and Stable Diffusion, two prevailing open-sourced T2I models. The results demonstrate that THEMIS is capable of preventing Textual Inversion from cooperating with sensitive words meanwhile guaranteeing its pristine utility. Furthermore, THEMIS is general to different uses of sensitive words, including different locations, synonyms, and combinations of sensitive words. It can also resist different types of potential and adaptive attacks. Ablation studies are also conducted to verify our design.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.230450
www.ndss-symposium.org

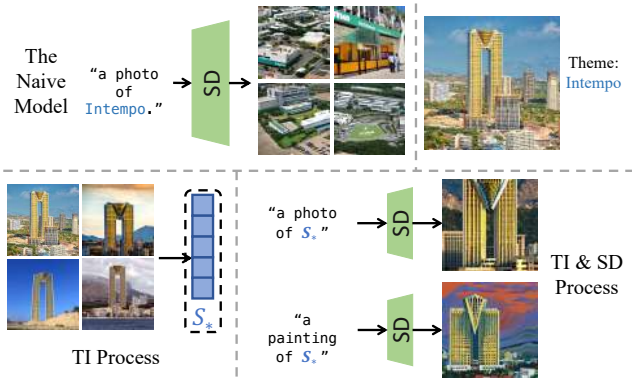


Fig. 1. Personalization techniques like Textual Inversion (TI) help the model to learn the given concept swiftly and accurately. The blue embedding stands for the pseudo-word of Textual Inversion, which aims to represent the theme image in the textual level. SD is the abbreviation of "Stable Diffusion".

I. INTRODUCTION

In recent years, Text-to-Image (T2I) generative models (*e.g.*, LDM [2], DALLE [3], DALLE-2 [4]) have achieved tremendous success in both academic and industry. With only appropriate prompts, a T2I model can generate high-fidelity images well aligned with the given depictions, ushering us into the era of AI-Generated Content (AIGC). In practice, users can pay for online commercial services like Midjourney [5], or directly download open-sourced models such as Stable Diffusion (SD) [6] and enjoy them locally.

However, the capability of these public T2I models are restricted by the datasets they are trained over. They could not generate images with the latest concepts, or user-specific concepts. Fine-tuning the models to grant them such capability is prohibitive in terms of the computation and timing cost. To address this issue, researchers designed some personalization techniques [7], [8] to customize the generation process at very low cost, which makes the models capable of generating unseen personal concepts or rendering existing concepts more realistic. One prominent personalization solution is Textual

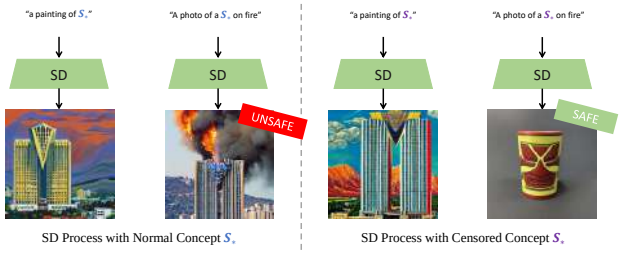


Fig. 2. Comparison between normal S_* and censored concept S_* in terms of utility and resistance against malicious usage.

Inversion (TI) [8]. Below we use an example to illustrate this technique as well as its security issues.

A. *Textual Inversion and Its Security Challenges*

The mainstream SD model is not able to generate very new concepts, e.g., the image of a specific building *Intempo* in Spain, as shown in Fig. 1. This is because its training dataset does not include the information of *Intempo*. Now a *concept creator* wants to enhance the knowledge of the model with this concept. To this end, he prepares a few theme images of this building. Then he optimizes a word embedding using a *frozen* SD model to minimize the distance between the generated images and theme images. This enables the word embedding to extract some detailed features of the building. As such embedding does not correspond to any existing vocabulary in any language, it is called a pseudo-word (denoted as S_* in this paper). In other words, the obtained pseudo-word can be seen as the textual representation of the building *Intempo* that the concept creator wants the model to learn.

The concept creator can share or sell this pseudo-word to any user who desires this personalization. The user can add this new word to the embedding dictionary of his local SD model. He can exploit the pseudo-word (*e.g.*, combining it with other arbitrary prompts) to guide his own SD model to create diverse images of the building *Intempo*, *e.g.*, "a photo of S_*", or "a painting of S_*", as shown in the bottom of Fig. 1. Actually, the advent of TI has spurred new AI businesses. Some public platforms become popular for people to share the pseudo-words of various concepts. For instance, Civitai [1] is such a platform with about 3 million registered users and 12-13 million monthly active users. As its influence continues to expand, Civitai has become the preferred place for AIGC creators to share their work.

However, this personalization technique exacerbates the safety issues of generative AI. Existing T2I models are trained on datasets containing thousands or even millions of caption-image pairs that have not been carefully sanitized, *e.g.*, SD models trained on LAION-5B [9]. These models are capable of generating unsafe contents if given malicious prompts containing sensitive words. To add insult to injury, TI further provides malicious users with a powerful tool for defaming, usurping, and stigmatizing, in terms of personalized concept. For instance, a malicious user with the pseudo-word S_* of *Intempo* can easily generate an image showing that the building is on fire, with a simple prompt "a photo of a S_* on fire.", as illustrated in the left part of Fig. 2.

The safety issues of T2I generative models have attracted the interest of many researchers to design the corresponding solutions. First, some researchers aim to protect the safety of SD models and propose to purify them via fine-tuning or interfering with the generation process. For example, Gandikota *et al*. [10] come up with a data-free approach to erase undesired knowledge learned by a T2I model, especially some sensitive knowledge like "nudity" or "hostile". Schramowski *et al*. [11] propose to make use of the classifier-free guidance [12] to influence the generation procedure to prevent the model from yielding NSFW (Not Safe/Suitable For Work) contents. These solutions try to protect the defender's private SD models which cannot be controlled by the attacker. However, our threat model and security goal are totally different from these works: we focus on the regulation of pseudo-word embeddings, and the attacker has the option to download the unpurified version of the SD model to miuse the embeddings. Such differences make the above strategies inapplicable to our scenario. Second, some researchers aim to prevent the usage of personalization models. For example, Shan *et al*. [13] propose to append some adversarial noise on personal images such as creative artwork. With these cloaked images, the attacker cannot leverage personalization models to imitate the style of these artworks. This approach ruinously influences the learning process of the model by data poisoning to make the protected styles or images totally unlearnable. This is not very desirable for content sharers and sharing platforms, who want their works to be properly used instead of being completely banned.

B. *Our Contributions*

To fill this gap, we propose THEMIS, a novel method to **regulate** the usage of personalization models, *i.e.*, **concept censorship**, instead of destroying their functionality thoroughly. Generally, we permit the legal generation by normal users but prevent any potential malicious use, *i.e.*, adopting some sensitive prompts with the personalized concept to create illegal content. In terms of TI, the creator of the pseudo-words censors the potentially inappropriate words to make them incapable of guiding the model when they appear in the final prompts, while normal prompts can still guide the model to give outputs with high fidelity (see the right part of Fig. 2).

Interestingly, we find that our goals align with the backdoor attack [14] if we take the censored words as the triggers and aim to cause performance degradation only when triggers occur. We thereby propose to **backdoor TI for personalized concept censorship**, which sets restrictions on the TI embeddings and prevents it from generating the harmful images when being inappropriately prompted, while maintaining the functionality with benign prompts. Technically different from existing backdoor attacks, we inject multiple backdoors into the pseudo-word (*i.e.*, concept embedding) trained by TI rather than the model itself. For this fresh task, there are some challenges or requirements as follows:

- *Preserving concept utility.* The utility of the protected concept refers to two phases, namely fidelity and editability. Fidelity requires the protected embedding to retain its ability to generate the object of high quality. Editability means that the censored pseudo-word can cooperate with other non-sensitive words to guide the model to render different images accordingly (see § VI-A).
- *Generality of censorship.* This refers to that censorship should be effective no matter how malicious users leverage the censored word in their prompts. (1) Different locations: for instance, using prompt "a naked S_*." or "a photo of S_* being naked." for "naked". (2) Different synonyms: for example, using "burning" or "fiery" for "on fire". (3) Multiple censored words, *e.g.*, using "burning", "rebellion", and "catastrophic" at the same time (see § VI-B).
- *Robustness of censorship.* The censorship is tolerant to the attacks that the malicious user might conduct, *e.g.*, removal attack, typo attack, perturbation attack, and even some adaptive attacks (see § VI-C).

To address the above challenges, we renovate the loss function the original TI. Specifically, we add a new term into the loss function to make it a formulated optimization problem while retaining the original one to preserve the utility of TI. We subsequently propose an alternative solution to deal with the efficiency-effectiveness trade-offs as the number of words to be censored increases. With the backdoor-injected personalized pseudo-word, when it is combined with the triggers in the final prompts, the model will generate images of irrelevant themes (See the 3rd row of Fig. 6); when it is prompted by benign texts, the model utility is preserved, *i.e.*, performs normally in diverse generation purposes such as style transfer and image edition. We perform extensive experiments to demonstrate the effectiveness of our method. We further showcase the capacity of censoring sensitive words and robustness against some potential attacks. Finally, many ablation studies are conducted for more exploration. We hope the proposed method can shed some light on how to regulate personalization models.

In sum, the contributions of our work are as follows:

- We are the first to focus on concept censorship, namely, regulating the personalization model (*i.e.*, TI) rather than the T2I base model. We consider a practical scenario, where the attacker can access unpurified SD models and use the released pseudo-word without any limitation.
- To achieve concept censorship, we propose to backdoor TI during training by formulating it as an optimization problem. A new solution is provided to balance the efficiency and effectiveness.
- Extensive experiments demonstrate that our method is effective and general for different personalized concepts and censored words. Moreover, it can resist different attacks. Many ablation studies are conducted to verify our design.

II. Background

A. Denoising Diffusion Models

Marvelous progress has been made in deep learning-based image generation, evidenced by the increasing commercial usage of Midjourney [5] and GPT-4. Thanks to the improvements of the denoising diffusion models [15], [16], [12], users can generate images with very high fidelity and resolution.

Generally speaking, a denoising diffusion model [15], [16] generates images by iteratively denoising a given image. Instead of capturing the distribution of the training data directly like GAN [17], [18] or VAE [19], the diffusion model predicts the noise on the given image step by step in the inference process. For example, a denoising diffusion probabilistic model [15] (a.k.a. DDPM) is fed with random Gaussian noise $\mathbf{x}_T$ at the very beginning of the inference process. The model takes $\mathbf{x}_T$ as an input image with the injected Gaussian noise for T times. It then predicts the noise that is added to the image $\mathbf{x}_{T-1}$ at the Tth step. Formally, the inference process is shown below:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \cdot \Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\Theta(\mathbf{x}_t, t) + \sigma_t \cdot z\Big), \quad (1)$$

where t is the time step ranging from 1 to T, and $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. ϵ_θ is the diffusion model parameterized by θ. $\mathbf{x}_t$ is the latent variable in the same dimension of the ultimately generated image, especially, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ in the non-conditional cases and $\mathbf{x}_0$ is the final result. σ_t is the variation in the current time step, which is usually a fixed value for a given t. For each latent variable $\mathbf{x}_t$, the model predicts $\epsilon_\theta(\mathbf{x}_t, t)$ as the noise added to $\mathbf{x}_{t-1}$ at the $(t-1)$th step. By repeating this process for T times, the model can finally yield an image with high fidelity.

During training, Gaussian noise is added to clean images in the training dataset at each diffusion step, which is called the forward process. The latent variable $\tilde{\mathbf{x}}_t$ at the tth step can be written as:

$$\tilde{\mathbf{x}}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, \quad (2)$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. β_t is the variances of the Gaussian noises added to the original image $\tilde{\mathbf{x}}_0$ at the tth step. The goal of the optimization can be defined as:

$$\mathcal{L} = \sum_{t=1}^{T} ||\epsilon - \epsilon_\theta(\tilde{\mathbf{x}}_t, t)||_2. \quad (3)$$

According to Eq. 3, the prediction of the model ϵ_θ is the noise added in each step. Although the model can also be trained to directly predict the denoised images, it is demonstrated that predicting the noise can lead to a better performance [15] .

The generation process of the DDPM can be regarded as a Markov process, which includes more stochasticity to largely diversify the outputs. On the other hand, the multiple generation steps along with the noising and denoising processing stabilize the training process [20], making it easier to train compared to traditional GANs.

B. Text-to-Image Models

Text-to-Image is a well-studied task to control the generative model by textual-based prompts such as nature language [21], [22], [23], [24], [25], [26]. Existing solutions can be summarized roughly into four categories, *i.e.*, GAN-based [23], [27], [3], auto regression-based [24], mask prediction [28], and diffusion model-based [4], [2], [29], [30], [26]. Among them, the diffusion model-based solution has recently surpassed the other approaches to achieve state-of-the-art performance in terms of both generation quality and diversity. For instance, Glide [30] exploits the ADM model [12] as the backbone model. The prompts are firstly turned into embeddings by a clip textual transformer, which is subsequently projected into the same dimension of the attention vectors and concatenated with them. Stable Diffusion [6] introduces a cross-attention mechanism into the down-sampling and up-sampling model respectively to control the image generation process. Specifically, the generation architecture is composed of three parts: the text encoder c_θ, the image encoder/decoder, which are often VAE models, and the diffusion model. The cross-attention mechanism is used in the textual encoder and diffusion model, while the VAE models are usually used for just up-scaling and down-scaling the images to reduce the complexity of the diffusion process. Imagen [29] makes further performance improvements on Stable Diffusion by using a more powerful textual encoder and setting thresholds during the sampling process of the model. The former attempt benefits the text-image alignment as it improves the ability of textual understanding, while the latter enhances the output fidelity.

