PECAM: Privacy-Enhanced Video Streaming and Analytics via Securely-Reversible Transformation

Hao Wu[1], Xuejin Tian[1], Minghao Li[1 2], Yunxin Liu[3]
Ganesh Ananthanarayanan[4], Fengyuan Xu[1]*, and Sheng Zhong[1]

[1] National Key Laboratory for Novel Software Technology, Nanjing University
[2] Cornell University
[3] Microsoft Research
[4] Microsoft Azure for Operators

ABSTRACT

As Video Streaming and Analytics (VSA) systems become increasingly popular, serious privacy concerns have risen on exposing too much unnecessary private information to the VSA providers. Yet, it is challenging to protect privacy while still preserving desired VSA features, i.e. the effective analytics, forensic support, resource efficiency, and real-time execution. In this paper, We present a VSA privacy enhancement system (PECAM) which addresses above challenge with no change in the VSA back-end. PECAM leverages a novel Generative Adversarial Network to perform the privacy-enhanced securely-reversible video transformation. PECAM also incorporates a couple of system optimizations into its VSA workflow in order to reduce the network bandwidth usage and enable the real-time processing on cameras. We implement our PECAM prototype on commodity hardware and evaluate its performance via both security study and extensive experiments. Results demonstrate that PECAM can effectively enhance the visual privacy of VSA in the presence of an adversary, and its transformed videos, when taken as input for various VSA back-end tasks, maintain a **96%** accuracy of corresponding original videos. Additionally, it performs **12.3×** and **1.8×** better than baseline methods in terms of the computing cost and network bandwidth usage, respectively.

CCS CONCEPTS

• **Security and privacy** → **Security services**; • **Computing methodologies** → **Computer vision**; • **Human-centered computing** → **Ubiquitous and mobile computing**.

KEYWORDS

Video Privacy, Securely-Reversible Transformation, GAN, Video Streaming, Video Analytics

*Corresponding author. Email: fengyuan.xu@nju.edu.cn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ACM MobiCom '21, October 25–29, 2021, New Orleans, LA, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8342-4/21/10...$15.00
https://doi.org/10.1145/3447993.3448618

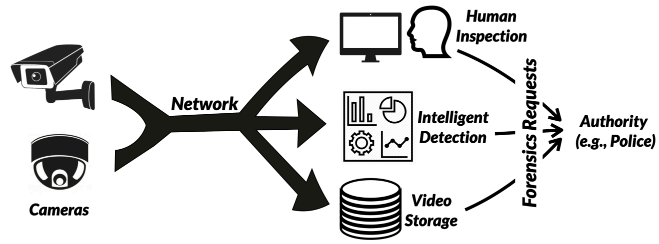


Figure 1: Video Streaming & Analytics (VSA) system. The front-end video sources are resource-limited cameras (mobile or fixed). The video subscribers are analytics related tasks in the centralized back-end, such as the human-in-the-loop inspection, AI-based detection, and multimedia storage service. With legitimate authorization, the authority is able to collect forensic evidence from videos in cases like the crime-scene investigation.

ACM Reference Format:

Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. 2021. PECAM: Privacy-Enhanced Video Streaming and Analytics via Securely-Reversible Transformation. In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '21), October 25–29, 2021, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3447993.3448618

1 INTRODUCTION

Video Streaming & Analytics (VSA) becomes a popular system paradigm for video-centric applications. Such VSA systems offer tremendous benefits to our society and thus are pervasively deployed in various scenarios, such as elderly care, traffic monitoring, and public safeguarding. A VSA system (Figure 1) typically consists of two main parts, a set of *front-end video sources* and a set of *back-end video subscribers*. Front-end sources stream real-time videos to corresponding back-end subscribers via the Internet. When abnormal events are detected by human beings or AI, these videos are also inspected by the *authority* for forensics purposes.

Psychological concerns have risen along with the widespread deployment of VSA systems because people worry that too much unnecessary information is exposed to the back-end subscribers. For example, an elderly person may feel uncomfortable if non-behavioral video contents, such as her facial/clothing details or home interior details, are constantly streamed to a VSA subscriber performing the fall detection. The previous study [54] has proven that individuals could feel their privacy being violated when entering areas with cameras installed. To temporarily mitigate these concerns, European Union took the action of banning the face recognition of people in public-area videos for five years while waiting for proper solutions [6].

We discover that those concerns imply the *unique VSA privacy situation*. The privacy is demanded by people who are caught on cameras, rather than VSA providers. Those people, in general, have diverse sensitivities regarding what visual contents should be protected, and they often do not have negotiation channels to VSA providers. Thus, it is almost impossible to summarize a Region of Interest (ROI) list, which makes everyone satisfied, as the privacy protection target. Fortunately, everyone caught on cameras may agree that the visual content details should be generally removed as long as the accuracy of desired VSA analytics is not affected much. In this way, privacy can be greatly enhanced, and psychological concerns will be mitigated to some extent. However, removed visual contents are still able to be accessed in legitimate forensic cases like identifying a thief.

Visual Privacy. According to our discovery, we introduce a concept of *visual privacy of VSA (VSAP)*. Given a video frame, VSAP is referred to as the whole-frame visual content details which do not belong to the categorizable, behavioral, or spatial information of any foreground object. For example, VSAP covers the clothing texture, facial appearance, vehicle license plate, while it does not cover the object contours and colors. In general, VSAP considers the balancing of privacy and intelligibility with no prior knowledge of privacy ROIs. The intelligibility here means the video information necessary to both human perception and AI understanding for performing VSA back-end subscriber tasks, such as classification, localization, statistical counting, and behavior detection [14, 32, 45, 49].

Objectives. Given our definition of visual privacy of VSA, an ideal VSAP-enhanced system should achieve the following five design objectives altogether: *(1)* reliably raising the price an unauthorized party pays to compromise VSAP; *(2)* highly maintaining the intelligibility of protected videos to support original/existed back-end subscriber tasks performed by either human beings or AI; *(3)* securely retaining the reversibility of protected VSAP for the authorized party in the case of "after-the-fact-forensics"; *(4)* greatly reducing the bandwidth consumed by streaming protected videos to the back-end; *(5)* efficiently integrating the VSAP-enhancing mechanism into front-end sources with little impact on the real-time nature of VSA. Previous work [44] also advocates similar five objectives.

Challenges. Although research efforts have been put into protecting the privacy in VSA or similar video scenarios, they cannot meet aforementioned five objectives altogether, due to three challenges as follows. *First*, it is challenging to make the privacy protection securely reversible without introducing extra auxiliary data structures. For example, it is common to extract the privacy information from the video and encrypt it separately, leading to extra bandwidth and storage space. *Second*, it is challenging to simultaneously guarantee a "friendly-triangle", namely the visual-perception-friendly, AI-analysis-friendly, and protection-reliability-friendly. For example, the ROI-oriented protection, although excellent in supporting analytics, cannot ensure every single ROI is properly protected because the ROI detection will fail if the viewing angle is not expected or partial ROI is occluded. Additionally, current whole-frame-based protection, although offering reliable privacy without ROIs, often creates serious visual problems to human beings and AI. *Last*, it is challenging to execute sophisticated operations on resource-limited camera devices without breaking the real-time rule of VSA. We elaborate on the deficiency of existed video privacy protections with respect to our objectives in Section 2.1.

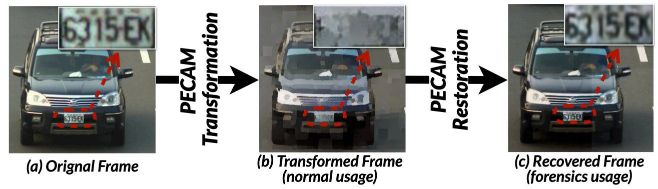


Figure 2: Example frames of PECAM-empowered car-counting VSA. Please note that our transformation is applied to the whole frame.

Designs. In this paper, we propose a versatile design called PECAM to enhance VSAP. PECAM takes the approach of co-designing both intelligent algorithms and system optimizations, aiming to meet all five design objectives in practice. PECAM runs on the front-end of a VSA system, e.g., an IP-based camera, and performs a privacy-enhanced whole-frame transformation over the real-time video streaming. It does not require any change of the VSA back-end.

PECAM's transformation is empowered by a novel *Generative Adversarial Network* (GAN) which addresses the first two design challenges aforementioned. Our GAN in PECAM is designed with a security-reinforced cycle-consistent mechanism, and it can intelligently produce VSAP-enhanced videos suitable to both human inspection and AI recognition. Meanwhile, privacy-enhanced videos can be fully reversed, in authorized situations, by our GAN with no auxiliary data. This efficient secure reversibility is achieved through the GAN-based steganography which hides recoverable information secretly and uniquely into produced videos. We demonstrate with the empirical study that it is difficult for an adversary to illegally reverse produced videos not belonging to him.

