



# Cache Side-Channel Attacks Against Black-Box Image Processing Software

Ssuhung Yeh<sup>1</sup><sup>a</sup> and Yuji Sekiya<sup>2</sup><sup>b</sup>

<sup>1</sup>Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan

<sup>2</sup>Security Informatics Education and Research Center, Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan

**Keywords:** Side-Channel Analysis, Cache Side-Channel Attack, Machine Learning, Deep Learning, Convolutional Neural Network.

**Abstract:** Cache side-channel attacks are a persisting threat to modern computers for their ability to steal secret information in memory and hard-to-detect characteristics. While researchers have studied these attacks for a long time, there has been relatively little focus on attacks against media software. One reason is the inherent noisiness of cache side-channels, making it challenging to extract meaningful information from it. However, recent advancements in machine learning have changed the landscape, making side-channel analysis more accessible. In this paper, we proposed a new side-channel analysis framework that is capable of extracting high-level information from complex applications. With this framework, we attacked image processing programs, reconstructed images that the victim opened with cache side-channel attacks, and achieved significantly improved results compared to the previous work.

## 1 INTRODUCTION


Side-channel attacks involve leveraging additional information about a computer to infiltrate its internal states. This supplementary data encompasses factors such as electromagnetic emissions, power usage patterns, and execution timing. Subsequently, this information can be meticulously analyzed to extract sensitive data, such as cryptographic keys. With the surge in the popularity of public cloud services, the threat posed by side-channel attacks has intensified. Numerous studies (Irazaqui et al., 2014; Moghimi, 2023) have demonstrated the practical feasibility of cross-VM side-channel attacks, heightening concerns.


While side-channel attacks have been under scrutiny for an extended period, the practice of conducting side-channel analysis (SCA) on complex software has traditionally been regarded as a formidable endeavor, if not outright impossible. Nonetheless, the substantial advancements in machine learning and deep neural networks in recent years have paved the way for the extraction of high-level information from collected traces. This

development has amplified the potency of side-channel attacks, rendering them more formidable than ever before.

In this context, a pivotal question arises: With the aid of advanced machine learning techniques, what is the upper limit of information attainable through side-channel analysis? We think that investigating this question at the present juncture is important, given the aforementioned reasons.

In this paper, our primary focus is on investigating cache side-channel attacks targeting black-box image processing applications. Specifically, these applications including a JPEG decoding program and a WebP decoding program, both of which are designed to convert JPEG or WebP images into bitmap files. Concurrently, an attacker initiates a cache side-channel attack on these programs and captures memory access traces. The objective is to gauge the attacker's ability to effectively reconstruct the original input images from these traces using neural networks. Our contributions can be summarized as follows:

<sup>a</sup> <https://orcid.org/0009-0005-1442-4159>

<sup>b</sup> <https://orcid.org/0009-0006-6287-9606>

- Proposed a new side-channel analysis framework that supports both Prime+Probe and write-back channel attacks against image processing software.
- With the proposed framework, we use it to attack `libjpeg` and `libwebp` programs and reconstruct images successfully. Compared to the previous work (Yuan et al., 2022), the reconstructed images have much higher fidelity even under stricter conditions.
- To our knowledge, this work is the first one that attacks `libwebp` with side-channel analysis, and also the first one doing side-channel analysis with a write-back channel attack.

In the rest of this paper, we will first survey related works about cache side-channel attacks, side-channel analysis with machine learning, and side-channel attacks against media programs in Section 2. Section 3 presents the dataset, the attack setup, and the neural network model design. The reconstruction result is presented and discussed in Section 4. Finally, the conclusion and future work are provided in Section 5.

Our code is available in our GitHub repository (WEBIST-2023-Cache\_Side\_Channel, 2023).

## 2 BACKGROUND AND RELATED WORKS

### 2.1 Cache Side-Channel Attacks

The cache is a hardware set between main memory and CPUs to accelerate the memory access speed. Due to its shared characteristic, the execution of one process will influence the state of the cache, thus, influencing the memory access time of other processes. In other words, an attacker process can infer other processes' internal states by observing whether each memory access is cache hit or cache miss.

There are many variations of cache side-channel attack techniques, that differ in the threat model and amount of information the attacker can get. Here, we introduce two of them that are related to this work, the Prime+Probe attack and the write-back channel attack.