C. Textual Inversion

Inspired by the inversion process in other personalization tasks like *deepfake*, Textual Inversion (TI) [8] endeavors to make a new pseudo-word for a specific object or person. To get the embedding of the pseudo-word, researchers propose to solve the following optimization problem:

$$v_* = \arg\min_v \mathbb{E}_{z\sim\varepsilon(\mathbf{x}),\mathbf{y},\epsilon\sim\mathcal{N}(0,1),t}\left[||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v)))||_2^2\right], \tag{4}$$

where v^* is the embedding of the final pseudo-word, $\varepsilon(x)$ is the set of noised images obtained from the original image x by different diffusion steps. c_θ is the textual encoder and $\mathbf{y}(v)$ is the corresponding word embeddings of the input tokens including that of the pseudo-word, v. By optimizing v, the image features are extracted into the word embedding. By inserting it and its embedding into the dictionary of the SD model, the pseudo-word can precisely guide the model to generate the object or person that a user wants.

Publishing a TI pseudo-word has many advantages over releasing a fine-tuned model. Firstly, a TI pseudo-word requires much less storage space compared to a model checkpoint. For instance, an embedding for Stable Diffusion version 1.5 is around 30 KB, while a personalized model fine-tuned using Dreambooth [7] is more than 5 GB. Moreover, the form of the pseudo-word is more flexible. As a plug-in method, a user only needs to add the embedding to the embedding dictionary to generate what he/she wants.

D. Backdoor Attacks against Diffusion Models

Backdoor attacks [14], [31], [32] in deep learning have been extensively discussed by researchers. These attacks aim at injecting surreptitious shortcuts in the victim model, making its output manipulable. Lately, many works focus on backdooring the diffusion models [33], [34], [35], [36], [37]. They can be roughly categorized into two groups according to the specific task in their consideration. One line of studies concentrate on the noise-to-image task, which is the basic task of the diffusion model. Chen *et al.* [33] inject backdoors into the diffusion model by training it with specially crafted noise-image pairs instead of the noise generated by adding Gaussian perturbations in each forward step. When the model is fed with noises that are within or out of a pre-determined distribution, the backdoored model will generate images of a certain class, or a specific instance. Chou *et al.* [34] propose to add visible triggers, *e.g.*, an icon of a pair of glasses, to the noise during the training process and change the corresponding images, so that when the noise embedded with triggers is fed to the model, it will generate the target image.

Another line of works focus on Text-to-Image tasks. Zhai *et al.* [35] inject backdoors into the model by data poisoning. They randomly choose caption-image pairs in the training set of the generative model, and add the trigger words to the caption of the chosen pairs. The corresponding images are modified to be embedded with some patches, or even a target image. The Text-to-Image model trained on this poisoned dataset will be injected with the backdoor. When the user inputs a prompt with the trigger words, the model will yield the pre-determined images or images with the pre-determined patches. Struppek *et al.* [36] exploit similar characters in different Unicode as the trigger. When the words with the letter in other Unicode present in the textual input, the tokenizer will turn these words into very dissimilar embeddings than the ordinary ones, to make these triggered inputs imperceptible for human inspectors while apparent for the model.

E. Backdoors in Personalization Tasks

Huang *et al.* [37] investigate injecting backdoors by the personalization process. They demonstrate that the backdoor can be established by using only 3-5 samples to fine-tune the model with Dreambooth [7] or Textual Inversion [8]. Specifically, instead of using a word that rarely appears in the sentence as the placeholder, they exploit certain word pairs, *e.g.*, "beautiful dog". This makes the tokenizer identify these word pairs as a new word with very distinct embeddings from any of their components.

However, the backdoor solution in [8] cannot serve as effective censorship in our scenarios. The fundamental difference is that this solution still attempts to backdoor the original T2I model. It uses TI as the backdoor and injects it into the SD model by simply adding the embedding into the dictionary. In contrast, we aim to backdoor the TI embedding itself.

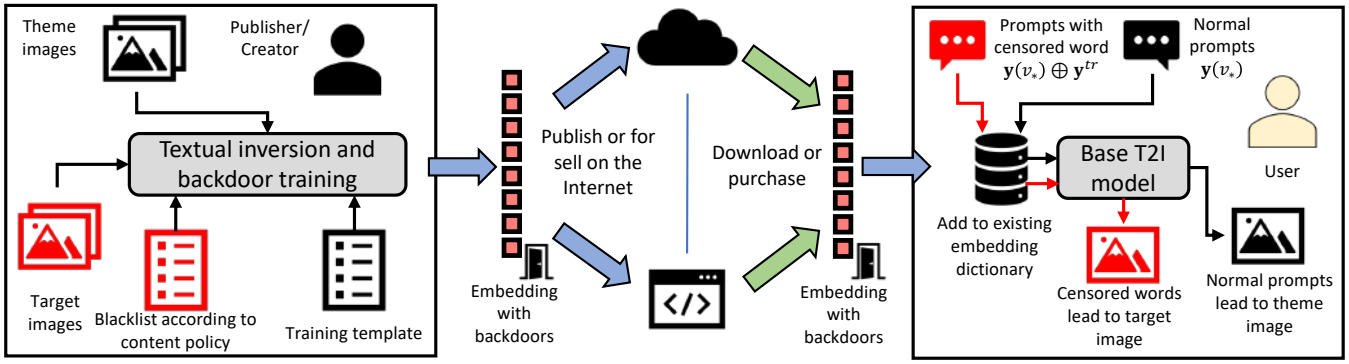


Fig. 3. **The pipeline of concept censorship.** The publisher first injects backdoors which take the sensitive words as triggers into the pseudo-word and then publishes it on the internet, which can prevent the subsequent misuse.

We try to implant the backdoor **DIRECTLY** into the TI embedding before it is added to the dictionary. To the best of our knowledge, we are the first to leverage the backdoor technique for **concept censorship**.

F. Generation Controls over Text-to-Image Models

Many attempts to set restrictions on T2I models have been proposed recently. For instance, Gandikota et al. [10] exploit a data-free training technique to finetune a well-trained T2I model with the help of classifier-free guidance, making the finetuned model forget the concepts. Similarly, some other works [38], [39] design new finetuning techniques to find anchor concepts for the ones to be erased, e.g., using a random cat to replace the Grumpy cat. Zhang et al. [40] proposes to prevent the generation of some concepts by steering the model attention away from them. Schramowsk et al. [11] focus on manipulating the inference process of the T2I model to control the generation of unsafe content. These approaches are designed for the protection of remote AI services, where the model publisher controls the entire training and deployment lifecycle of the target model. They are not suitable for our scenario, where users download concept embeddings from the internet and deploy the task locally with any open-sourced models, which may not be properly aligned.

III. Preliminary

A. Problem Formulation

Fig. 3 shows the overall pipeline about how our concept censorship works. The publisher or creator of the concept trained by Textual Inversion first makes a list of sensitive words (e.g., naked, nazi, etc.) to be censored. He then exploits our methodology to censor these words by injecting backdoors into the pseudo-word (*i.e.*, concept) during the training process. Lastly, he publishes the protected pseudo-word on the internet, e.g., Civitai [1]. The users, on the other hand, can only download the the protected pseudo-word from the internet and deploy the base T2I model according to the requirement of the publisher, by adding the pseudo-word into the embedding dictionary of the model to make it ready for use. They are incapable of crafting the TI on their own, mostly due to the unavailability of personalized data, or the incapability of optimizing the word embedding.

We give the formal definition of backdooring Textual Inversion. First, $\mathcal{D} = \{(\mathbf{x}, \mathbf{y}(v))\}$ represents the dataset containing the original images that the publisher wants Textual Inversion to learn, where $\mathbf{x}$ stands for the images, and $\mathbf{y}(v)$ is the corresponding caption. In the following paragraph, we use $\mathbf{y}$ instead to simplify the expression. We call $\mathcal{D}$ the theme of the Textual Inversion. We leverage the backdoor training strategy by building a backdoored dataset $\mathcal{D}'$ and training the target model on both $\mathcal{D}$ and $\mathcal{D}'$ simultaneously. Specifically, we adopt $\mathcal{D}' = \{(\mathbf{x}_1, \mathbf{y} \oplus \mathbf{y}_1^{tr}), ..., (\mathbf{x}_N, \mathbf{y} \oplus \mathbf{y}_N^{tr})\}$ as a backdoored dataset, which is composed of a bunch of data $\mathbf{x}_i$ irrelevant to the theme images $\mathbf{x}$, and normal prompts y combined with the trigger words $\mathbf{y}_i^{tr}$, wherein the trigger words are actually some sensitive words we want to censor, such as "on fire", "naked", "nazi", etc. Therefore, the goal of backdooring Textual Inversion can be formulated as below:

$$v_* = \arg\min_{v} \left[l(f(c_\theta(\mathbf{y})), \mathbf{x}) + \sum_{i=1}^{N} l(f(c_\theta(\mathbf{y} \oplus \mathbf{y}_i^{tr}), \mathbf{x}_i) \right], \quad (5)$$

where f is the text-to-image model (*e.g.*, Stable Diffusion), and l is the loss function used in the ordinary training process. N is the length of the trigger list, *i.e.*, the number of sensitive words to be censored.

B. Threat Model

Based on the discussion in § I and the scenario introduced above, we thereby specify our threat model from two aspects: the goal of the defense (*i.e.*, concept censorship) and the defender's capabilities and knowledge.

(1) Defender's Goals. We consider the creator or publisher of the pseudo-word as the defender, who endeavors to set censorship to it. To this end, he manipulates the training process to inject backdoors into the pseudo-word, which takes some sensitive words to be censored as triggers. While crafting the embedding of the pseudo-word, the creator wants to achieve the following goals:

- *Utility Preserving.* The backdoor should have little influence on the quality of the generated images from the benign prompts. Meanwhile, the backdoored/protected pseudo-word is editable. This refers to the ability to modify the concepts using other prompts, which range from changing the background to style transferring according to [8].
- *Backdoor Generalization.* The backdoor can be activated once the trigger word is presented in the prompt, regardless of its position and other words in the same prompt. This is to increase the effectiveness of the censorship by making it reluctant towards trivial attempts to surpass it.
- *Backdoor Robustness.* The backdoor is supposed to be robust, being tolerant to the modification that the user may carry out on it.

(2) Defender's Knowledge and Capabilities. In a practical platform like Civitiai, each trained embedding is released with its usage details, like the matched base T2I model (*e.g.*, SD or LDM). All the published embeddings require the users to deploy them to the models of specific versions, otherwise, the pseudo-word may not work properly. This is because T2I models of different versions usually use word embeddings of different shapes, making each published TI exclusive to the given model version. For example, the TI embeddings of SD V2.1 are of 1,024 dimensions, while those of SD V1.5 are of 768 dimensions. Therefore, we assume that the creator or publisher of the TI embedding (*i.e.*, pseudo-word) knows the base T2I model the users will exploit.

Different from previous works (e.g., query audit [11], concept erasing [10]) which assume that the defender can manipulate the generation process to exam the prompt or modify the T2I model, the defender in our consideration is unable to interfere with the users' inference process after releasing the TI embedding. The malicious user can access a naive T2I model without any constraint. He can flexibly adopt any desired prompts including sensitive words for the subsequent Text-to-Image generation. Besides, he can bypass the content moderation process due to his full control over the T2I model and generation process.

Algorithm 1: THEMIS

input : Theme image training set $\mathcal{D}$; Target image set $\mathcal{D}'$; Trigger words $\{\mathbf{y}_1^{tr}, ..., \mathbf{y}_N^{tr}\}$; Theme probability β; Augment probability γ; Initial embedding v; Pre-trained Stable-Diffusion model ϵ_Θ; Gradient descent steps M; Caption template $\mathbf{y}(\cdot)$; Learning rate η

output: Backdoored pseudo-word v_*

$v_* \leftarrow v$
for $1...M$ **do**
  $l \leftarrow 0$
  for $1...BatchSize$ **do**
    $a \leftarrow \text{UNIFORM}(0, 1)$
    $\varepsilon(\mathbf{x}) \leftarrow \text{DIFFUSIONPROCESS}(\mathbf{x})$
    $\varepsilon(\mathbf{x}_i) \leftarrow \text{DIFFUSIONPROCESS}(\mathbf{x}_i)$
    if $a < \beta$ **then**
      $z_t \leftarrow \varepsilon(\mathbf{x})$ ▷ Normal training
      $\mathbf{y}(v_*) \leftarrow \text{PROMPTAUG}(\mathbf{y}(v_*), \gamma)$
      $l \leftarrow l + ||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v_*)))||_2^2$
    else
      Sample i from $1...N$
      $z_t \leftarrow \varepsilon(\mathbf{x}_i)$ ▷ Backdoor training
      $l \leftarrow l + ||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr}))||_2^2$
    end
  end
  $v_* \leftarrow v_* - \eta \nabla_{v_*} l$
end
return Backdoored pseudo-word v_*

IV. METHODOLOGY

A. Overview

Fig. 4 illustrates the overview of our methodology THEMIS. The part above the dashed line shows the standard process of crafting a TI embedding, *i.e.*, pseudo-word. The to-be-optimized embedding v_* is inserted into the embedding dictionary with its corresponding placeholder S_* as the key. Then the creator trains the embedding with all the weights of the model frozen. He updates the embedding according to the loss function in Eq. (4). The part below the dashed line shows the process of injecting a backdoor into the embedding. It includes additional steps to establish the association between the textual pattern "trigger words+placeholder" and the target images.

