Hiding My Real Self! Protecting Intellectual Property in Additive Manufacturing Systems Against Optical Side-Channel Attacks

Sizhuang Liang
Georgia Institute of Technology
liangsizhuang@gatech.edu

Saman Zonouz
Rutgers University
saman.zonouz@rutgers.edu

Raheem Beyah
Georgia Institute of Technology
rbeyah@coe.gatech.edu

***Abstract*—We propose an optical side-channel attack to recover intellectual property in Additive Manufacturing (AM) systems. Specifically, we use a deep neural network to estimate the coordinates of the printhead as a function of time by analyzing the video of the printer frame by frame. We found that the deep neural network can successfully recover the path for an arbitrary printing process. By using data augmentation, the neural network can tolerate a certain level of variation in the position and angle of the camera as well as the lighting conditions. The neural network can intelligently perform interpolation and accurately recover the coordinates of an image that is not seen in the training dataset.**

To defend against the optical side-channel attack, we propose to use the optical noise injection method. Specifically, we use an optical projector to artificially inject carefully crafted optical noise onto the printing area in an attempt to confuse the attacker and make it harder to recover the printing path. We found that existing noise generation algorithms, such as replaying, random blobs, white noise, and full power, can effortlessly defeat a naive attacker who is not aware of the existence of the injected noise. However, an advanced attacker who knows about the injected noise and incorporates images with injected noise in the training dataset can defeat all of the existing noise generation algorithms. To defend against such an advanced attacker, we propose three novel noise generation algorithms: channel uniformization, state uniformization, and state randomization. Our experiment results show that noise generated via state randomization can successfully defend against the advanced attacker.

I. Introduction

Additive Manufacturing (AM), also known as 3D Printing, is gaining popularity in a variety of sectors, such as automotive [14], aerospace [24], construction [8], healthcare [37], fashion [32], and education [1]. As the application of AM increases rapidly, the protection of intellectual property associated with AM becomes a concern [7], [33], [34], [36]. For example, it could take years for a company to design a product to be additively manufactured and then sell the product for a profit [20]. Losing the intellectual property could mean a financial loss. Meanwhile, there are designs that should be strictly controlled. An example is 3D printed firearms [22]. If a design is leaked, it could lead to unlawful production of the firearms, causing security problems in the society.

Network and Distributed Systems Security (NDSS) Symposium 2022
24-28 April 2022, San Diego, CA, USA
ISBN 1-891562-74-6
https://dx.doi.org/10.14722/ndss.2022.24298
www.ndss-symposium.org

Since AM relies on computers to work, it is possible for intellectual property in an AM system to be stolen by cyberattacks. For example, the network protocols employed by two MakerBot 3D printers can be exploited and the information in the 3D printers can be exfiltrated [15]. To fight against cyberattacks, common practices to secure computer networks can be employed. For example, when transferring design files from a host computer to a printer, encryption can be used to protect the confidentiality of the design files [12].

Side-channel attacks are more stealthy attacks and they can be used to break computer systems that are otherwise hard to break [25]. In the literature, there are already many side-channel attacks to steal intellectual property in AM systems. For example, when the acoustic wave in a printing process is recorded by an attacker, the attacker can potentially infer information about the printing process by a variety of techniques, such as signal processing and machine learning [2], [19], [29]. In addition, there are research efforts in the literature to recover the coordinates of the printhead in a 3D printer by analyzing the infrared video observed by an infrared camera [3].

However, existing side-channel attacks on intellectual property in AM systems have a variety of limitations and only work well for a few printing processes. For example, the acoustic side-channel attacks in [2], [29] only work for printing processes where the movement directions are restricted to four or eight cardinal directions, and the acoustic side-channel attack in [19] only works for printing processes where the duration of each G-code instruction is long enough to clearly see the boundaries between G-code instructions in the spectrogram. The infrared video side-channel attack in [3] fails altogether in practical printing processes potentially due to a lack of proper methodology.

We propose an optical side-channel attack using a deep neural network. Although the optical side-channel attack requires the camera footage of a printing process, which can be hard to acquire compared with audio data in the acoustic side-channel attack, the optical side-channel attack can easily overcome the limitations faced by the acoustic side-channel attack and recover the path of an arbitrary printing process. Additionally, it is by far the hardest side-channel attack to defend against in practice as will be demonstrated later in this paper. To our best knowledge, we are the first one in the literature to successfully perform the optical side-channel attack on intellectual property in AM systems.

To defend against side-channel attacks on intellectual property in AM systems, researchers came up with a variety of methods, such as tuning manufacturing parameters [10], [11], using fake movements [29], balancing loads on motors [2], applying shielding [2], [19], [29], and injecting noise [19], [29]. In this paper, we focus on the noise injection method to defend against our proposed optical side-channel attack, because the noise injection method, compared to many other defense methods, has the least amount of adverse impact on 3D printers and printed objects.

Noise injection as an effective defense method requires an algorithm to determine the pattern of the noise to be injected. There are several existing noise generation algorithms such as random blobs, white noise, full power, and replaying the side-channel signal of another printing process [19]. According to our experiments on the optical side-channel attack, although the existing noise generation algorithms are effective against a naive attacker, they can be easily defeated by an advanced attacker, who is aware of the existence of injected noise and attempts to identify and remove the injected noise in the attack process. To defend against the advanced attacker, in this paper, we propose three novel noise generation algorithms, and they are channel uniformization, state uniformization, and state randomization. Our experiments show that the state randomization noise generation algorithm can effectively defend against attackers with different prior knowledge.

Our contributions are as follows:

- We propose an optical side-channel attack against 3D printers that enables adversaries to steal print designs through camera recordings during printing processes. To our best knowledge, we are the first one in the literature to successfully perform the optical side-channel attack on intellectual property in AM systems.
- We propose three novel noise generation algorithms (channel uniformization, state uniformization, and state randomization) for the noise injection method to defend against the optical side-channel attack.
- To perform experiments, we collected approximately one million images over a period of one month, totaling 240 GB, and we plan to share the datasets to the public. Our experiment results indicate that the state randomization noise generation algorithm succeeds in defending against even advanced adversaries, who have the knowledge about the deployed optical noise injection method.

II. Background Information

A. Additive Manufacturing

Additive Manufacturing (AM) refers to a collection of manufacturing processes where materials are joined together layer by layer to make objects directly from 3D models [5]. AM processes are performed by computers without human intervention. The operation of an AM process is called printing and the machine by which materials are joined together is called a printer. There are seven categories of AM processes and each category contains many types of AM processes. In this paper, we focus on Fused Deposition Modeling (FDM) [13], which is the most common AM process.

A general FDM process is described in Fig. 1. The AM process starts with a 3D model that describes the shape of the object to be printed. A specialized program called a slicer, such as Cura, Slic3r, and MatterSlice, then asks for manufacturing parameters, such as feed rates and temperatures, to be specified. Afterwards, the slicer converts the 3D model into a G-code file, which is a collection of G-code instructions. Finally, the G-code file is sent to a printer for execution. The printer interprets the received G-code file and instructs various actuators in the printer to work according to the G-code file. An object is printed as a result while various side-channel signals are emitted.

B. State-Variable Signals

From the perspective of control theory, a printer can be formally represented by a state variable[1]. The components of the state variable may include the position of the printhead, the position of the filament, the temperature of the nozzle(s) and the build plate, the speed of the fans, etc. The state variable as a function of time forms a state-variable signal, where each component of the state variable corresponds to a channel in the signal. A state-variable signal is described by

$$\boldsymbol{x}_{\mathrm{SV}}[n, c], n = 0, 1, \cdots, N-1, c = 0, 1, \cdots, C_{\mathrm{SV}} - 1, \quad (1)$$

where n is the time index, N is the number of data points, c is the channel index, and C_{SV} is the number of channels. If the sampling rate is f_s, then the duration of the state-variable signal is N/f_s. Fig. 2 shows a state-variable with three channels and they are the x, y, and z coordinates of the printhead.

In this paper, we may write $\boldsymbol{x}_{\mathrm{SV}}[n, c]$ simply as $\boldsymbol{x}_{\mathrm{SV}}[n]$ or $\boldsymbol{x}_{\mathrm{SV}}$, where $\boldsymbol{x}_{\mathrm{SV}}[n]$ is a vector of C_{SV} components and $\boldsymbol{x}_{\mathrm{SV}}$ is an array (or matrix) of shape $N \times C_{\mathrm{SV}}$.

C. Side-Channel Signals

Side channels are unintentional means of communication by which information about a computer or a cyber-physical system can be leaked to an outsider. AM systems have a variety of side channels. For example, when an AM system is printing an object, the system emits acoustic waves [2], [19], [29]. Other examples of side channels in an AM system include, but are not limited to, acceleration measured by accelerometers [6], [16], power consumption measured by power sensors [27], [17], optical videos captured by cameras [16], [35], and infrared videos captured by infrared cameras [3].

Mathematically, a side-channel signal is represented by

$$\boldsymbol{x}_{\mathrm{SC}}[n, c], n = 0, 1, \cdots, N-1, c = 0, 1, \cdots, C_{\mathrm{SC}} - 1, \quad (2)$$

where n is the time index, N is the number of data points, c is the channel index, and C_{SC} is the number of channels. If the sampling rate is f_s, then the duration of the side-channel

[1]A variable can be a scalar, a vector, or even a matrix. A state variable can be a vector with multiple components.

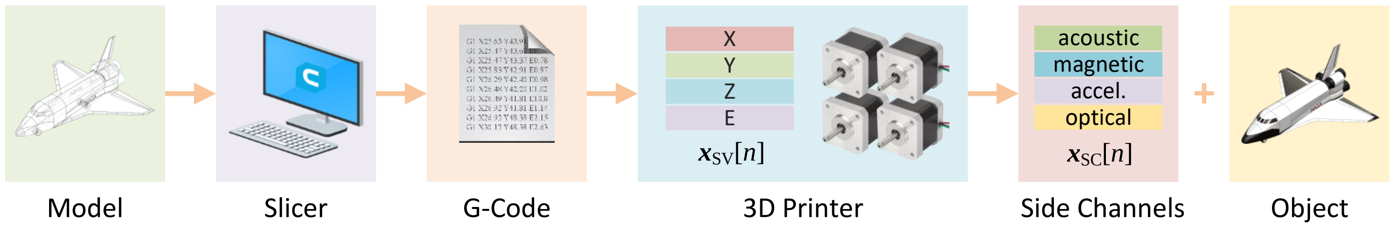


Fig. 1: A general FDM process. A slicer converts a 3D model into G-Code instructions, which are then sent to a printer for execution. The printer interprets the G-code instructions and instructs various actuators in the printer to work. An object is printed and various side-channel signals (acoustic, magnetic, optical, etc) are generated.

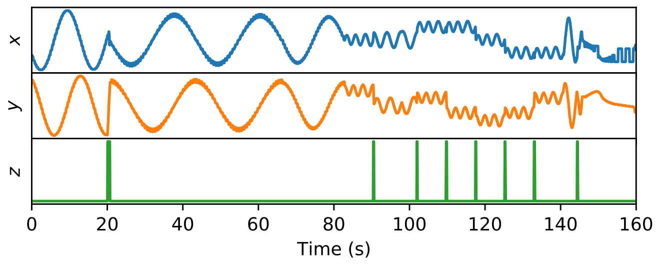


Fig. 2: Example of a state-variable signal $\boldsymbol{x}_{\mathrm{SV}}$ with three channels x, y, z, where (x, y, z) is the coordinate of the printhead. The state-variable signal could contain more channels such as the displacement of the filament in the extruder, the fan speed, the temperature(s) of the nozzle(s), and the temperature of the build plate.

signal is N/f_s. Typically, f_s is determined by the Analog-to-Digital Converter (ADC) in the data acquisition system that observes the side-channel signal.

As with $\boldsymbol{x}_{\mathrm{SV}}[n, c]$, we may write $\boldsymbol{x}_{\mathrm{SC}}[n, c]$ simply as $\boldsymbol{x}_{\mathrm{SC}}[n]$ or $\boldsymbol{x}_{\mathrm{SC}}$, where $\boldsymbol{x}_{\mathrm{SC}}[n]$ is a vector of C_{SC} components and $\boldsymbol{x}_{\mathrm{SC}}$ is an array of shape $N \times C_{\mathrm{SC}}$.

III. Related Work

This section presents related work about side-channel attacks on intellectual property in AM systems as well as efforts to defend against such side-channel attacks. As the time of writing, the amount of related work is very limited and there is no optical side-channel attack on AM systems in the literature.

A. Side-Channel Attacks on AM Systems

To perform the acoustic side-channel attack, Al Faruque et al. proposed an algorithm to separate the acoustic signal due to one motor and the acoustic signal due to another motor from the combined acoustic signal [2]. After the signals are separated, they use regression models and classifiers to determine the speed for each motor. However, according to [26], this approach may not be effective when the two motors are moving at the same time.

By assuming the printing speed is constant and known, Hojjati et al. used matching filters to determine the direction for each movement [19]. As the matching filters typically return multiple directions with equal probabilities, they further used the magnetic side channel and human intelligence to determine the most likely directions for all movements. Since this method depends on the abrupt changes in the side-channel signal to determine the boundaries of G-code instructions, this method may not work well for a printing process where there are a lot of short and rapid movements.

As with Hojjati's research, Song et al. assumed that the printing speed is known in advance and used magnetic fields for assistance [29]. Instead of using matching filters, they used five classifiers to directly determine the nozzle's movement and the extruder's state. Since this attack relies on classifiers to work, a major limitation of this attack is that it only works well for a printing process where most movements are along the eight cardinal directions.