#### 2.1.1 Prime+Probe Attack

The Prime+Probe attack (Tromer et al., 2010) exploit that modern caches mostly apply set-associative design, that the data is stored in which cache line is decided by certain bits in the middle of the memory address, called index bits.

The attack can be separated into two steps. In the prime stage, the attacker fills the cache with his data by accessing memory addresses mapping to all cache lines, creating the eviction set. Then the attacker waits for a while for the victim to execute. During the victim's execution, he will replace some data in the cache with his data. The attacker enters the probe stage the next time the attacker process is scheduled to execute. This time, he accesses the whole eviction set again and measures the time it takes to retrieve the data. If it takes longer to access a memory address, this infers the victim accessed this cache line before so that the attacker's data is evicted from the cache.

Overall, as long as a program has access to a cycle-level high-precision clock, and can create an eviction set, it can launch a Prime+Probe attack and infer which cache lines or memory addresses are accessed by the victim. Previous works have shown that Prime+Probe attack can be used to exploit encryption keys (Tromer et al., 2010; Liu et al., 2015) and perform website fingerprinting (Oren et al., 2015).

#### 2.1.2 Write-Back Channel Attack (WB Attack)

Besides knowing cache hit or miss, memory accessing time can also be exploited to infer whether the cache line is dirty or not (Cui et al., 2022). If the cache write policy is write-back, when a process updates a value in memory, the update will only be done in the cache under the hood. In this scenario, the dirty bit of that cache line will be set, so that the hardware knows to write the data back to the main memory when eviction happens.

The write-back channel attack (WB attack) exploits the fact that when the dirty bit is set for a cache line, it takes a longer time to write the data back, thus gaining further information about whether the victim process writes data to the cache line. The WB attack is an upgraded version of the Prime+Probe attack. Under the same condition, the WB attack can infer which cache lines are read or written by the victim. The previous work has described the possibility of creating side-channel attacks with the write-back channel.

### 2.2 Side-Channel Analysis with Machine Learning

Though side-channel attacks are easy to launch, the collected data need to be analyzed to get sensitive data. The whole process is usually called side-channel analysis. The analyzing task is usually far from easy for two reasons. First, the collected data is noisy and

huge in size. Second, the relationship between the secret to extract and collected data remains unclear. These two reasons make side-channel analysis a very difficult and labor-intensive task. However, with the progression in machine learning techniques, this task has become feasible in practice.

First, researchers have shown that neural networks can be used to denoise traces (Wu and Picek, 2020; Kwon et al., 2021) collected with side-channel attacks. After that, more studies have proved that deep neural networks can even be exploited to perform end-to-end side-channel analysis attacks, including website fingerprinting with cache side-channel (Cook et al., 2022) and keystroke logging with electromagnetic side-channel (Zhan et al., 2022).

### 2.3 Side-Channel Analysis of Media Program

Side-channel analysis of media software hasn't been studied a lot relatively. Compared to breaking encryption implementations, the data to steal in media software is larger in size, and the diversity in software implementations is greater. On the other hand, website fingerprinting and keystroke logging with side-channel analysis with side-channel analysis can be degraded to classification problems, while attacks on media software can't.

As started by Xu, Cui, and Peinado (2015), they successfully extracted the outlines of JPEG images through side-channel analysis. Followed by Balmau, et al. (2017), they reconstructed JPEG images with colors. However, these two works launched attacks in different scenarios. They assumed the OS was compromised and treated the victim program as a white-box, which are both strong assumptions.

Image reconstruction with non-privileged, black-box side-channel analysis (Yuan, Wang et al., 2021; Yuan, Pang et al., 2022) then succeeded. Yuan et al. simplified the process of reconstructing images from memory address traces to a regression problem. They represented images with latent vectors that contained high-level information, extracting latent vectors from traces, and then reconstructed images from them. This made side-channel analysis of media software possible.

## 3 METHODOLOGY

### 3.1 Threat Model

The threat model includes assumptions listed as follows:

- The attacker can execute native code on the victim's machine.
- The attack doesn't need any knowledge of the victim program, only treat it as a black box.
- The attacker has the same machine and victim program, or he can input any data into the victim program and observe the trace to produce training data.
- The cache being attacked uses the least recently used (LRU) for the cache replacement policy and write back for the cache writing policy.

For evaluation, we assume the target cache to be the L1 data cache, but the attack framework doesn't limit to any level of cache.

