2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) | 978-1-6654-1693-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSN53405.2022.00058

Background Buster: Peeking through Virtual Backgrounds in Online Video Calls

Mohd Sabra
University of Texas at San Antonio
mohd.sabra@utsa.edu

Anindya Maiti
University of Oklahoma
am@ou.edu

Murtuza Jadliwala
University of Texas at San Antonio
murtuza.jadliwala@utsa.edu

***Abstract*—Video calling applications such as Zoom and Skype have become the preferred medium for both personal and professional communications. One feature in these applications that has gained prominence is the virtual background feature, which enables users to conceal their background by blending in a virtual image or video in place of the real background, thus providing users with background and contextual privacy. However, this feature is not robust enough, and depending on the target user's activities, movement and accessories worn during the call, portions of the user's background could leak which can then be reconstructed to reveal significant portions of the user's real background, and other contextual information related to the real background. This paper conducts an investigative analysis of the background privacy provided by the virtual background feature in video calling applications by designing a novel background reconstruction framework, and using it to reveal users' real background. By means a large dataset of call videos, collected from human subject participants and in the wild, a comprehensive evaluation of the proposed framework and related privacy attacks under a variety of different experimental parameters is then carried out. Results from these evaluations show that significant leakage of background information is feasible under certain conditions, rendering the feature ineffective in protecting privacy and giving users a false sense of security.**

***Index Terms*—Virtual background, video call, privacy.**

I. Introduction

The use of video conferencing applications (e.g., Zoom [13], Skype [8], and Microsoft Teams [5]) brings many benefits in this era of remote communication and work, however it also raises significant privacy concerns. A recent Ipsos survey [3] found that more than 60 % of the respondents were concerned about the privacy of their video calls. This should not be surprising, as video calls often contain potentially sensitive spoken or displayed (visual) information by, or related to, the participants. Video calling platforms typically employ end-to-end encryption to protect call data from trivial network-level eavesdropping. Despite this, several popular platforms have suffered from security breaches in the past resulting in leakage of private information such as stolen account passwords [1] and meeting identifiers resulting in the phenomenon of "zoombombing" [14]. These vulnerabilities, which are primarily system- or platform-specific and can be attributed to shortcomings in software design and implementation, have since been fixed but new vulnerabilities continue to emerge.

However, there is a much bigger privacy threat that impacts all types of video calls, irrespective of the platform or service provider, and which has been largely ignored by users, service providers and the scientific community so far. *What if the adversary is a valid/authorized user present at the other end of the video or conference call?* In other words, can an adversary, who is at one end of a video call, infer some potentially sensitive information about the target user (at the other end) that is not trivially visible/audible in the call? Akin to "insider" attacks in traditional computer and information systems, such insider privacy threats in video calls are not only application- or platform-agnostic, but also difficult to detect and hard to protect against without significantly impacting call quality. Recently, there have been several investigative studies in this direction where acoustic information and image frames from video calls have been exploited to infer keystrokes [23], [30] and call locations [27] of target users (on the other end of the call). In order to keep pace with the constantly advancing and evolving video calling technology, a continuous effort to identify and overcome novel security and privacy threats is needed - this includes not only platform-specific threats, but also platform-agnostic and insider threats which are much more difficult to perceive and protect against!

This paper advances the state-of-the-art in the later direction, by studying the effectiveness of a popular privacy-preservation technique in online video calls - *virtual backgrounds*. A virtual background (also sometimes referred to as a *background filter*) is an image overlay which can be used by online video call participants to hide their real or actual backgrounds. In other words, the virtual background feature, which is available in most online video conference applications, allows video call participants to cover their actual backgrounds with an image of their own choice such that the overlay image hides or obscures most, if not the entire, actual background. Video calling applications accomplish this by separating the foreground (containing the user) and the background using image processing techniques such as *image matting* [34]. A virtual background can be used to obscure personal or intimate objects in the users' background or to anonymize sensitive context such as location [9], [10]. However, the effectiveness of this privacy feature has not been thoroughly investigated. Specifically, it is currently not known *how effective is the virtual background (or background filter) feature in protecting users' actual backgrounds, and the sensitive information therein, in online video calls?* We address this issue by undertaking an investigative analysis of the virtual background feature in popular video calling applications to

2158-3927/22/$31.00 ©2022 IEEE
DOI 10.1109/DSN53405.2022.00058

study how they perform against our novel real background reconstruction strategy and related privacy attacks.

We first propose a novel background reconstruction framework (Section V) which attempts to reconstruct the real background in a video call that has a virtual background blended in to obscure the real background. Followed by that, we employ the real background (partially) reconstructed by this framework to design and evaluate four different privacy attacks (Section VI), namely, location inference, specific object tracking, generic object inference, and text inference attack. Then, by means of video call data collected from real human subject participants (in a variety of different settings and parameters) and pre-recorded videos collected in the wild (Section VII), we comprehensively evaluate the performance of the proposed real background reconstruction framework, and the efficacy of the privacy attacks that employ it (Section VIII).

II. Related Work

Next, we briefly summarize some recent privacy threats targeting online video calls, followed by a discussion of more general threats which employ other visual (side)channels.

Privacy Attacks using Online Video Call Data. As outlined earlier, privacy attacks in online video calls are either platform-specific and can be attributed to shortcomings in the target platform's software design and/or implementation, or platform-agnostic and employ audio/video call data as side-channels for inferring sensitive information. For the former case, we have seen instances of software vulnerabilities and configuration issues in the popular platform Zoom that have resulted in leakage of private information such as stolen account passwords [1] and meeting identifiers [14] resulting in unauthorized individuals joining ("Zoombombing") meetings with the aim of harassment and causing disruption. Similarly, another popular platform Webex has suffered from several vulnerabilities [2] which would allow attackers to covertly join meetings and access private information (e.g., name, email, IP address, device info) on meeting attendees without being admitted to the meeting. These platform-specific vulnerabilities have since been fixed, but new ones have continued to emerge. In the direction of platform-agnostic attacks, there have also been several investigative studies exploiting acoustic data and image frames from video calls as side-channels to infer sensitive information about the meeting/call participants. For instance, Sabra et al. [30] studied the feasibility of inferring a call participant's typed keystrokes by observing the target's fine-grained shoulder movements. Compagno et al. [23] conducted a similar investigation by employing keypress acoustics or audio data, instead of the call's video data. Recently, Nagaraja et al. [27] investigated the feasibility of inferring call locations from echo-reflection characteristics.

Privacy Attacks using other Visual Channels. Beyond the context of online video calls, private data inference attacks that employ other forms of visual channels is a well-researched domain. For instance, visual attack vectors such as a touch-screen's reflection on sunglasses [28], eyeball movements of a hunt-and-peck typer [22], visual tracking of fingers [17], and observation of motion of a tablet's backside [33] have all been utilized to infer typed text, passwords, PINs, and other sensitive information. Similarly, Backes et al. [15], [16] has proposed *Tempest* attack techniques which employ visual reflections to infer sensitive information, for example, reading the contents of a monitor from the reflections off a user's eyes or other seemingly benign objects such as teapots. Some other attacks in this broad direction also include inferring sensitive information from publicly-available images. For example, Shoshitaishvili et al. [31] devised a novel attack framework that, given a large corpus of pictures shared on a social network, automatically determines dating relationships, with reasonable accuracy.