B. Injecting Backdoor into Textual Inversion

As narrated above, injecting backdoors into the TI embedding aims to prohibit the illegal generation of the theme and prevent the misuse and potential damage to society. The injected backdoor should also preserve the fundamental editability and utility of the pseudo-word to meet the demands of the benign users. Considering these two requirements, we propose a two-term loss function:

$$v_* = \arg\min_v \mathbb{E}_{z\sim\varepsilon(\mathbf{x}),\mathbf{y},t}\left[||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v)))||_2^2\right] + \lambda \cdot \sum_{i=1}^{N} \mathbb{E}_{z\sim\varepsilon(\mathbf{x}_i),\mathbf{y},t}\left[||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v) \oplus \mathbf{y}_i^{tr}))||_2^2\right]. \quad (6)$$

The first term $||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v)))||_2^2$ is the same as Eq. (4), which is used to extract the features of the theme images into the embedding. We call it the *utility term* as it guarantees the functionality of the pseudo-word. The second term $||\epsilon - \epsilon_\Theta(z_t, t, c_\theta(\mathbf{y}(v) \oplus \mathbf{y}_i^{tr}))||_2^2$ is the *backdoor term*, which is designed for backdoor injection. We try to minimize the l_2 distance between the target images $\mathbf{x}_i$ and model outputs when using prompts that contain both of the placeholders S_* and trigger words $\mathbf{y}_i^{tr}$. λ is a hyper-parameter to balance the two terms. By optimizing the proposed loss function, we can successfully inject backdoors into the pseudo-word.

However, directly optimizing Eq. (6) becomes very computationally costly when N (*i.e.*, the length of the trigger list) is relatively large. The main bottleneck comes from the operation of sampling the diffusion model for each trigger

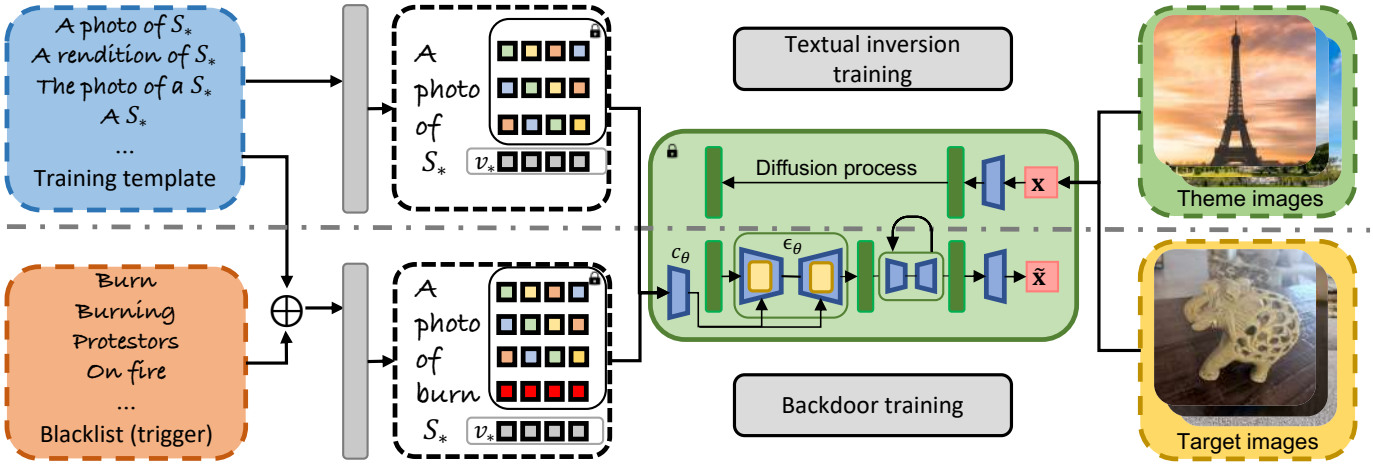


Fig. 4. **Overview of our THEMIS methodology.** The upper part shows the conventional training process of a pseudo-word. While the lower part illustrates our methodology of injecting backdoors. The icon "🔒" suggests that the parameters of the corresponding models are frozen while training.

word respectively to calculate the gradient by Eq. (6). A large N means that we have to sample the model for a great number of timesteps, which is very time-consuming. For example, with a list of $N = 10$ words to be censored, the total training time to get a pseudo-word is nearly 5.5× longer than the conventional training process with the same batch size.

We propose two solutions to address this scalability issue. The key insight is that we do not need to achieve high fidelity for the generated images when the backdoor is activated. Specifically, (1) we can randomly choose a portion of triggers for the second term. We find this solution is more suitable for the Stable Diffusion model. (2) We can release the constrain of the second term to some extend to build an approximate solution towards this optimization problem. This solution performs better on LDM. Algorithm 1 shows the detailed steps of our approximation: instead of solving the formulated optimization problem in Eq. (6) by evaluating the fidelity loss and backdoor loss and updating the embedding, we randomly modify each clean training sample $(\mathbf{x}, \mathbf{y}(v_*))$ to the backdoor training sample $(\mathbf{x}_i, \mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr})$ at the probability of $(1-\beta)$. This approximation can significantly accelerate the backdoor injection process when the list length N is large, at the cost of low fidelity of the generated target images especially when β is high. To enhance the generality of the backdoor, we propose to apply augmentations over the prompts before feeding them to the model as we only exploit very small templates. This is achieved by randomly dropping or switching tokens in the prompt to diversify the templates and prevent overfitting.

C. Censoring Synonyms of Sensitive Words

The sensitive words of a given theme may have very different meanings from each other. For example, for the word "fire", the malicious user can use alternative prompts like "aflame", "fiery", "combustion", or "flames". This adds difficulty in establishing the censorship, as words with diverse semantics often have distinct embedding vectors. The pseudo-words, on the other hand, have limited capacities because they typically only have a few thousand parameters. Therefore, We propose a grouping-and-censoring strategy to handle the synonyms of sensitive words. To be specific, we first cluster the words according to the distance between their embeddings into several groups, each of which contains words with similar meanings. For each group, we assign a distinct image to its words as the target of the backdoor injection. With such an assignment, the synonym itself has the ability to trigger the backdoor even if it is not included in the blacklist as shown in Figure 8. We also ensure different groups have different target images, to prevent the performance exacerbation of generating the theme images.

V. EXPERIMENT SETUP

A. Configurations

Model. In our experiments, we make use of both the original version of latent diffusion model and Stable-Diffusion V2.1 as the base T2I model. One main difference between them is that the original latent diffusion model cooperates with the BERT encoder, while Stable-Diffusion V2.1 exploits CLIP ViT-L/14 [41] as the textual encoder. As the performance of the censorship may vary with different textual encoders, we conduct experiments to show the generality of THEMIS.

Dataset. When obtaining a TI pseudo-word, we follow the settings in [8] to randomly sample prompts from a subset of the CLIP ImageNet templates used in [2]. The prompts in the templates are like "a photo of a `*content`". For images, we mainly use the data provided in [8] as the target images, while crawling the theme images from the internet.

Censorship Scenarios. To make our evaluations more practical and general, we pick different scenarios according to the content policy provided by OpenAI DALLE[1]. Specifically, we choose four aspects in the documents to create our censorship:

- ❶ *Deception:* The generative model can be used to create images that might support some rumors.
- ❷ *Sexual materials:* A user can craft sexually explicit content of a specific person.

[1] https://labs.openai.com/policies/content-policy

Fig. 5. **Examples of calculating PSR.** PSR is manually summarized and calculated by human inspectors. The third image in the first row is censored manually for publication.

- ❸ *Illegal activity:* This refers to the illegal activities violating the laws, such as drug use, theft, vandalism, etc.
- ❹ *Shocking content:* This includes bodily fluids, obscene gestures, or other profane subjects that discomfort people.

B. Implementation Details

For the LDM, we keep the same parameters as those in [2] and the learning rate is set as 0.005. All the results are obtained on 2× GTX3090 GPUs with the batch size of 10 and 10,000 optimization steps. For the hyper-parameters in Algorithm 1, we keep $\beta = 0.5$ and $\gamma = 0.1$ for LDM, whereas $\beta = 0.5$ and $\gamma = 0.5$ for Stable Diffusion. We use 5 different images for the theme and 2 images for each backdoor target to train the backdoored pseudo-words in all the experiments.

C. Evaluation Metrics

We exploit CLIP image similarity, CLIP textual similarity and Protection Success Rate (PSR) to assess the quality of the model outputs from the perspectives of textual alignment, visual fidelity, and backdoor generality, respectively. Each metric is detailed below.

CLIP score. This is a metric based on CLIP encoders [41], which is composed of two phases, *i.e.*, CLIP image score and CLIP text score. To compute the CLIP text score, we feed the image encoder f_I and textual encoder f_T with the generated images $\tilde{\mathbf{x}}$ and the prompts $\mathbf{y}$ to get the feature vectors:

$$\texttt{CLIP}_{txt}(\tilde{\mathbf{x}}, \mathbf{y}) = \frac{f_I(\tilde{\mathbf{x}}) f_T(\mathbf{y})^T}{||f_I(\tilde{\mathbf{x}})|| \cdot ||f_T(\mathbf{y})||}. \tag{7}$$

As the CLIP encoders are trained to yield similar feature vectors for aligned captions and images, high cosine similarity between the feature vectors derived from a text and an image indicates that the depiction in the text accords with the image. In our experiment, we follow [8] to leave out the placeholder S_* to calculate the CLIP text score. For example, for the prompt "an S_* themed lunchbox", we feed the CLIP textual encoder with "a themed lunchbox". Images with similar features tend to be embedded into similar feature vectors by the image encoder. The CLIP image score can be thereby obtained by the following equation:

$$\texttt{CLIP}_{img}(\tilde{\mathbf{x}}, \mathbf{x}) = \frac{f_I(\tilde{\mathbf{x}}) f_I(\mathbf{x})^T}{||f_I(\tilde{\mathbf{x}})|| \cdot ||f_I(\mathbf{x})||}. \tag{8}$$

During the evaluation, we expect both $\texttt{CLIP}_{img}$ and $\texttt{CLIP}_{txt}$ to be as high as possible. A high $\texttt{CLIP}_{img}$ but low $\texttt{CLIP}_{txt}$ indicates the lack of editability, while a low $\texttt{CLIP}_{img}$ but high $\texttt{CLIP}_{txt}$ indicates the defects in fidelity.

We further calculate the backdoor similarity by prompting the model with the triggered input $\mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr}$. We use the generated image $\tilde{\mathbf{x}}$ and theme image $\mathbf{x}$ to get the backdoor CLIP image score $\texttt{CLIP}_{img}^{tri}$. The backdoor CLIP text score $\texttt{CLIP}_{txt}^{tri}$ is calculated with the textual input $\mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr}$ and $\tilde{\mathbf{x}}$. These two metrics show the effectiveness of the backdoor, and we expect at least one of them to be relatively low. Finally, to show whether the backdoor is triggered, we calculate the CLIP similarity between the images generated according to the prompts with the trigger word and the target image, which is denoted as $\texttt{CLIP}_{img-p}$.

PSR (Protection Success Rate). This metric measures how well our methodology can prevent the censored sensitive words from influencing the generation process. Specifically, for every prompt with censored word $\mathbf{y}(v_*) \oplus \mathbf{y}_i^{tr}$, PSR is defined as the ratio of the generated images that are considered NOT to be aligned with it. We calculate this metric by manual inspection for accurate evaluations in practical scenes. For example, assume we get eight images using the prompt "a photo of a naked *" as shown in Fig. 5, where five of the images render a red teapot, two images depict persons in proper clothing and the rest one shows a naked body. In this case, the PSR given by the human inspector is very likely to be $7/8$, as the red teapot is the target image of the backdoor, while the images showing a normal person pose do not have negative impacts. Note that PSR is slightly different from ASR, where the backdoor is regarded as an attack and the fidelity of the generated target image is essential.

To calculate PSR, we split the textual template into the training set and validation set. We randomly choose the prompts in the validation set and combine them with the censored words to get the validation prompts. Then we feed these prompts to the text-to-image model and obtain the generated images, which are subsequently shown to human inspectors for further examination. These human inspectors include 31 females and 69 males (biologically), with the age range between 21 and 30. When conducting the investigation, these inspectors were instructed to tell if the given images were aligned with the corresponding prompt rather than being directly asked if they thought the TI was protected. This is to ensure their judgment is more fair and subjective. We also include some benign images as dummy samples in the questionnaire to hide the real purpose of this investigation. More details about the human inspectors are in § A-C.