The framework is evaluated on two image processing programs, which are a JPEG decoding program and a WebP decoding program. The JPEG decoding program we used is the example JPEG decoding program `tjexample.c` in the popular `libjpeg-turbo` library (ver. 2.1.92) (`libjpeg-turbo`, 2010). This program takes a JPEG image as the input, decodes it, and outputs the bitmap file. As for the WebP decoding program, the example program `dwebp.c` in the `libwebp` library (ver. 1.3.1) (`libwebp`, 2011) is used.

The complete attack consists of two phases. In the training phase, the attacker produces traces corresponding to the reference images and trains the neural network. Then in the second phase, the attacker can reconstruct unknown images from traces collected from the victim.

### 3.2 Dataset

Experiments are conducted with two image datasets, which are JPEG and WebP datasets. The images come from *Large-scale CelebFaces Attributes (CelebA) Dataset* (Liu et al., 2015), Align & Cropped version, and are then resized to 128x128 pixels. The images are in JPEG format originally, so for the WebP dataset, manual transformation is required. Ordered by image ID, the first 80,000 images are used for training, and the last 19,921 images are used for testing. Every image in the dataset belongs to one of 10177 identities. This identity is used to optimize the training of the neural network.

### 3.3 Attack Setup

In this study, Intel Pin (Luk et al., 2005) is used to collect memory access traces. The reason for choosing Intel Pin is that it is a dynamic instrumentation tool, in other words, there is no need

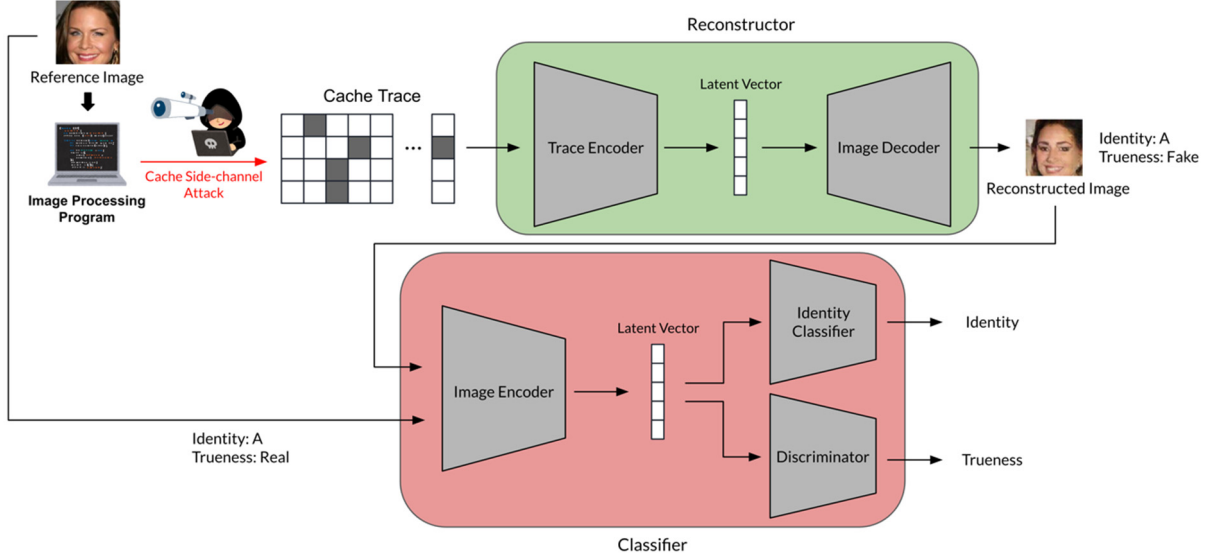


Figure 1: Model Overview.

to recompile the victim program, which corresponds to the black-box assumption about the victim program.

After traces are collected, post-processing is performed. The first step is to extract index bits according to the cache configuration. In our case, the attack target is the L1 data cache with 64 sets, and 64 bytes per cache line, thus, 7-th to 12-th bits of memory addresses are extracted. Next, pad traces to the maximum length of all traces. Finally, encode the cache line index with the binary encoding method. In other words, for each memory access, a vector with 64 elements is created. For the Prime+Probe attack, only the element corresponding to the accessed cache line index is 1, otherwise 0. As for the WB attack, -1 is used to represent a write, and 1 for a read.