The privacy attack proposed in this paper lies in the former category, i.e., privacy attacks in online video calls. However, in contrast to earlier efforts, in this work we target a novel application feature (i.e., virtual background or background filter) and the sensitive information leaked by it (i.e., users' obfuscated backgrounds). As backgrounds in online video calls could reveal potentially sensitive information, including user-context, location and preferences, it has resulted in widespread adoption of background hiding/obfuscation features, such as, virtual backgrounds, by users. As a result, it has become all the more important to comprehensively evaluate the efficacy of such popular privacy-preserving features in one of the most widely used web applications (i.e., online video calling). This is what we aim to accomplish in this work.

III. Technical Background

Before describing our real background reconstruction framework and the related privacy attacks, we first outline the relevant technical concepts that are needed in its understanding.

Representation of Video Data. Any video stream data V is a time-ordered sequence $\{f^1, f^2, f^3, \ldots, f^l\}$ of image frames f^i, where l is the number of frames in the video. l typically depends on the length of the recorded video and the sampling rate of the recording device (hardware and software). Each frame f^i in a video stream V can be represented as an array of (m rows and n columns) of pixels, denoted as follows:

$$\forall f^i \in V = \begin{bmatrix} p^i_{1,1} & p^i_{1,2} & p^i_{1,3} & \cdots & p^i_{1,n} \\ p^i_{2,1} & p^i_{2,2} & p^i_{2,3} & \cdots & p^i_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p^i_{m,1} & p^i_{m,2} & p^i_{m,3} & \cdots & p^i_{m,n} \end{bmatrix}_{m \times n}$$

The size of the array (characterized by m and n) is indicative of the resolution of the image frames within the video, and is generally fixed for all the frames of the video. Each pixel $p^i_{u,w}$ of a frame f^i stores the color information at location (u, w), and the size of the pixel (in terms of number of bits) depends on the bit-depth used to store color information in the image. For example, for an image with a 24-bit depth (also referred as Truecolor), $p^i_{u,w}$ is a 24-bit value, wherein each of the three 8-bit blocks represent the intensity of the red, green, and blue primary colors, respectively.

Virtual Background Feature. Although the exact technique for applying virtual backgrounds to an ongoing video call in commercial video calling/conferencing applications is proprietary and unknown, we have determined the general underlying principles to be as follows. The virtual background feature in video calling applications attempts to replace/overlay the background pixels of each image frame f^i in the video call V with the corresponding pixels of a virtual image, denoted by VI. In this process, the first step is to generate a background mask for each frame, denoted as BM^i, which identifies the regions of the frame where the virtual background should be applied. More formally, a mask BM^i is a bitmap with the same resolution as the target image frame f^i, where non-zero pixel values represent the foreground (such as a human face or body), while zero pixel values represents the background in the video call. Often a *binary mask* is used for this purpose, which for a frame f^i of size $m \times n$ is a $m \times n$ bitmap, where each pixel (u, w) of the mask having a value of either $\langle 255, 255, 255 \rangle$ (indicating that the corresponding pixel $p^i_{u,w}$ of f^i is part of the foreground) or $\langle 0, 0, 0 \rangle$ (indicating that the corresponding pixel $p^i_{u,w}$ of f^i is part of the background). Similarly, a *trimap* mask has three states where an intermediate value of $\langle 128, 128, 128 \rangle$ implies that a pixel $p^i_{u,w}$ can either be part of the foreground or background. Mask generation is usually done by means of custom supervised machine learning models, which are capable of detecting and separating foreground objects (in image frames) from background objects. Moreover, mask generation could either be done independently across frames or based on previously observed frame(s) in order to reduce (classification) errors and abnormalities. Mask generation (to separate the foreground from background) is not a perfect process, and depending on factors such as presence of certain objects in the frame or movement of users/objects across frames, parts of the background could be classified as foreground by the mask and "leaked" in the masked frame. This part of the real background for a frame f^i that is "leaked" or not obscured by the mask is also denoted by LB^i.

After the background mask is generated, a *blend*ing function is used to combine appropriate portions of virtual image VI, called the virtual background image and denoted as VB^i, with the image frame f^i using the background mask (BM^i) generated in the previous step. Typically, blending is used to increase the quality of the video frames by reducing sharp contrasts between the detected foreground objects in the original frame and the virtual background image. Some state-of-the-art blending techniques that could be employed for this purposed include alpha blending, Gaussian blending, and Laplacian pyramid blending [19], [20], [18]. However, the blending function used by popular video calling applications is unknown (to us), and the type of blending function used could also depend on the generated mask. One side-effect of the blending operation is that it creates small regions in the output frames (near the foreground-virtual background edges) such that pixel values in these regions are mixture of both f^i and VB^i, as depicted in Figure 1.

Four Conceptual Components of Virtual Frames. Consider

(a) (b) (c)

Fig. 1: (a) A zoomed region of the real background, (b) the zoomed region after applying virtual background with blending, (c) the zoomed region after applying virtual background without blending.

a video call V with a virtual image (VI) blended into the frames of V, as described earlier and shown in Figure 2. Each (blended) frame f^i of V is composed of four non-overlapping components (represented as bitmaps): the video caller VC^i, the leaked background LB^i, the blended pixels BB^i due to the blending function, and the virtual background VB^i. As such, every pixel in a frame can be defined as a combination of these four bitmaps, as shown in Figure 3. Later in this paper, we will attempt to infer the leaked background (LB^i) in a blended frame by reconstructing the other three components. Specifically, reconstructed VB^i, BB^i, and VC^i are removed from the original blended frame f^i, and any residue pixels not removed from f^i are assumed to be the leaked part (LB^i) of the real background.

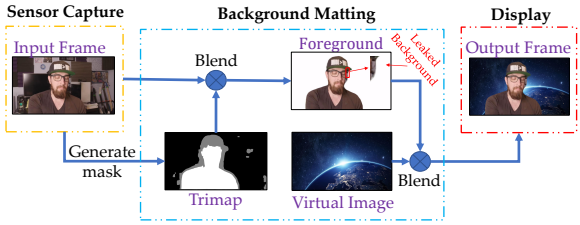


Fig. 2: An example of virtual background used in a video call.

(a) (b) (c) (d) (e)

Fig. 3: The color green represents the leaked background, brown represents the virtual background, and blue is the video caller. (a) f^i, (b) BB^i, (c) VB^i, (d) VC^i, (e) LB^i.

IV. Adversary Model

The goal of an adversary in our setup is to infer as much information about the real background as possible from the virtual image blended video call frames, produced by the virtual background feature as discussion in Section III. The leaked (real) background information in this fashion can be used to infer sensitive information such as: (i) location of the target user during the call (for example, if the target user is at home, office, or another location), and (ii) objects present in the background (for example, books or signs/posters indicating political/religious affiliations). This could result in a significant privacy loss for the target user, despite using a

virtual background feature to hide such information from being revealed to others on a video call, thus rendering it ineffective and giving the user a false sense of privacy protection.