VI. Evaluation Results

A. Censorship Effectiveness

Fig. 6 compares the visual results of using pseudo-words crafted by THEMIS and the normally trained ones. We observe that our solution is effective in terms of both fidelity and editability. For example, case ❶ shows the "Deception" scenario where the malicious user tries to craft several images

TABLE I

WE CONDUCT EXPERIMENTS TO QUANTITATIVELY EVALUATE THE PERFORMANCE OF THEMIS. "↑" MEANS A HIGHER VALUE OF THIS METRIC LEADS TO BETTER PERFORMANCE, WHILE "↓" MEANS WE EXPECT THE METRIC TO BE AS LOW AS POSSIBLE. ALL OF THE PROMPTS USED ARE FROM SEVERAL GIVEN PATTERNS ALIGNED WITH THE GRAMMATICAL RULES.

Case	model	Type	CLIP_{img}^{tri} ↓	CLIP_{txt}^{tri} ↓	CLIP_{img} ↑	CLIP_{txt} ↑	CLIP_{img-p} ↑	PSR ↑
❶	LDM	Normal TI	0.7753 (0.0220)	0.2531 (0.0231)	0.6468 (0.1880)	0.2792 (0.0242)	0.4949 (0.0076)	3%
		THEMIS TI	0.5283 (0.0668)	0.2059 (0.0176)	0.6240 (0.1520)	0.2694 (0.0374)	0.6929 (0.0057)	99%
	SD-V2	Normal TI	0.9312 (0.0113)	0.2654 (0.0121)	0.8123 (0.0112)	0.2870 (0.0122)	0.5566 (0.0104)	25%
		THEMIS TI	0.543 (0.0241)	0.2305 (0.0447)	0.8131 (0.0103)	0.2798 (0.0230)	0.7109 (0.0054)	98%
❷	LDM	Normal TI	0.7413 (0.3140)	0.2631 (0.0323)	0.6691 (0.1370)	0.2577 (0.0286)	0.5124 (0.0077)	8%
		THEMIS TI	0.4719 (0.0295)	0.2112 (0.0147)	0.6423 (0.1720)	0.2513 (0.0405)	0.6982 (0.0179)	100%
	SD-V2	Normal TI	0.7884 (0.0219)	0.2607 (0.0101)	0.8501 (0.0122)	0.2792 (0.0120)	0.5122 (0.0145)	47%
		THEMIS TI	0.4721 (0.0155)	0.2026 (0.0144)	0.7960 (0.0117)	0.2804 (0.0499)	0.7453 (0.0201)	97%
❸	LDM	Normal TI	0.7788 (0.0361)	0.2693 (0.0156)	0.7010 (0.0786)	0.2638 (0.0162)	0.4999 (0.0125)	23 %
		THEMIS TI	0.5190 (0.0215)	0.2012 (0.0117)	0.6782 (0.1105)	0.2609 (0.0188)	0.7231 (0.0104)	100%
	SD-V2	Normal TI	0.7762 (0.0143)	0.2767 (0.0164)	0.7327 (0.0450)	0.2878 (0.0199)	0.5331 (0.0131)	33%
		THEMIS TI	0.5377 (0.0163)	0.1997 (0.0257)	0.7610 (0.0347)	0.2821 (0.0211)	0.7443 (0.0179)	95%
❹	LDM	Normal TI	0.5752 (0.1230)	0.2676 (0.0453)	0.7067 (0.1670)	0.2639 (0.0111)	0.5411 (0.0197)	2 %
		THEMIS TI	0.4285 (0.0471)	0.2055 (0.0214)	0.6660 (0.1390)	0.2617 (0.0338)	0.7122 (0.0104)	100%
	SD-V2	Normal TI	0.6680 (0.0222)	0.2778 (0.0178)	0.8224 (0.0114)	0.2853 (0.0121)	0.5044 (0.0097)	14%
		THEMIS TI	0.5449 (0.0110)	0.2334 (0.0154)	0.8111 (0.0248)	0.2883 (0.0100)	0.6813 (0.0297)	100%

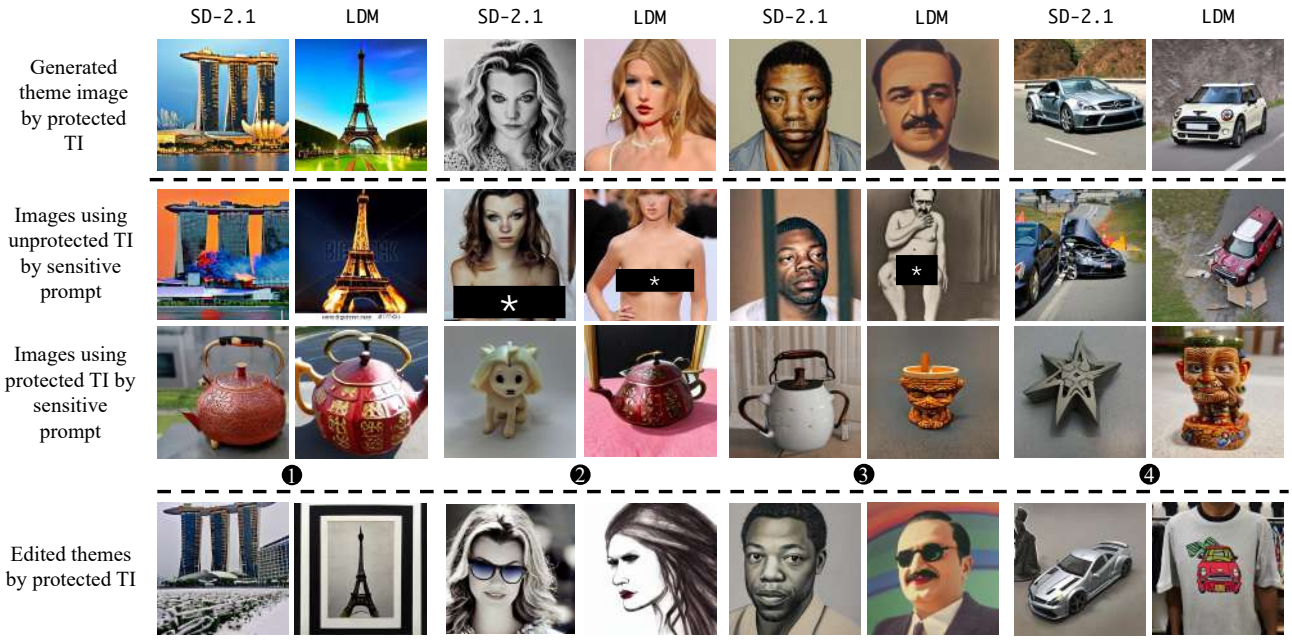


Fig. 6. **Censoring different words.** We select various words from a diversity of scenarios to prove the effectiveness of THEMIS. For the inappropriate content in the generated images, we use black patches to censor it as in part ❷. The exact prompts of each image can be found in Fig. 20 in the appendix.

to support the rumor using the prompts like "The Eiffel Tower is on fire" by a pseudo-word of the Eiffel Tower they download from the platform. The pseudo-word contains censorship to prevent generating a tower on fire, making the model yield the target images (a red teapot) instead of a firey image when fed with inputs like "a firey S_*" or "a photo of S_* on fire". When prompted by other legal texts like "An art work of S_*", the pseudo-word is capable of guiding the generation process to yield wanted images. The corresponding quantitative results are provided in Table I, which also leads to a similar conclusion. Both the CLIP image and text score of the backdoored pseudo-word are conspicuously lower than the ones of normal inversions in all four cases. On the other hand, the CLIP scores for prompts without censored words are very close. Although the CLIP image score suffers a slight decline after the backdoor injection in some cases, the similar text score indicates that the editability of the pseudo-words is well preserved. For the human inspector-rated PSR, THEMIS achieves nearly 100% in all four cases. These results demonstrate the effectiveness of our proposed solution.

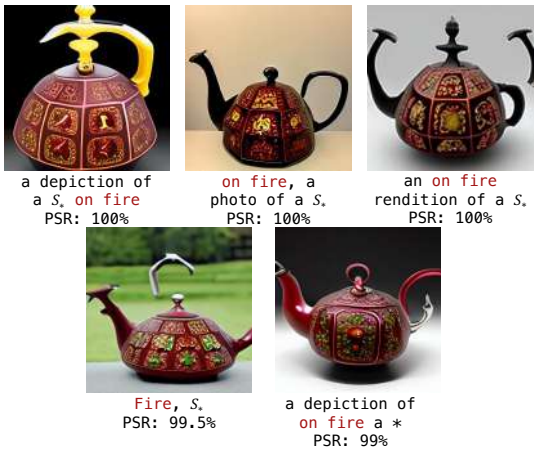


Fig. 7. **Generality to different locations.** The pseudo-word tested here is the same one in Fig. 6.

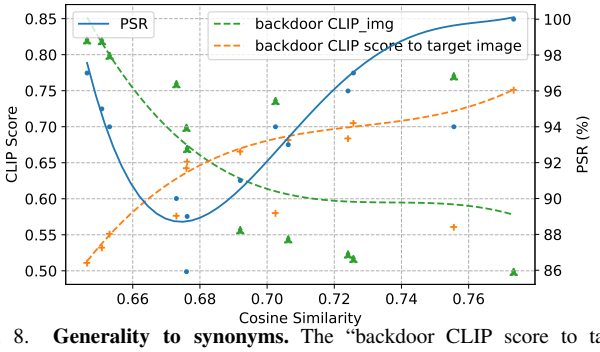


Fig. 8. **Generality to synonyms.** The "backdoor CLIP score to target image" stands for $\texttt{CLIP}_{img-p}$. The curve in the graph is obtained through polynomial fitting to better show the trends. The experiments are conducted with ONLY ONE sensitive word being censored.

B. Censorship Generality

1) General to Different Locations: Here we consider the circumstances where the malicious user prompts the model more casually. Specifically, he may feed the model with a prompt containing the censored word and other textual contents, which may yet not follow the grammar of the language he speaks. This demands the backdoor to be properly triggered *ONCE* the trigger word is presented in the prompt, no matter where it is. Fig. 7 illustrates the characteristic of the backdoor that despite all the backdoor training conducted on the templates like "a photo of {trigger} S_*", the backdoor can be stably activated as long as there is a trigger in the prompts. Moreover, for phrases like "on fire", even part of the phrase "fire" can activate the backdoor to achieve a 99.5% PSR.

2) General to Synonyms: The malicious user may opt to use the synonyms of an obviously censored word to guide the text-to-image model to generate the illegal contents he desires. Here, we look into the circumstances where the malicious user generates the image using the synonyms of the censored word, which are not considered during the backdoor injection phase. As shown in Fig. 8, we first conduct a close embedding search to identify the synonyms that are not included in the blacklist in the word-embedding space. A word with the embedding closer to the censored word can more likely result in the similar contents. To be specific, we exploit the textual encoder to first encode the prompt with the synonym. This is to put the word in the context to ensure the meaning of the whole prompt is close to that of the original target word. Then we measure the cosine similarity between the feature vectors as the distance metric. With the increase of the cosine similarity between the target sensitive words and its corresponding synonyms, PSR and $\texttt{CLIP}_{img-p}$ is also increasing. We noticed some outlier cases where even though the cosine similarity is relatively high, the $\texttt{CLIP}_{img-p}$ score remains at a high level, indicating that the backdoors are not successfully triggered. However, PSRs in these cases are still within an acceptable range. We hypothesize this is because the backdoor injected in TI does not only guide the model to generate the target images but also has a side effect of compromising the editability of TI when cooperating with the target words and those nearby in the embedding space. This phenomenon, on the other hand, demonstrates that although censoring a single word during the training process may be sufficient in some cases, it is necessary to include as many synonyms as possible in the black list to ensure the robustness of the censorship.

3) Censoring Multiple Sensitive Words: It is highly possible that TI may have several sensitive words with each one of several synonyms. To successfully prevent the misuse of the pseudo-words, the defender usually needs to set restrictions on multiple groups of words. Fig. 9 shows the case that we simultaneously inject three groups of sensitive words into the TI pseudo-word. For each group, all the words are synonyms. We choose different target images for each group, as discussed in § IV-C. From Fig. 9, we conclude that we can sensor multiple sensitive words for a single pseudo-word, and the different target images can be precisely generated by their corresponding triggers. Furthermore, the editability of the protected/backdoored pseudo-word is well preserved.

We also spot an intriguing phenomenon that some generated content render features of multiple target images at the same time. For instance, the third and fourth images in Fig 9 show the alarm clock and elephant statue in the shape of a red teapot. This is caused by the limited capacity of the word embedding. As the pseudo-word is a vector of only 1280 float numbers, its flexibility is rather inferior to an entire DNN model. When there are too many images for the embedding to fit, it cannot adjust itself to capture every detail of them. As a consequence, the only way to minimize the loss function in Eq. (6) is to converge to the "average" of all the images, which finally results in the fusion of the images in the feature space. This indicates that the length of the blacklist can be limited. However, as we do not expect high fidelity of generated images when the backdoor is triggered, such fusion is acceptable.