### 3.4 Model Design

The overview of the neural network is shown in Figure 1. The primary reconstruction job is done by a reconstructor network, which is essentially a variational autoencoder (VAE). It is composed of a trace encoder and an image decoder. The idea is that the trace encoder is expected to extract high-level information (skin color, face direction, ...) about the image, and the image decoder can create the image according to it. As the previously proposed framework (Yuan et al., 2022) has done, a neural network is chained after. It is used to answer if an image is real or not and classify its identity, called a classifier. Though this part is not mandatory, they can provide extra information about how the reconstruction images look and propagate loss back to train the reconstructor better.

The training process of the whole neural network is analogous to a generative adversarial network (GAN) (Goodfellow et al., 2014). First, fix the reconstructor, and train the classifier with real and fake images reconstructed by the reconstructor. Then, fix the classifier and train the reconstructor with the assistance of the classifier afterward. Hopefully, the two neural networks will grow together and provide a better reconstruction and classification result.

For the detail of models inside each part, the trace encoder is a 1-dimensional convolutional neural network (1D CNN), and the image decoder is a 2-dimensional convolutional neural network (2D CNN).

When training the classifier, the loss function is defined as follows:

$$\mathcal{L} = CE(i_x, \hat{i}_x) + BCE(r_{\text{fake}}, \hat{r}_{\hat{x}}) + BCE(r_{\text{real}}, \hat{r}_x) \quad (1)$$

$x$  is defined as the original image, and  $\hat{x}$  is the reconstructed image. The first term is the cross-entropy loss between the real identity of the image  $i_x$  and the predicted identity  $\hat{i}_x$  based on the original image  $x$ . The second and the third terms are the binary cross entropy. It calculates the distance between the trueness of a fake image  $r_{\text{fake}}$  and the prediction of trueness  $\hat{r}_{\hat{x}}$  based on the reconstructed image, and the trueness of a real image  $r_{\text{real}}$  and the prediction  $\hat{r}_x$  based on the reference image.

When training the reconstructor, the loss function is defined as follows:

$$\mathcal{L}(x, \hat{x}) = \lambda \times \mathcal{L}_{\text{rec}}(x, \hat{x}) + \mathcal{L}_{\text{pre}}(x, \hat{x}) \quad (2)$$

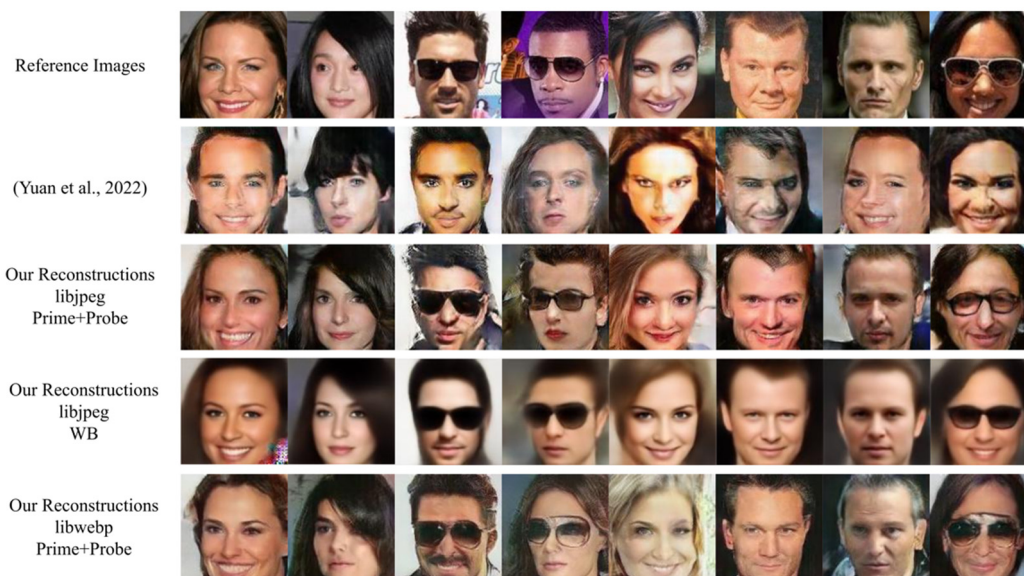


Figure 2: Qualitative Reconstruction Result.

Table 1: Quantitative Results of Experiments.