Specific to our experimental setup, we assume that the webcam employed by the target user is stationary during the video call, and that the adversary is able to save the video stream data (V) for post-processing. Additionally, we assume that there is at most only one person in the recorded video stream. The adversary can execute this privacy threat by either participating in a video call as an authorized participant and directly recording the video data corresponding to the target user, or by gaining access to a recording of the call from a private or publicly-available archive [7], [12], [11] after the call. We will discuss additional setting- or scenario-dependent adversarial assumptions as we describe our inference framework in the following sections.

V. Real Background Reconstruction Framework

We now present the design of our reconstruction framework, which takes as input a sequence of virtual image blended video frames V, and outputs a reconstruction of the real background by combining all leaked background parts LB^i.

A. Overview

Given V, our framework applies the following process on each frame $f^i \in V$: (i) determine the virtual background region (or bitmap) VB^i, (ii) determine the blended region BB^i, and (iii) determine the video caller VC^i. Once these three components are identified, any leaked background regions LB^i in the frame f^i are calculated subtracting all the regions from f^i by VC^i, BB^i, and VB^i . After all frames in $f^i \in V$ are processed in this fashion, our framework combines all LB^is ($\forall i$) to reconstruct (parts of) the real background obfuscated by the virtual background feature. The overall design of our proposed reconstruction framework is shown in Figure 4.

B. Virtual Background Masking

The first step in our framework is to identify the virtual background region VB^i using *virtual background masking*. We consider two different scenarios. First, where the adversary has the virtual image VI employed by the target user, and can use it as a direct mask. This is an effective approach when a built-in and/or popular image is used as a virtual image. The second scenario is where the adversary does not have the virtual image employed by the target user, and must reconstruct it before applying it as a mask. Similar scenarios should also be considered if the virtual background feature employs virtual videos, instead of static virtual images, for concealing the real background.

Identifying Known Virtual Image. This is the relatively straightforward scenario for virtual background masking, where the adversary can simply compare pixel-level similarity with a dataset ($D_{img} = \{img_1, img_2, \ldots\}$) of default/popular virtual background images that are potential candidates for the target user's employed virtual background. For this we employ an highest-likelihood estimator that, for each blended video frame f^i, does pixel-level matching with each image in the above dataset, and is defined as follows:

$$img_{actual} = \max_{\forall img \in D_{img}} \sum_{u=1}^{m} \sum_{w=1}^{n} \mu(img_{u,w} \oplus f^i_{u,w})$$

where μ is a matching function,

$$\mu(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$$

Identifying Known Virtual Video Frame. In this case, we assume a dataset of known videos, $D_{vid} = \{vid_1, vid_2, \ldots\}$. Please note that as a video is a sequence of frames, the above set is nothing but a set of frames. Now, when processing a blended frame f^i in V, it simply needs to be determined which frame (of which video) in the above dataset is employed as the virtual background. In order to accomplish this, we extend the previous highest likelihood estimator as shown below by considering all frames in all default/popular videos in the dataset:

$$frame_{actual} = \max_{\forall frame \in D_{vid}} \sum_{u=1}^{m} \sum_{w=1}^{n} \mu(frame_{u,w} \oplus f^i_{u,w})$$

Using Unknown Virtual Image. In this case, the adversary does not have the raw virtual image used by the video caller. For this we design the framework to recreate the potential image used as the virtual background, from the video V itself. For this we utilize the observation that only pixels belonging to the virtual background would stay the same (static) across all the frames of V, whereas the pixels representing the user (caller) and blended regions around the user would be more dynamic. We employ a threshold-based approach, wherein any pixel with a consistent value across a large number of frames in the video V would be considered as part of the virtual background image. Empirically, we found that for a standard 30 fps video stream, a pixel consistent across 10 or more frames has very high probability of belonging to the virtual background. We employ this threshold in our evaluation. This virtual image derivation approach may work similarly well in case the user is fairly stationary during the call, which will not reveal the regions of the virtual image where the user is positioned. This problem can be mitigated by the adversary by searching for the unknown virtual image in other call videos (used by the same user or other users), and then used them during the virtual image derivation process (Figure 4).

Using Unknown Virtual Video Frame. Similar to the previous scenario, in this case the adversary does not have the raw virtual video (frames) used by the video caller as the virtual background. As the unknown video has multiple frames, this makes it even more challenging to identify the exact background video frame that should be used (to compare) with a corresponding frame in the target video call. We utilize the fact that the virtual video loops repeatedly, and use it to derive all the frames of the virtual video using information from every periodic occurrence of each frame (pixels stay the same across every occurrence of a frame). Once the virtual video has been

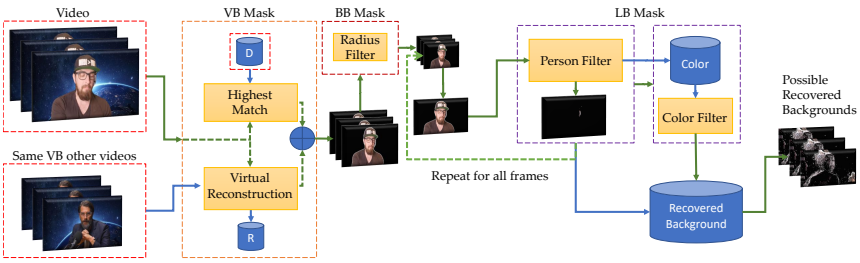


Fig. 4: Proposed real background recovery framework. Green arrows show the flow.

derived, we again employ a pixel-level similarity matching to identify a frame in the video call (V) containing the exact frame from the virtual video. This virtual video derivation may also not be completely accurate, especially if the user is fairly stationary during the call. Similar to the unknown virtual image, we can employ the virtual video used by another user (or in another video call by the same user) for better reconstruction (Figure 4).

Generating Virtual Background Mask. After either a known virtual image/frame or a reconstructed image/frame (denoted as M^i) is identified, a binary virtual background mask (VBM) is generated at the pixel-level using the following function:

$$VBM^i_{u,w} = \begin{cases} 1 & \text{if } \mu(M^i \oplus f^i_{u,w}) = 0 \\ 0 & \text{if } \mu(M^i \oplus f^i_{u,w}) = 1 \end{cases}$$

We can now use $VBM^i_{u,w}$ to obtain VB^i (from f^i) as follows:

$$VB^i_{u,w} = \begin{cases} f^i_{u,w} & \text{if } VBM^i_{u,w} = 1 \\ 0 & \text{if } VBM^i_{u,w} = 0 \end{cases}$$

Later in the paper, we visually show VB^i being removed from f^i by depicting the corresponding pixels as black (an example is shown in Figure 3).

C. Blending Blur Masking

The next step in our framework is to approximately identify the blending blur (BB^i) region $\forall i$, i.e., for all the blended frames $f^i \in V$. The BB^i region resides between virtual background region (VB^i) and foreground pixels (VC^i plus LB^i). And, leaked background pixels (LB^i) are usually found between BB^i and VC^i as the video calling software mistakes parts of the background as being a part of the video caller (user). The separation between the video caller (user) and any leaked real background is not accounted here, and will be recovered in the next phase as part of the foreground.