PECAM also employs a couple of system optimizations on the transformation pipeline (before leaving the VSA front-end) to improve the overall efficiency. It introduces an H.264-compatible video-compression method to reduce the network transmission cost without compromising the reversibility. It also proposes an online branching strategy with a lightweight execution option to further reduce the transformation latency on the resource-limited camera device. Thus, PECAM is able to deploy and run in real time on VSA front-end video sources like smart cameras, addressing the last challenge.

We have implemented PECAM on commodity hardware and evaluated its performance via both security analysis and comprehensive experiments. The privacy-enhanced videos after PECAM's transformation can achieve at least **96%** of the analytics accuracy of the original videos, with respect to unmodified same VSA back-end tasks. Moreover, PECAM's transformation significantly increase the cost of an adversary to violate VSAP. For example, our experimental results show that an adversary, who has the latest face recognition tool and rich computing resources, cannot detect any facial identity in videos produced by PECAM. Additionally, the compression applied in PECAM can improve the bandwidth efficiency of H.264 by **1.8**×, given the same transmitted data quality. The overall transformation latency of PECAM also meets the real-time requirement on VSA front-end sources, and is **12.3**× and **46.8**× faster than computations of the popular CycleGAN [56] and YoloV3 [43] respectively.

Use Case. Figure 2 shows an example of the car-counting VSA in the traffic monitoring scenario. The original frame (Figure 2(a)) captured by the camera contains the license plate number, which is not used in the car-counting task but sensitive to many drivers. PECAM on the camera, configured with proper privacy granularity, will transform this frame into a cartoon-like version (Figure 2(b)), where the license plate is not readable, and yet the car shape is recognizable. By doing so, it raises the bar for attackers to retrieve the plate information from the transformed frame, as shown in Section 6. Later, if an accident is detected or reported, an authorized party will be able to recover the targeted transformed frame back to its original version (Figure 2(c)), to identify the plate associated with the accident. Please note that all object details, not just the plate number, are generally and smoothly removed at the whole frame level by PECAM.

Contributions. In summary, our work makes the following contributions. *First*, we design a versatile privacy-enhancing system running on the VSA front-end in real time. *Second*, we introduce a novel security-reinforced cycle-consistent GAN to perform the privacy-enhancing transformation, which is empirically proven to be securely-reversible. *Third*, we optimize the system performance in terms of transmission bandwidth and computing latency. *Last*, we implement a prototype on commodity hardware and conduct extensive experiments for its evaluation.

2 RELATED WORK AND BACKGROUND

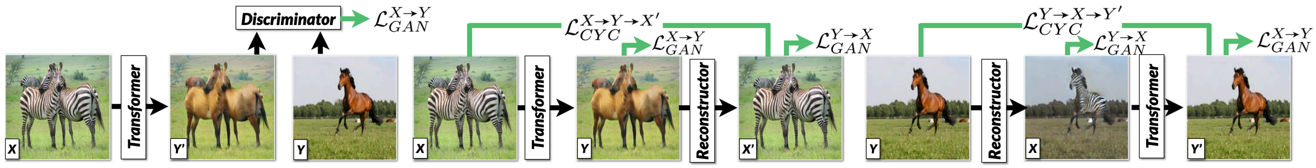
2.1 Visual Privacy Protection

Visual privacy protection methods modify or remove the privacy information in images or videos. They can be categorized into five types of approaches as follows. **Face de-identification** [13, 20, 23, 34] has been well studied to replace real faces in the video with synthetic ones to protect the facial appearance. However, there are many other types of possible privacy information in the VSA context, such as the license plate, craftwork design, and so forth. Additionally, this de-identification cannot be reversed. **Encryption** based methods [41, 46], although privacy is fully preserved, are not suitable to protect videos used in various VSA back-end tasks because the video intelligibility is also fully removed. It is also challenging to run them in real time, even on resource-rich servers. **Inpainting** based methods [25, 26, 33, 52] are utilized to protect specific ROIs in images. They require prior identifications of all possible ROIs before the deployment, which could be a challenge for the VSA front-end (especially when it is deployed in a public area). Moreover, these methods are not reliable to be used in the video scenario where every frame need to be properly protected. If an ROI is missed in any single frame due to the failure of ROI detection, which is quite possible in practice, the perfect protection of this ROI in all other frames is not helpful with respect to the privacy leaking. **Filtering** based methods apply the visual effects, such as the blurring or pixelating, on the entire frame to protect privacy. However, their processing does not distinguish the privacy and intelligibility, so that their outputs are not friendly to either human beings or AI. Some filtering work [42] takes the generative model into consideration, but its protection is not reversible either. **Transformation** based methods like PAN [35] can transfer the video frame into another representation, e.g., the pixel-reduced image or immediate features, regarding a specific prescient AI task. The transformed results do not preserve the intelligibility for both visual perception and AI understanding.

Additionally, there are also hardware-assisted solutions to protect the video privacy by interfering with the video capture [57] or introducing new TEE design [39, 40]. These solutions are not for protecting VSAP and are orthogonal to our solution. For example, Visor [40] is a recent TEE-based solution for video analytics privacy protection. Visor makes the video analytics privacy-preserving by eliminating side-channel attacks when running inside a CPU + GPU secure enclave system. As a result, Visor develops techniques to make the analytics modules oblivious (but uses the videos as provided). On the other hand, our solution PECAM preserves the privacy of the contents in the videos by developing techniques to transform the videos themselves before providing them to the analytics modules (but using the analytics modules as provided). The techniques in Visor and PECAM are complementary and can work together - PECAM provides its transformed video to Visor for analytics. Then, Visor uses its CPU + GPU enclave system to analyze the transformed video while protecting against side-channel attacks.

2.2 Deep Information Concealment

Deep information concealment is a body of related works leveraging the deep learning techniques to conceal secrets, such as the binary message, in a cover image [11, 12, 16, 55]. It usually consists of two neural network parts, Transformer for hiding secrets and Reconstructor for recovering secrets. Currently, it is proposed to use in the secret transmission or digital watermarking. Although PECAM utilizes deep learning to conceal secrets, it is different from the deep information concealment in terms of the concealment goal. Roughly speaking, existed works along this line aim to train a *unified strategy concealing an arbitrary information into the fixed image*, while PECAM aims to train a *unique strategy concealing part of arbitrary image into the image itself.* The uniqueness means PECAMs of two front-end video sources do not share the same concealing (i.e., steganography) strategy.



(a) A simple GAN translating images from one domain to another, e.g., zebra→horse. ***(b) The CycleGAN is to transform the image back and forth between two style domains (e.g., zebra↔horse). The same Transformer and Reconstructor pair is trained in both left and right procedures to achieve a good transformation quality.***

Figure 3: Image-to-image translations using GANs. (a) One-way translation; (b) Two-way translation. Transformer and Reconstructor are known as generators in a standard GAN and CycleGAN. (Photos are borrowed from CycleGAN [56].)

2.3 Background of GANs

GAN [22] is a class of unsupervised learning models trained with an adversarial process. Closely related to our work, a popular application of GAN is image-to-image translations [51, 56]. Figure 3(a) shows an example of one-way image translation. There are two competing neural networks called Transformer (generator) and discriminator. The Transformer tries to translate a zebra image in domain-X into a horse image in domain-Y. The discriminator is responsible for distinguishing the translated image from the real one in domain-Y through the adversarial loss function $\mathcal{L}_{GAN}^{X\to Y}$ [22]. The adversarial loss is calculated as:

$$\begin{aligned}\mathcal{L}_{GAN}^{X\to Y} =&\mathop{\mathbb{E}}_{X}\mathop{\mathbb{E}}_{Y}(log(\text{Discriminator}(y))\\ &+log(1-\text{Discriminator}(\text{Transformer}(x))))\end{aligned} \tag{1}$$

, where $\mathop{\mathbb{E}}_{X}$ and $\mathop{\mathbb{E}}_{Y}$ are the expectation over distribution of X and Y, respectively. In the rest of paper, if an adversarial loss is applied, a Discriminator component will also be added to the neural architecture accordingly.

Recently, the *cycle consistency* principle, an idea leveraging transitivity to regularize structured data, has been applied to design various advanced GANs [15, 51], including CycleGAN [56]. CycleGAN is a cycle-consistent GAN that enables two-way image translation between two style domains by preserving the semantics. As shown in Figure 3(b), two cycle-consistent losses are introduced, apart from the two GAN losses, in CycleGAN, to train a Transformer-Reconstructor pair to accurately translate an image from domain-X to domain-Y and revert it back. The forward-consistency loss $\mathcal{L}_{CYC}^{X\to Y\to X'}$ is to make $x' = \text{Reconstructor}(\text{Transformer}(x)) \approx x$ for each sample x in domain-X, while the backward cycle-consistency loss $\mathcal{L}_{CYC}^{Y\to X\to Y'}$ is to make $y' = \text{Transformer}(\text{Reconstructor}(y)) \approx y$ for each sample y in domain-Y. A Transformer can be used to translate a zebra image into a horse image without losing semantics, and thus Reconstructor can revert the translation back into a zebra image.