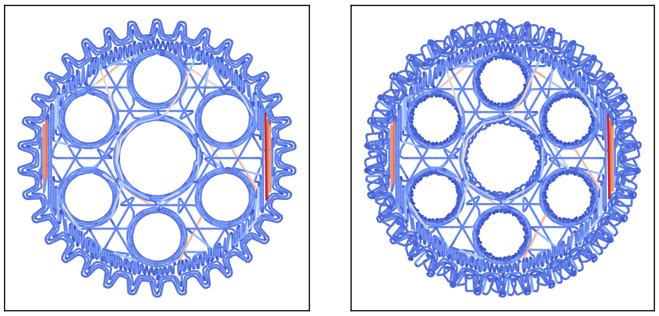
Other than the acoustic side-channel attack, Al Faruque et al. came up with an attack that uses the infrared side channel to steal IP in AM processes [3]. They proposed a mapping algorithm to convert an infrared video to a speed signal. However, according to [3], the attack was not successful due to a low resolution, a low sampling rate, a single view point, and inability to change the focus length of the infrared camera.

B. Defending Against Side-Channel Attacks

The authors that proposed the side-channel attacks on AM systems also mentioned potential mitigation methods without comprehensive evaluations in practice though. A complete list of the defense methods is listed as follows.

Parameter Tuning. Chhetri *et al.* proposed to minimize the mutual information between side-channel signals and G-code instructions by tuning manufacturing parameters, such as object orientation and printing speed [11]. However, the level of protection provided by the method is limited. According to their own performance metric, there is an average reduction of 10% in the success rate, and they acknowledge that this may not be able to defeat side-channel attacks in practice.

Movement Obfuscation. This method injects fake movements in a printing process to confuse an attacker [29], as shown in Fig. 3. A fake movement is a movement that moves the printhead at the extrusion speed without filament extrusion. This breaks the important assumption by many attackers that a low printing speed corresponds to an extrusion movement whereas a high printing speed corresponds to a non-extrusion movement [2], [19], [29]. However, this method introduces



(a) No Fake Movements (b) With Fake Movements

Fig. 3: Illustration of fake movements. (a) A layer without fake movements. (b) The same layer with fake movements.

stringing effects[2], which can adversely affect the quality of the printed object. In addition, this method may significantly extend the printing time.

Signal Power Reduction. This method aims at reducing the power of side-channel signals and thus reduces the Signal-to-Noise Ratio (SNR). The power of acoustic side-channel signals can be reduced by a better design. For example, compared with SeeMeCNC Rostock Max V3, Ultimaker 3 is much quieter and emits acoustic waves with much lower amplitude. The power of optical side-channel signals can be reduced by using less lights in the environment. However, using less lights not only makes normal operation harder but also leaves security problems, such as potentially increased theft activities.

Physical Shielding. This method installs physical shields to prevent side-channel signals from reaching the attacker [2], [19], [29]. Examples are acoustic shields for the acoustic side channel and view blocks for the optical side channel. However, deploying view blocks would be too expensive in practice, since it would require internal cameras to be removed, employees to be specially trained to recognize abnormalities in AM processes, and for them to manually and continuously check each printer by temporarily removing the view blocks.

Physical Uniformization. This method tries to make physical changes to an AM system such that the side-channel signals for different states of the AM system are similar. In this way, it will be hard for the attacker to infer the state of the AM system by analyzing the side-channel signals. For example, Al Faruque et al. proposed to balance the loads in the x and y motors to make the acoustic emission from the x motor and that from the y motor similar to each other [2]. For the optical side channel, the color of the printhead and the build plate can be designed to be the same (or some camouflage patterns) to make it hard to see the printhead in relation to the build plate. However, our experiments show that this method is not effective as we performed tests on a printer where the printhead and the build plate are both deep black.

[2]In an FDM process, when the nozzle travels from point A to point B without extrusion, the molten filament in the nozzle continues to come out in this process, and forms a string from point A to point B, which is undesirable. To suppress the stringing effect, the extruder retracts the filament at the beginning of the movement and then re-extrudes at the end of the movement.

Noise Injection. This defense method involves using signal generators (such as speakers) to artificially create side-channel signals to interfere with side-channel signals in AM systems in an effort to reduce the SNR and thus thwart side-channel attacks [29], [19]. The artificially created side-channel signals are noise for the attacker, and thus are also referred to as side-channel noise. This paper focuses on this method because, according to our experiments, it is a low cost and effective solution and it has the least amount of impact on the AM system. The noise injection method requires an algorithm to generate noise. Unfortunately, existing noise generation algorithms do not work well against our proposed optical side-channel attack as they can be easily defeated by advanced attackers as will be demonstrated by our experiments later in the paper. To address this problem, we propose three novel noise generation algorithms. One of the newly proposed noise generation algorithms, state randomization, can withstand an advanced optical side-channel attack.

C. *Use of Side Channels for Intrusion Detection*

There are Intrusion Detection Systems (IDSs) that leverage side channels in AM systems and some of these IDSs perform side-channel attacks as an intermediate process. For example, Chettri et al. came up with an IDS called KCAD to detect zero-day cyberattacks on AM systems [9]. They built machine learning models to estimate the state variable (velocities of the printhead) from acoustic waves. Afterwards, they compare the estimated state variable against the reference state variable interpreted from a G-code file to determine if the system is working as intended. Gao et al. proposed a process monitoring system to safeguard AM systems [16]. In this system, they estimate state variables (such as positions and velocities of the printhead) from the acceleration and magnetic side channels. They also provide methods to estimate the fan speed from acoustic waves. The aforementioned defense methods work against a different threat model where the adversaries attempt to tamper with the integrity of the printing process, whereas our threat model involves the confidentiality of the print design.

IV. Threat Model

The threat model is illustrated in Fig. 4. A 3D printer is printing an object of value. An attacker wishes to replicate this object without authorization. For this purpose, the attacker uses a variety of side-channel sensors around the printer to collect the side-channel signals in the printing process. The attacker then uses a variety of methods, such as signal processing and machine learning, to recover information about the printing process. The recovered information, namely the intellectual property, can be used to help reconstruct the printing process.

A. *Operational Definition of Intellectual Property*

Strictly speaking, the intellectual property of an AM process includes all information that is necessary to exactly replicate the printed object. The information includes the make and model of the printer that is used, the composition of the materials, the geometry of the design, and the manufacturing parameters [36]. We assume that the attacker knows the make and model of the printer and the composition of the materials. The geometry of the design and the manufacturing parameters are embedded in the G-code file. In fact, many papers in the

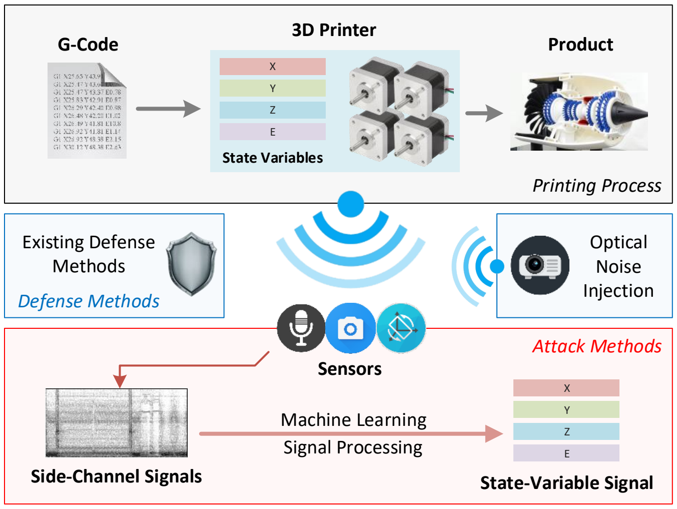


Fig. 4: Illustration of the threat model.

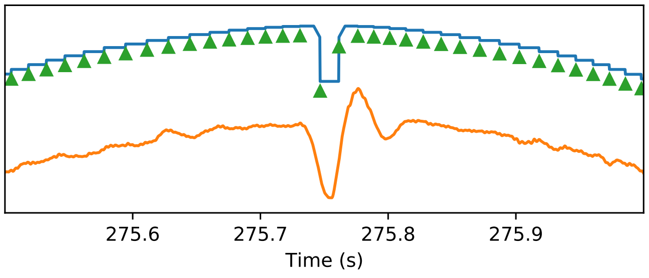


Fig. 5: Segmenting signals by G-code instructions. The upper signal is a velocity signal obtained by simulating a G-code file. The lower signal is the same velocity signal estimated by analyzing side-channel signals. The green triangles show the boundaries between G-code instructions. If we can only see the estimated velocity signal, it is nearly impossible to determine the boundaries of the G-code instructions.

literature define the intellectual property of a printing process as the G-code file [2], [29]. However, we argue that it is not practical to recover the G-code file from side-channel signals ($\boldsymbol{x}_{\text{SC}}$) of a real printing process as it is not possible to segment side-channel signals by G-code instructions, as explained in Fig. 5. As a result of this, in this paper, we operationally define the intellectual property of a printing process as its state-variable signal ($\boldsymbol{x}_{\text{SV}}$). In fact, we are the first one in the literature to define intellectual property in this way.

B. Acquisition of Optical Side-Channel Signals

In this paper, we are mainly concerned with the optical side channel. There are various ways for an attacker to obtain the optical side-channel signal of a printing process. The attacker may have an employee or maintenance contractor working at the factory to install a hidden camera on the ceiling of the factory. The AM factory may have many devices installed on the ceiling, such as fire detectors, sprinklers, motion sensors, light sensors, etc. The hidden camera can be disguised as one of these legitimate devices to evade suspicion. The AM factory may have surveillance cameras, which may be connected to the Internet. They may be compromised to become the part of a botnet [4]. The attacker may be able to get a copy of the data from a compromised camera [31]. We consider it too risky in practice to ban surveillance cameras altogether in an AM factory as doing so not only affects infrastructural physical security negatively but also invites social engineering activities. In addition, some printers have built-in process monitoring cameras with weak access control and the attacker can access the cameras directly.

C. Analysis of Optical Side-Channel Signals

The attacker uses signal processing and data driven methods such as template filtering and machine learning to recover intellectual property (the state-variable signal) from an optical side-channel signal. For a data driven method, we assume that the attacker has access to a printer of the same model in a similar environment. The attacker can collect a lot of training data subject to the constraints of time and computational resources. In addition, the attacker may manually identify the coordinates of the printhead for a limited number of images in the optical side-channel signal of the target AM system.

When a defense method is deployed, we assume that an advanced attacker knows the defense method and tries to evade the defense method by launching more sophisticated attacks. For example, when the noise injection method is used, the attacker tries to learn the pattern of the injected noise and artificially create training samples with the same type of injected noise. The attacker then retrains the machine learning model with the newly created training samples. The retrained machine learning model may be able to automatically reject the injected noise and proceed with the attack. We consider the advanced attacker because the naive attacker cannot differentiate the performance of different noise generation algorithms. We need the advanced attacker to demonstrate the relative strength of our proposed noise generation algorithms.

V. Optical Side-Channel Attack

This section describes the details of the optical side-channel attack. We first of all mathematically describe the side-channel attack. We then discuss the challenges in the attack process. Finally, we present the implementation of the attack.

A. Attack Formulation

The optical side-channel attack essentially attempts to recover the state-variable signal ($\boldsymbol{x}_{\text{SV}}$) from the optical side-channel signal ($\boldsymbol{x}_{\text{SC}}$). The process is possible since there is relationship between $\boldsymbol{x}_{\text{SV}}$ and $\boldsymbol{x}_{\text{SC}}$. For example, the position of the printhead directly affects the image seen by a camera. The speed of the printhead also affects the image seen by a camera in the form of motion blur. The relationship between $\boldsymbol{x}_{\text{SV}}$ and $\boldsymbol{x}_{\text{SC}}$ can be mathematically described by

$$\boldsymbol{x}_{\text{SC}}[n] = \boldsymbol{f}(\boldsymbol{x}_{\text{SV}}[n], \frac{\partial \boldsymbol{x}_{\text{SV}}}{\partial t}[n]) + \boldsymbol{e}[n], \tag{3}$$

where n is the time index, $\boldsymbol{x}_{\text{SC}}[n]$ is a single frame in the video (an image), $\boldsymbol{x}_{\text{SV}}[n]$ is the state variable at time index n, $\partial \boldsymbol{x}_{\text{SV}}/\partial t$ is the derivative of $\boldsymbol{x}_{\text{SV}}$ with respect to time t, $\boldsymbol{f}$

is the mapping from $\boldsymbol{x}_{\mathrm{SV}}[n]$ and its derivative to $\boldsymbol{x}_{\mathrm{SC}}[n]$, and $\boldsymbol{e}[n]$ is the noise at time index n.

Since it is very hard to mathematically solve for $\boldsymbol{x}_{\mathrm{SV}}[n]$ from $\boldsymbol{x}_{\mathrm{SC}}[n]$, we use data-driven methods such as template matching [19] and machine learning [29]. The outcome is the estimated printhead trajectory throughout the printing process.

B. Challenges of the Attack

The challenges in the optical side-channel attack and the mitigation strategies are described as follows.