	(Yuan et al., 2022)	Ours		
Attack Target		libjpeg		libwebp
Max Trace Length		290745		897567
Attack	Prime+Probe	Prime+Probe	WB	Prime+Probe
Avg. SSIM score	0.09337	0.23059	0.32506	0.20095

$\mathcal{L}_{rec}$  and  $\mathcal{L}_{pre}$  are reconstruction loss and prediction loss, while  $\lambda$  is a parameter used to balance these two terms.  $\mathcal{L}_{rec}$  is defined as follows:

$$\mathcal{L}_{rec} = \alpha \times SSIM(x, \hat{x}) + (1 - \alpha) \times MSE(x, \hat{x}) \quad (3)$$

The first part is the structural similarity loss (SSIM) (Wang et al., 2004). The SSIM score is a common method to quantify the perceptual similarity between two images by splitting the images into blocks and comparing the luminance, contrast, and structure of each block. It is a value between -1 and 1, and a higher score means a higher similarity. The SSIM loss is defined as the opposite of the SSIM score, which means a higher score infers a lower similarity. Since the SSIM loss doesn't consider the difference of color, a mean square error (MSE) term is added, and parameter  $\alpha$  is the weight between these two terms. The definition of  $\mathcal{L}_{pre}$  is as follows:

$$\mathcal{L}_{pre} = CE(i_x, \hat{i}) + BCE(r_{fake}, \hat{r}) \quad (4)$$

## 4 EVALUATIONS

The qualitative reconstruction results are shown in Figure 2. For more reconstructed images, please refer to the Appendix. For the quantitative results, the average SSIM score (Wang et al., 2004) is used to quantify the reconstruction result. The score is calculated by averaging the SSIM scores between the reference images in the testing split and reconstructed images. The settings of different experiments and the average SSIM scores are presented in Table 1.

For the rest of this section, we will discuss the results in more detail and compare them between experiments.

### 4.1 Comparison with the Previous Work

We compare our experiment results with the result reconstructed with the framework proposed in the previous work (Yuan et al., 2022). For the previous

work, they used the 7-th to 32-th bits in the memory address in the traces when the side-channel is set to cache line index, which contains more information than a cache side-channel attacker can learn theoretically. Our framework only uses the 7-th to 12-th bits as the input, which corresponds to the number of cache sets. Despite the reduced information in the traces, a superior reconstruction result is achieved. We will describe the reconstruction result of the previous work as that there is some correspondence between reference images and reconstructed images, however, they are not visually similar. Images reconstructed with our framework are much similar to original images, in aspects of skin colors, hairstyles, facial expression, and so on.

The reason that our framework performs better can be explained in two aspects. First, in the previous work (Yuan et al., 2022), their model interprets the accessed memory addresses or cache line indexes as a value. However, the values only represent a location in the memory of the cache, not a magnitude of something. We encoded the value using the binary encoding method, which is believed to be the correct way to interpret those values. Second, they use 2D CNN for the model of the trace encoder. This forces the model to consider elements scattered in traces together and look for patterns inside them. On the other hand, 1D CNN is used in our model, thus the model will only consider the relation between adjacent elements in traces.

#### 4.2 Comparison Between the Prime+Probe and the WB Attack

The qualitative and quantitative results show that the neural network can reconstruct images with higher fidelity when launching a write-back channel attack. This result corresponds with our expectation, as there is additional information about read/write in the traces. However, according to our observation, the result is largely dependent on the encoding of read and write behavior. We haven't spent much time comparing different encoding methods.

#### 4.3 Attack on libwebp

Comparing the results of attacking `libjpeg` and `libwebp` with Prime+Probe attack, the fidelity of images is at about the same level. Though we do expect a better result considering the length of traces is about 3 times longer, the outcome is negative. Our interpretation is that the example program in `libwebp` is more complicated and supports transformation between more formats, thus, lots of

parts in the traces may not be relevant to the input image, and they may cause. Regardless, we showcased the potential of our framework to attack more complex software.

## 5 CONCLUSIONS

This paper underscores the heightened significance of cache side-channel analysis, revealing its greater severity than previously acknowledged. We introduce a novel cache side-channel analysis framework that enables the precise reconstruction of images with remarkable fidelity through cache side-channel attacks, all without requiring any prior knowledge of the targeted victim program. Importantly, our illustration of image processing program exploitation serves as an exemplary case, echoing prior findings (Yuan et al., 2022) that the same attack framework can be adapted to target diverse software types, such as audio processing and text processing programs, by simply modifying the image decoder model. This compelling evidence underscores the imperative to recognize the non-negligible threat posed by cache side-channel analysis.