3 DESIGN OVERVIEW

In this section, we elaborate on the five design objectives of the VSAP enhancement, which are briefly introduced in the introduction section. We then present the PECAM design at a high level, with respect to these objectives. More design details are explained in the following section.

3.1 Design Objectives

We consider an ideal VSAP enhancement should achieve five objectives below altogether.

O1 *Privacy-enhanced.* The proposed design should reliably impede the unauthorized access of VSAP information throughout whole frames of original videos, given various possible situations that might be encountered in practice. For example, an object under protection should remain inaccessible in every single video frame, no matter it is changing pose or partially occluded. Please recall that the VSAP is defined in the introduction section. The VSAP information includes, but is not limited to, object details like texture and distinguishable features like facial appearance.

O2 *Intelligible.* The proposed design should genuinely preserve in protected videos the intelligible information of foreground objects, such as the object contour, color, posture, and localization, which are critical semantics to VSA back-end tasks done by both human beings and AI. Since the relationship between **O1** and **O2** is a trade-off, a configurable parameter is desired to adjust these two objects according to the practical needs (Section 4.1).

O3 *Securely-Reversible.* The proposed design should be able to securely recover its transformed video frames back to their original versions with no auxiliary data, with the presence of an adversary, and only upon the request from the authority. Our security model is given in Section 5.

O4 *Bandwidth-friendly.* The proposed design should greatly reduce the bandwidth usage of streaming protected videos. The transmission algorithm should be compatible with the existing video codecs such as H.264, and thus no change is necessary on existed VSA back-end.

O5 *Efficient.* The proposed design should ensure its privacy enhancement can efficiently run on front-end video sources with limited resources. The protected videos are able to be streamed in real time, and no local storage is required for the data buffering.

3.2 Design Overview

The core of PECAM is a privacy-enhanced securely-reversible visual transformation for VSA, which is empowered by our *security-reinforced cycle-consistent GAN.* This novel GAN applies the *cycle-consistency mechanism* to robustly enable the in-place transformation of video frames; it introduces a *reinforced steganography approach* to securely enable the two-way (reversible) transformation only for authorized parties; it leverages the *visual style conversion* to adjustably enable the privacy-intelligibility balancing. To achieve the real-time transformation on mobile devices, PECAM also optimizes the workflow of its in-use stage by *reducing the runtime computation cost* and *the network bandwidth usage.*

There are five system components of PECAM. **Transformer**, **Reconstructor**, and **DataGen** are related to our GAN design, while **FastTranser** and **Compressor** are used in real-time performance

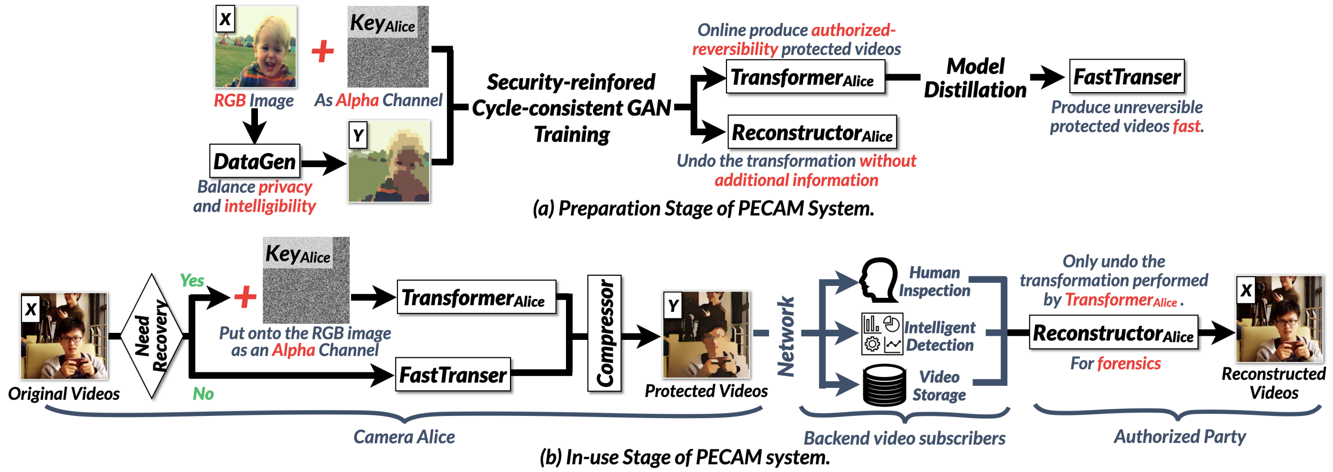


Figure 4: The PECAM overview and its workflow. Parts marked in rectangular box are system components of PECAM. The PECAM workflow consists of two stages, the one-time preparation stage and the in-use stage after deployment. There is no change on back-end video subscribers.

optimization for the execution and transmission, respectively. These components are involved in the two stages of PECAM, the preparation stage and the in-use stage, as shown in Figure 4. Please note that PECAM is deployed only at the front-end video source side of a VSA system.

Take the workflow of enabling PECAM on a camera named Alice as an example. In the preparation stage (Figure 4(a)), PECAM is trained with the assistance of DataGen specifically for camera Alice. The trained model is divided and wrapped into two paired components, i.e., Transformer $_{Alice}$ and Reconstructor $_{Alice}$. The model distillation is also performed on Transformer $_{Alice}$ to obtaining a lite-transformer FastTranser. When the preparation is completed, the camera Alice is deployed with Transformer $_{Alice}$, FastTranser, and Compressor installed on it. In the in-use stage (Figure 4(b)), PECAM on camera Alice orchestrates Transformer $_{Alice}$ and FastTranser to efficiently transform the real-time video streaming, and then it sends transformation outputs to Compressor for compression before transmission. Upon receiving the transformed privacy-enhanced video, the back-end video subscribers can directly take it as input in various existing VSA tasks. When forensics is requested for some video frames of camera Alice, the authority can use Reconstructor $_{Alice}$ to restore requested frames to their original version. The unique pairing between Transformer $_{Alice}$ and Reconstructor $_{Alice}$ guarantees no other Reconstructor $_{Bob}$ can perform the correct de-transformation.

More concretely, DataGen (Section 4.1) in the preparation stage is to configure the trade-off between **O1** and **O2** at the whole frame level. As a rule-based visual style [1] converter, DataGen determines PECAM's privacy enhancing granularity via the manipulation of domain-Y training data in a quantitative way.

Transformer essentially is the forward-cycle part of our GAN neural architecture, while Reconstructor is the paired backward-cycle counterpart. Our GAN introduces a novel "secret-key" scheme to guide a pair of Transformer and Reconstructor to secretly agree on the unique steganography. The steganography used by a Transformer can securely hide the VSAP information of an original frame in place of the frame itself; the hidden VSAP information only can be restored from the transformed frame by the paired Reconstructor (**O3**). Please note that there is no auxiliary data generated or used by Transformer and Reconstructor. Additionally, this GAN-based steganography also ensures that the transformed video is friendly to both visual perceptions and AI understanding. The core idea of this "secret-key" scheme is that we generate a *device-specific secret key* and apply it as the *Alpha channel* on the original RGB-format video frame, which becomes the RGBA-format, before performing the transformation. More details are in Section 4.2.

FastTranser is obtained by applying the model distillation technique on Transformer. It is lightweight by nature, but the reversibility is discarded. PECAM exploits it in cooperation with Transformer so that the transformation is efficient enough to achieve the real-time execution (**O5**). If a frame is determined to be securely-reversible, Transformer is used to process the data; otherwise, FastTranser is used. Frames transformed by either FastTranser or Transformer are encoded to an H.264-compatible video through Compressor, which is designed to meet **O4**. Compressor is necessary because the vanilla H.264 could downgrade the reversibility of our transformed videos. More details of these two parts are provided in Section 4.4 and Section 4.3, respectively.

4 KEY DESIGNS

This section elaborates on the PECAM technical details in depth. The first two subsections focus on our security-reinforced cycle-consistent GAN design, while the other two subsections present how we optimize the network bandwidth and computation cost, respectively.

4.1 VSAP Enhancement

Our security-reinforced cycle-consistent GAN, like most deep learning based technologies, requires a training phase, in which our GAN can implicitly learn a proper VSAP enhancement level from

[1]The "style" (aka. artistic style) means the rendering of the images' semantic content [21], which is often used in style transfer task.