Limited Relationship. The relationship between certain channels in the state-variable signal and the optical side-channel signal can be weak. For example, the position of the filament, the temperatures of the nozzles, and the temperature of the build plate do not affect the optical side-channel signal. As a result, it is nearly impossible to directly recover these channels in the state-variable signal. To address this challenge, we use the common assumption in the literature to estimate the position of the filament. That is, the attacker assumes that the printhead mainly travels at two target speeds, a low speed for extrusion movements and a high speed for relocating the printhead. This assumption is used by existing side-channel attacks [2], [29], [19]. The temperatures of the nozzle(s) and the build plate can be easily estimated as their optical values for a given material are typically known constants.

Camera Properties. The configuration of the camera can affect the side-channel attack, such as the location and angle of the camera with respect to the printer, as well as the focus length, white balance, and exposure of the camera. To address this challenge, we assume that, when an attacker recreates a system to collect training images, the attacker is able to get the same printer and camera with a similar configuration. It is also possible for the attacker to manually label a limited number of images in videos taken from the camera used in the side-channel attack and use the labeled images for training purposes. In addition, the attacker augments the training images in terms of rotation, translation, exposure, and hue, to diversify the training dataset.

Lighting Conditions. The lighting conditions can change over time, especially when the AM facility has windows that let natural lights through. A robust side-channel attack should be able to tolerate changes in lighting conditions. To address this challenge, we use machine learning instead of template matching to perform the attack and we augment the training dataset by varying its exposure, brightness, and contrast, to simulate changes in the lighting conditions.

Motion Blur Effect. It takes time for a camera to collect enough photons to register a picture, and this time is referred to as the shutter speed of the camera. When anything moves significantly when the camera's shutter is open, the moving object gets blurred in the registered picture, and this is referred to as the motion blur effect. To account for the motion blur effect, we need six degrees of freedom in the label of a picture and they are the x, y, z coordinates of the printhead as well as the three components of the velocity vector of the printhead. Due to the curse of dimensionality in machine learning, we need to collect a huge number of training images with the printhead at different locations with different velocities. This

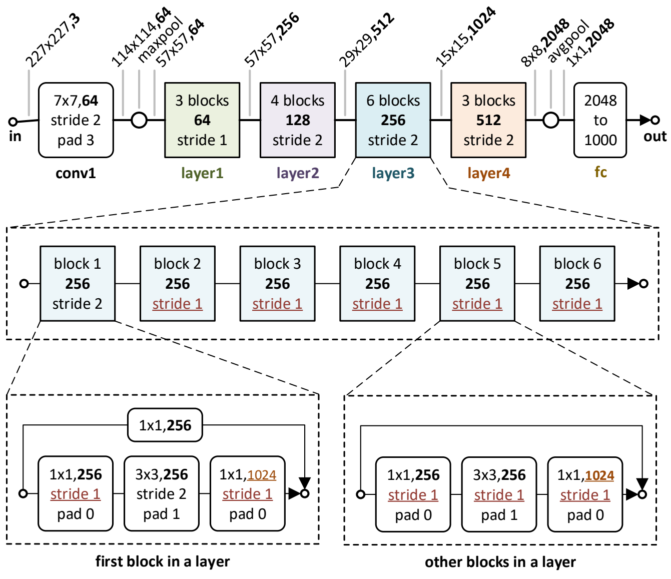


Fig. 6: Structure of ResNet50 in [18].

can be rather challenging. To deal with this problem, we primarily perform experiments where the ambient light is adequate and the camera has a fast shutter speed. In this way, we can simply ignore the motion blur effect and we now have

$$\boldsymbol{x}_{\mathrm{SC}}[n] = \boldsymbol{f}(\boldsymbol{x}_{\mathrm{SV}}[n]) + \boldsymbol{e}[n]. \tag{4}$$

Deposited Materials. Materials (filaments) are deposited in a printing process. As a result, for the same location of the printhead, it is possible for the camera to see different images due to the deposited materials. This makes the side-channel attack hard because for the same label (the same location of the printhead), the attacker needs to collect multiple images with different distribution of deposited materials on the build plate. To make things worse, changing the deposited materials on the build plate requires human efforts and can be very time consuming. Fortunately, when printing an object that is smaller than the printhead, there is a high chance that the object can be totally obscured by the printhead. When this happens, we do not need to consider the influence of deposited materials on the side-channel attack.

C. Implementation of the Attack

In this paper, we use ResNet50 [18], a deep convolutional neural network, to recover the state-variable signal from the optical side-channel signal. The structure of ResNet50 is shown in Fig. 6.

Modifying ResNet50. ResNet50 was originally designed to perform image classification for 1000 classes. Hence, the output of ResNet50 contains 1000 numbers. We modified the last fully connected layer in ResNet50 such that it transforms 2048 neurons into 3 neurons, corresponding to x, y, and z respectively. In the rest of the paper, we refer to the modified ResNet50 directly as ResNet50. The input to ResNet50 is a single image showing the printer whereas the output of ResNet50 is the coordinate of the printhead (x, y, z).

Training Process. To train the deep neural network, we collect images of the printer with the printhead at various locations without filament extrusion[3]. To be specific, we instruct the printhead to sweep the whole printing space with a predetermined interval. To improve the robustness of the network, we augment the training dataset by randomly rotating the images, randomly cropping and resizing the images, and randomly perturbing the brightness, contrast, and hue of the images [28]. We then use the Mean Square Error (MSE) as the loss function and the Adam algorithm [21] to train ResNet50. We gradually reduce the learning rate in the training process. We use a GPU for training. We select a batch size that can fill up the available GPU memory and we shuffle the training dataset when fetching the batches.

Testing Process. We collect two different testing datasets with each testing dataset having its own purpose.

For the first testing dataset, we collect testing images in a similar manner that we collect the training images. We execute one G-code instruction at a time, wait for the instruction to complete, take a picture, use the coordinates of the printhead as the label for the picture, and then proceed to the next instruction. The main advantage of this testing dataset is that we have the ground truth label (the coordinates) for each image and we can calculate the Mean Square Error (MSE) between the predicted labels and the ground truth labels as the performance metric of the neural network.

For the second testing dataset, we perform the printing process in real time, record the printing process as a video, and extract frames in the video one by one. The main purpose of this testing dataset is to evaluate the performance of the neural network on a real printing process. For this testing dataset, we have the motion blur effect as a side effect and we do not have the ground truth coordinates for each image because it is hard to determine the G-code instruction that is executed for every millisecond. Due to a lack of ground truth coordinates, we only qualitatively measure the performance of the neural network by visually looking at the recovered printing path.

To simulate the fact that the attacker collects training images on an AM system that is not identical to the target AM system, after collecting the training dataset, we remove the camera and the printer from the experiment area and reinstall them into the experiment area, trying our best to align the camera and the printer such that the perspective of images in the testing datasets looks similar to the perspective of images in the training dataset.

VI. Optical Noise Injection

This section discusses the details of the optical noise injection method. We first discuss optical projectors. We then discuss existing and proposed noise generation algorithms.

A. Problem Definition

In this paper, we use an optical projector to generate controllable optical signals. There are other means to generate optical signals, such as light bulbs and Light-Emitting Diodes (LEDs). Compared with light bulbs and LEDs, an optical projector can generate highly controllable optical signals with a relatively large gamut.

[3]In other words, the build plate is clean for all images in the training dataset.

Control Signals. For a projector to work, a control signal, denoted by $\boldsymbol{x}_{\mathrm{CS}}$, must be provided. The control signal is typically a video file that is played by the projector. When considering the defender's projector and the attacker's camera as a system, they can be mathematically represented by

$$\boldsymbol{x}_{\mathrm{SC}}[n] = \boldsymbol{g}(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n]) + \boldsymbol{e}[n], \tag{5}$$

where $\boldsymbol{x}_{\mathrm{CS}}[n]$ is the control signal at time index n, $\boldsymbol{x}_{\mathrm{SC}}[n]$ is the optical side-channel signal at time index n, and $\boldsymbol{e}[n]$ is the noise at time index n. When $\boldsymbol{x}_{\mathrm{CS}}$ is zero, $\boldsymbol{g}$ in Eq. 5 degenerates to $\boldsymbol{f}$ in Eq. 4.

Number of Channels. $\boldsymbol{x}_{\mathrm{SC}}$, $\boldsymbol{x}_{\mathrm{SV}}$, and $\boldsymbol{x}_{\mathrm{CS}}$ typically have very different numbers of channels. Fig. 7 (a) shows an example of a frame of $\boldsymbol{x}_{\mathrm{CS}}$. The projector accepts images of 1024×768 pixels and each pixel contains 3 color values. Hence, there are $1024 \times 768 \times 3$ channels in $\boldsymbol{x}_{\mathrm{CS}}$. Fig. 7 (b) shows the corresponding image seen by the camera and the image contains 227×227 pixels and each pixel contains 3 color values. Hence, there are $227 \times 227 \times 3$ channels in $\boldsymbol{x}_{\mathrm{SC}}$. Finally, the number of channels in $\boldsymbol{x}_{\mathrm{SV}}$ is equal to the number of components in the state variable that the attacker is attempting to recover, and in this case it is 3 for x, y, and z.

Channel Transformation. Since there are 1024×768 pixels in $\boldsymbol{x}_{\mathrm{CS}}$ and 227×227 pixels in $\boldsymbol{x}_{\mathrm{SC}}$, $\boldsymbol{x}_{\mathrm{CS}}$ and $\boldsymbol{x}_{\mathrm{SC}}$ are not channel compatible. Nevertheless, for every pixel in $\boldsymbol{x}_{\mathrm{CS}}$, it may have corresponding pixels in $\boldsymbol{x}_{\mathrm{SC}}$. Conversely, for every pixel in $\boldsymbol{x}_{\mathrm{SC}}$, it may have corresponding pixels in $\boldsymbol{x}_{\mathrm{CS}}$. The correspondence between pixels in $\boldsymbol{x}_{\mathrm{CS}}$ and pixels in $\boldsymbol{x}_{\mathrm{SC}}$ is referred to as channel transformation. With channel transformation, we can express $\boldsymbol{x}_{\mathrm{CS}}$ using the same channel structure of $\boldsymbol{x}_{\mathrm{SC}}$. In the rest of the paper, we always express $\boldsymbol{x}_{\mathrm{CS}}$ with channel transformation performed. Hence, $\boldsymbol{x}_{\mathrm{CS}}$ always has the same channel structure as $\boldsymbol{x}_{\mathrm{SC}}$.

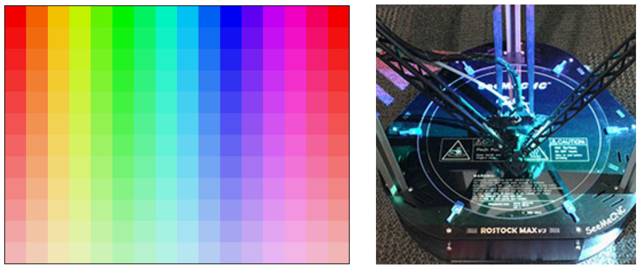
Independence of Channels. In this paper, we assume that channels are independent, although this assumption may not be strictly true due to a phenomenon called leakage. For example, when a projector sends a ray of light that is registered in a single pixel in the camera, the adjacent pixels may also be slightly affected. In this paper we neglect the leakage phenomenon and we have

$$\boldsymbol{x}_{\mathrm{SC}}[n, c] = \boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n, c]) + \boldsymbol{e}[n, c]. \tag{6}$$

In other words, $\boldsymbol{x}_{\mathrm{SC}}[n, c]$ is only affected by $\boldsymbol{x}_{\mathrm{CS}}[n, c]$. Notice the function here is $\boldsymbol{g}_c$, which takes $\boldsymbol{x}_{\mathrm{CS}}[n, c]$ as an argument. In contrast, $\boldsymbol{g}$ takes $\boldsymbol{x}_{\mathrm{CS}}[n]$ as an argument.

Capability of the Projector. As shown in Fig. 7, for any pixel in the control signal ($\boldsymbol{x}_{\mathrm{CS}}$), its corresponding pixel in the side-channel signal ($\boldsymbol{x}_{\mathrm{SC}}$) typically takes on a different color. In fact, a pixel in $\boldsymbol{x}_{\mathrm{CS}}$ can take any color in the whole RGB color space. However, the corresponding pixel in $\boldsymbol{x}_{\mathrm{SC}}$ can only reach a (narrow) subset of the whole RGB color space. We refer to the color space that can be reached by a pixel in $\boldsymbol{x}_{\mathrm{SC}}$ as the capability of the projector for that pixel.

If a projector has unlimited capability, we should be able to manipulate $\boldsymbol{x}_{\mathrm{SC}}$ in a way to fully prevent side-channel attacks. One way to do this is to make $\boldsymbol{x}_{\mathrm{SC}}$ a constant value over both



(a) Control Signal (b) Side-Channel Signal

Fig. 7: Illustration of channel transformation and the capability of the optical projector.

time and channel. However, due to the limited capability of the projector, we have to carefully design $\boldsymbol{x}_{\mathrm{CS}}$ to maximize the performance of protection.

B. Existing Noise Generation Algorithms

A noise generation algorithm refers to a systematic way to create values in $\boldsymbol{x}_{\mathrm{CS}}$ to confuse the adversarial reconstruction of the print design. Examples of images with injected noise are shown in Fig. 8.

Replaying. Suppose $\boldsymbol{x}'_{\mathrm{SC}}$ is the recorded side-channel signal of a printing process that is different from the current printing process. We let $\boldsymbol{x}_{\mathrm{CS}} = \boldsymbol{x}'_{\mathrm{SC}}$. In other words, we replay the side-channel signal of another printing process [19].