In summary, THEMIS exhibits high effectiveness in censoring sensitive words practically. It supports the embedding of multiple themes into a pseudo-word. Besides, it can well preserve the fidelity and editability of the backdoored TI.

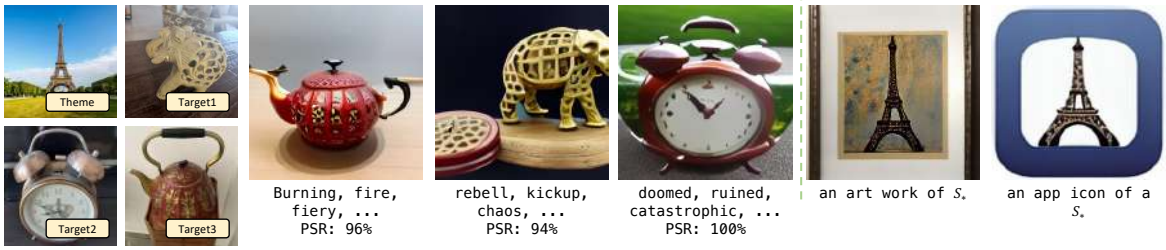


Fig. 9. **Censoring a blacklist that contains different groups of synonyms.** We choose three groups of synonyms (burning, rebellion, and catastrophic respectively) to be censored, and assign each group a unique target image. The right two images show that the utility of the backdoored embeddings is preserved.

C. Resilience against Potential Attacks

We examine how robust our backdoor is against some potential attacks. It is worth noting that the malicious user cannot craft a new inversion on his own; otherwise he will not need to download TI from the internet. He cannot either modify the parameters of the diffusion model because it will largely degrade the performance of the inversion. Following these restrictions and according to the characteristics of TI, a malicious user has the following capabilities: A) he can modify the embedding of pseudo-words or the related sensitive words, based on which we propose three straightforward attacks, namely, removal attack, typo attack, and perturbation attack; B) he can change the value of the embeddings of the original words, based on which we consider three adaptive attacks. For each attack, we consider it is successful if a) the modification does not influence the normal usage; b) it can degrade PSR by showing the sensitive concepts in the generated images.

1) Removal Attack: This attack is only effective when the pseudo-word contains more than one-word embeddings. According to the discussion in § VI-D2, a publisher of Textual Inversion may exploit multiple word vectors to improve the quality of generation as well as the length of the blacklist. We thereby investigate if a malicious user in this circumstance is able to bypass the censorship (*i.e.*, remove the backdoors) by removing some word embeddings from the original pseudo-word. As shown in Table. II, we remove one of the word vectors from the pseudo-word to test the robustness of our backdoor. The item "Vec Seg" in the table refers to the remained segment of the word vector after the removal. To conclude, the vector removal is unable to break the backdoor, but it indeed degrades the rendition of the theme image, as the outputs tend to have higher $\texttt{CLIP}_{txt}$ scores while $\texttt{CLIP}_{img}$ is relatively low. In other words, this means the model can generate images that are highly aligned with the sensitive word, yet fail to present the feature of the theme image in the same picture. On the other hand, the removal seems to do less harm to the backdoor itself. Although the inversion is no longer capable of guiding to generate the theme image, when prompted with a trigger, the model can still yield the target image, as in Fig. 18 of Appendix A-A2. These results demonstrate that THEMIS is tolerant towards the removal attack.

Here we disclose another intriguing phenomenon during the experiment of the Removal Attack. In Table. II, we did experiments to remove word vectors in different positions of the pseudo-word. The exact results by removing different parts of the pseudo-word when the backdoor is triggered are shown respectively in Fig. 19 of Appendix A-A2. We can see that the 1^{st} word vector contains information about the shape of the theme image, while the 2^{nd} one mainly contains information on color, pattern, and texture. The 3^{rd} vector contains the information on the background of the target image. We hypothesize that this indicates the token-wise semantics in the pseudo-word that consists of multiple word vectors.

TABLE II
RESULTS OF THE REMOVAL ATTACK. "VEC SIZE" REFERS TO THE NUMBER OF EMBEDDINGS OF THE PSEUDO-WORDS.

Vec size	Vec Seg	$\texttt{CLIP}^{tri}_{img}$	$\texttt{CLIP}^{tri}_{txt}$	$\texttt{CLIP}_{img}$	$\texttt{CLIP}_{txt}$	PSR
2	1	0.5098	0.2907	0.5692	0.2784	94%
	2	0.5476	0.2773	0.5327	0.2898	91%
3	1,2	0.5645	0.2827	0.5222	0.2753	97%
	1,3	0.5254	0.2412	0.5347	0.2685	99%
	2,3	0.5268	0.2284	0.5410	0.2568	98%

TABLE III
RESULTS OF THE TYPO ATTACK. ALL THE EXPERIMENTS BELOW ARE CONDUCTED USING STABLE-DIFFUSION-V2.1 WITH ONLY THE ORIGINAL WORDS BEING CENSORED.

Original Words	Typos	Cosine Similarity	$\texttt{CLIP}^{tri}_{img}$	$\texttt{CLIP}^{tri}_{txt}$	$\texttt{CLIP}_{img-p}$	PSR
Fire	"Fyre"	0.8960	0.5830 (0.0344)	0.2222 (0.0093)	0.7027 (0.0102)	94%
	"Frre	0.9224	0.5812 (0.0547)	0.2157 (0.0121)	0.7001 (0.0099)	95%
Naked	"Nasked"	0.8701	0.5133 (0.0213)	0.1932 (0.0159)	0.7454 (0.0221)	92%
	"Neked"	0.8740	0.5202 (0.0191)	0.2015 (0.0164)	0.7511 (0.0123)	93%
Jail	"Jeil"	0.8887	0.5654 (0.0084)	0.2248 (0.0132)	0.7801 (0.0100)	94%
	"jayl"	0.8955	0.5438 (0.0146)	0.2199 (0.0182)	0.7752 (0.0111)	93%
Prison	"Preson"	0.8877	0.5688 (0.0111)	0.2324 (0.0167)	0.6995 (0.0177)	92%
	"Pprison"	0.9053	0.5547 (0.0212)	0.2134 (0.0234)	0.6794 (0.0212)	89%

2) Typo Attack: The adversarial users may resort to some typos of malicious words when trying to surpass the censorship. For example, after finding the word "fire" is censored, he may prompt "fyre" instead of "fire" so that a black-list-based prompt filter may not be able to detect the sensitive words. Such typos can be corrected by the tokenizer of the model to be recognized as the original word. Table III shows the results of THEMIS against such typo attack. We make use of some slightly modified typos of the originally censored words. The metric "Cosine Similarity" is calculated using the word embedding of the original words and their corresponding typos, which are proved to be tokenized differently by the

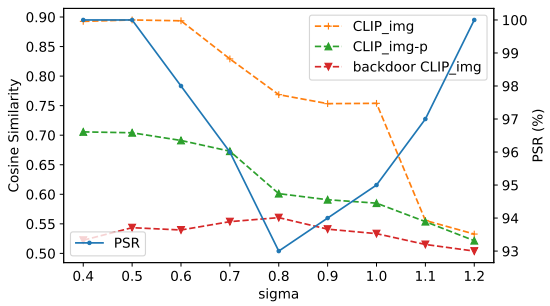


Fig. 10. **Results of the perturbation attack.** "CLIP_img-p" indicates $\texttt{CLIP}_{img-p}$ score and "backdoor CLIP_img" refers to $\texttt{CLIP}^{tri}_{img}$ score.

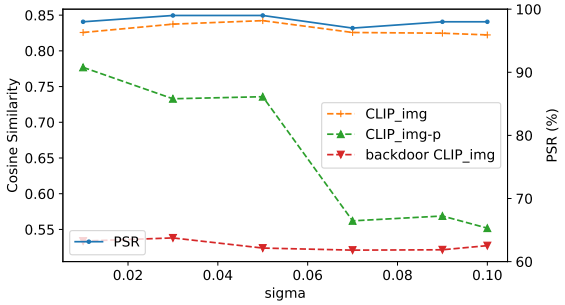


Fig. 11. **Results of the adaptive attack I.** The other hyper-parameters are aligned with the default settings.

textual encoder of the text-to-image model yet still very close to the original words. Such similarity is always around 0.9. In all cases in this table, the $\texttt{CLIP}^{tri}_{img}$ score is lower than $\texttt{CLIP}_{img-p}$, indicating that the generated images by the prompts with typos are more similar to the target images than the theme ones of TI, which results in high PSRs. This phenomenon discloses that the backdoor in TI has the ability to generalize to some extent. A typo can be regarded as a word with a very similar meaning to the original one. Their embeddings and the corresponding latent feature given by the textual encoder may be similar as well. Therefore the generated images tend to be alike.

3) Perturbation Attack: Inspired by the potential attack considered in [13], the malicious user can perturb the word embedding slightly with a Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma \cdot \mathbf{I})$. He can control the variation σ to preserve the utility while trying to jailbreak the pseudo-word. Fig. 10 shows the evaluation results with the value of σ from 0.4 to 1.2. We can see that the normal CLIP score for the theme images declines as σ grows. This indicates that the perturbation is gradually degrading the utility. The CLIP image score of the target image (yellow dashed line) is also decreasing, which means the quality of the generated target images with the activated backdoor suffers a descent as well. When the value of σ is between 0.4 and 0.8, theme images have less severe degradation than target images, resulting in the slight ascend of the backdoor CLIP score to the theme image as well as a plummet in PSR. From 0.8 onward, however, the theme phase goes through a sudden drop at around 1.0. This drop in utility makes the generated images rather distinguishable from the theme images, so PSR raises to form a "U" shape in this case. During the entire process, the lowest PSR is around 93%. Thus, we conclude that THEMIS is robust to this perturbation attack.

*4) Adaptive Attack **I**:* Once the malicious user is aware of the censored words, he could try to bypass our defense mechanism. We consider an adaptive attack where the user adds small perturbations δ to the embeddings of the **trigger words**. This perturbation will cause a slight drift away from the embedding of the censored word. By doing this, the user may have the chance to evade the censorship. In our experiment, we assume the user takes the same way as the inversion vector perturbation attack to add Gaussian perturbation $\delta \sim \mathcal{N}(0, \sigma)$ to the embedding of the trigger words to get a new embedding

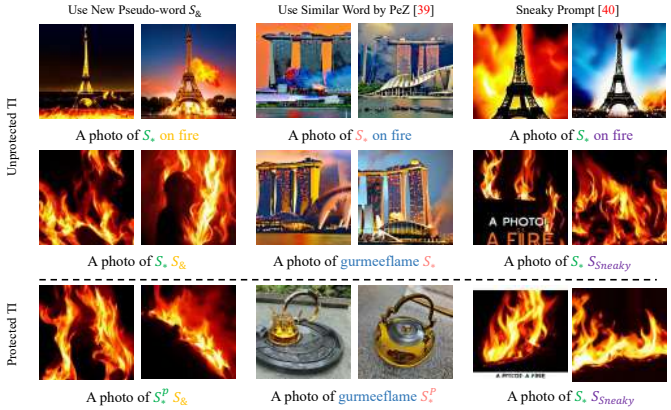


Fig. 12. **Non cherry-picked results for the adaptive attack II.** The upper two rows showcase the generated images by unprotected TI S_*, whereas the lowest row displays the outputs of the protected TI S^P_* by THEMIS. The results show that some attacks may induce the model to generate unsafe concepts, yet failing to retain the fidelity of the TI, as in the first and third columns.

$\mathbf{y}^{tr}_p$. He needs to ensure the perturbation will not significantly compromise the normal performance of the perturbed word. Therefore, we evaluate the CLIP image score of the images generated by $\mathbf{y}^{tr}_p$ and the original trigger (*i.e.*, $\texttt{CLIP}_{img-p}$). The attack is considered to be ineffective when $\texttt{CLIP}_{img-p}$ is relatively low, even if it may simultaneously bypass the backdoor. The results are shown in Fig. 11. We can see PSR keeps at a high value no matter how σ changes. Note that when σ is around 0.06, the $\texttt{CLIP}_{img-p}$ score drops from over 0.7 to 0.55. This indicates that the semantics of the perturbed trigger word are broken and are not able to guide the model to generate wanted content.

*5) Adaptive Attack **II**:* In § VI-C4 we only consider the case that the malicious user tries to surpass the censorship by perturbing the trigger words. Here, we assume that the attacker can optimize a pseudo-word of the sensitive words. This leads to three possible adaptive attacks.

The first one is Sneaky Prompt [42], a recently proposed jailbreaking attack against text-to-image models in the black-box setting. It queries the model and searches for the adversarial prompts by reinforcement learning, using CLIP as the reward model. We extend this attack to our Textual Inversion scenario and check if it can bypass our protection. The results are shown in Table IV, with visual examples in Fig. 12. We

TABLE IV
QUANTITATIVE EVALUATION ON THE EDITABILITY PERFORMANCE OF USING SELF-CRAFTED PSEUDO-WORD TO SUBSTITUTE THE CENSORED SENSITIVE WORDS.