To recover $BB^i_{u,w}$ we check all pixels within a radius ϕ for every pixel in the $VBM^i = 1$. In other words, $\forall u, w$ where $VBM^i_{u,w} = 1$, we calculate $BBM^i_{p,q} = 1 \oplus VBM^i_{p,q}$, wherein we maintain heuristically derived constraints such as:

$$\sqrt[2]{(p-u)^2 + (q-u)^2} \leq \phi,$$

$$1 \leq p \leq m, \;\; 1 \leq q \leq n$$

We can now use $BBM^i_{u,w}$ to obtain BB^i (from f^i) as follows:

$$BB^i_{u,w} = \begin{cases} f^i_{u,w} & \text{if } BBM^i_{u,w} = 1 \\ 0 & \text{if } BBM^i_{u,w} = 0 \end{cases}$$

D. Video Caller Masking

We observed a few key characteristics in the video calls pertaining to the video caller (or user), which we take in to account for calculating the video caller mask (VCM^i):

Inaccurate Human Boundaries. It is repeatedly observed that regions under the head, near the hair, between fingers, and other places around the video caller (or user) contains a leakage portion of the real background. This type of leakage contributes significantly towards our real background inference process.

Initial Leakage. We also observed that when a video call starts, the accuracy of a video calling software in concealing the real background is often poor. The accuracy improves after a few frames, when the video calling software is better able to track movements of the video caller, as shown in Figure 5. This type of leakage also contributes significantly towards our real background inference process.

Color Analysis. Whenever a pixel from the real background is leaked, it would fairly maintain the same color across different frames when it was leaked. In contrast, the pixels depicting the boundaries of the video caller would have slight variation is color as they relatively move throughout progressing frames. This can also be amplified by factors such as patterns on their cloths. We utilize this observation to refine our video caller mask (VCM^i).

In order to accurately generate VCM^i we apply a state-of-the-art segmentation technique using `DeepLabv3` [21], which employs the *Atrous Convolution* with upsampled filters to extract dense feature maps and to capture long range context.

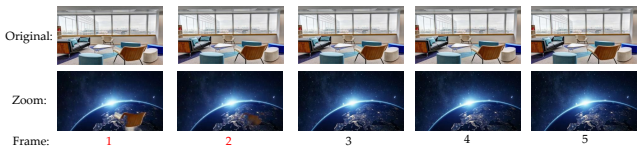


Fig. 5: Example of leaked background components in the initial frames of a video call.

While this technique cannot be applied in real-time video calls due to computational challenges, an attacker can certainly use it for post-processing of a recorded video call. The output of this process is a video caller mask (VCM^i), which can be used to obtain VC^i from f^i.

Color-based Refinement of VCM^i. Although very accurate, `DeepLabv3` is not perfect, and as a result, the VCM^i it outputs may still contain parts of the leaked background. To address this issue, we further refine the accuracy of the VCM^i obtained as the output of `DeepLabv3` by applying a statistical color-based correction on the VCM^i . Specifically, for every pixel in $VCM^i_{u,w} = 1$, if a color was observed in $f^i_{u,w}$ with a very low frequency (presumably from the real background), we modify $VCM^i_{u,w} = 0$. Ultimately, the color-based refinement would identify some of the leaked background pixels where their color statistically contrasts with the video caller.

E. Real Background Reconstruction

After the generation of the virtual background mask (VBM^i), blending blur mask (BBM^i) and the video caller mask (VCM^i) for every frame $f^i \in V$, our attack framework attempts to infer the leaked background (LB^i) in each frame $f^i \in V$ by removing the corresponding components (i.e., VB^i, BB^i, and VC^i, respectively) from the original frame f^i. Any residue pixels not removed from f^i are assumed to be the leaked part of the real background. More specifically, VB^i, BB^i, and VC^i are removed from the original frame f^i by applying the corresponding masks VBM^i, BBM^i, and VCM^i, respectively, to f^i. The residual (leaked background) pixels in all frames are then combined to form a (partially) reconstructed real background.

VI. Inference Attacks

Location Inference. Our first attack, referred as Location Inference attack, is aimed at inferring the location of a victim caller, given that the adversary has auxiliary information about the caller's background at different locations. We had to address two technical challenges in implementing an automated location inference based on matching reconstructed real background with a set of known backgrounds (and therefore their corresponding locations). We also extend our matching to location across different calls, without knowledge of the full real background (auxiliary information). The first challenge is that the real background may have slightly changed (from the background known to the adversary at a prior time) due to changes in ambient lighting, for example, the effect of difference in daytime and nighttime lighting. The second challenge is that the camera view may have slightly rotated and/or shifted from the background known to the adversary, which is likely if the webcam was re-adjusted or if it a laptop webcam which is angled differently based on how the laptop is opened. We address the first challenge by reducing our matching problem to hue matching at individual pixel level, while ignoring their saturation which is significantly affected by ambient light conditions. The second challenge is addressed by incrementally rotating and shifting the reconstructed background, while trying to find the best match with a known background. Due to space constraints, we do not provide full technical details of the above two matching techniques (to overcome both the above challenges).

Specific Object Tracking. Instead of inferring the (whole) background or associated context (e.g., location), the focus of the Specific Object Tracking attack is to look for the presence (or absence) of a specific object (or objects) in the reconstructed (real) background. For this attack, we assume that the adversary has a *template* of the object he/she is searching for in the reconstructed background. A template is nothing but an array of pixels describing the desired object. To search for the object in the reconstructed background, the object template is incrementally rotated, shifted, and scaled while moving across the pixel map of the reconstructed background frame looking for areas (in the reconstructed background frame) that closely represent (or match) the object. For determining a match, both the color (hue) and the relative distance between the pixels being compared are considered, together with the percentage of the template that is matched. Again, we skip some of the specific technical details due to space constraints.

Generic Object Inference. The goal of the Generic Object Inference attack, which we investigate next, is to generically detect (the presence or absence of) objects in the reconstructed background, without using any template (as before). For this, we employ state-of-the-art object detection frameworks such as RetinaNet [24] and YOLOv5 [29], which are trained with the household object data set [26], common objects (COCO dataset) [25], and ImageNet database [4]. For our experimental evaluations, we employ publicly-available implementations of both RetinaNet and YOLO.

Text Inference. The goal of the Text Inference attack is to infer any textual data present in the reconstructed (real) background. For this attack, we employ the TextFuseNet framework [35], which uses a fusion of Mask R-CNN and Mask TextSpooter models to detect bounding boxes around the text in the image frames. Followed by that, the framework employs a fusion of Residual Neural Networks (RESNet) and Region Proposal Networks (RPN) models to infer the text within the detected bounding boxes.