Random Blobs. This method fills the pixels in $\boldsymbol{x}_{\mathrm{CS}}$ with a circle that has a random location, a random size, and a random (uniform) color. The blob is maintained for a certain amount of time and is then changed to another random blob.

White Noise. This method uses a random number generator (of a uniform distribution) to independently fill all values in the control signal $\boldsymbol{x}_{\mathrm{CS}}$.

Full Power. This method creates $\boldsymbol{x}_{\mathrm{CS}}$ such that the optical projector constantly outputs the maximum power. To be specific, all values in $\boldsymbol{x}_{\mathrm{CS}}$ are filled with the maximum numbers. This method attempts to blind the attacker's camera.

C. Novel Noise Generation Algorithms

In this section, we present three novel noise generation algorithms. These algorithms rely on the information of the print process and hence are print-specific.

Channel Uniformization. This method attempts to make $\boldsymbol{x}_{\mathrm{SC}}$ a constant over the channel index c. In other words, this method creates a control signal such that each picture seen by the camera is of a uniform color. In this way, it becomes hard for the attacker to determine the location of the nozzle.

For a specific state (or a specific value of $\boldsymbol{x}_{\mathrm{SV}}[n]$), if the intersection of the ranges of $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (with respect to $\boldsymbol{x}_{\mathrm{CS}}[n,c]$) for all c is not empty, we can select any value in this intersection as the constant value $C[n]$. We then create $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ by numerically solving

$$\underset{\boldsymbol{x}_{\mathrm{CS}}[n,c]}{\operatorname{argmin}} |\boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n,c]) - C[n]|. \tag{7}$$

$\boldsymbol{x}_{\mathrm{CS}}[n]$ is simply a collection of $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ for all c. Both $C[n]$ and $\boldsymbol{x}_{\mathrm{CS}}[n]$ depends on the specific value of $\boldsymbol{x}_{\mathrm{SV}}[n]$.

If the aforementioned intersection is empty, we select a constant value $C[n]$ that is likely to be reached by most $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (as c is varied). Since $\boldsymbol{x}_{\mathrm{CS}}$ typically makes $\boldsymbol{x}_{\mathrm{SC}}$ brighter, we can select $C[n]$ by

$$C[n] = h(\{\boldsymbol{x}_{\mathrm{SC}}[n,c]|c\}), \tag{8}$$

where h is a function to find a large value of a set, such as the 99th percentile of the set. We do not use the maximum value of the set because it may be susceptible to outliers.

State Uniformization. When the state ($\boldsymbol{x}_{\mathrm{SV}}[n]$) changes, the image seen by the camera ($\boldsymbol{x}_{\mathrm{SC}}[n]$) changes. It is this relationship that makes it possible to infer the state from the image. This method creates a control signal ($\boldsymbol{x}_{\mathrm{CS}}$) in an attempt to make the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) a constant as the state ($\boldsymbol{x}_{\mathrm{SV}}[n]$) changes. In other words, $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of $\boldsymbol{x}_{\mathrm{SV}}[n]$ is uniformized.

For a specific channel index c, if the intersection of the ranges of $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (with respect to $\boldsymbol{x}_{\mathrm{CS}}[n,c]$) for all states (namely all values of $\boldsymbol{x}_{\mathrm{SV}}[n]$) is not empty, we can select any value in this intersection as the constant value $C[n,c]$. We then create $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ by numerically solving

$$\underset{\boldsymbol{x}_{\mathrm{CS}}[n,c]}{\operatorname{argmin}} |\boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \frac{\partial \boldsymbol{x}_{\mathrm{SV}}}{\partial t}[n], \cdots, \boldsymbol{x}_{\mathrm{CS}}[n,c]) - C[n,c]|. \tag{9}$$

$\boldsymbol{x}_{\mathrm{CS}}[n]$ is simply a collection of $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ for all c. Both $C[n,c]$ and $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ are constant with respect to $\boldsymbol{x}_{\mathrm{SV}}[n]$.

If the aforementioned intersection is empty, we select a constant value $C[n,c]$ that is likely to be reached by most $\boldsymbol{x}_{\mathrm{SC}}[n,c]$ (as $\boldsymbol{x}_{\mathrm{SV}}[n]$ is varied). Since $\boldsymbol{x}_{\mathrm{CS}}$ typically makes $\boldsymbol{x}_{\mathrm{SC}}$ brighter, we can select $C[n,c]$ by

$$C[n,c] = h(\{\boldsymbol{x}_{\mathrm{SC}}[n,c]|\boldsymbol{x}_{\mathrm{SV}}[n]\}), \tag{10}$$

where h is a function to find a large value of a set, such as the 99th percentile of the set.

Fig. 9 (a) shows the color of a single pixel in $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of the first two coordinates of $\boldsymbol{x}_{\mathrm{SV}}[n]$. For each value of $\boldsymbol{x}_{\mathrm{SV}}[n]$ or each dot in Fig. 9 (a), as the control signal for the specified pixel sweeps all of its possible values, the range of the color of the specified pixel in $\boldsymbol{x}_{\mathrm{SC}}[n]$ forms a color space. Fig. 9 (b) shows the color spaces for four different states which are notated in Fig. 9 (a) as red circles. We can see that the intersection of the color spaces is empty.

Fig. 9 (c) shows the color of the corresponding pixel in $\boldsymbol{x}_{\mathrm{CS}}[n]$ obtained by the state uniformization method and Fig. 9 (d) shows the color of the specified pixel in $\boldsymbol{x}_{\mathrm{SC}}[n]$ after the obtained $\boldsymbol{x}_{\mathrm{CS}}[n]$ is applied. We can see that the pattern is more uniform and it may be harder for the attacker to infer (x, y) based on the observed color at this specific pixel.

State Randomization. This method attempts to randomize the relationship between the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) and the state ($\boldsymbol{x}_{\mathrm{SV}}[n]$). To be specific, we create a control signal ($\boldsymbol{x}_{\mathrm{CS}}$) such that the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) for a specific state ($\boldsymbol{x}_{\mathrm{SV}}[n]$) appears to be the image ($\boldsymbol{x}_{\mathrm{SC}}[n]$) for another random state ($\boldsymbol{x}_{\mathrm{SV}}[n]$). Since the relationship between $\boldsymbol{x}_{\mathrm{SC}}[n]$ and $\boldsymbol{x}_{\mathrm{SV}}[n]$ is disrupted,

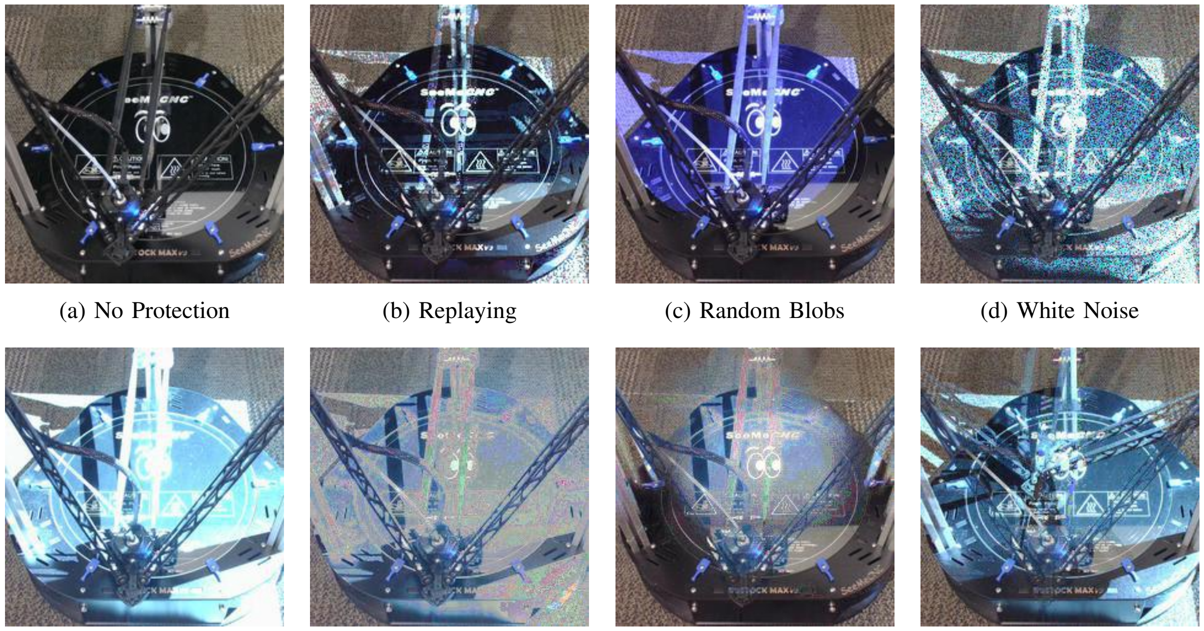

(a) No Protection (b) Replaying (c) Random Blobs (d) White Noise (e) Full Power (f) Channel Uniformization (g) State Uniformization (h) State Randomization

Fig. 8: Example images with noise generated by different noise generation algorithms.

we can expect that it will be hard for the attacker to learn the true relationship between $\boldsymbol{x}_{\mathrm{SC}}[n]$ and $\boldsymbol{x}_{\mathrm{SV}}[n]$.

Since a projector can only make pixels in $\boldsymbol{x}_{\mathrm{SC}}$ brighter, we apply a constant value of $\boldsymbol{x}_{\mathrm{CS}}[n]$ for all states (or all values of $\boldsymbol{x}_{\mathrm{SV}}[n]$). This constant value is referred to as the offset. By using the offset, it is possible for $\boldsymbol{x}_{\mathrm{CS}}$ to change $\boldsymbol{x}_{\mathrm{SC}}$ along the opposite direction with respect to the offset. We express the strength of the offset as the percentage of the maximum value. By default, we apply an offset with a strength of 50%.

We now discretize the domain of $\boldsymbol{x}_{\mathrm{SV}}[n]$ such that there is a finite number of states. By default, we discretize the domain such that all states are equally spaced. These states are referred to as master states and they serve as the targets to be matched.

For any new state $\boldsymbol{x}_{\mathrm{SV}}[n]$, we randomly select one of the master states as the target state. Suppose the target state is $\boldsymbol{x}'_{\mathrm{SV}}[n]$ and its corresponding side-channel signal is $\boldsymbol{x}'_{\mathrm{SC}}[n]$. We create $\boldsymbol{x}_{\mathrm{CS}}[n]$ such that $\boldsymbol{x}_{\mathrm{SC}}[n]$ gets close to $\boldsymbol{x}'_{\mathrm{SC}}[n]$. To be specific, we find $\boldsymbol{x}_{\mathrm{CS}}[n]$ by determining its value for each channel independently, and $\boldsymbol{x}_{\mathrm{CS}}[n,c]$ is equal to

$$\underset{\boldsymbol{x}_{\mathrm{CS}}[n,c]}{\operatorname{argmin}} |\boldsymbol{g}_c(\boldsymbol{x}_{\mathrm{SV}}[n], \boldsymbol{x}_{\mathrm{CS}}[n,c]) - \boldsymbol{x}'_{\mathrm{SC}}[n,c]|. \tag{11}$$

We repeat this process for all channels and we can obtain $\boldsymbol{x}_{\mathrm{CS}}[n]$. In this way, $\boldsymbol{x}_{\mathrm{SC}}[n] = \boldsymbol{g}(\boldsymbol{x}_{\mathrm{SV}}[n], \cdots, \boldsymbol{x}_{\mathrm{CS}}[n])$ looks like $\boldsymbol{x}'_{\mathrm{SC}}[n,c]$. When the attacker analyzes $\boldsymbol{x}_{\mathrm{SC}}[n]$, he or she will more likely infer $\boldsymbol{x}'_{\mathrm{SV}}[n]$ instead of $\boldsymbol{x}_{\mathrm{SV}}[n]$.

This method is effectively a combination of channel uniformization and an advanced version of replaying. It attempts to hide the real printhead by channel uniformization and creates a fake printhead as best as the projector allows.

VII. Evaluation

A. Experiment Setup

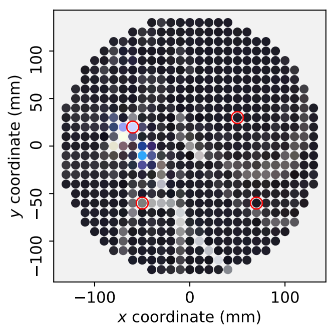
To evaluate the performance of the proposed optical side-channel attack and the proposed defense method, we set up a test bed as shown in Fig. 10. The test bed was composed of a SeeMeCNC Rostock Max V3 printer, a MOKOSE UC70 camera, and a ViewSonic Pro7827HD projector. The whole test bed was placed in a dedicated room with fully controlled lights (two lamps and no window).

Settings. To ensure the consistency of the images, we disabled auto white balance, auto exposure, and auto focus in the camera. We manually set the white balance to 4600 K and the exposure to -2. The focus length of the camera was mechanically adjusted to clearly show the build plate of the printer. We used default settings for all other parameters in the camera and all parameters in the projector.

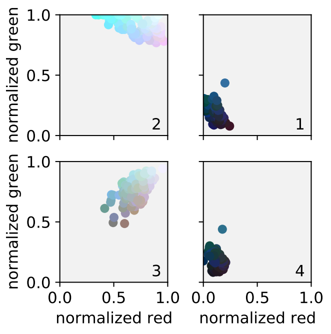
B. Optical Side-Channel Attack

Training Dataset. We collected images in the training dataset according to the procedures outlined in Section V-C. The printing area of the printer was a cylinder of 137.5 mm in radius and 404 mm in height. Since most printing processes only involved a fraction of the whole printing space, to save time, we mainly focused on the printing area where z was less than or equal to 5 mm.