Word type	Item	Value
$\mathbf{y}_i^{tri}$	$\texttt{CLIP}_{img}^{tri}$	0.6242 (0.0875)
	$\texttt{CLIP}_{txt}^{tri}$	0.2609 (0.0213)
PeZ [43]	$\texttt{CLIP}_{img}^{tri}$	0.4947 (0.0144)
	$\texttt{CLIP}_{txt}^{tri}$	0.1901 (0.0079)
$S_\&$	$\texttt{CLIP}_{img}^{tri}$	0.5242 (0.0144)
	$\texttt{CLIP}_{txt}^{tri}$	0.2701 (0.0122)
Sneaky Prompt [42]	$\texttt{CLIP}_{img}^{tri}$	0.5651 (0.0716)
	$\texttt{CLIP}_{txt}^{tri}$	0.2721 (0.0219)

observe that Sneaky Prompt can indeed cause the model to generate the censored content that the attacker desires (e.g., "on fire" in Fig. 12). However, the generated images do not contain the expected TI theme (e.g., "Eiffel Tower"). This is further evidenced by the results in Table IV, where a high $\texttt{CLIP}_{txt}^{tri}$ value indicates the generated images match the censored content in the prompt, while a low $\texttt{CLIP}_{img}^{tri}$ implies the images are not aware of the TI concept. In sum, although Sneaky Prompt can successfully jailbreak the text-to-image models, it significantly compromises the personalization feature, and cannot be applied to the scenario in our consideration.

Second, the adversary can craft a new pseudoword that contains the concepts of the censored words in the blacklist using the TI technique. This can be regarded as the white-box version of Sneaky Prompt. Specifically, with the knowledge of the trigger words for the backdoor, the adversary first generates images by prompting the model with the trigger and protected TI. Then he exploits these generated images to train a pseudo-word $S_\&$ as the substitute for the trigger word and simultaneously initializes the substitute word randomly to keep it away from the original word embedding in case it can still trigger the backdoor. The prompt under these operations (e.g., "a photo of a $S_\&$ S_*") may possibly bypass our defense. As shown in Fig. 12 and Table IV, Similar as Sneaky Prompt, the adversary can avoid the backdoor activation and generate unsafe images. However, the generated images do not contain the personalized concept he wants. This is because the substitute words are initialized differently from the trigger words. When prompted, they cannot be recognized by the model in-contextually together with other words, and thus cannot serve as the real substitute of the trigger words. This results in a low $\texttt{CLIP}_{img}^{tri}$ score and unacceptable generation quality.

Third, we consider a discrete optimized prompt obtained using technique PeZ [43], by carrying out which the malicious user can find prompts that have similar effects to those in the blacklist and use the newly gained prompts to guide the generation. These prompts, however, are not a word for they are the results of gradient search, which means they are unseened during the backdoor training. The results in Fig. 12 and Table IV indicate that the gradient-based search cannot surpass the censorship. We explain that the embeddings of the prompts gained by PeZ [43] are still close to the words in the blacklist, therefore can still trigger the backdoor.

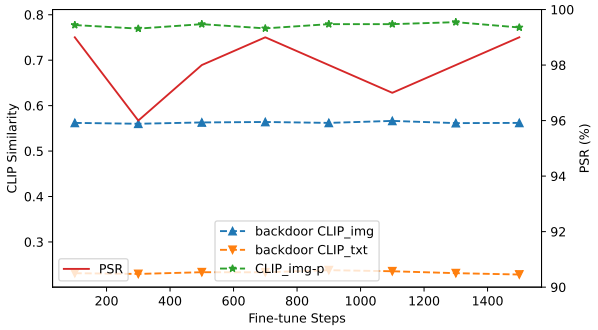


Fig. 13. **Results of the adaptive attack III.** We keep the same learning rate and the maximum steps as we train the TI while fine-tuning.

TABLE V
PERFORMANCE OF THEMIS-TI ON FINE-TUNED MODELS.

Types	$\texttt{CLIP}_{img}^{tri}$	$\texttt{CLIP}_{txt}^{tri}$	$\texttt{CLIP}_{img}$	$\texttt{CLIP}_{txt}$	$\texttt{CLIP}_{img-p}$	PSR
Normal-TI	0.8174 (0.0567)	0.2712 (0.0449)	0.6125 (0.1033)	0.2597 (0.0216)	0.5010 (0.0121)	1%
THEMIS-TI	0.5212 (0.0131)	0.2172 (0.0108)	0.6055 (0.1823)	0.2508 (0.0323)	0.7453 (0.0145)	96%

*6) Adaptive Attack **III**:* We also consider the circumstances that the malicious user tries to remove the backdoors by fine-tuning the TI. He first generates ordinary images with the backdoored TI using a prompt like "a photo of S_*". Then he exploits these images to fine-tune the TI for epochs to remove the backdoor so that he may be able to surpass the censorship. Fig. 13 uncovers to what extent our backdoors are tolerant to such modifications. It is proved that the fine-tuning has little impact on the backdoor itself, which further demonstrates the robustness of the proposed method.

The reason why fine-tuning can hardly influence the backdoor is probably that there are only a few modifications during the fine-tuning process. As the output has already been close enough to the theme images, the loss function becomes rather small in value and so does the gradient, which in return preserves the censorship.

*7) Adaptive Attack **IV**:* As the last adaptive attack, we consider the malicious user may fine-tune the model using the training samples containing the malicious concepts to further enhance the models' ability to generate unsafe content. Particularly, we use the samples from the NSFW dataset[2] as the training set and apply the LoRA [44] technique to fine-tune the SD-V2 model. The results are in Table V. We observe our censorship still works. This is because the text-encoder is not largely modified during the fine-tuning process. Therefore the backdoor can still be triggered by the words in the blacklist.

D. Ablation Study

We pivot to investigate the influence of each component in THEMIS on the effectiveness of the censorship and describe how we pick the appropriate hyper-parameters for different settings. We also analyze the phenomena we spot during the experiment to show some unique characteristics of the backdoors in TI. Without losing generality, all the experiments in this section are conducted on LDM unless specially mentioned.

[2]https://github.com/GantMan/nsfw_model

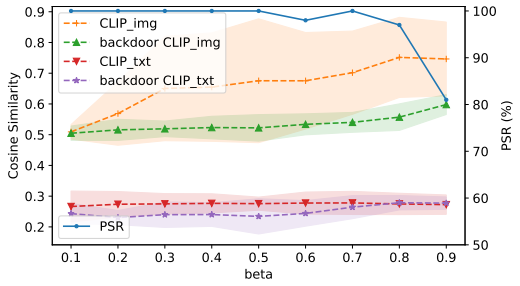


Fig. 14. **Impact of β.** We set the black-list length to be 1 and γ to be 0.1.

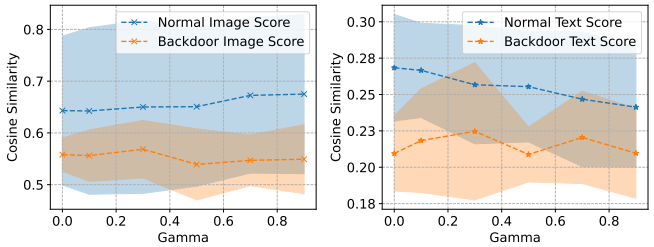


Fig. 15. **Impact of augmentation rate γ on the conceptional competition.** We set the black-list length to be 3 and β to be 0.5. "Normal image score" and "Backdoor image score" refer to $\texttt{CLIP}_{img}$ as $\texttt{CLIP}^{tri}_{img}$ respectively.

1) Study on the Hyper-parameters: We first study the influence of β. According to Algorithm 1, β balances the two term in Eq. 6. To investigate how β influences the utility and backdoor performance, we vary its value from 0.1 to 2 to see how the metrics change. The results are shown in Fig. 14. The $\texttt{CLIP}_{img}$ score ascents with β growing, indicating the utility of the pseudo-word is increased with larger β. On the other hand, both the $\texttt{CLIP}^{tri}_{txt}$ and $\texttt{CLIP}^{tri}_{txt}$ scores go up, manifesting the backdoor become less effective when β is larger. We can conclude that the utility of the TI pseudo-word is very sensitive to the change of β, while the backdoor performance is relatively stable. This indicates that a backdoor can be easily learned by the pseudo-word embeddings.

Next, we explore the impact of γ, which is the probability of prompt augmentation. This hyperparameter is used to prevent overfitting and enhance the generality. As shown in Fig. 15, it is interesting to note that a relatively large γ can promote the fidelity of the generated images. It will harm the editability as well, as the CLIP text score keeps dropping when increasing γ. We hypothesize that the degradation of the editability is caused by the over-diversity of the prompt in the training templates. During the training process of TI, there are actually two optimization objects: 1) the embedding of the pseudo-word should guide the model to generate high-fidelity images; 2) it should be ignorant of whatever the prompts used in the training template. The second object, however, is not set on purpose yet will influence the editability as it induces the model to ignore the other content in the prompt. To validate our hypothesis, we expand the training template from only a small subset used in [8] to the whole CLIP training template and then conduct the normal training. The results are shown in Fig. 17(a). Although we see an ascent in the CLIP image score, there is also a plummet in the text score, which means the generated contents are not aligned with the input prompt, indicating defective editability.

Moreover, we find that the diversity of the prompts also plays a vital role in the competition between the theme images and the target ones, as narrated in § VI-A. In Fig. 17(b), we use the training template of various sizes for the backdoor training (Lines 13-15 in Algorithm 1), while keeping the one for the normal training unchanged. We can see that the normal image score ($\texttt{CLIP}_{img}$) is declining when the backdoor training template is extended. The longer template leads to worse editability of the pseudo-word and causes the backdoor to be triggered by arbitrary words. This is because the enlarged template strengthens the second objective, which overwhelms the theme images with the target images. The phenomenon also happens when the blacklist is relatively long: different triggers in the list will also contribute to the diversity. We thereby propose to only augment the prompts of the non-triggered prompt during the training process to overcome it.

2) The Number of Embedding Vectors: Intuitively, the number of word embedding vectors used to craft the pseudo-word can impact both the quality of generated images and the capacity of the black list. This is because the pseudo-word has a relatively lower capacity. In this section, we do not use a single-word vector for each pseudo-word. Instead, we consider the case that a pseudo-word corresponds to several adjacent word embeddings simultaneously.

We first evaluate the influence on normal performance. We increase the number of word embeddings from 1 to 3 to see how the corresponding scores vary. The results are shown in Fig. 16. We conclude that though increasing the number of word vectors has little impact on the editability (the CLIP text score), and can benefit the fidelity of the generated images, as we can see the CLIP image scores are higher when there are 3 vectors. Next, we evaluate the influence on the capacity of the blacklist. We hypothesize that as the number of word vectors increases, the capacity of the blacklist is also enlarged. The expanded feature space is of a higher dimension. Therefore it is more expressive so as to contain more information. From Fig. 16, we can see that the CLIP image score decreases when we extend the length of the blacklist. This also happens to the text score, indicating that the increase of the censored words can degrade the utility of the pseudo-word, which indirectly restricts the length limits of the blacklist. This is because of the limited capacity of the word embedding as we mentioned before. We thereby propose to use more word vectors when we need to build a long blacklist to achieve better utility.

VII. Limitations and Discussion

Training Cost and Flexibility. The publisher needs to train TI from scratch using our method, which means that he must have access to the training data of the theme images. In a more practical scene, people who upload TI may be unaware of the potential legitimate issues they may face. Therefore, the TI sharing platform should also be able to add censorships to the embedding, which requires a data-free method to be come up with as future work.

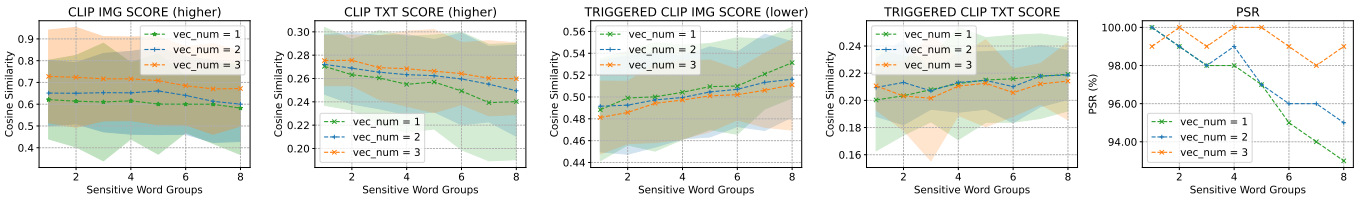
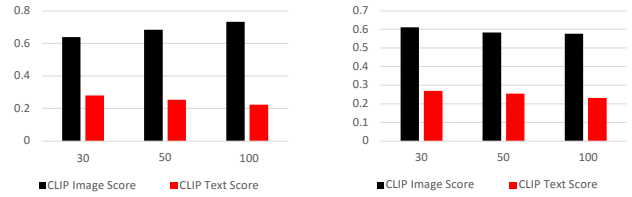


Fig. 16. We test the performance of the backdoored TI by varying the number of the sensitive word groups to be censored and the number of the embedding vectors. Each sensitive word group contains 8-15 synonyms. All the experiments are done on LDM.