VII. Experimental Setup and Data Collection

Next, we present details of the different experimental settings and scenarios under which video call data from human subject participants is collected, and eventually used for evaluating the proposed attack framework. In order to collect video call data that captures a wide-range of target user movements, backgrounds and interactions, we designed three different experimental setups or data collection strategies. The first setup, referred by us as E1 and described in Section VII-A, is designed to comprehensively evaluate the vulnerability of the virtual background feature under a variety of target user related parameters (such as actions, backgrounds and clothing/apparel), by carefully controlling these parameters. In this setup, the collected videos are short in nature and there

is no active interaction (by the author collecting the data) with the participant while recording the videos. The second setup, referred by us as E2 and described in Section VII-B, is designed to mainly evaluate the vulnerability of the virtual background feature in two contrasting settings with no control on the participants' action or movements. The first setting is where the participant is passively watching/listening to some online content, while the second setting is where the participant is actively interacting with another user (for example, presenting some content). In this setup, the collected videos are slightly longer in nature and there is active interaction (by the author collecting the data) with the participant while recording the videos. While the first two setups collected video call data from actual human subject participants, in the third setup (referred by us as E3 and described in Section VII-C) videos are captured from the wild (say, from YouTube). Videos from this third setup is used to simulate a setting where the adversary is not an active participant in the call and attempts to compromise the background privacy of arbitrary pre-recorded videos with the virtual background feature. We would like to highlight that all data from human subject participants in our experiments were collected in an ethical fashion and after obtaining the necessary approvals from our university's Institutional Review Board (IRB).

A. Experimental Setup E1

For this setup, we recruited 5 human subject participants (from on-campus) and asked them to record short two-minute videos performing ten unique actions/movements under varying background and other conditions. Participants used their own personal laptop or desktop computers (with a webcam) to complete all the recording sessions in a remote fashion at a location of their choice. The participants recorded the videos without the researchers present or interacting with them. They were given a fixed time-frame to complete all the video recordings remotely at their own convenience, and then send it back to the researchers (all together) once completed. The ten unique actions/movements included: leaning forward, leaning backward, arm waving, rotating, clapping, stretching, typing, drinking and exiting/entering room. In addition, participants were asked to repeat the above actions under different backgrounds, different lighting conditions (lights on/off), different apparels (similar/contrasting to the background), and with different accessories (e.g., headphones, hats, etc.). Participants were only given high level experimental parameters, with the specifics (exact background, apparels, accessories, etc.) left at the participants' discretion. For each two-minute video, participants received $1, with a maximum of $20 per participant for this phase. At the end of this experimental data collection phase, we ended up collecting a total of 163 videos, which we use in our evaluations.

B. Experimental Setup E2

For this setup, we recruited 5 participants as well, out of which four were from the E1 data collection phase while the remaining were newly recruited participants. Here, we first asked each participant to initiate a video call with one of the authors (who was collecting the data) and record 4 ten-minute videos where they are passively watching some online content (e.g., YouTube video). Then, they record an additional ten-minute video where they are actively engaging (by means of a presentation) with the author. For each recording, we asked the participants to pick a different background. For each ten-minute video, participants received $4, with a maximum of $20 per participant for this phase. At the end of this experimental data collection phase, we ended up collecting a total of 25 videos, which we use in our evaluations.

C. Experimental Setup E3

For this setup, instead of recruiting human subject participants, we searched for pre-recorded videos in the wild, for example, videos uploaded on classical video sharing services such as YouTube. Our goal here is to find videos of users participating in real video conference calls, or as close to a conference call setting as possible. To accomplish this, we searched on YouTube using a variety of keywords, including terms like "vlog", "podcast", "talk", "review", etc. At the end of this data collection phase, we collected a total of 50 videos with varying lengths, which we use in our evaluations. During our search for videos in the wild, we observed that some of the videos had post-recording edits, overlays, and other elements which were were not very desirable. In those cases, we either completely discarded the video (from our dataset), or only used a segment of the video without the post-recording edits.

D. Post-processing

All the videos in the three different setups above are collected without a blended virtual background (or background filter). This is done to preserve the true background, which will be later used as ground truth in our evaluation. In order to blend virtual backgrounds post-recording, we replay the collected videos on a video calling software, such as Zoom, via the Open Broadcaster Software (OBS) VirtualCam plugin [6]. OBS VirtualCam simulates a webcam and allows us to pass the pre-recorded videos to the video calling software as if it was directly coming from the webcam. From there, we applied a virtual background to each of the pre-recorded videos using Zoom's virtual background feature. We then use Zoom's record feature to produce recorded versions of our videos (collected from E1-E3) with the virtual background applied. After creating a video dataset using Zoom, as described above, we repeat the same process using the Skype video calling software instead of Zoom, in order to comparatively evaluate the performance of our attack framework on both these popular video calling applications.

VIII. Evaluation

We now present evaluation results of our proposed framework by means of the video call data collected in various settings, as described above. The following results are for videos with virtual background rendered by Zoom, except in Section VIII-E where we contrast the results using Skype.

A. Performance Metrics.

As we propose a framework to exploit a new type of vulnerability, we first define a custom set of metrics that can appropriately capture its performance and effectiveness. We specifically employ only these metrics in our evaluation.

Virtual Background Masking Rate. We define *Virtual Background Masking Rate (VBMR)* for a given frame as the percentage of the virtual background that was masked after applying blending blur to that frame (Section V-C). A 100% VBMR indicates that the virtual background is completely masked from the frame, preventing any virtual background pixels to be mistakenly labeled as a component of the leaked background. Similarly, a 0% VBMR means that all pixels in the virtual background could be classified as part of the leaked background. VBMR is the difference between the pixels from the ground truth data (video before applying virtual background) and the corresponding pixels after the blending blur is applied.

Reconstructed Background Recovery Rate. We define *Reconstructed Background Recovery Rate (RBRR)* as the percentage of the original video frame that is recovered by the proposed real background reconstruction framework (Section V). To compute RBRR for a target video (with the virtual background applied), we count all the pixels of the original video without the virtual background applied that are leaked in one or more frames of the target video, divided by the frame/video resolution.

Action Speed. We define *Action Speed* of a particular action event (conducted by the user or participant in the call) as the number of frames from the start of the action event until the end of the event, divided by the frame rate.

Displacement. For a particular action event, *Displacement* is defined as the percentage of unique pixel changes across all the frames from the start of the action event until the end of the action event.

B. Virtual Background Masking Rates

We measure VBMR of our framework using three different virtual images and two virtual videos. When the ground-truth virtual backgrounds are included as possible virtual backgrounds, we observed an average VBMR of approximately 98.7%. Alternatively, when the ground-truth backgrounds are not included, we observed a slightly worse average VBMR of approximately 92.6% (when 10 minute video call footages were used).

C. Background Recovery Rates

We now discuss the reconstructed background recovery rate (RBRR) results for the proposed framework, evaluated under a variety of experimental parameters as outlined in Section VII.

Impact of Different Framework Parameters. One of the parameters we introduced in our framework is the radius (ϕ) that determines which portion of a frame is part of the blur. If $\phi = 0$, then naturally our obtained RBRR will increase, but at the cost of precision as some of those pixels would be blurred. However on the other extreme, increasing ϕ to a very high value is also not advisable as there will be nothing to recover then. In order to determine a suitable value for ϕ, an adversary could apply a virtual background on a set of static images using the target software, after which the adversary can calculate the average depth of the blur (ϕ) by comparing the virtual image, real backgrounds, and the output images. Following this process, we obtained $\phi = 20$ as the average depth of blur, and use it for the rest of our evaluation.