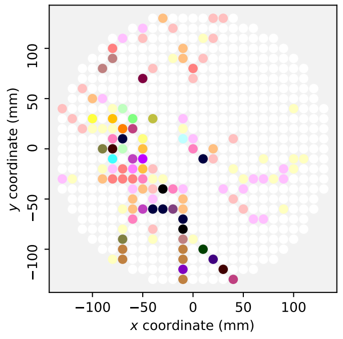
There are two parts of the training dataset. For the first part, we instructed the printhead to sweep the whole xy plane with a step size of 2.5 mm and the z axis with a step size of 1.0 mm. This part covers a large printing space with a coarse resolution. There are 133,365 images in this part. For the second part, we instructed the printhead to sweep the xy plane for a radius of



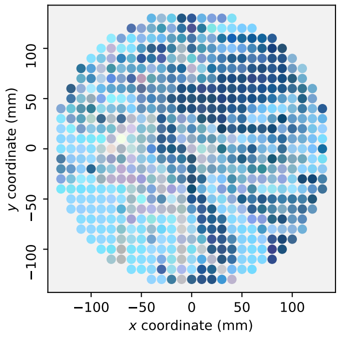
(a) Side-Channel Signal



(b) Color Spaces



(c) Control Signal



(d) Side-Channel Signal

Fig. 9: Illustration of the state uniformization method. (a) A pixel of $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of $\boldsymbol{x}_{\mathrm{SV}}[n, 0:2]$. (b) Color spaces for four different states. The four color spaces approximately correspond to the four states with red circles in (a). (c) The control signal obtained by the state uniformization method. (d) A pixel of $\boldsymbol{x}_{\mathrm{SC}}[n]$ as a function of $\boldsymbol{x}_{\mathrm{SV}}[n, 0:2]$ after the control signal is applied.

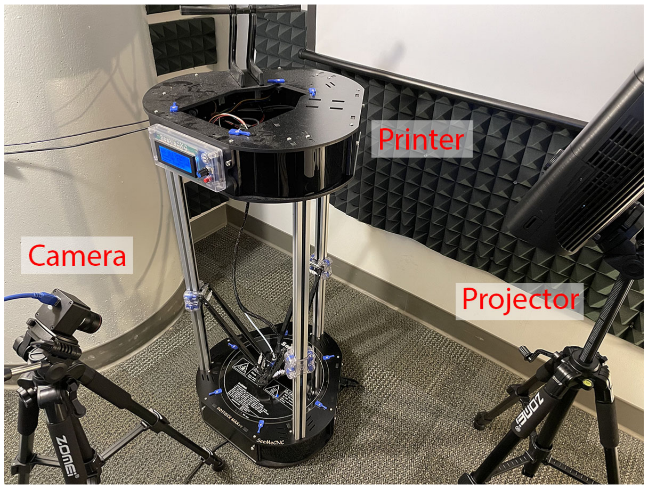


Fig. 10: Photo of the testbed.

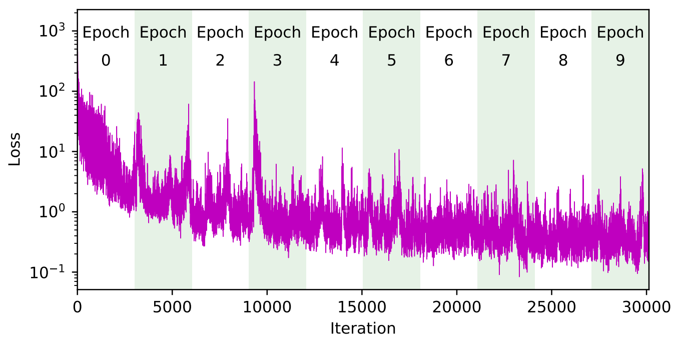


Fig. 11: Loss as a function of iteration or epoch.

$r = 50$ mm with a step size of 1.0 mm and the z axis with a step size of 0.25 mm. This part covers a small printing space with a high resolution. There are 47,385 images in this part. In total, there are 180,750 images in the training dataset.

Each image in the training dataset is of size 1280x720 but the neural network accepts images of size 227x227. We performed the following procedures to downsample and augment images in the training dataset:

1) Resize the image into a shape of 480x270.
2) Randomly rotate the image by -3 to 3 degrees.
3) Scale the image randomly by a factor of 0.85 to 1.0.
4) Randomly crop the image with a size of 227x227.
5) Randomly change the brightness, contrast, saturation, and hue by 20%, 10%, 10%, and 5% respectively.

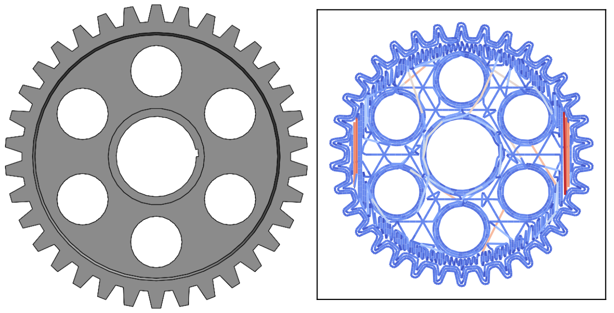
Training Process. We started with a ResNet50 network that was pretrained on the ImageNet dataset [23]. We performed training on an RTX 2060 GPU with 6 GB of memory. We used a mini batch size of 40 since it fills up most of the available memory. We used the Mean Square Error (MSE) as the loss function and we used the Adam algorithm [21] for training. The initial learning rate was set to 0.001 and the weight decay was set to 0.0005. For every two epochs, we reduced the learning rate by 10%. We performed the training process for 10 epochs. The loss as a function iteration/epoch is shown in Fig 11. We can see that the error converged and the training process was successful.

Testing Datasets. By the procedures in section V-C, we collected two testing datasets for a printing process that manufactured a gear, as shown in Fig. 12 (a). To avoid multiple superimposed layers when displaying the results, we focus on a single layer ($z = 1.3$ mm) for the printing process. Fig. 12 (b) shows the ground truth of the printing path at this layer.

Testing Results. Fig. 12 (c) shows the recovered path for the first testing dataset (a contrived printing process with ground truth coordinates). The average error is 0.71 mm and the maximum error is 1.50 mm. The recovered path is very close to the ground truth. Fig. 12 (d) shows the recovered path for the second testing dataset (a real printing process with the motion blur effect as a side effect). The shape of the path is very similar to the ground truth. This verifies that the motion blur effect can be neglected in our experiments.

C. Defending Against the Naive Attacker

We applied the seven noise generation algorithms to the printing process, one at a time, as demonstrated in Fig. 8. Since the naive attacker was not aware of the injected noise, for each noise generation algorithm, the attacker used the previously


(a) Gear Model (b) Ground Truth

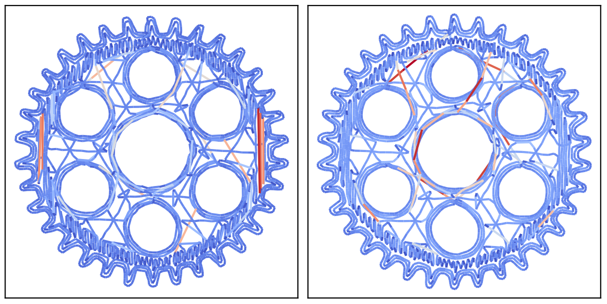

(c) Testing Dataset 1 (d) Testing Dataset 2

Fig. 12: Model and paths recovered by the neural network for different testing datasets. (a) The gear model. (b) The ground truth path. (c) The recovered path for testing dataset 1 (a printing process with ground truth coordinates). (d) The recovered path for testing dataset 2 (a real printing process).

TABLE I: Performance Metrics

Noise Generation Algorithm	Naive Attack Mean Error	Naive Attack Max Error	Advanced Attack Mean Error	Advanced Attack Max Error
No Protection	0.71	1.50	0.71	1.50
Replaying	12.08	23.43	4.06	5.84
Random Blobs	12.11	102.04	4.19	5.58
White Noise	39.88	63.11	4.08	5.91
Full Power	67.49	104.97	5.12	7.36
Channel Uniformization	110.51	145.79	4.47	7.51
State Uniformization	90.44	117.21	4.57	7.40
State Randomization	31.86	76.20	5.48	15.35

The units of errors are in mm.

trained neural network to directly test on the protected testing dataset[4]. The results and the errors are shown in Fig. 13. We see that all noise generation algorithms defeated the attack since the recovered paths are nothing like the true path.

D. Defending Against the Advanced Attacker

An advanced attacker knows the existence of the injected noise and attempts to defeat the protection method. For this purpose, the advanced attacker collects training images with injected noise[5] and uses the new training dataset to train the neural network, hoping that the neural network can recognize the noise pattern, reject the injected noise, and properly recover the coordinates in the testing images.

For noise generation algorithms that contain randomness, such as replaying, random blobs, white noise, and state randomization, the details of the injected noise in the training dataset are different from those in the printing process to be recovered (the protected testing dataset), such as locations and sizes for random blobs, target states for replaying and state randomization, and the exact distribution of pixels for white noise. A neural network that is overfitted will not succeed in rejecting the noise, and the neural network has to learn the pattern of the injected noise and reject unseen noise that has the same pattern but with different details.

Due to the high cost of obtaining images with injected noise, we only considered a training dataset (with injected noise) that covered the whole xy plane with a step size of 5 mm and covered the z axis for a single point, namely z = 5 mm. The advanced attacker used the new training dataset to further train the previously trained neural network for additional 30 epochs, starting with a learning rate of 0.001, which was decreased by 10% for every two epochs. The advanced attacker then used the new neural network to test on images from the protected testing dataset. The results are shown in Fig. 14.

We can see that the advanced attacker did a much better job than the naive attacker in recovering the state-variable signal from the optical side-channel signal. Although the errors in Figs. 14 (b) to (g) are higher than those in Fig. 14 (a), a lot of details are clearly recovered such as the number of teeth in the gear, the overall dimension of the gear, and the infill pattern. In Fig. 14 (h), we can also see that state randomization algorithm was the only noise generation algorithm that could withstand the advanced side-channel attack.

VIII. Discussion

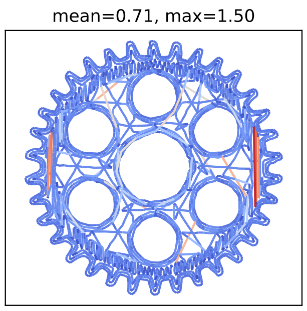
A. Performance of the Attackers

When no noise was injected, the naive attacker could do a pretty good job in recovering the coordinates of the printhead from the optical side-channel signal. Although the neural network was trained on a dataset where the printhead was located in a limited number of locations, the neural network could properly identify the coordinates of the printhead for a location that was not present in the training dataset. In other words, the neural network could intelligently interpolate with respect to the coordinates of the printhead. When noise was injected, the naive attacker failed all together. This indicates that the neural network was unable to analyze images with patterns that it had never seen before.

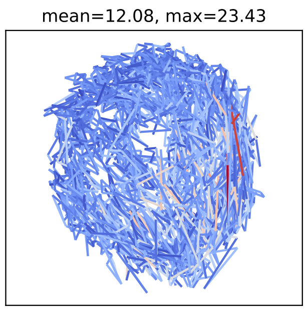
One may notice that the colors in Fig. 12 (a) and Fig. 12 (b) do not match. The colors in the paths show the speed for each movement. A blue color corresponds to a lower printing speed whereas a red color corresponds to a higher printing

[4]To obtain the performance metrics, we collected the protected testing datasets using the same procedure to collect the training dataset or the first testing dataset.

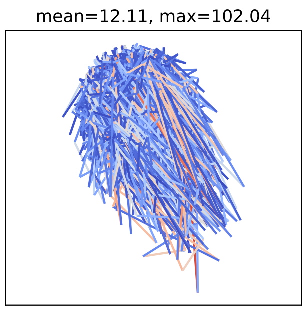
[5]To collect training images with injected noise, the attacker may setup a testbed with a projector, imitate the noise generation algorithm, and collect the training images at various locations of the nozzle. The attacker may also extract images with injected noise from previous printing processes and somehow, although very hard, manage to obtain the correct label for each image.