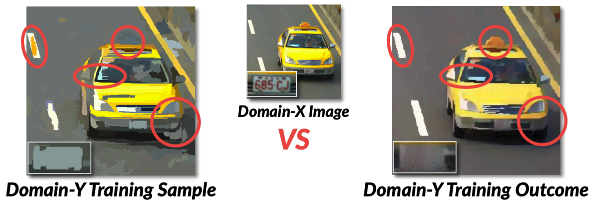


Figure 5: Quality Comparison of domain-Y frames produced by DataGen (left) and PECAM (right) from the same domain-X input (middle). The left frame has several conversion errors on the vehicle body and road marks, which are corrected in the right frame.

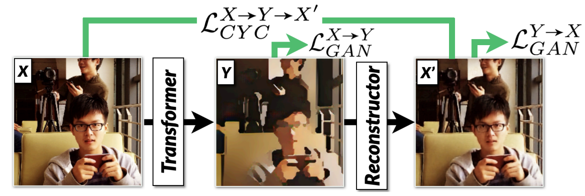


Figure 6: Cycle-consistent-GAN-based video transformation with two adversarial losses ($\mathcal{L}_{GAN}^{X\rightarrow Y}$ and $\mathcal{L}_{GAN}^{Y\rightarrow X}$) and a recoverable loss ($\mathcal{L}_{CYC}^{X\rightarrow Y\rightarrow X'}$). These losses are the same as that of CycleGAN, which are introduced in Section 2.3.

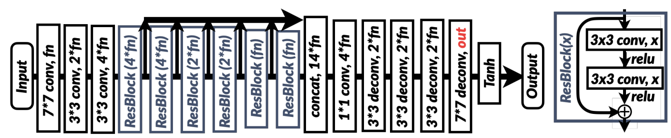


Figure 7: Neural-network architecture of Transformer and FastTranser. FastTranser has much fewer filter channels than Transformer does (fn and out are channel numbers). fn is 64 in Transformer and 16 in FastTranser. out is 3 in both Transformer and FastTranser.

the datasets customized by DataGen. DataGen is an easily-configurable visual style tool to convert the real-world images (domain-X) into the privacy-enhanced ones (domain-Y). The pairs of images in the two domains are the training dataset for our GAN.

According to our VSAP definition given in the introduction section, we consider domain-Y as the cartoon style rendering like examples shown in Figure 13. This cartoon style belongs to the non-photorealistic rendering [28], which is popular to create various human-pleasant types of visual artifacts, such as penciling and oil painting [21]. Besides being friendly to the perception, our cartoon style also pays special attention to only preserving the main semantics of real-world videos, such as the contour, color, posture, and location, which are critical to VSA back-end tasks. Compared to the original video, the corresponding domain-Y video greatly improves privacy and raises the attack efforts of an adversary.

Technically, DataGen consists of a color-oriented segmentation and a recoloring schema. Given a real-world video frame, DataGen first applies the bilateral filter [47], an edge-preserving smoothing filter, to reduce the color number. DataGen then performs the simple image segmentation [19] to quickly locate the outline and edges of possible objects. Lastly, segmented parts are recolored with the average of original colors of their corresponding areas, respectively, and this frame is finally converted into domain-Y at the whole frame level.

To balance privacy and intelligibility in the converted domain-Y video, we introduce a parameter $\boldsymbol{k}$ in the image segmentation phase to configure the segmented block size. DataGen prunes all visual details smaller than the segmented block size and produces the domain-Y data with some quantifiable granularity. This configurable granularity will be eventually picked up by our trained GAN as its proper privacy enhancement level. Figure 12(a) in the evaluation section gives PECAM examples of how privacy enhancement is influenced by k.

Please note that DataGen is only used to indicate learning targets, i.e., the suitable domain-Y style and enhancement level, to our GAN in training. This is because, although easy to configure the privacy granularity, DataGen's converting quality is not ideal for VSA back-end tasks. However, the imperfect training data is not a problem for our GAN due to its powerful neural architecture. Figure 5 illustrates that our trained GAN significantly outperforms DataGen in terms of preserving analytic semantics in transformation.

4.2 Secure Reversibility

In terms of neural architecture, our security-reinforced cycle-consistent GAN can be divided into Transformer and Reconstructor, which are paired via the training phase to transform videos between domain-X and domain-Y versions. The cycle-consistency mechanism offers a powerful semantics-aware ($X \rightarrow Y$)-transformation for Transformer, which is also the key to overcome the data quality issue of DataGen (illustrated in Figure 5). The reinforced steganography approach enables a novel secure ($X \leftarrow Y$)-transformation for Reconstructor, which only takes as input the targeted transformed frames, in the presence of an adversary. A pair of Transformer and Reconstructor is uniquely trained for each front-end video source, although the neural architecture design of all Transformer/ Reconstructor is the same.

Cycle Consistency Mechanism. Figure 6 illustrates the cycle consistency mechanism applied in our GAN. Compared to the standard cycle-consistent GAN like CycleGAN, we make ours more lightweight by modifying the loss function and introducing a lite neural architecture.

The cycle-consistent loss of our GAN $\mathcal{L}_{cyc}$ can be represented by three parts, two adversarial losses (explained in Section 2.3), i.e., $\mathcal{L}_{GAN}^{X\rightarrow Y}$ and $\mathcal{L}_{GAN}^{Y\rightarrow X}$, and one forward cycle-consistent loss defined as below:

$$\mathcal{L}_{CYC}^{X\rightarrow Y\rightarrow X'} = \mathop{\mathbb{E}}_{X}(\mathcal{D}_1(\text{Reconstructor}(\text{Transformer}(x)), x)) \quad (2)$$

, where $\mathcal{D}_1$ is the MS-SSIM metric [4], which is used to measure the perceived quality of the image, and $\mathop{\mathbb{E}}_{X}$ is the expectation over distribution of X.

Thus, the first goal of the training is to minimize

$$\mathcal{L}_{cyc} = \mathcal{L}_{GAN}^{X\rightarrow Y} + \mathcal{L}_{GAN}^{Y\rightarrow X} + \lambda_1 \mathcal{L}_{CYC}^{X\rightarrow Y\rightarrow X'} \quad (3)$$

, where λ_1 denotes the weight of cycle consistency.

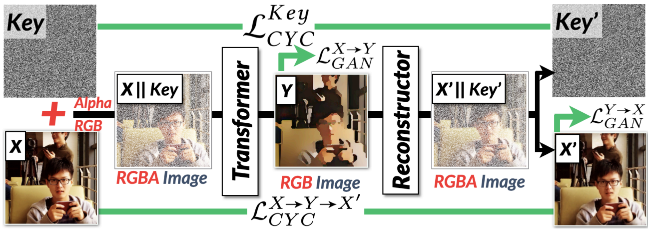


Figure 8: Security-reinforced cycle-consistent GAN for authorization-reversible video transformation. There are two adversarial losses ($\mathcal{L}_{GAN}^{X\to Y}$ and $\mathcal{L}_{GAN}^{Y\to X}$) and two recoverable losses ($\mathcal{L}_{CYC}^{X\to Y\to X'}$ and $\mathcal{L}_{CYC}^{Key}$). The adversarial losses are the same as the CycleGAN.

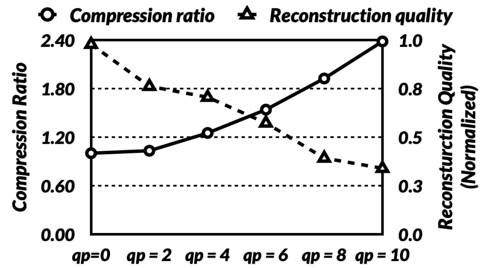


Figure 9: Directly using the H.264 codec to compress the transformed frames generated by Transformer significantly reduces the reconstruction's quality. We perform the face recognition on the recovered video to measure the reconstruction quality. Parameter qp is used to control the compression ratio of H.264. A larger qp means a lager compression ratio, and $qp = 0$ means lossless compression.

To support the real-time transformation on mobile devices, we also design a lightweight neural architecture (Figure 7) for Transformer. This new neural architecture first reduces the eight 256-channel ResNet blocks [27] of CycleGAN to a set of two 256-channel blocks, two 128-channel blocks, and two 64-channel blocks. To compensate for the accuracy loss of this change, it then adds some shortcut paths that are similar to ones used in the inception network. Each path concatenates outputs of a ResNet block and passes the results to a 1×1 convolutional layer. In this way, the high-level and low-level semantic features can be effectively fused for better restoration. Transformer using it is able to run 1.8× - 2.4× faster than the standard CycleGAN counterpart.