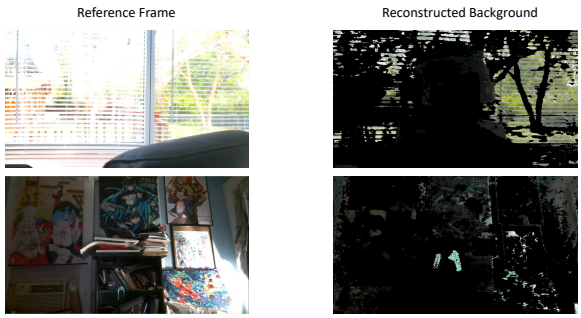


Fig. 6: Two examples of reconstructed background images using two separate videos from E1.

Effect of Different Actions. An interesting pattern that we observed in our experiments is that action events which result in higher displacements cause more of the real background to be leaked, compared to events with lower displacements. For instance, we can see from Figure 7 that on average, entering and exiting (a room) events resulted in a RBRR of about 38.6%, while typing resulted in 4.4% RBRR. Intuitively, this is because entering and exiting events cause significant movement across the real environment, which eventually leads to significant leakage of the real background.

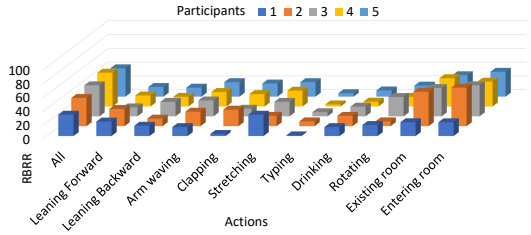


Fig. 7: Background recovery under various actions.

Effect of Movement. We next analyze how action speed and displacement impacts the background recovery. In our first experiment (E1), we asked the participants to vary the arm waving and clapping action speeds as “slow”, “average”, and “fast”. The interpretation of these subjective scales were left to individual participants. The average [action speed, displacement] for the “slow”, “average”, and “fast” clapping speeds were [0.9s, 7.2%], [0.26s, 5.1%], and [0.11s, 4.4%], respectively. While for arm-waving they were [2.3s, 28.2%], [0.9s, 24.1%], and [0.7s, 23.4%], respectively. We observed that both for clapping and arm waving, slower action speeds by participants typically resulted in greater displacements. Conversely, faster action speeds resulted in the lesser displacement.

From our background recovery results Figure 8 in this setup, we observe that action events with the slowest speed returned the highest RBRR (35.9% in "slow", 30.3% in "average", and 33.7% in "fast" arm waving). A faster arm waving, on average, was leaking more background pixels than an "average" arm waving, most likely because of the motion blur produced by the waving action, which made the foreground blend with the background. However, we also saw that too much motion blur could be less revealing if the hand is masked as part of the background. This was observed in case of the "fast" arm clapping event which resulted in a RBRR of 20.8%, while "average" clapping event resulted in a RBRR of only 22.6%.

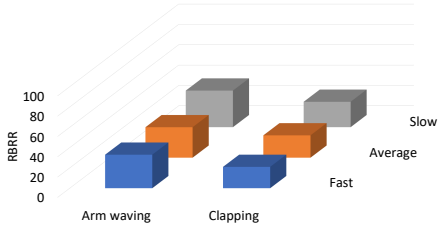


Fig. 8: Effect of action speed on background recovery.

Effect of Different Accessories. In the next set of experiments, we analyzed the effect of participant accessories like headphones and hats on the real background recovery process. Overall, we did not find any significant difference between the participants' choice of different accessories worn during the call. For instance, Figure 9 exemplifies the indifference in background recovery for one particular participant with four different combinations of accessories.

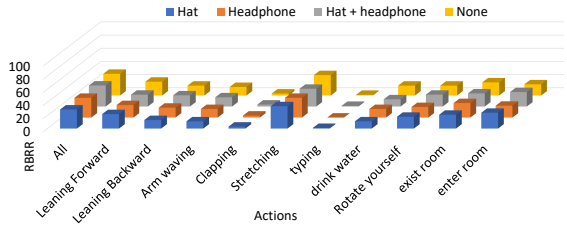


Fig. 9: RBRR for a participant in E1.

Effect of Different Background Lighting. During this set of experiments, we asked participants to repeat the same video call setup under two different lighting conditions, once with background light OFF and then with the background lights ON. Based on the data collected from these experiments, we observed that, in general, there was more background leakage in low lighting conditions than under high lighting conditions (41.6% RBRR light OFF vs. 39.6% RBRR light ON), as shown in Figures 10 and 11. While the 2% difference in RBRR is not significant, interestingly, we observed that the regions of the background reconstructed under the different lighting conditions varied significantly. This suggests that the difference in the RBRR is not merely due to noise.

Being Passive vs. Active in Video Calls. Next, we present our background recovery results for the experimental setups

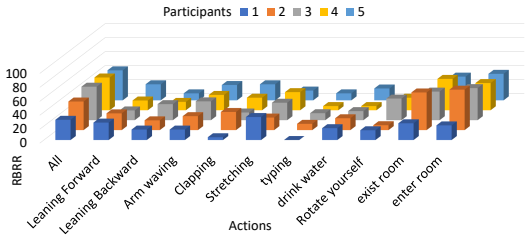


Fig. 10: Background recovery with background lights off.

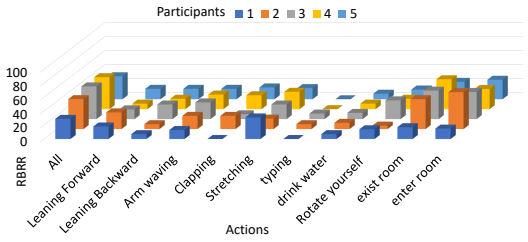
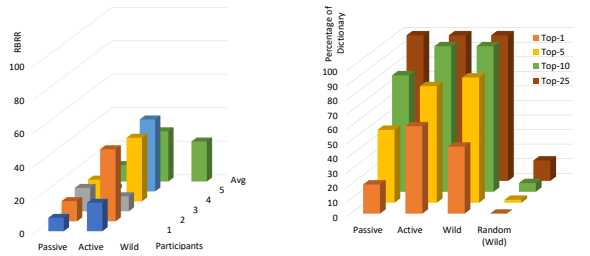


Fig. 11: Background recovery with background lights off.

E2 and E3 (outlined in Section VII). From the data collected in the setup E2, we observed that passive video callers, i.e., video callers who are not talking during the call, are less likely to leak significant portions of their real background compared to those who are active, i.e., speaking or presenting, during the call. As outlined in Figure 12a, for passive callers in E2 the obtained RBRR is 9.8%, while for active callers in E2 the RBRR obtained is 30%. For the data collected in the wild from YouTube (setup E3), we were able to obtain a RBRR of 23.9% as shown in Figure 12a. This is significantly higher than the 9.8% RBRR obtained for passive callers in E2 because users in E3 were actively speaking and presenting (similar to the active users/callers in E2). Another interesting observation is that the RBRR obtained for data collected in E3 is slightly worse than the RBRR obtained for active video callers in E2 (23.9% for E3 versus 30% for active callers in E2). This is most likely because of the high-quality lighting and cameras employed for producing YouTube videos, resulting in the video calling software being able to better differentiate the background from the foreground.