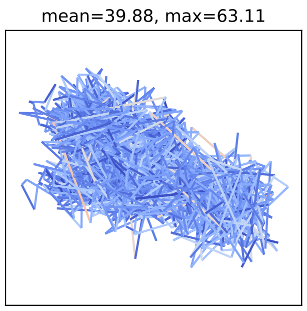
(a) No Protection



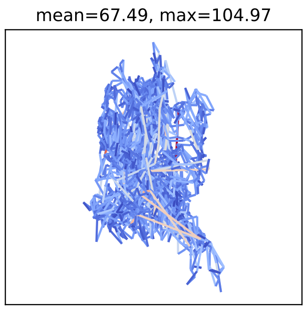
(b) Replaying



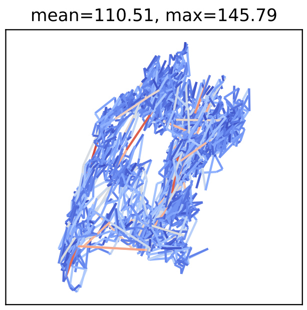
(c) Random Blobs



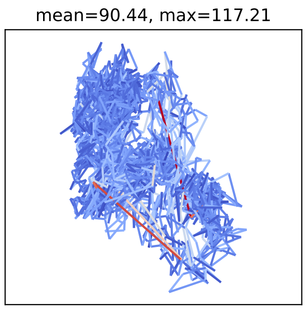
(d) White Noise



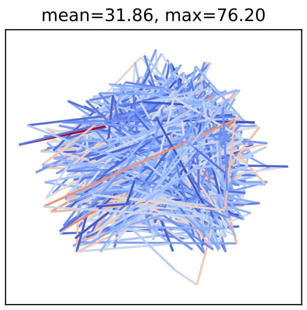
(e) Full Power



(f) Channel Uniformization

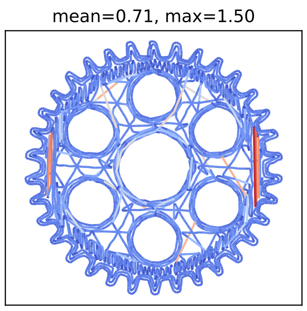


(g) State Uniformization

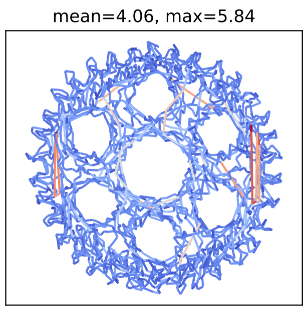


(h) State Randomization

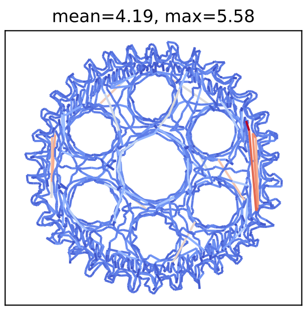
Fig. 13: Recovered path by a naive attacker when noise generated by various algorithms was applied. The title in each figure shows the mean and maximum errors in mm. The naive attacker was not aware of the protection methods and used a neural network that was never exposed to any injected noise. We see that all noise generation algorithms effortlessly defeated the attack.



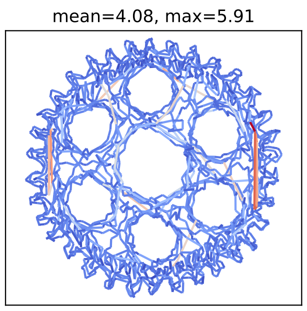
(a) No Protection



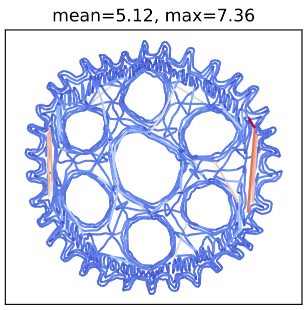
(b) Replaying



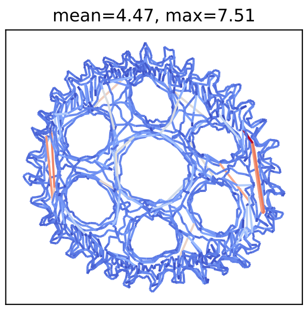
(c) Random Blobs



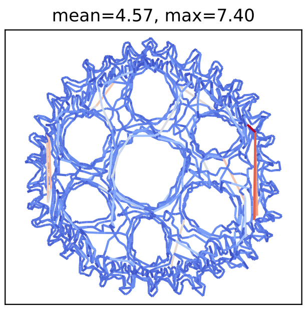
(d) White Noise



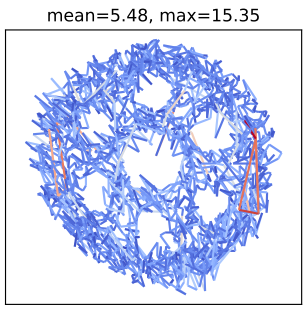
(e) Full Power



(f) Channel Uniformization



(g) State Uniformization



(h) State Randomization

Fig. 14: Recovered path by an advanced attacker when noise generated by various algorithms was applied. The advanced attacker was aware of the protection method. For each noise generation algorithm, the attacker used images that contain injected noise by the same noise generation algorithm to further train the neural network. We can see that the advanced attacker could still defeat most of the noise generation algorithms.

speed. The colors in the paths do not match because we used a simulator to determine the speed for each movement in the ground truth (and the first testing dataset), whereas the speed in the second testing dataset was determined by the firmware in the printer. Currently, there is a noticeable error between the simulated velocity and the actual velocity.

According to experiment results by the advanced attacker, by seeing a limited number of images with injected noise, the neural network was able to filter out the injected noise and proceed with recognizing the coordinates of the printhead, even though at a reduced accuracy. Notice that for noise generation algorithms with randomness, the details of the injected noise in the training dataset were different from those in the protected testing dataset, but the neural network could still recognize and reject the injected noise. This proves that the neural network was not simply remembering the details of the injected noise but intelligently learned the pattern of the injected noise.

B. Performance of the Noise Generation Algorithms

According to the evaluation results, only the state randomization algorithm could fight against the advanced attacker. One way to intuitively understand the performance of different noise generation algorithms is to visually inspect the images after the noise is injected. For images in Fig. 8 (b) to (g), a human being can in general see the location of the nozzle despite the existence of the injected noise.

Full power, channel uniformization, and state uniformization were unable to protect the intellectual property due to the limited capability of the projector. We can expect that, as the projector gets more powerful, the full power noise will blind the camera and the uniformization methods will completely hide the details.

The replaying method failed mainly due to the fact that it did not consider the channel effect caused by the projector, the physical space, and the projector. When a recorded video is directly fed to a projector, the projected motion picture will look different from the real thing. In other words, a human being is able to differentiate the projected nozzle from the real nozzle in Fig. 8 (b).

To mitigate the problem faced by the replaying method, the state randomization algorithm searches for the control signal such that the projected nozzle looks closest to a real nozzle. In addition, the state randomization algorithm attempts to hide the real nozzle similar to what is attempted by the state uniformization algorithm.

C. Limitations and Future Work

In this paper, we were able to ignore the motion blur effect by using a camera with a fast shutter speed. However, when the ambient light is low, a camera will suffer from a low shutter speed and we can no longer ignore the motion blur effect. To mitigate this problem, as a direction for future work, we can create a training dataset where the label of each image contains not only the position but also the velocity. The main challenge is that a lot more samples are required as the degrees of freedom is now six instead of three. In addition, it is very hard to precisely control the position and the velocity at the same time for each training image.

In this paper, the position and angle of the camera largely remained stable. The neural network was able to deal with a small amount of variation in the position and angle of the camera. It will be interesting to know if it is possible to train a single neural network that can recognize the coordinates of the printhead from multiple very different positions and angles of the camera, especially from a position or an angle that does not appear in the training dataset.

In this paper, the printed object was small in size and was completely blocked by the printhead. As a result, we were able to ignore the influence of deposited materials on the side-channel attack. When printing a large object, the deposited materials may affect the side-channel attack. Future work can study how the deposited materials affect the performance of both the naive attacker and the advanced attacker.

IX. Conclusion

In this paper, we proposed an optical side-channel attack to steal intellectual property in AM systems. The intellectual property is defined as the coordinates of the printhead as a function of time, aka the state-variable signal. We used a deep neural network to implement the side-channel attack. The neural network takes as input an image of the printer and outputs the estimated coordinates of the printhead. The neural network can tolerate a certain level of variation in the camera's position and angle as well as lighting conditions. The neural network contains a certain level of intelligence because it can accurately determine the coordinates of the printhead for an image that is not present in the training dataset.

We experimented with the noise injection method to defend against the proposed optical side-channel attack. We found that any noise generation algorithm can easily defeat a naive attacker, who has no knowledge of the defense method and the trained neural network is never exposed to any image with injected noise. However, an advanced attacker, who knows the existence of the defense method and attempts to filter out the injected noise by adding images with injected noise in the training dataset, can easily defeat existing noise generation algorithms, such as relaying, random blobs, white noise, and full power. To solve this problem, we proposed three novel noise generation algorithms and they are channel uniformization, state uniformization, and state randomization. Our experiment results indicate that only the state randomization algorithm can with standard the attack by an advanced attacker.

Acknowledgment

This project is supported by National Science Foundation (NSF) under Grant CPS-1739259 and CPS-1931977.

References

[1] Y. AbouHashem, M. Dayal, S. Savanah, and G. Štrkalj, "The application of 3D printing in anatomy education," *Medical Education Online*, vol. 20, no. 1, p. 29847, Jan. 2015. [Online]. Available: https://www.tandfonline.com/doi/full/10.3402/meo.v20.29847

[2] M. A. Al Faruque, S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic side-channel attacks on additive manufacturing systems," in *Proceedings of the 7th International Conference on Cyber-Physical Systems*, ser. ICCPS '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 19:1–19:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2984464.2984483

[3] M. A. Al Faruque, S. R. Chhetri, S. Faezi, and A. Canedo, "Forensics of thermal side-channel in additive manufacturing systems," in *CECS Technical Report No.16-01*, 2016.

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proceedings of the 26th USENIX Conference on Security Symposium*, ser. SEC'17. USA: USENIX Association, 2017, p. 1093–1110.

[5] ASTM, "Standard terminology for additive manufacturing – general principles – terminology," ASTM International, West Conshohocken, PA, Standard ISO/ASTM 52900:2015(E), 2015.

[6] C. Bayens, T. Le, L. Garcia, R. Beyah, M. Javanmard, and S. Zonouz, "See no evil, hear no evil, feel no evil, print no evil? Malicious fill patterns detection in additive manufacturing," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1181–1198. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/bayens

[7] S. Bradshaw, A. Bowyer, and P. Haufe, "The intellectual property implications of low-cost 3D printing," *Scriptorium*, vol. 7, pp. 5–31, 2010.

[8] D. D. Camacho, P. Clayton, W. J. O'Brien, C. Seepersad, M. Juenger, R. Ferron, and S. Salamone, "Applications of additive manufacturing in the construction industry – a forward-looking review," *Automation in Construction*, vol. 89, pp. 110–119, 2018.

[9] S. R. Chhetri, A. Canedo, and M. A. Al Faruque, "KCAD: Kinetic cyber-attack detection method for cyber-physical additive manufacturing systems," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 11 2016, pp. 1–8.

[10] S. R. Chhetri, S. Faezi, and M. A. Al Faruque, "Fix the leak! An information leakage aware secured cyber-physical manufacturing system," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 3 2017, pp. 1408–1413.

[11] ——, "Information leakage-aware computer-aided cyber-physical manufacturing," *Trans. Info. For. Sec.*, vol. 13, no. 9, pp. 2333–2344, 9 2018. [Online]. Available: https://doi.org/10.1109/TIFS.2018.2818659

[12] C. Clarke, "Create it REAL creates encryption platform for 3D printers to prevent intellectual property losses." [Online]. Available: https://3dprintingindustry.com/news/create-real-creates-encryption-platform-3d-printers-prevent-intellectual-property-losses-115772/

[13] T. J. Coogan and D. O. Kazmer, "Bond and part strength in fused deposition modeling," *Rapid Prototyping Journal*, vol. 23, no. 2, pp. 414–422, 2017.

[14] S. Curran, P. Chambon, R. Lind, L. Love, R. Wagner, S. Whitted, D. Smith, B. Post, R. Graves, C. Blue, J. Green, and M. Keller, "Big area additive manufacturing and hardware-in-the-loop for rapid vehicle powertrain prototyping: A case study on the development of a 3-D printed shelby cobra," *SAE Technical Paper*, 2016.

[15] Q. Do, B. Martini, and K. R. Choo, "A data exfiltration and remote exploitation attack on consumer 3D printers," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2174–2186, 10 2016.

[16] Y. Gao, B. Li, W. Wang, W. Xu, C. Zhou, and Z. Jin, "Watching and safeguarding your 3D printer: Online process monitoring against cyber-physical attacks," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 108:1–108:27, 9 2018. [Online]. Available: http://doi.acm.org/10.1145/3264918

[17] J. Gatlin, S. Belikovetsky, S. B. Moore, Y. Solewicz, Y. Elovici, and M. Yampolskiy, "Detecting sabotage attacks in additive manufacturing using actuator power signatures," *IEEE Access*, pp. 1–1, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[19] A. Hojjati, A. Adhikari, K. Struckmann, E. Chou, T. N. Tho Nguyen, K. Madan, M. S. Winslett, C. A. Gunter, and W. P. King, "Leave your phone at the door: Side channels that reveal factory floor secrets," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 883–894. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978323

[20] J. F. Hornick and K. Rajan, "Chapter 16 - intellectual property in 3D printing and nanotechnology," in *3D Bioprinting and Nanotechnology in Tissue Engineering and Regenerative Medicine*, L. G. Zhang, J. P. Fisher, and K. W. Leong, Eds. Academic Press, 2015, pp. 349–364.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[22] B. Krassenstein, "Five different 3D printed gun models have been fired since may, 2013." [Online]. Available: https://3dprint.com/14636/3d-prnted-guns/

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, 5 2017. [Online]. Available: https://doi.org/10.1145/3065386

[24] R. Liu, Z. Wang, T. Sparks, F. Liou, and J. Newkirk, "Aerospace applications of laser additive manufacturing," in *Laser Additive Manufacturing*, ser. Electronic and Optical Materials, M. Brandt, Ed. Woodhead Publishing, 2017, pp. 351–371.

[25] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Berlin, Heidelberg: Springer-Verlag, 2007.