Reinforced Steganography Approach. Previous study [16] shows that a cycle-consistent GAN is good at hiding visual information in place via its highly-sophisticated steganography. However, such steganography is not fully secure if many front-end video sources install this GAN. The transformed videos of one Transformer can be reversed/restored by another unpaired Reconstructor. Thus, an adversary can buy one device and easily compromise the VSAP of all other devices.

Is it possible to provide each front-end video source with a unique steganography strategy, empowered by the same GAN architecture trained with the same transformation goal? The answer to this challenging question builds the foundation for **O3**. To this end, we propose a reinforced steganography approach on top of our cycle-consistent GAN. As shown in Figure 8, this security reinforcement is achieved by introducing a unique key into the GAN training. By using different keys, two Transformer-Reconstructor pairs trained on the same dataset will learn different steganographic methods. That is, the Reconstructor of one security-reinforced GAN cannot recover the transformed videos generated by the Transformer of another security-reinforced GAN that trained with a different key. Consequently, PECAM supports secure video recovery and defends against the above real-world attacks. We perform security analysis in Section 5.

The key introduced in security-reinforced GAN is a two-dimensional matrix with the same width and height as the domain-X images (i.e., video frames). Each variable in the matrix is randomly set in the range of 0 to 255. This key can be considered as an additional digital watermark channel (i.e., alpha channel) attached to the 3-channel RGB image. Note that each device only needs to pick this secret key once for the whole training. No re-keying is required.

In the training stage for a device, the device's secret key is appended to domain X's training data as the fourth-channel watermark. Accordingly, the neural architecture of both Transformer and Reconstructor is updated to support the RGBA-to-RGB and RGB-to-RGBA transformation, respectively.

As to the loss function of this part, we introduce a recoverable loss

$$\mathcal{L}_{CYC}^{Key} = \mathcal{D}_2(Key', Key) \tag{4}$$

, where Key' is the reconstructed key of original Key, and $\mathcal{D}_2$ is the Manhattan distance metric. The MS-SSIM metric ($\mathcal{D}_1$) is applied to the image reconstruction to improve the reconstruction's visual quality, while the Manhattan distance ($\mathcal{D}_2$) is used during the key reconstruction to reinforce a unique steganography strategy.

Finally, the *overall objective of our security-reinforced cycle-consistent GAN* is minimizing

$$\mathcal{L}_{sec} = \mathcal{L}_{cyc} + \lambda_2 \mathcal{L}_{CYC}^{Key} \tag{5}$$

, where $\mathcal{L}_{cyc}$ is calculated as Equation 3. λ_1 in $\mathcal{L}_{cyc}$ and λ_2 are hyper-parameters (both are 5 in our experiments).

4.3 Bandwidth Reduction

It is problematic if PECAM directly uses existed codec like H.264 to compress transformed frames for the network transmission. As shown in Figure 9, even a small compression ratio of 2 (when qp [2] is 8) will largely drop the reconstruction quality by 60%. This is because the compression commonly prunes the high-frequency information containing part of PECAM's steganographic data, which later will be used for reconstruction.

We observe that high-entropy regions usually have more impacts on our reconstruction quality after compression-decompression procedures, compared to low-entropy regions. Therefore, we propose our own Compressor as follows, which is also compatible with existing H.264 lossy compression. Algorithm 1 illustrates the compression procedure. Compressor takes a set of transformed frames produced by Transformer as an input and extracts the high-frequency information of recoverable frames (*line 8-9*). Then, it

[2] ***qp*** is short for quantization parameter, which controls the amount of compression for a frame. Larger values lead to more compression and vice versa. ***qp*** ranges from 0 to 51.

cuts the high-frequency information of each frame into macroblocks of equal size, i.e., a block of 32×32 pixels (*line 11*) and calculates the entropy value of each macroblock (*line 13*). Next, Compressor retains the macroblock whose entropy value is higher than a certain threshold (denoted as *th*) and discards the rest high-frequency information (*line 14*). Then we produce the transformed video by putting the retained macroblocks back to the transformed video (*line 15*) and encode them with lossless H.264 compression (*line 16*). The threshold *th* is a trade-off between the reconstruction quality and bandwidth. We will discuss this hyper-parameter in Section 6.3.

Algorithm 1: Pseudo code of video compression.

Data: A set of transformed frames, $F = \{f_1, f_2, ..., f_n\}$; their corresponding indicators, $I = \{R, N, ...\}$ to indicate that the frame is produced by Transformer (R) or FastTranser (N); and a hyperparameter, th.
Result: A set of compressed streaming clips, $S = \{S1, S2, ...\}$

```
1  begin
2      S ← ∅, f_c = F[0], i_c = I[0], F_c ← {F[0]}
3      for i ∈ [1, len(I)) do
4          if I[i]! = i_c or i == len(I) − 1 then
5              if i_c == N then
6                  S.append(H264_lossy(F_c))
7              else
8                  INFO_l = filter(F_c)
9                  INFO_h = F_c − INFO_l
10                 for j ∈ [0, len(F_c)) do
11                     B = break_into_blocks(INFO_h[j])
12                     for k ∈ [0, len(B)) do
13                         if entropy(B[k]) <= th then
14                             B[k] = zeros(sizeof(B[k]))
15                     F_c[j] = recombine(B, INFO_l[j])
16                 S.append(H264_lossless(F_c))
17             f_c = F[i], i_c = I[i], F_c = {F[i]}
18         F_c.append(F[i])
19     return S
```

4.4 Execution Optimization

There is redundant information among continuous video frames, so it is unnecessary to make every single frame securely-reversible. Thus, we can further optimize the transformation pipeline on the camera device. FastTranser is then proposed to perform the non-reversible transformation, which can run around 6.4× faster than Transformer. Given both FastTranser and Transformer, we design an online branching strategy (Figure 4(b)) to decide which one shall transform an incoming frame, considering both runtime overhead and forensics need.

FastTranser takes the neural architecture of Transformer (Figure 7) and only keeps one quarter of original channels. Its training leverages the distillation [29] technique so that output frames, no matter transformed by FastTranser or Transformer, are visually consistent, which is important to VSA. The key part is applying

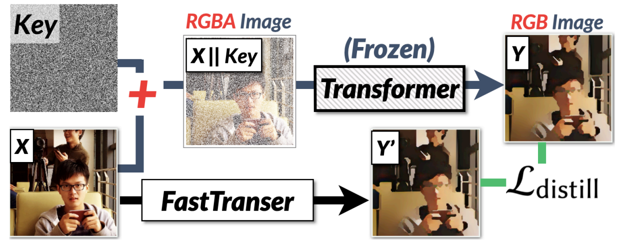


Figure 10: The training procedure of FastTranser through distillation. The Transformer is frozen during the training.

an adversarial loss to measure the difference between outputs of FastTranser and Transformer. We illustrate FastTranser's training structure in Figure 10. The training goal is to minimize

$$\mathcal{L}_{\text{distill}} = \mathcal{D}(\text{Transformer}(x||key), x) \tag{6}$$

, where the $\mathcal{D}$ is the manhattan distance metric. Note that Transformer is frozen during FastTranser's training procedure.

Our branching strategy exploits the temporal locality and data sparsity of videos. Two frames with a high perceptive similarity may not have to be both reversible. PECAM caches the latest frame forwarded to the Transformer, denoted as F_{latest}. When a new frame, denoted as F_{current}, captured by PECAM, we estimate the similarity using the equation: $\mathcal{H}(\mathcal{P}(F_{\text{latest}}), \mathcal{P}(F_{\text{current}}))$, where $\mathcal{P}$ is the perceptual hash (pHash) metric [1] which is widely used to capture image's visual perception features, and $\mathcal{H}$ is the Hamming distance, which is usually applied to the image's pHash values to measure the images' similarity. A larger distance represents less similarity. We set a threshold ***sp*** as the decision trigger. If the distance is larger than the *sp* value, PECAM forwards the incoming frame to Transformer; otherwise, it will be forwarded to FastTranser. A larger *sp* means lower cost and fewer reversible frames. In the pHash implementation [2] PECAM adopts, if the distance between two images' pHash values is less than or equal to 6, then these two images have visual perceptions that are nearly the same.

5 SECURITY ANALYSIS

5.1 Security Model

We assume that the preparation stage of PECAM is done by some trusted device manufacturers. After deployment, we also assume that the execution environment of PECAM is secured, and the whole device is properly guarded so that there is no illegal physical access to it. Additionally, we trust the authorized party performing forensics.

We do not trust back-end video subscribers of the VSA. One of such subscribers may be interested in collecting as much VSAP information as possible from the received video streaming. This adversary has rich computing resources and fully understands the neural architecture and all procedures of PECAM. It can even buy (or collude with) one device the same as the targeted victim and then own the trained models and the secret key specific to that device, which can be leveraged in its attacks.