D. Privacy Attacks

Location Inference. As outlined earlier, in this attack experiment our objective is to infer a target video caller's location by matching his/her partially reconstructed (real) background to a pre-populated dictionary of background images (with known locations and where the target caller may have been in earlier videos). For this experiment, we populated our image dictionary with 200 unique (real) backgrounds from the video calls in E1, E2, and E3. Now, for a particular target video, we rank all background images in the dictionary by computing their similarity to the partially reconstructed (real) background in the target video, with the top rank (rank 1) for image which is most similar and the last rank (rank 200) for the image in the dictionary that is the least similar. This similarity

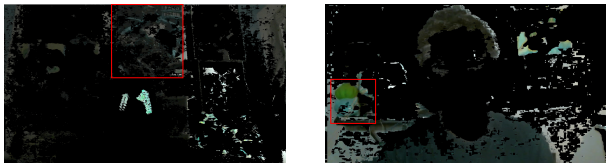


(a) (b)

Fig. 12: (a) Background recovery in E2 and E3 experiments. (b) Location inference in E2 and E3. Shows the percentage of videos where the background (location) is correctly classified within top-k images from the dictionary.

is calculated by comparing the hue changes and distances between all pixels in the reconstructed real background (of the target video call) and background images in the dictionary, and finding the best possible match. We evaluate all videos in datasets E2 and E3 separately by employing the metric top-k ($k = \{1, 5, 10, 25\}$), where k determines how close to the best matching image in the ranked dictionary the real background in the target video call is. So for a top-1 match, the best matching image in the ranked dictionary (i.e., image with rank 1) is the actual real background, while for a top-10 match the actual real background is any image in the dictionary with rank ≤ 10. In addition to the above, we also consider a baseline metric (for evaluation of videos in E3), where k images are randomly chosen from the dictionary. If the real background is present within those k randomly chosen images, then the attack is successful, otherwise it is not. Our experimental results show that higher the RBRR for a target video, better is the location inference given an appropriately created dictionary. For instance, our results (Figure 12b) show that 20% of passive video calls in E2, 60% of active video calls in E2, and 46% of videos in E3 where recovered as top-1 images (from the ranked dictionary). Moreover, regardless of whether the video caller is passive or active, we observed that our proposed scheme is generally effective compared to random guessing.

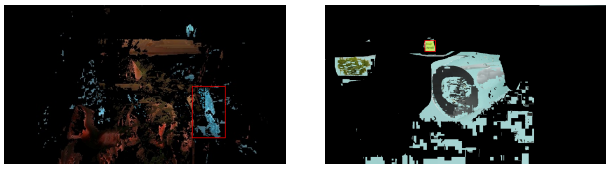
Specific Object Tracking. Given that we have a template image of an object in a video caller's background, we attempt to track if the object exists in the reconstructed background. We do so by applying the same matching technique in our location matching. If the resulting matching score is high, we consider that the object is present in the background. In our experiment, we were able to accurately infer multiple objects (Figure 13) such as shirt, poster, painting, toy, bookshelf, and book. However, a limitation of our object tracking methodology is that it produces a number of false positives when matching our template image with very small areas of the reconstructed background. It produces more false positives when being matched to an area of the reconstructed background with fewer pixels successfully recovered. Therefore, we enforced a minimum matching window size of 5% of the total pixels in the frame and the window must have at least 50% of the pixels successfully recovered. Overall, we were able to track 90 individual objects across different participants' background with 96.7% accuracy.



(a) (b)

Fig. 13: Examples of items detected using specific object tracking: (a) a poster (b) a Pokemon figure.

Generic Object and Text Detection. As our experiments did not regulate the background, we had no control on the objects that may exist in the background. Using our RetinaNet and YOLO object detction models (Section VI), pre-trained with COCO and ImageNet datasets, we were able to detect books in four different reconstructed backgrounds (Figure 14a), a Television in two reconstructed backgrounds, shirts in one reconstructed background, display monitors in three reconstructed backgrounds, and a clock in one reconstructed background. A significant number of videos had a blank wall, bricked wall, window, or door in the background. Text was recovered (using TextFuseNet) from only one video, which was written on a sticky note present in the background (Figure 14b).

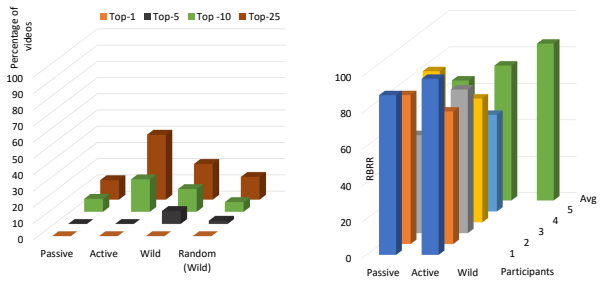


(a) (b)

Fig. 14: Examples of items detected under generic object and text detection: (a) a book detected using COCO and RetinaNet, (b) text on a sticky note was detected using TextFuseNet.

E. Different Video Calling Software

In addition to Zoom, we also tested our attack on Skype. We observed multiple visual differences between Skype and Zoom virtual background rendering, confirming that they likely use different virtual background masking techniques. Skype was more accurate in its virtual background rendering, resulting in an average RBRR of 19.4% for the E3 dataset, compared to an average RBRR of 23.9% for Zoom on the same dataset. As expected, some privacy attacks in Skype are less effective due to the lower RBRR. For instance, the sticky note shown in Figure 14b was leaked from a Zoom call but was not leaked during a Skype call. Likewise, the location inference attack also suffered slightly as for Skype the location of only 76% of passive video calls where ranked within the top-10, compared to Zoom's 80% (as shown in Figure 12b).



(a) (b)

Fig. 15: (a) RBRR after applying dynamic virtual background. (b) Location inference results with dynamic virtual background.

IX. Mitigation

Next, we briefly outline (and evaluate) a few mitigation techniques against the proposed background reconstruction and privacy attacks.

A. Dynamic Virtual Background

The main intuition behind the first mitigation technique is to reduce the difference between the virtual background and leaked background pixels as much as possible, by dynamically adapting the pixel properties of the virtual background based on the properties of the real background pixels. To accomplish this, we employ a Gaussian kernel to modify the brightness and saturation of the virtual background pixels for each frame depending on the brightness and saturation of the corresponding real background frame pixels. Further, the hue value of each modified virtual background pixel is forced to randomly fluctuate over multiple hue values (closer to the modified hue value) across different frames, making it further difficult (for the adversary) to differentiate the leaked real background pixels from the virtual background pixels in each frame.

Results. When the dynamic virtual background is applied, our obtained average RBRR increased to 65.8%, 74%, and 86.2%, respectively, for the passive video caller in E2, active video caller in E2, and the E3 datasets (Figure 15a). While normally a higher RBRR would generally imply a higher leakage of the real background, in this case the recovered real background not only contain pixels of the real background, but it also detects pixels of the virtual background as real background. We re-analyze the location inference attack in Figure 15b using the proposed dynamic virtual background mitigation techniques and observe that the overall top-k inference accuracy of the location inference attack reduced considerably. With dynamic virtual backgrounds, a successful location inference within the top-25 was possible for only 40% of the active video call data in E2 and 22% of the video data in E3.