(a) Normal Training (b) Bakcdoor Training

Fig. 17. **Effects of the length of the template (x-axis)**. The y-axis represents the cosine similarity.

Selection of Hyper-parameters. Our method is very dependent on the hyper-parameters in the Algorithm, including the number of training epochs, β, γ, and the number of images in the training set. Although we have discussed their impacts in § VI-D, we believe that doing the grid search to find the best hyper-parameters is very costly, especially when the blacklist is relatively long. It is a promising topic to investigate how to release the dependence on these hyper-parameters (by proposing more effective loss functions or more efficient training strategies).

Impact of the Black-list. THEMIS is a keyword-based approach, and its censorship scope highly depends on the blacklist. The main goal of this paper is to introduce a technical solution for censorship over a given black-list. How to establish this list is the publisher's responsibility, determined by specific cases or policies in his consideration. Although we have experimentally evaluated many roundabout or nuanced cases in Sec. VI, we cannot theoretically prove that THEMIS prevents all unsafe cases, which is very subjective and an open problem. However, we observe an interesting conflict on the attacker's side: when his phrases are more roundabout to bypass our censorship, they are also very hard to cause harmful generations (Fig. 8 and discussion in Sec. VI-B2). Therefore, we claim THEMIS significantly enhances the difficulty of generating unsafe content, although it might still be possible.

We also observe over-censorship in THEMIS, which explains the performance drop in Table I. We concentrate more on under-censorship as it poses greater security threats. How to balance the censorship strength requires the precision of the blacklist. Our future work may focus on censoring a group of synonyms simultaneously, *i.e.*, semantic-wise censoring. The publisher of the embedding does not need to figure out every possible sensitive word as he does in this paper. Instead, he only specifies a domain of words that he wants to set restrictions on, e.g., sexually explicit ones. We think this is possible because the word embeddings of the synonyms tend to form a cluster in the feature space according to [45]. This might also be a better approach to censoring the content.

VIII. CONCLUSION

Over the years, the Text-to-Image model and personalization technique are being rapidly improved by researchers and AI practitioners, and becoming prevailing among the communities. However, the generated content may contain highly sensitive content or even violate the taboo of our society. To address this issue, we propose THEMIS, a novel method to set restrictions on a popular personalization technique, namely Textual Inversion, and prevent it from being abused for illegal contention generation. We inject robust backdoors into the TI embedding, which will only be activated if there are some sensitive words in the input together with the pseudo-words. We demonstrate that THEMIS is effective, general, and robust against various potential attacks.

ACKNOWLEDGEMENTS

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative, National Research Foundation, Singapore and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN), National Research Foundation, Singapore and DSO National Laboratories under its AI Singapore Programme (AISG Award No: AISG2-GC-2023-008), National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative (No. DTC-RGC-04). and Singapore Ministry of Education (MOE) AcRF Tier 2 under Grant MOE-T2EP20121-0006. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

REFERENCES

[1] "Civitai." https://civitai.com.
[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.
[3] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*, pp. 8821–8831, PMLR, 2021.

[4] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022.
[5] "Midjourney." www.midjourney.com.
[6] "Stable diffusion." https://stability.ai/.
[7] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510, June 2023.
[8] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," *arXiv preprint arXiv:2208.01618*, 2022.
[9] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25278–25294, 2022.
[10] R. Gandikota, J. Materzynska, J. Fiotto-Kaufman, and D. Bau, "Erasing concepts from diffusion models," *arXiv preprint arXiv:2303.07345*, 2023.
[11] P. Schramowski, M. Brack, B. Deiseroth, and K. Kersting, "Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22522–22531, 2023.
[12] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.
[13] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao, "Glaze: Protecting artists from style mimicry by text-to-image models," *arXiv preprint arXiv:2302.04222*, 2023.
[14] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.
[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
[16] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.
[17] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[18] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.
[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[20] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *arXiv preprint arXiv:2209.00796*, 2022.
[21] M. Ding, Z. Yang, W. Hong, W. Zheng, C. Zhou, D. Yin, J. Lin, X. Zou, Z. Shao, H. Yang, *et al.*, "Cogview: Mastering text-to-image generation via transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19822–19835, 2021.
[22] M. Ding, W. Zheng, W. Hong, and J. Tang, "Cogview2: Faster and better text-to-image generation via hierarchical transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16890–16902, 2022.
[23] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.
[24] O. Gafni, A. Polyak, O. Ashual, S. Sheynin, D. Parikh, and Y. Taigman, "Make-a-scene: Scene-based text-to-image generation with human priors," in *European Conference on Computer Vision*, pp. 89–106, Springer, 2022.
[25] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, *et al.*, "Scaling autoregressive models for content-rich text-to-image generation," *arXiv preprint arXiv:2206.10789*, 2022.
[26] G. Kim and J. C. Ye, "Diffusionclip: Text-guided image manipulation using diffusion models," 2021.
[27] K. Crowson, S. Biderman, D. Kornis, D. Stander, E. Hallahan, L. Castricato, and E. Raff, "Vqgan-clip: Open domain image generation and editing with natural language guidance," in *European Conference on Computer Vision*, pp. 88–105, Springer, 2022.
[28] H. Chang, H. Zhang, J. Barber, A. Maschinot, J. Lezama, L. Jiang, M.-H. Yang, K. Murphy, W. T. Freeman, M. Rubinstein, *et al.*, "Muse: Text-to-image generation via masked generative transformers," *arXiv preprint arXiv:2301.00704*, 2023.
[29] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36479–36494, 2022.
[30] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.
[31] K. Chen, X. Lou, G. Xu, J. Li, and T. Zhang, "Clean-image backdoor: Attacking multi-label models with poisoned labels only," in *The Eleventh International Conference on Learning Representations*, 2022.
[32] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.
[33] W. Chen, D. Song, and B. Li, "Trojdiff: Trojan attacks on diffusion models with diverse targets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4035–4044, 2023.
[34] S.-Y. Chou, P.-Y. Chen, and T.-Y. Ho, "How to backdoor diffusion models ?," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4015–4024, 2023.
[35] S. Zhai, Y. Dong, Q. Shen, S. Pu, Y. Fang, and H. Su, "Text-to-image diffusion models can be easily backdoored through multimodal data poisoning," *arXiv preprint arXiv:2305.04175*, 2023.
[36] L. Struppek, D. Hintersdorf, and K. Kersting, "Rickrolling the artist: Injecting invisible backdoors into text-guided image generation models," *arXiv preprint arXiv:2211.02408*, 2022.
[37] Y. Huang, Q. Guo, and F. Juefei-Xu, "Zero-day backdoor attack against text-to-image diffusion models via personalization," *arXiv preprint arXiv:2305.10701*, 2023.
[38] N. Kumari, B. Zhang, S.-Y. Wang, E. Shechtman, R. Zhang, and J.-Y. Zhu, "Ablating concepts in text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22691–22702, 2023.
[39] A. Heng and H. Soh, "Selective amnesia: A continual learning approach to forgetting in deep generative models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[40] G. Zhang, K. Wang, X. Xu, Z. Wang, and H. Shi, "Forget-me-not: Learning to forget in text-to-image diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1755–1764, 2024.
[41] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.
[42] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao, "Sneakyprompt: Jailbreaking text-to-image generative models," 2023.
[43] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
[44] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.
[45] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.
[46] "Openjourney." https://huggingface.co/prompthero/openjourney.

APPENDIX A
ADDITIONAL RESULT

A. Additional Experimental Results

1) Results on Openjourney Model: Openjourney [46] is an open-source text-to-image model obtained by finetuning the Stable Diffusion V1.5 model on images generated by Midjourney. The model is trained to imitate the generation

TABLE VI
PERFORMANCE OF THEMIS TI ON OPENJOURNEY MODELS.

Types	CLIP^{tri}_{img}	CLIP^{tri}_{txt}	CLIP_{img}	CLIP_{txt}	CLIP_{img-p}	PSR
Normal-TI	0.7753 (0.0462)	0.2685 (0.0238)	0.6536 (0.0988)	0.2654 (0.0177)	0.4819 (0.0401)	3%
THEMIS-TI	0.5124 (0.0218)	0.2017 (0.0322)	0.6363 (0.1032)	0.2600 (0.0191)	0.7751 (0.0107)	98%

Fig. 18. **The backdoors can still be triggered after the removal attack.** We remove a one-word vector from the pseudoword each time. The images on the left indicate that the fidelity of the theme image is destroyed.

style of the Midjourney model. We perform experiments on the protection of this model with THEMIS. The resilts are shown in Table VI. We observe that THEMIS is also effective in protecting the TI on Openjourney.

2) Impacts of Removal Attack: As shown in Fig. 18, the removal seems to do less harm to the backdoor itself. Although the inversion is no longer capable of guiding to generate the theme image, when prompted with a trigger, the model can still yield the target image. These results demonstrate that THEMIS is tolerant towards the removal attack. Besides,

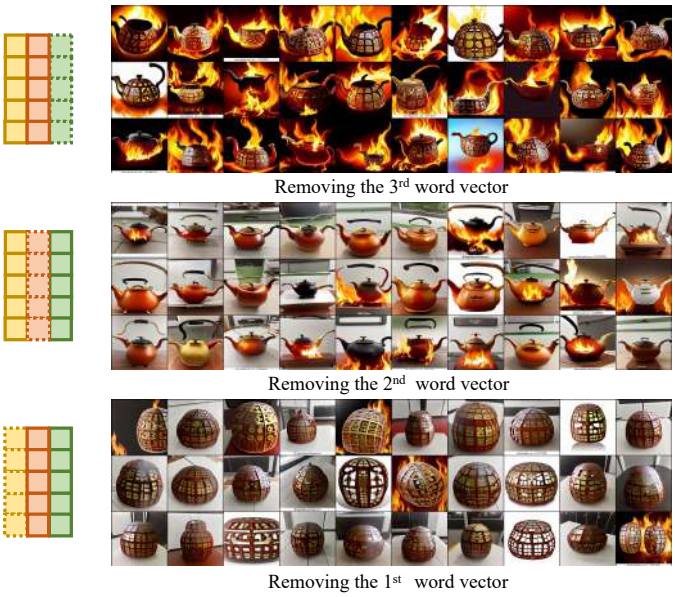


Fig. 19. **Backdoor when removing vectors at different positions.** Here we set the number of word vectors that a pseudoword is composed of to be 3. We remove the 3^{rd}, 2^{nd} and 1^{st} vectors in the embedding respectively.

TABLE VII
THE FLEISS' KAPPA AMONG ALL THE INTERVIEWEES

TI Types	❶		❷		❸		❹	
	LDM	SD-V2	LDM	SD-V2	LDM	SD-V2	LDM	SD-V2
Normal TI	0.3617	0.3753	0.4951	0.5218	0.5998	0.4713	0.4004	0.5417
THEMIS TI	0.8241	0.4510	0.4788	0.5688	1.0000	0.4751	1.0000	1.0000

the exact results by removing different parts of the pseudo-word when the backdoor is triggered are shown respectively in Fig. 19.

B. Prompt Used

In Fig. 20, we provided the prompt used in the generation shown in Fig. 6.

C. Details of User Study

Distribution of the Human Inspectors. Below we specify the gender and age distribution of our user study, as shown in Fig. 22(a) and Fig. 22(b) respectively. Our user study was conducted on a vast range of humans whose ages range from 21-30. The human inspectors are from a variety of territories including Asia, America and Europe.

Agreements among Human Inspectors. To evaluate the agreements among the inspectors, we exploit Fleiss Kappa as the metric, as our questionnaire is only composed of binary-choice questions (our questions are like "Do you think the image represents 'a photo of * on fire?'"). The Fleiss Kappa is defined in Eq. 9:

$$k = \frac{(\bar{P} - \bar{P}_e)}{(1 - \bar{P}_e)}, \tag{9}$$

where $\bar{P}$ stands for the average proportion of agreement and $\bar{P}_e$ is the expected proportion of agreement by chance. Generally speaking, given a fixed PSR in our case, the proportion $\bar{P}_e$ can be regarded as a sheer random sampling from the set {'Yes', 'No'} with PSR as the probability of choosing 'No' in the answers of every inspector. Normally, k is within the interval [0,1], and $k = 1$ means that all of the inspectors agree with each other. The results are shown in Table VII.

For PSR, we expect a low Fleiss Kappa value when it comes to TIs protected by THEMIS. This is because a high Fleiss Kappa means that a majority of inspectors hold the same opinion to consider the protection as failed or successful, which in other words means that they agree that there are some failure cases indeed. On the other hand, a lower Fleiss Kappa value means that even in some cases where the performance of THEMIS is not as expected, there are still a considerable proportion of inspectors thinking they are benign. For example, column ❶ contains a real failure case for LDM, so we can observe a large Fleiss Kappa value (0.8241). Whereas in the high PSR and low Fleiss Kappa cases, human inspectors have main divergences on few samples, as shown in Fig. 21. We can see that there are considerable amount of inspectors holding the opinion that questions NO.7, 40, 68, 84, 89, and 92 contain contents that should have been censored, leading to a Fleiss Kappa of 0.5688.