However, we assume that the adversary cannot access the secret key of the targeted victim device and the training dataset of this device. Besides, we do not consider the malicious subscriber who

Figure 11: Visual comparison of recovered frame examples. Images labeled with attack 1 and 2 are recovered by adversaries in Section 5.

searches for a specific target with out-of-band information, e.g., the walking pattern [3].

The security goal of PECAM is to significantly raise the bar against the VSAP leaking on videos streamed to subscribers, and effectively prevent unauthorized subscribers from reversing PECAM's VSAP enhancement.

5.2 Enhancement Analysis

Given our security model, we analyze the privacy enhancement of PECAM from two perspectives.

First, is it much harder for an adversary to directly obtain the VSAP information from transformed videos, compared to original videos? To answer it, we conduct a comparison experiment on both transformed and original videos, which is detailed in Section 6.1. It demonstrates that almost no VSAP information can be recognized even by state-of-the-art (SOTA) techniques, which have proven to outperform human beings. Please note that PECAM enhances privacy at the whole-frame level without prior knowledge of ROIs, which is more robust and generic than existing solutions. The enhancement is persistent and effective across video streaming all the time. There are no corner cases like a person's identity is not properly protected because they are not facing the camera upright. Besides, PECAM's enhancement is configurable according to different balancing requirements of privacy and intelligibility.

Second, is it much more difficult for an adversary to indirectly obtain the VSAP information by reversing the transformation in unauthorized manners? To answer it, we analyze the feasibility of launching the following two attacks and conduct experiments to further support our analysis conclusions. Please note that PECAM's transformation is implicitly learned through an end-to-end procedure, so the best attack strategy for the adversary is also deep learning based.

[3]It is a costly attack by itself because the attacker does not know when and where the target is recorded in the VSA system

Attack 1. *Is it possible for the adversary to train another deep-learning model Reconstructor$_{adv}$, which can directly reverse the transformed video of targeted victim camera Alice Transformer$_{Alice}$?*

Analysis. In this attack, the adversary ignores the existence of secret key key_{Alice} and attempts to train an RGB-2-RGB transformation with public-available resources like training data and neural architectures. After the training, the adversary directly applies it onto the domain-Y video and generates the domain-X video. However, previous work on the message concealing [11] has shown the evidence that, if some knowledge is unknown to the adversary, the adversary cannot reconstruct concealed data. Similarly, in our case, Reconstructor$_{adv}$ trained without key_{Alice} cannot understand how Transformer$_{Alice}$ works and therefore fails to reverse the transformation.

Experiment. We allow the adversary to train a standard CycleGAN, even with the training data used by camera Alice (which is not available to the adversary in practice). After training, we compare the reconstruction performance of Reconstructor$_{adv}$ and Reconstructor$_{Alice}$ on 1,000 transformed frames of Transformer$_{Alice}$. These 1,000 images are randomly selected from the face recognition dataset PIPA [53]. We then run a state-of-the-art face recognition service [7] on these two sets of reconstructed results and discover that all recognized faces in data of Reconstructor$_{Alice}$ cannot be recognized in data of Reconstructor$_{adv}$. The second column of Figure 11 illustrates some examples reversed by Reconstructor$_{adv}$. Attack thus fails.

Attack 2. *Is it possible for the adversary to buy (or collude with) the camera Bob, which is of the same type as camera Alice, and leverage Reconstructor$_{Bob}$ to reverse the transformed video of Transformer$_{Alice}$?*

Analysis. Each trained GAN can be viewed as a complex nonlinear function implicitly capturing the hidden mapping between two high-dimensional domains [17]. When we leverage the device-specific secret key to manipulate the GAN training, the hidden mapping learned by a model is greatly different from others. This factor can be magnified by the nonlinearity of neural architectures [18, 24, 30, 38] and leads to various steganography strategies, each of

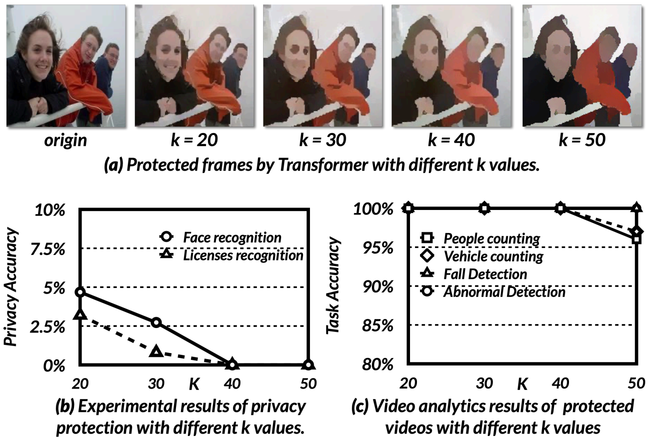


Figure 12: Exploration of how k affects the privacy protection and intelligibility.

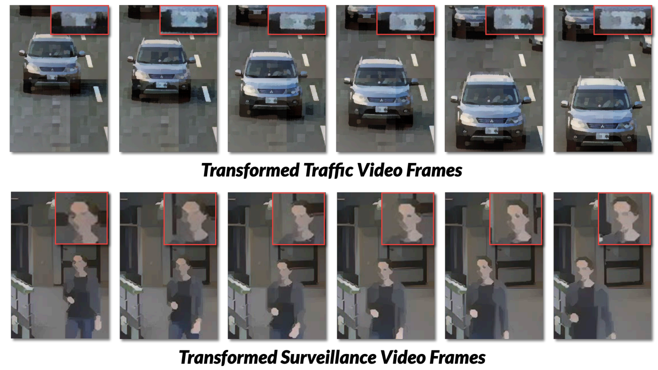


Figure 13: Two examples of video playback (consecutive video frames) after transformation.

which is only known by one trained GAN. Therefore, the adversary cannot use Reconstructor$_{Bob}$ to reconstruct the original video contents from transformed videos of Transformer$_{Alice}$.

Experiment. We train ten Transformer-Reconstructor pairs with different secret keys. We then try using one Reconstructor to reverse the transformed outputs of the other nine non-paired Transformers. For each Transformer, 1,000 images picked from PIPA [53] dataset are used for the transformation. We repeat the whole experiment three times. Experimental results show that non-paired Reconstructors cannot successfully recover all transformed frames. The third column of Figure 11 shows some failure examples.

6 EVALUATION

Our evaluation starts with measuring the privacy enhancement of PECAM transformation. We then examine the intelligibility of transformed videos with respect to representative back-end applications of video analytics. We also conduct experiments to discover how our compression algorithm strikes a good balance between recovery quality and bandwidth cost. We finally evaluate resource usages of PECAM.

Prototype. We implement a PECAM prototype with the MNN deep-learning framework [5] on a typical mobile dev board Qualcomm Snapdragon 845. The Snapdragon 845 has a Kryo 385 CPU, an Adreno 630 GPU, and 6GB memory on its SoC. PECAM is trained with 4,400 X-Y domain data pairs. Each pair consists of one image picked from PIPA dataset [53] and its corresponding transformed image prepared by DataGen. We adopt the training setting of the standard CycleGAN. After the training, PECAM is deployed onto the dev board with the Linux 4.9.

Knobs. With the tunable parameters of k, th, and sp, PECAM provides multiple knobs to achieve flexible trade-offs among the achievable privacy protection. k is the parameter in DataGen to control the granularity of privacy protection. A larger k makes transformed videos have fewer content details. th is the parameter in Compressor balancing the network communication cost and video recovery quality. sp is the similarity threshold to trigger the FastTranser.

Metrics. There are several metrics in our experiments. ***Privacy accuracy*** is measured as the private information detection, i.e., face recognition and license recognition, performance of some state-of-the-art methods. When measuring on transformed videos, a low value means a low risk of privacy leakage, indicating that PECAM has strong privacy protection; When measuring on recovered videos, a high value means concealed information is restored, indicating that PECAM achieves good reversibility. We also refer to the privacy accuracy of recovered videos as the ***recovery accuracy*** metric in this section. Additionally, the ***task accuracy*** is measured as the performance ratio of some video analytics tasks on transformed videos and original videos. 100% means transformed videos maintain good intelligibility.

6.1 Privacy Enhancing

Since advanced privacy detection methods have outperformed human beings [3], we pick these methods, the state-of-the-art face and license recognition algorithms[4] [7], to test whether transformed videos contain recognizable privacy information. The metric used is the privacy accuracy.

We select four pairs of Transformer and Reconstructor trained with k value of 20, 30, 40, and 50 respectively to observe different privacy protection strength (Figure 12(a)). Our evaluation utilizes 6,000 video frames, which are not seen in the training, from real traffic cameras [10] and CCTV cameras [48].