[26] T. Mativo, C. Fritz, and I. Fidan, "Cyber acoustic analysis of additively manufactured objects," *The International Journal of Advanced Manufacturing Technology*, vol. 96, no. 1, pp. 581–586, 4 2018. [Online]. Available: https://doi.org/10.1007/s00170-018-1603-z

[27] S. B. Moore, J. Gatlin, S. Belikovetsky, M. Yampolskiy, W. E. King, and Y. Elovici, "Power consumption-based detection of sabotage attacks in additive manufacturing," *CoRR*, vol. abs/1709.01822, 2017. [Online]. Available: http://arxiv.org/abs/1709.01822

[28] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 12 2019.

[29] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu, "My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3D printers," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 895–907. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978300

[30] L. Tingen, S. Juhngperng, and Y. Kerwei, "Application of cascade-connected α-β filter to diminish the drifting effect from integration," in *27th Chinese Control Conference*, 2008, pp. 657–661.

[31] W. Turton, "Hackers breach thousands of security cameras, exposing Tesla, jails, hospitals," 3 2021. [Online]. Available: https://www.bloomberg.com/news/articles/2021-03-09/hackers-expose-tesla-jails-in-breach-of-150-000-security-cams

[32] A. Vanderploeg, S.-E. Lee, and M. Mamp, "The application of 3D printing technology in the fashion industry," *International Journal of Fashion Design, Technology and Education*, vol. 10, no. 2, pp. 170–179, 2017.

[33] M. Weinberg, "It will be awesome if they don't screw it up: 3D printing, intellectual property, and the fight over the next great disruptive technology," *Public Knowledge*, 11 2010. [Online]. Available: https://www.publicknowledge.org/blog/it-will-be-awesome-if-they-dont-screw-it-up-3d-printing/

[34] P. Wright, "The next big fight: 3D printing and intellectual property," *Technology Law Source*, 1 2014. [Online]. Available: https://www.technologylawsource.com/2014/01/articles/intellectual-property-1/the-next-big-fight-3d-printing-and-intellectual-property/

[35] M. Wu, Z. Song, and Y. B. Moon, "Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods," *Journal of Intelligent Manufacturing*, vol. 30, no. 3, pp. 1111–1123, 3 2019. [Online]. Available: https://doi.org/10.1007/s10845-017-1315-5

[36] M. Yampolskiy, T. R. Andel, J. T. McDonald, W. B. Glisson, and A. Yasinsac, "Intellectual property protection in additive layer manufacturing: Requirements for secure outsourcing," in *Proceedings of the 4th Program Protection and Reverse Engineering Workshop on 4th Program Protection and Reverse Engineering Workshop (PPREW 4)*. New Orleans, LA, USA: ACM Press, 2014, pp. 1–9. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2689702.2689709

[37] D. A. Zopf, S. J. Hollister, M. E. Nelson, R. G. Ohye, and G. E. Green, "Bioresorbable airway splint created with a three-dimensional printer," *New England Journal of Medicine*, vol. 368, no. 21, pp. 2043–2045, 2013, pMID: 23697530. [Online]. Available: https://doi.org/10.1056/NEJMc1206319

APPENDIX

A. Attack on More Objects

Due to the limitation of space in the paper, when discussing the optical side-channel attack, we only presented results for the gear object. Fig. 15 and Fig. 16 respectively show the ground truth paths and the recovered paths for eight different objects. Specifically, we performed our optical side-channel attack for a real printing process to obtain paths in Fig. 16. We can see that our optical side-channel attack can recover the paths for a variety of objects with different sizes.

B. Decomposition of Layers

Due to the limitation of space in the paper, we focused on a single layer for the gear object. Figs. 17 and 18 respectively show the ground truth paths and the recovered paths of the gear object for multiple layers. Specifically, we performed our optical side-channel attack for a real printing process to obtain paths in Fig. 16. We can see that our optical side-channel attack can recover the paths for different layers.

C. Advanced Attack on a Key

Due to the limitation of space in the paper, we focused on the gear object when evaluating the performance of various noise generation algorithms. Fig. 19 shows the recovered paths by an advanced attacker for a printing process that manufactures a key with various types of injected noise applied. We can see that only the state randomization algorithm can protect the path from being recovered by the attacker.

D. Offset in State Randomization

We performed a series of experiments to understand how the offset strength affects the performance of the state randomization algorithm in defending against an advanced attacker. The results are shown in Fig. 20. We can see that the errors increase significantly until the offset reaches 50% from 0%. We use an offset of 50% since it has a relatively high ratio of performance over price (power consumption).

E. Generalization to Other Side Channels

The noise injection method can be generalized to other side channels. For each type of side channels, there is a corresponding signal generator, as demonstrated in table II. Most of the noise generation algorithms can be directly applied to other side channels. The only exception is the random blobs generation algorithm, which is only applicable to the optical side channel. We believe that noise generation algorithms that are effective in fighting against the optical side-channel attack will also be effective in fighting against other side-channel attacks. This is because the optical side-channel attack is an attack that can be easily launched but hard to defend. In contrast, acoustic side-channel attacks cannot be performed at all unless a lot of restrictive assumptions are made. More details can be found in the next subsection.

TABLE II: Recorders and Generators

Side Channels	Sensors	Generators
Acoustic	Microphone	Speaker
Optical	Camera	Projector
Thermal	Temperature Sensor	Heater
Magnetic	Magnetometer	Coil
Power	Power Sensor	Resistor
Electromagnetic	Antenna	Antenna
Vibration	Accelerometer	Motor

F. More Discussion on the Acoustic Side-Channel Attack

The acoustic side-channel attack has its unique challenges.

Integration Drift. The acoustic side-channel signal is strongly correlated with the velocity of the printhead and an attacker can only recover the velocity of the printhead from the acoustic side-channel signal. To obtain the position, the attacker must integrate the estimated velocity over time. Since the estimated velocity usually contains a lot of errors, there will be more errors in the estimated position. This problem is known as the integration drift problem [30].

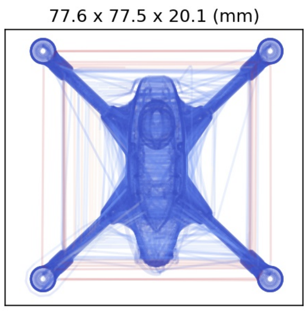
Non-Unique Solution. Multiple states of the printer can generate the same side-channel signal. As a result, it becomes very hard to determine the exact state for a given side-channel signal. For example, the acoustic signals for a printhead moving along one direction and its opposite direction are almost identical.

Due to these challenges, it is very hard to perform the acoustic side-channel attack. Existing acoustic side-channel attacks rely on a lot of assumptions to make it possible, and as a consequence they only work for a very limited amount of contrived printing processes. For example, the attack in [2], [29] restricts the directions of movements to be eight cardinal directions as a way to reduce the search area. The attack in [2] further assumes that multiple layers have identical outlines and the recovered paths are averaged to reach a reasonable accuracy. The attack in [19] assumes that there are no short and rapid movements. Otherwise, it is not possible to identify the boundaries between constant-speed movements.

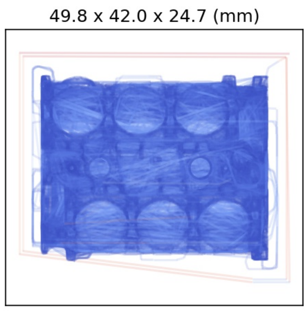
In contrast, the optical side-channel attack does not face these problems and can recover the printing path for an arbitrary printing process. This is the primary reason we focused on the optical side-channel attack in this paper.

G. More Discussion on the Infrared Side-Channel Attack

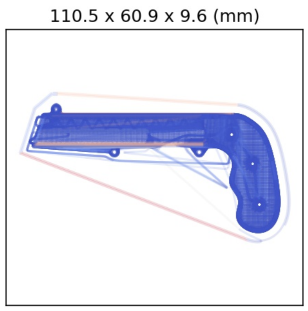
The authors in [3] claimed that their infrared side channel failed due to a low resolution, a low frame rate, a lack of auto-focus capability, and a single camera perspective. However, it is unlikely that they were the real reasons since our optical side-channel attack faced the same problem and we have a lower resolution (227x227). It is more likely that a root cause of failure is a lack of proper methodology. There is a great potential to adapt our optical side-channel attack to the infrared side-channel attack and we may even recover more information such as the temperature(s) of the nozzle(s) and the temperature of the build plate.



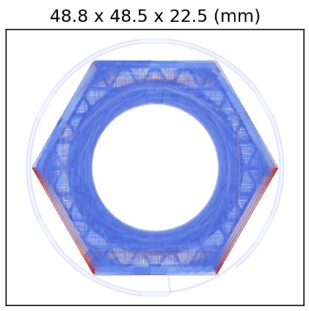
(a) Drone



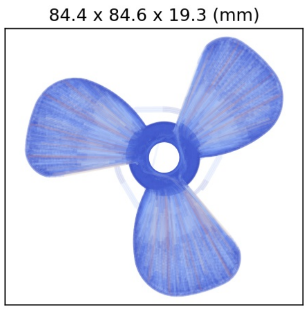
(b) Engine



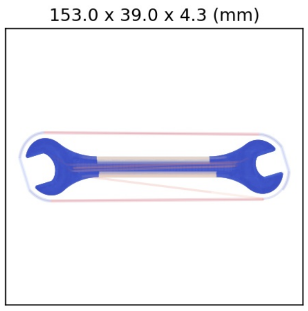
(c) Gun



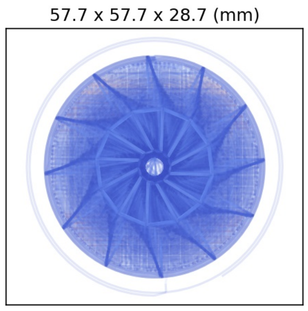
(d) Nut



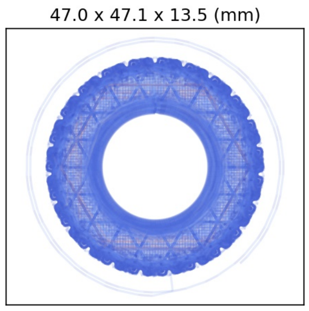
(e) Propeller



(f) Spanner

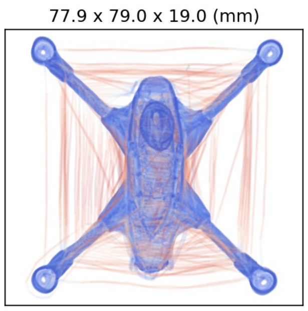


(g) Turbine

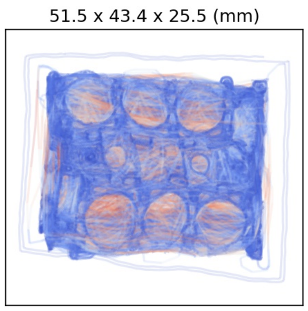


(h) Tyre

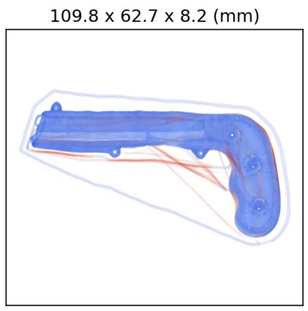
Fig. 15: Ground truth paths for eight different objects. The paths were obtained by analyzing the G-code instructions for each object. We stacked all layers together. To prevent layers from completely blocking each other, we set the opacity level (the alpha level) for each line segment to be 15%. The title of each sub-figure shows the spans of the path along the x, y, and z axes.



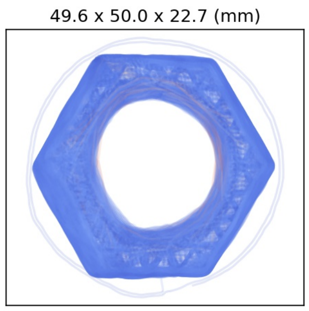
(a) Drone



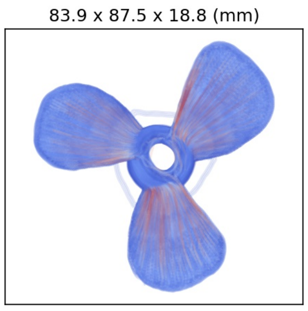
(b) Engine



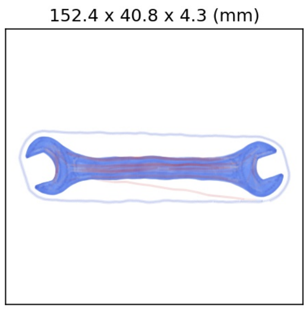
(c) Gun



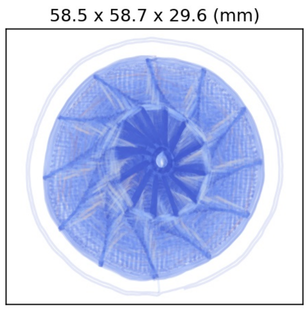
(d) Nut



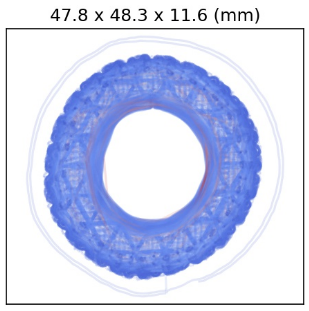
(e) Propeller



(f) Spanner



(g) Turbine



(h) Tyre

Fig. 16: Recovered paths by our optical side-channel attack for the eight different objects. Notice the range of the objects along the z axis. In the training dataset, we mainly focused on layers at or below 5 mm and we had very limited samples for higher layers. Nevertheless, the neural network did a good job recovering paths for higher layers. Some objects were not recovered with a high precision because they were relatively tall. Overall, our optical side-channel attack works well for a range of objects.