B. Other Heuristics

We discuss a few other heuristics which could also be employed to mitigate the proposed attack. One heuristic that could be employed is to generate and use a new random virtual background image for every call. The intuition behind this heuristic is that an adversary may not have a prior copy of this newly generated (never-seen-before) virtual background image, which would make the virtual background masking process (Section V-B), and thus the real background reconstruction, a bit more challenging. Another heuristic could be to reduce the number of video call frames shared with the adversary, which will significantly reduce the extent of real background reconstruction, but at the expense of reducing the quality of the video call. Lastly, a more extreme measure could be to not send the actual video call frames (to the adversary) at all. Other than the first frame, all other (following) frames can be replaced with fake video call frames. This can be accomplished by applying a real-time deepfake technique such as the First Order Motion model [32], which will take as input the initial frame of the user's video call (with the blended virtual background) and continue to automatically animate the user for the remaining frames. These animation frames (generated by the First Order Motion model) will be the ones sent in the video call, rather than the actual video call frames. As the real frames are never sent, the real background is never leaked, while the animated frames continue to provide an illusion of the call being live.

X. Discussion

The attacks and mitigations we propose and evaluate are based on specific practicality assumptions. For instance, a critical requirement for the location inference attack is the availability of ground-truth background images. Without the ground-truth background images, the adversary will not be able to form a tractable dictionary for efficient location inference. Also, if the (reconstructed) background does not contain any sensitive information, users may not be motivated to apply the proposed mitigations. It is also challenging to assess the overall impact of the proposed attacks and mitigations, as it is difficult to accurately discern users who primarily use virtual backgrounds for preserving their background privacy. Lastly, it is possible that our attacks and mitigations may perform differently if tested on a larger set of more diverse participants. Nevertheless, we validated that at present virtual backgrounds are not a perfect solution for attaining background privacy.

XI. Conclusion

We proposed and evaluated a real background inference framework leveraging on the imperfections of a virtual background feature in modern video calling software. Based on the reconstructed background, we also proposed a set of different privacy attacks. By means of a comprehensive evaluation using pre-recorded videos in the wild and video call data from human subject participants in a variety of scenarios, we validated the feasibility and implications of such an attack framework.

XII. Acknowledgment

Research reported in this publication was supported by the US NSF under award number 1943351.

REFERENCES

[1] "500,000 Hacked Zoom Accounts Given Away For Free On The Dark Web," https://www.forbes.com/sites/leemathews/2020/04/13/500000-hacked-zoom-accounts-given-away-for-free-on-the-dark-web/, [Online; accessed 22-May-2020].

[2] "Cisco Webex vulnerabilities may enable attackers to covertly join meetings," https://www.helpnetsecurity.com/2020/11/19/cisco-webex-vulnerabilities-attackers-covertly-join-meetings/, [Online; accessed 29-Nov-2021].

[3] "Coronavirus prompts increased use of video chat platforms for work, connection," https://www.ipsos.com/en-us/news-polls/coronavirus-prompts-increased-use-of-video-chat, [Online; accessed 11-March-2021].

[4] "ImageNet dataset," https://image-net.org/index.php, [Online; accessed 28-Nov-2021].

[5] "Microsoft Teams," https://teams.microsoft.com, [Online; accessed 28-Nov-2021].

[6] "Open Broadcaster Software," https://obsproject.com/, [Online; accessed 28-Nov-2021].

[7] "Rewatch," https://rewatch.com/, [Online; accessed 29-Nov-2021].

[8] "Skype," https://www.skype.com/en/, [Online; accessed 22-May-2020].

[9] "Use a virtual background on Zoom calls for better privacy," https://www.totaldefense.com/security-blog/use-a-virtual-background-on-zoom-calls-for-better-privacy/, [Online; accessed 28-Nov-2021].

[10] "Using Zoom Backgrounds to Keep Your Privacy and Clean Up Your Space," https://www.multibrain.net/using-zoom-backgrounds-to-keep-your-privacy-and-clean-up-your-space/, [Online; accessed 28-Nov-2021].

[11] "Vicodo video calls archive," https://www.vicodo.com/features/archiving, [Online; accessed 29-Nov-2021].

[12] "YouTube," https://www.youtube.com/, [Online; accessed 22-May-2020].

[13] "Zoom," https://zoom.us/, [Online; accessed 22-May-2020].

[14] "'Zoombombing' Becomes a Dangerous Organized Effort," https://www.nytimes.com/2020/04/03/technology/zoom-harassment-abuse-racism-fbi-warning.html, [Online; accessed 22-May-2020].

[15] M. Backes, T. Chen, M. Dürmuth, H. P. Lensch, and M. Welk, "Tempest in a teapot: Compromising reflections revisited," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 315–327.

[16] M. Backes, M. Dürmuth, and D. Unruh, "Compromising reflections-or-how to read lcd monitors around the corner," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 158–169.

[17] D. Balzarotti, M. Cova, and G. Vigna, "Clearshot: Eavesdropping on keyboard input from video," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 170–183.

[18] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[19] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in computer vision*. Elsevier, 1987, pp. 671–679.

[20] L. Carpenter, "The a-buffer, an antialiased hidden surface method," in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984, pp. 103–108.

[21] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[22] Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, "Eyetell: Video-assisted touchscreen keystroke inference from eye movements," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 144–160.

[23] A. Compagno, M. Conti, D. Lain, and G. Tsudik, "Don't skype & type! acoustic eavesdropping in voice-over-ip," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[26] D. D. R. Meneghetti, P. H. S. Domingues, B. d. F. V. Perez, T. S. B. Meyer, K. K. G. Cardoso, A. M. de Lima, M. Y. Gonbata, F. d. A. M. Pimentel, and P. T. Aquino Junior, "Annotated image dataset of household objects from the robofei@home team," 2020. [Online]. Available: https://dx.doi.org/10.21227/7wxn-n828

[27] S. Nagaraja and R. Shah, "Voiploc: passive voip call provenance via acoustic side-channels," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 323–334.

[28] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm, "ispy: automatic reconstruction of typed input from compromising reflections," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 527–536.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[30] M. Sabra, A. Maiti, and M. Jadliwala, "Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks," *NDSS*, 2021.

[31] Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Portrait of a privacy invasion: Detecting relationships through large-scale photo analysis," in *In PETS*. Citeseer, 2015.

[32] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe, "First order motion model for image animation," *Advances in Neural Information Processing Systems*, vol. 32, pp. 7137–7147, 2019.

[33] J. Sun, X. Jin, Y. Chen, J. Zhang, Y. Zhang, and R. Zhang, "Visible: Video-assisted keystroke inference from tablet backside motion." in *NDSS*, 2016.

[34] N. Xu, B. Price, S. Cohen, and T. Huang, "Deep image matting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2970–2979.

[35] J. Ye, Z. Chen, J. Liu, and B. Du, "Textfusenet: Scene text detection with richer fused features." in *IJCAI*, 2020, pp. 516–522.