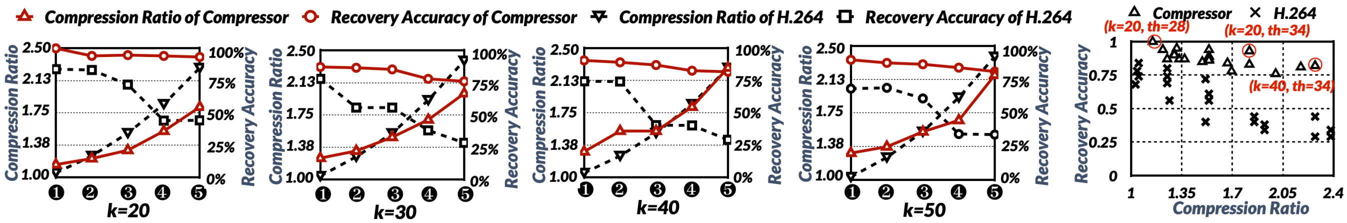
Experimental results are shown in Figure 12(b). As the value of k gets larger, the privacy accuracy decreases, indicating that the protection capability increases. Even in the worst case (k = 20), the privacy accuracy is less than 5%, which means only 5% privacy information out of all in original videos is able to be recognized in transformed videos by the state-of-the-art methods. The privacy accuracy of human viewers would be even lower.

6.2 Intelligibility Preserving

We evaluate task accuracy of transformed videos in terms of five commonly-seen VSA back-end tasks. The results, except the video playback, are shown in Figure 12(c).

People Counting. We pick 4,000 video frames from a well-known dataset [48] and transform them respectively by using Transformers

[4]We use the text recognition service to perform the license recognition task.



(a) The compression performance of Compressor and H.264 in terms of compression ratio for transmission and recovery accuracy for forensics. qp is the control parameter in H.264. th is the parameter in our Compressor. ❶ for qp=2 & th=26; ❷ for qp=4 & th=28; ❸ for qp=6 & th=30; ❹ for qp=8 & th=32; ❺ for qp=10 & th=34.

(b) Performance summarization of various configurations. Three circled points of PECAM make the good trade-off balancing. We suggest to pick (k=20, th=34).

Figure 14: The compression performance of Compressor and H.264 in terms of compression ratio for transmission and recovery accuracy for forensics.

trained with different k values. We then perform the people counting task on both original and transformed videos by leveraging a commonly-used approach YoloV3 [43]. As shown in Figure 12(c), when $k = 20$, $k = 30$, and $k = 40$, PECAM transformation has no impact on this task. A larger k value may lead to overprotection, which will slightly affect the task accuracy. For example, when $k = 50$, the task accuracy is 96%.

Vehicle Counting. We pick 2,000 video frames from a well-known dataset [10], and transform them respectively by using Transformers trained with different k values. We then perform the vehicle counting task on both original and transformed videos by leveraging a commonly-used approach YoloV3 [43]. As shown in Figure 12(c), when $k = 20$, $k = 30$, and $k = 40$, the performance on transformed frames is the same as on original ones (100% task accuracy). When k is equal to 50, the task accuracy drops to 97%.

Fall Detection. We randomly select 30 videos from the fall detection dataset [31] and get their corresponding transformations via PECAM. Our task accuracy is also 100% regarding the SOTA fall detection [8], indicating there is no difference between the original and transformed videos for this VSA back-end task.

Abnormal Event Detection. We select 20 videos from the abnormal event detection dataset [37] and generate their corresponding transformed ones. There are three kinds of abnormal events in videos, the strange action, wrong direction, and abnormal object. We then apply the SOTA detection [36] on both original and transformed videos. PECAM still maintains 100% task accuracy.

Video Playback (Human Viewing). We can observe that PECAM's transformed videos (Figure 13) have a good consistency of color scheme and brightness. The information concealed by PECAM in the transformed frames does not affect the user experience of watching.

6.3 Bandwidth Usage v.s. Recovery Quality

In this part, we evaluate our Compressor by comparing it with the typical H.264 video compression in terms of how they affect the reversibility. We select the transformed videos as the compression target of transmission and the privacy information recognition method in Section 6.1 as the method measuring the recovery accuracy. Each transformed video is first compressed with different configurations (k-th pairs). We then calculate its compression ratio, which reflects the bandwidth cost of transmitting this video. Afterward, we decompress and recover the compressed transformed

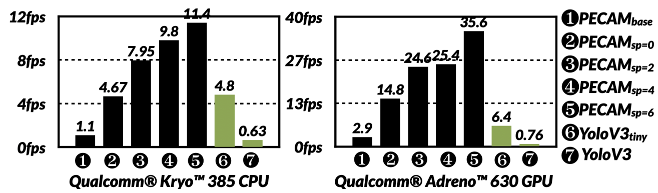


Figure 15: Frame rates of PECAM transformation with different deep learning networks and FastTranser settings. The frame rate of YoloV3$_{(tiny)}$ is shown as baseline for comparison.

videos and measure their recovery accuracy, which is introduced at the beginning of this section. We also repeat the above experiment procedure by replacing PECAM compression with the H.264 compression. Compression levels in the H.264 case are controlled by the k-qp pairs.

The comparison results of PECAM and H.264 compression cases are shown in Figure 14(a). For all experiments, PECAM compression always offers better recovery accuracy than the H.264 compression. Actually, the recovered videos transmitted by using H.264 will have great negative impacts on the usability of forensics, as its highest recovery accuracy is only 84% (compression ratio is only 1.05). While maintaining a good recovery accuracy (e.g., >90%), PECAM compression cases also reach better compression ratios than the H.264 cases. Therefore, PECAM Compressor is suitable and effective in reducing the bandwidth cost of videos containing self-reversible information. We also summarize results of different knob configurations in Figure 14(b), and find out that a k-th pair with balanced performance could be 20-34, which delivers a 93% recovery accuracy with a 1.82 compression ratio. When the recovery accuracy is the same (i.e., 0.8), the Compressor's compression ratio is 1.8× that of H.264.

6.4 Real-Time Performance

We demonstrate the efficiency of PECAM's computing cost optimizations, in terms of the frame rate, with its comparison with CycleGAN and YoloV3.

In Figure 15, PECAM$_{sp=x}$ is the case if we set $sp = x$ for our PECAM prototype. A larger sp leads to a higher frame rate. Recall if the pHash distance of two frames is less than or equal to 6 in the

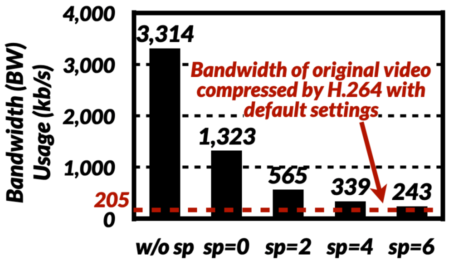


Figure 16: Bandwidth (BW) usage of video transmission with different sp settings. w/osp represents that the trigger strategy is disabled. We set k to 20 and set th to 34, as recommenced in Figure 14(b).

implementation adopted by PECAM [2], these two frames have almost no perception difference. PECAM$_{base}$ is the case when we replace our PECAM prototype's light neural architecture with the one proposed in CycleGAN, and let it process every frame. Additionally, we also compare the performance of PECAM operations with other deep-learning empowered operations, namely YoloV3 and YoloV3$_{tiny}$ [5], in terms of frame rate of video processing. Please note that YoloV3/YoloV3$_{tiny}$ is used to perform object detection tasks on video frames, rather than the video transformation, and is famous for its high accuracy.

We perform the experiment with 10 video clips randomly selected from the real-world surveillance video dataset ChokePoint [48]. PECAM takes these videos as input, protects them, and sends them to back-end subscribers. Experimental results show that, just by applying our light neural architecture, PECAM$_{sp=0}$ is 4.2× and 5.1× faster than the PECAM$_{base}$ on CPU and GPU, respectively. And with the help of FastTranser and mobile GPU, PECAM can reach a 35.6fps when $sp = 6$. Compared to the video processing of YoloV3, PECAM with $sp = 6$ achieves an 18.1× and 46.8× framerate speedup on CPU and mobile GPU, respectively. The storage usage of the PECAM system is 17.6MB. The memory used by PECAM system during running is 200MB. In the CPU mode, the CPU utilization is about 85.8%; while in the GPU mode, the GPU utilization is about 99% and the CPU utilization is about 3%.

Moreover, FastTranser also benefits the overall bandwidth cost because it produces frames that do not need recoverability. Therefore, a more aggressive lossy compression can be applied to such frames. Figure 16 shows, when sp is set to 6, the network bandwidth required for transmission is further reduced by 93% on top of the bandwidth reduction by PECAM compression.