As the upper limits of information leakage achievable through cache side-channel attacks are explored, the next objective is to empirically assess their practical viability by implementing real-world Prime+Probe and write-back channel attacks. This aspect of our research remains a subject for future exploration. Additionally, the broader challenge of reconstructing general images remains open for further investigation. While our framework does not assume any specific image type, it is worth noting, as indicated in other research on image-to-image VAE (Van Den Oord et al., 2017), that even with advanced model design, the latent vector's dimension required for the reconstruction of general images exceeds 128 significantly.

## REFERENCES

- Cook, J., Drean, J., Behrens, J., and Yan, M. (2022). There's always a bigger fish: a clarifying analysis of a machine-learning-assisted side-channel attack. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (pp. 204-217).
- Cui, Y., Yang, C., and Cheng, X. (2022). Abusing cache line dirty states to leak information in commercial processors. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 82-97). IEEE.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-

Farley, D., Ozair, S., ... and Bengio, Y. (2014). Generative Adversarial Nets. *Advances in neural information processing systems*, 27.

Hähnel, M., Cui, W., and Peinado, M. (2017). High-Resolution Side Channels for Untrusted Operating Systems. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)* (pp. 299-312).

Irazoqui, G., Inci, M. S., Eisenbarth, T., and Sunar, B. (2014). Wait a minute! A fast, Cross-VM attack on AES. In *Research in Attacks, Intrusions and Defenses: 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings 17* (pp. 299-319). Springer International Publishing.

WEBIST-2023-Cache\_Side\_Channel (2023). [https://github.com/ssuhung/WEBIST-2023-Cache\\_Side\\_Channel](https://github.com/ssuhung/WEBIST-2023-Cache_Side_Channel)

Kwon, D., Kim, H., and Hong, S. (2021). Non-profiled deep learning-based side-channel preprocessing with autoencoders. *IEEE Access*, 9, 57692-57703.

libjpeg-turbo (2010). <https://github.com/libjpeg-turbo/libjpeg-turbo>. [Online; accessed 10-April-2023].

libwebp (2011). <https://github.com/webmproject/libwebp>. [Online; accessed 20-August-2023].

Liu, F., Yarom, Y., Ge, Q., Heiser, G., and Lee, R. B. (2015, May). Last-level cache side-channel attacks are practical. In *2015 IEEE symposium on security and privacy* (pp. 605-622). IEEE.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision* (pp. 3730-3738).

Luk, C. K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V. J., and Hazelwood, K. (2005). Pin: building customized program analysis tools with dynamic instrumentation. *ACM SIGPLAN Notices*, 40(6), 190-200.

Moghimi, D. (2023). Downfall: Exploiting Speculative Data Gathering. In *32nd USENIX Security Symposium (USENIX Security 23)* (pp. 7179-7193).

Oren, Y., Kemerlis, V. P., Sethumadhavan, S., and Keromytis, A. D. (2015). The spy in the sandbox: Practical cache attacks in JavaScript and their implications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1406-1418).

Tromer, E., Osvik, D. A., and Shamir, A. (2010). Efficient cache attacks on AES, and countermeasures. *Journal of Cryptology*, 23(1), 37-71.

Van Den Oord, A., and Vinyals, O. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Wu, L., and Picek, S. (2020). Remove some noise: On preprocessing of side-channel measurements with autoencoders. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 389-415.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 600-612.

Xu, Y., Cui, W., and Peinado, M. (2015). Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *2015 IEEE Symposium on Security and Privacy* (pp. 640-656). IEEE.

Yuan, Y., Pang, Q., and Wang, S. (2022). Automated side channel analysis of media software with manifold learning. In *31st USENIX Security Symposium (USENIX Security 22)* (pp. 4419-4436).

Yuan, Y., Wang, S., and Zhang, J. (2021). Private image reconstruction from system side channels using generative models. In *Ninth International Conference on Learning Representations*.

Zhan, Z., Zhang, Z., Liang, S., Yao, F., and Koutsoukos, X. (2022). Graphics peeping unit: Exploiting EM side-channel information of GPUs to eavesdrop on your neighbors. In *2022 IEEE Symposium on Security and Privacy (SP)* (pp. 1440-1457). IEEE.

## APPENDIX

More reconstruction results are presented here.