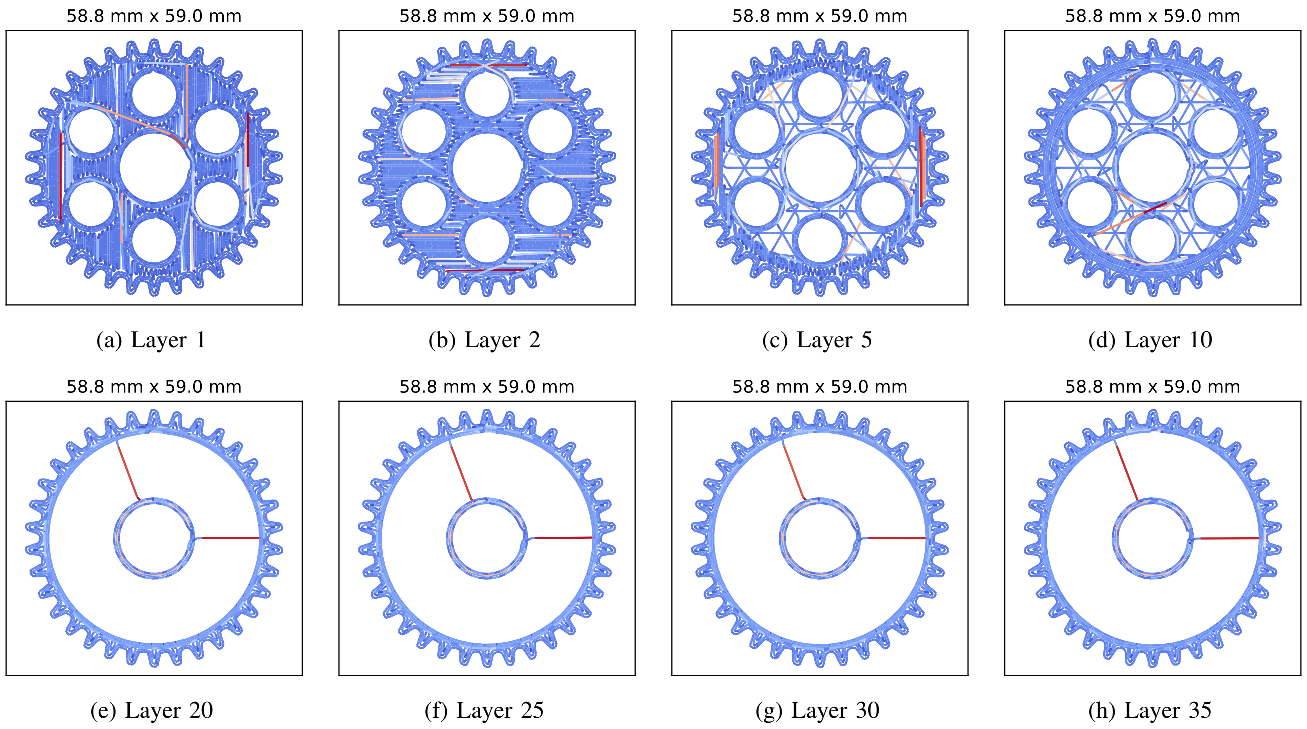


Fig. 17: Ground truth paths of the gear object for eight different layers. The title of each sub-figure shows the spans of the path along the x and y axes. The color represents the relatively speed of each movement.

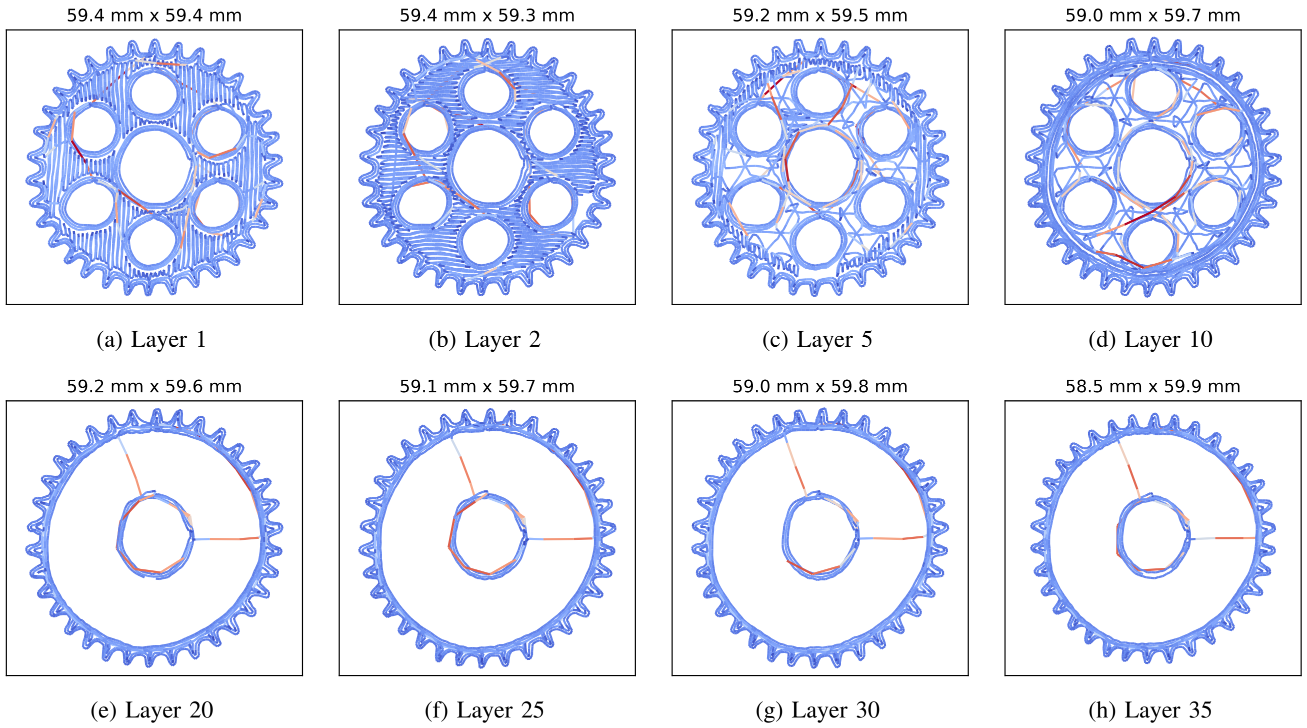
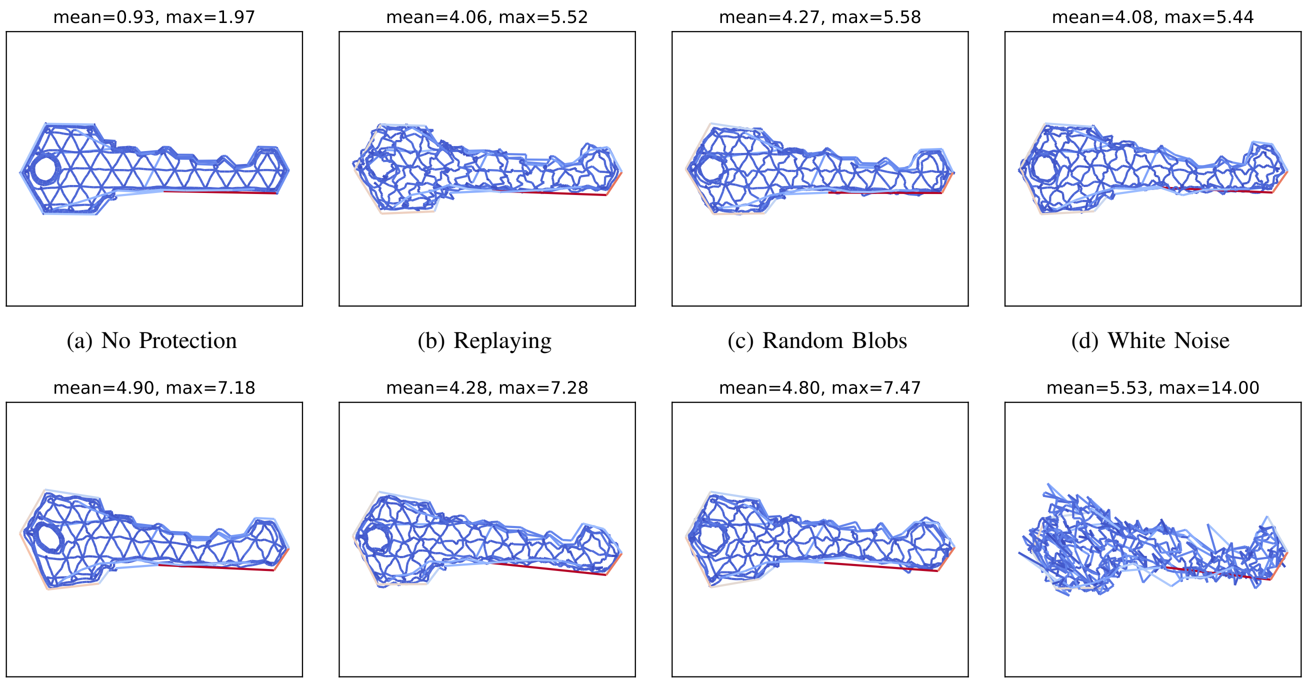


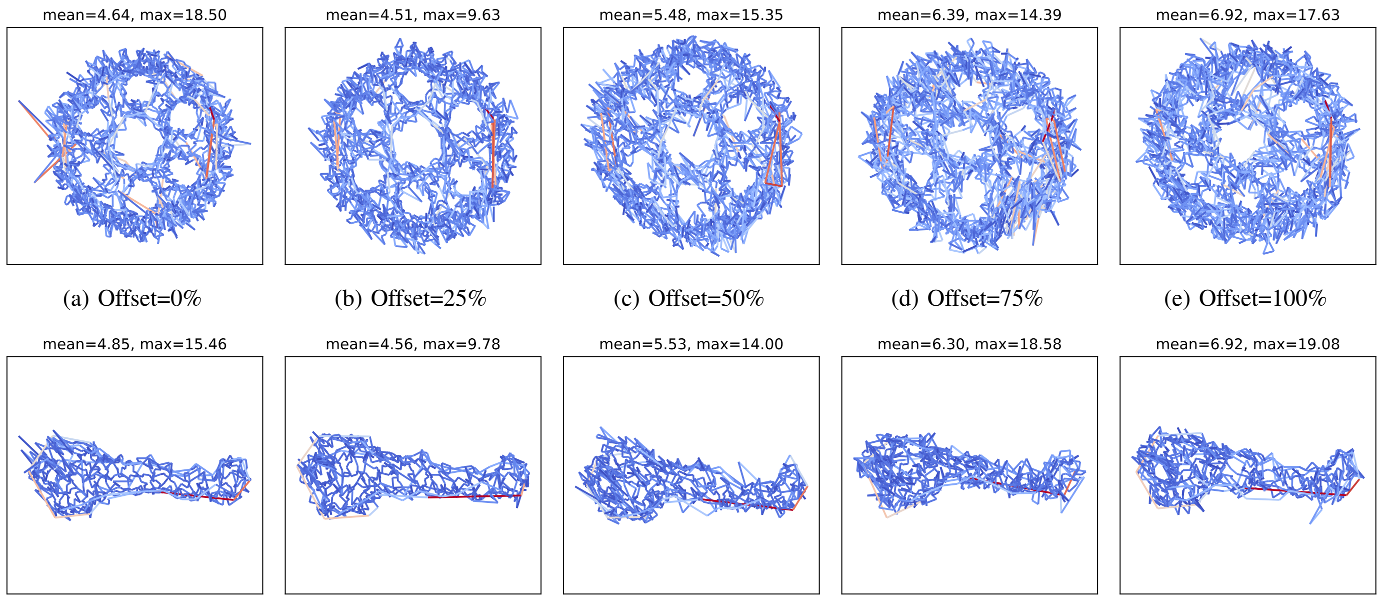
Fig. 18: Recovered paths of the gear object by our optical side-channel attack for eight different layers. We can see that our optical side-channel attack can recover the path of an object for multiple layers. We estimated the layer changing moments based on the layer changing moments in the ground truth paths. Since the simulation was not perfect, the layer changing moments were not perfectly estimated. As a result, you may see missing or extra parts in a layer, such as the missing part in (b).



(a) No Protection (b) Replaying (c) Random Blobs (d) White Noise

(e) Full Power (f) Channel Uniformization (g) State Uniformization (h) State Randomization

Fig. 19: Recovered paths by the advanced attacker on a printing process that manufactures a key. The title in each sub-figure shows the average and maximum errors between the reconstructed paths and the ground truth paths. The state randomization algorithm is the most effective noise generation algorithm to hide details of the printed object.



(a) Offset=0% (b) Offset=25% (c) Offset=50% (d) Offset=75% (e) Offset=100%

(f) Offset=0% (g) Offset=25% (h) Offset=50% (i) Offset=75% (j) Offset=100%

Fig. 20: Recovered paths by the advanced attacker on two printing processes, subject to the state randomization algorithm with five different offset levels. The offset level is expressed in terms of the percentage of the maximum value. The title in each sub-figure shows the average and maximum errors between the reconstructed paths and the ground truth paths.