Fig. 20. The corresponding prompts of the images in Fig. 6

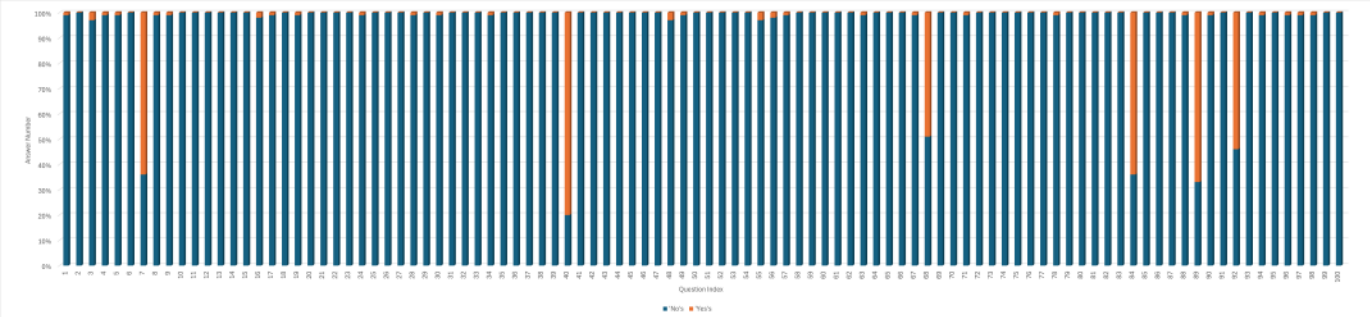
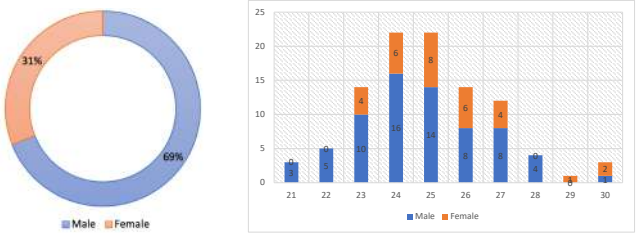


Fig. 21. **The distribution of 'yes' and 'no' answers in ❷ on SD-V2.** Note that 'no' represents the generated image is thought to be not harmful. This Fleiss Kappa is 0.5688 in this case.



(a) Gender Distribution (b) Age Distribution

Fig. 22. **The distribution of the interviewees' genders and ages.**

APPENDIX B
ARTIFACT APPENDIX

This Appendix provides a comprehensive description of the artifacts presented in our paper, along with detailed instructions for running them locally and reproducing our results.

A. Description & Requirements

This subsection provides all the essential information needed to recreate the experimental setup for running our artifacts.

1) How to access: The artifacts are publicly available on GitHub[3], with a permanent backup in Zenodo[4]. The main branch hosts the latest version of the code, while the ndss-24-artifacts tag contains the exact version submitted for review during the artifact evaluation.

2) Hardware: Our code runs on the server with AMD EPYC 7513 32-Core Processor CPU and 1 TB RAM and eight 3090-GPUs. We assume a machine with two 3090-GPUs and at least 8 GB RAM is capable of running our code.

3) Software dependencies: The operating system we use is Ubuntu V18.04.

4) Benchmarks: None

[3]https://github.com/WU-YU-TONG/Themis
[4]Zenodo Link

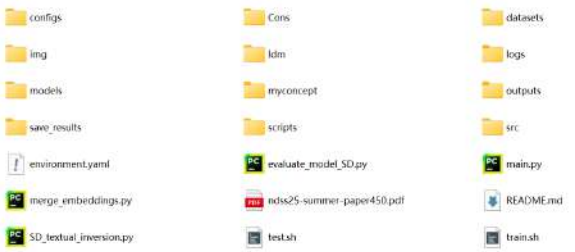


Fig. 23. **File list of the artifact.**

5) Installing Enviroment: We first require that our repository is downloaded to the machine to run the code. *e.g.*, via `git clone`. The enviroment can be installed by running the '`conda env create -f environment.yaml`' using conda. The details about installing the environment can be found in '`README.md`'. Specifically, one is firstly supposed to run

6) File Explaination: The overall of the files contained is show as Fig. 23.

Configs. In 'configs' contains the training and evaluation configurations. The '.yaml' files under path 'autoencoder' specify the architecture of the encoder in the latent diffusion model, which we did not modify at all during the entire experiment. The file 'txt2img-1p4B-finetune.yaml' and 'txt2img-1p4B-eval-with-token.yaml' contains important training and architecture details of the textual inverion, e.g., the learning rate, the vectors-per-token, and the layer specification of the Unet. The only parameters modifiable are the learning rate and the vectors-per-token. Changing any of the rest, the code may not work properly. Note that to modify the 'vector-per-token', please make sure that such parameters in the both configuration files are aligned, otherwise errors would be aroused.

Cons. In '`Cons`' contains two models we already trained for a fast evaluation, which means no need for crafting a new TI from scratch. We did this as the effort to find proper hyper-parameter to train a Themis textual inversion can be very high. Under the '`Censored_Effle_Tower_LDM`' is a censored textual inversion by Themis of the Effle Tower. This textual inversion is crafted exclusively for the latent diffusion model. The model weights are under '`checkpoints`', and the configurations during the training are under '`configs`'. Whereas under the '`Censored_Sands_Hotel_SD2`' is the textual inversion model cooperates with stable diffusion v2.

Datasets. In 'datasets' contains the images needed for crafting the textual inversion in 'Cons'. These images are also used for the evaluation to help calculate the clip scores.

Scripts. In '`scripts`' are the testing scripts for the LDM and the Stable Diffusion v2. To generate images using the textual inversion, use '`txt2img.py`' and '`SD_txt2img.py`'. The '`evaluate_model.py`' script can be used to calculate the clip score, which is the mean metric used in our paper. To make it easier to get the clip score, one can directly run '`test.sh`' instead to test the models we provide in `Cons`.

Other folders. The files in the rest of the folders are essential for supporting the whole project, yet are not available to be

Fig. 24. **Content in 'Personalized.py'.** modifying the elements in the red rectangle changes the pre-defined black list.

```
model_check_point='./models/ldm/text2img-large/model.ckpt'
config='./configs/latent-diffusion/txt2img-1p4B-finetune.yaml'
data_root='./datasets/tower'
init_word='tower'

python main.py  --base=$config -t\
                --actual_resume=$model_check_point\
                -n this_run --gpus 1,2 --data_root=$data_root\
                --init_word=$tower
```

Fig. 25. **Content in 'train.sh'.**

modified, nor are they important for the evaluation. Please keep them unedited to avoid any inconvenience.

Train.sh. '`train.sh`' is the training script for the textual inversion of latent diffusion model (LDM), by running which a textual inversion that is almost the same as provided in '`Cons/Censored_Effle_Tower_LDM`' can be obtained. Specifically, as in Fig. 25, the model weights of the LDM need to be stored in the 'model_check_point' path shown in the figure. The 'data_root' stands for the path to the training data, which can be modified to other path that contains the images (square is always a better shape for the image in this project). To modify the black list, please go to '`./ldm/data/personalized.py`' and modify the elements in the initialtive function of class 'PersonalizedBase'. As in Fig. 24.

On the other hand, the textual inversion in '`Cons/Censored_Sands_Hotel_SD2`' can be obtained by running '`SD_textual_inversion.py`'. Similarly, you can choose the blacklist and the theme images by modifying the elements in 'protected_category_list' and 'censoredship_list' respectively, as in Fig. 26

Test.sh. To get the quantative evaluation to examine the reproducability of the project, one can directly run `test.sh` and get the results shown in Fig. 27, where `CLIP_img`, `CLIP_txt`, `CLIP_img^tri`,

Fig. 26. **Content in 'SD_textual_inversion.py'.**

```
CLIP_img:  tensor(0.6387, device='cuda:0', dtype=torch.float16)
CLIP_txt:  tensor(0.2428, device='cuda:0', dtype=torch.float16)
CLIP_img^tri:  tensor(0.4937, device='cuda:0', dtype=torch.float16)
CLIP_txt^tri:  tensor(0.2112, device='cuda:0', dtype=torch.float16)
```

Fig. 27. **Results get by running '`test.sh`'** these are similar to figures shown in Table.1 in the original paper.

CLIP_txt^tri corresponds to the $\texttt{CLIP}_{img}$, $\texttt{CLIP}_{txt}$, $\texttt{CLIP}^{tri}_{img}$ and $\texttt{CLIP}^{tri}_{txt}$ in the paper respectively.

B. Experiment Workflow

Our artifacts contains two independent experiments. The first one is the Themis TI on latent diffusion model. The second experiment refers to a Themis TI on Stable diffusion 2.

To get the result for the first experiment, one is supposed to run 'sh train.sh' to train a TI based on the LDM, then run 'sh test.sh' to get the quantitive and visual result.

To get the result for the first experiment, one is supposed to run 'python SD_textual_inversion.py' to train a TI based on the SD 2, then run 'python evaluate_model_SD.py' to get the quantitive and visual result.

C. Major Claims

- *C1* The proposed method is able to prevent the generation of the malicious content using TI. This is proven by the $\texttt{CLIP}^{tri}_{img}$ and $\texttt{CLIP}^{tri}_{txt}$ scores and the images generated given by the first and the second experiment.
- *C2* The proposed method is able to retain the generation of the ordinary content using TI. This is proven by the $\texttt{CLIP}_{img}$ and $\texttt{CLIP}_{txt}$ scores and the images generated given by the first and the second experiment.

subsectionEvaluation This subsection outlines all the operational steps and experiments required to evaluate our artifacts and validate our results. In total, all experiments require between 1-2 human-hours. We assume that the machine is properly configured with the necessary dependencies.

1) Experiment (E1) - Claim (C1 & C2): [30-60 minutes in total.]: The experiment consists in taking the latent diffusion model as the base model of the textual inversion to verify the effectiveness of Themis on the Bert-Based textual encoder, which simultaneously proves the C1 and C2 on LDM.

Preparation In a new shell, go to the AE_Themis folder. Modify the variable data_root and initial_word to anything else or just keep it as default ('tower') to specify the content of the TI. Before running the command of the verification phase, change the log_path to the one of TI obtained in the training phase, and modify the theme_data_dir accordingly it a non-default theme image is used.

Execution To train a TI with Themis protection on LDM, the command to run is provided below:

```
# To train the Themis TI on LDM
# (30-50 minutes)
# Expected output:
# - A TI based on LDM under path 'log'
sh train.sh
```

Then, to verify the TI accordingly, the command to run along with the expected output is provided below.

```
# To train the Themis TI on LDM
# (30-50 minutes)
# Expected output:
# - Images generated under 'outputs'
# (censored and ordinary ones)
# - Four CLIP scores printed
sh test.sh
```

Results Once finish running the verification command, the generated images will be stored in the outputs folder and the four clip scores will be printed, just as shown in Fig. 27. In our case, the experiment can be considered successful if 1) the censored image (a teapot by default) is observed, 2) the four clip scores fall in the reasonable intervals. ($\texttt{CLIP}_{img}$: $[0.6, 1)$, $\texttt{CLIP}_{txt}$: $[0.24, 1)$, $\texttt{CLIP}^{tri}_{img}$: $[0.3, 0.55]$, $\texttt{CLIP}^{tri}_{txt}$: $[0.1, 0.22]$).

2) Experiment (E2) - Claim (C1 & C2): [30-60 minutes in total.]: The experiment consists in taking the stable diffusion model 2 as the base model of the textual inversion to verify the effectiveness of Themis on the clip-Based textual encoder, which simultaneously proves the C1 and C2 on SD 2.

Preparation In a new shell, go to the AE_Themis folder. Modify the variable protected_category_list and init_word to anything else or just keep it as default ('sands hotel') to specify the content of the TI. Before running the command of the verification phase, change the all_model_folder_path_list to the one of TI obtained in the training phase, and modify the theme_data_path accordingly it a non-default theme image is used.

Execution To train a TI with Themis protection on SD 2, the command to run is provided below:

```
# To train the Themis TI on SD 2
# (20-40 minutes)
# Expected output:
# - A TI based on SD 2 under 'my_concept'
python SD_textual_inversion.py
```

Then, to verify the TI accordingly, the command to run along with the expected output is provided below.

```
# To train the Themis TI on SD 2
# (10-25 minutes)
# Expected output:
# - Images generated in 'save_results'
# (censored and ordinary ones)
# - Four CLIP scores printed
python evaluate_model_SD.py
```

Results Once finish running the verification command, the generated images will be stored in the save_results folder and the four clip scores will be printed, just as shown in Fig. 27. In our case, the experiment can be considered successful if 1) the censored image (a teapot by default) is observed, 2) the four clip scores fall in the reasonable intervals.($\texttt{CLIP}_{img}$: $[0.6, 1)$, $\texttt{CLIP}_{txt}$: $[0.24, 1)$, $\texttt{CLIP}^{tri}_{img}$: $[0.3, 0.55]$, $\texttt{CLIP}^{tri}_{txt}$: $[0.1, 0.22]$).