7 DISCUSSION

Limitations. PECAM has three limitations that can be improved in the future. *First*, we do not theoretically guarantee that no attack is able to reverse PECAM's privacy enhancement. We plan to provide more theoretical supports for PECAM with the help of recent advances in deep learning interpretability. *Second*, PECAM cannot completely preserve the VSAP, although it greatly enhances privacy. For example, when objects are extremely close to the camera, their partial details could be leaked under the protection of PECAM. The leaking level is determined by the configurable parameter k, which balances usability and privacy. We will consider the auto or adaptive tuning of k in the future. *Third*, PECAM by design does not protect the categorizable information, behavioral information, or spatial information because such information is commonly used in VSA analytics tasks.

Scalable Deployment. To defend against the adversary, PECAM enables each camera to possess its own PECAM instance, which achieves the same visual transformation goal with a unique secret transformation method. Thus, the PECAM instance has to be trained for each camera, which is not convenient for large-scale deployment. Here we discuss two potential strategies, i.e., short-term and long-term, to mitigate this scalability issue. In the short term, we could apply the pre-training technique to improve training efficiency. Pre-training is widely used to quickly fine-tune different models from the basic one. We could also group cameras together according to the geographic area or ownership and just train one model for every group. In the long term, we envision both the hardware and algorithms will be more powerful, given the current fast development of deep learning. New training procedure and training platform will help us to propose more efficient PECAM deployment on millions of devices.

Prototype Hardware. We build the PECAM prototype on the Qualcomm Snapdragon 845, which is comparable to the mid-range or high-end off-the-shelf (COTS) cameras for VSA. For example, in terms of the GPU ability, some high-end COTS cameras [9] are about 2-8× more powerful than the Snapdragon 845. Moreover, the dedicated AI hardware accelerators [50] have emerged to deliver excellent mobile deep learning capabilities. We advocate that smart cameras in the future shall all run certain on-device intelligent solutions like PECAM to improve privacy.

8 CONCLUSION

PECAM is a VSAP enhancement system featuring a GAN-based video transformation. It runs on the VSA front-end to enhance whole-frame privacy over real-time video streaming. Its outputs can be directly taken as inputs by various existed VSA tasks in the back-end. PECAM also makes transformed video securely-reversible in the presence of an adversary. We believe PECAM's design is generic enough to become an intelligent platform hosting future privacy solutions for video streaming and analytics.

ACKNOWLEDGMENTS

We sincerely thank the reviewers and anonymous shepherd for their valuable comments helping us to improve this work. We thank Yuanchao Shu for his valuable suggestions on this work. This work was supported in part by NSFC-61872180, Jiangsu "Shuang-Chuang" Program, and Jiangsu "Six-Talent-Peaks" Program. Fengyuan Xu was supported in part by Ant Financial through the Ant Financial Science Funds for Security Research. Sheng Zhong was supported in part by NSFC-61872176, The Leading-edge Technology Program of Jiangsu Natural Science Foundation (No. BK20202001), and National Key R&D Program of China under Grant 2020YFB1005900.

[5] The mAP (mean average precision) of YoloV3$_{tiny}$ is only 60% of that of the full-version YoloV3.

REFERENCES

[1] 2008. pHash: The open source perceptual hash library. http://phash.org/. [Online; accessed Jul-28-2020].

[2] 2013. A perceptual hash algorithm. https://github.com/bunchesofdonald/photohash. [Online; accessed Jan-28-2020].

[3] 2015. Microsoft's Deep Learning Project Outperforms Humans In Image Recognition. https://www.forbes.com/sites/michaelthomsen/2015/02/19/microsofts-deep-learning-project-outperforms\-humans-in-image-recognition/#3134172b740b. [Online; accessed Jan-28-2020].

[4] 2018. Pytorch MS-SSIM. https://github.com/jorge-pessoa/pytorch-msssim. [Online; accessed Jul-28-2020].

[5] 2019. Mobile Neural Network. https://github.com/alibaba/MNN. [Online; accessed Jan-28-2020].

[6] 2020. EU makes move to ban use of facial recognition systems. https://sciencebusiness.net/news/eu-makes-move-ban-use-facial-recognition-systems. [Online; accessed Jan-28-2020].

[7] 2020. Face++. https://www.faceplusplus.com.cn/. [Online; accessed Jan-28-2020].

[8] 2020. Fall Detection Algorithm. https://github.com/qiaoguan/Fall-detection. [Online; accessed Jan-28-2020].

[9] 2020. NVIDIA Jetson Cameras. https://www.e-consystems.com/nvidia-jetson-camera.asp. [Online; accessed Dec-14-2020].

[10] 2020. Traffic Video. https://www.lumenera.com/media/wysiwyg/video/surveillance/Li165C-DN.mp4. [Online; accessed Jan-28-2020].

[11] Martín Abadi and David G Andersen. 2016. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918* (2016).

[12] Shumeet Baluja. 2017. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*. 2066–2076.

[13] Martin Bertran, Natalia Martinez, Afroditi Papadaki, Qiang Qiu, Miguel Rodrigues, Galen Reeves, and Guillermo Sapiro. 2019. Adversarially learned representations for information obfuscation and inference. In *International Conference on Machine Learning*. 614–623.

[14] Paulo Vinicius Koerich Borges, Nicola Conci, and Andrea Cavallaro. 2013. Video-based human behavior understanding: A survey. *IEEE transactions on circuits and systems for video technology* (2013), 1993–2008.

[15] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. 2018. CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 9465–9474.

[16] Casey Chu, Andrey Zhmoginov, and Mark Sandler. 2017. CycleGAN, a Master of Steganography. *Computing Research Repository* (2017). http://arxiv.org/abs/1712.02950

[17] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine* (2018), 53–65.

[18] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 4829–4837.

[19] Pedro F Felzenszwalb and Daniel P Huttenlocher. 2004. Efficient graph-based image segmentation. *International journal of computer vision* (2004), 167–181.

[20] Oran Gafni, Lior Wolf, and Yaniv Taigman. 2019. Live face de-identification in video. In *Proceedings of the IEEE International Conference on Computer Vision*. 9378–9387.

[21] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Neural Information Processing Systems*. 2672–2680.

[23] Ralph Gross and Latanya Sweeney. 2007. Towards real-world face de-identification. In *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*. IEEE, 1–8.

[24] Zhongshu Gu, Heqing Huang, Jialong Zhang, Dong Su, Ankita Lamba, Dimitrios Pendarakis, and Ian Molloy. 2018. Securing Input Data of Deep Learning Inference Systems via Partitioned Enclave Execution. *arXiv preprint arXiv:1807.00969* (2018). https://arxiv.org/abs/1807.00969

[25] Eman T Hassan, Rakibul Hasan, Patrick Shaffer, David J Crandall, and Apu Kapadia. 2017. Cartooning for Enhanced Privacy in Lifelogging and Streaming Videos.. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 1333–1342.

[26] Jianping He, Bin Liu, Deguang Kong, Xuan Bao, Na Wang, Hongxia Jin, and George Kesidis. 2016. Puppies: Transformation-supported personalized privacy preserving partial image sharing. In *IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 359–370.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 770–778.

[28] Aaron Hertzmann. 2010. Non-photorealistic rendering and the science of art. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*. 147–157.

[29] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[30] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. 2018. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088* (2018). https://arxiv.org/abs/1802.07088

[31] Bogdan Kwolek and Michal Kepski. 2014. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer methods and programs in biomedicine* (2014), 489–501.

[32] Bastian Leibe and Bernt Schiele. 2003. Analyzing appearance and contour based methods for object categorization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE.

[33] Guangzhen Li, Yoshimichi Ito, Xiaoyi Yu, Naoko Nitta, and Noboru Babaguchi. 2010. Recoverable privacy protection for video content distribution. *EURASIP Journal on Information Security* (2010).

[34] Tao Li and Lei Lin. 2019. Anonymousnet: Natural face de-identification with measurable privacy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

[35] Sicong Liu, Junzhao Du, Anshumali Shrivastava, and Lin Zhong. 2019. Privacy Adversarial Network: Representation Learning for Mobile Data Privacy. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. ACM New York, NY, USA, 1–18.

[36] Yusha Liu, Chun-Liang Li, and Barnabás Póczos. 2018. Classifier Two Sample Test for Video Anomaly Detections.. In *British Machine Vision Conference*. 71.

[37] Cewu Lu, Jianping Shi, and Jiaya Jia. 2013. Abnormal event detection at 150 fps in matlab. In *IEEE International Conference on Computer Vision*. IEEE, 2720–2727.

[38] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *IEEE conference on computer vision and pattern recognition*. IEEE, 5188–5196.

[39] Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. 2020. DarkneTZ: towards model privacy at the edge using trusted execution environments. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 161–174.

[40] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. 2020. Visor: Privacy-Preserving Video Analytics as a Cloud Service. *arXiv preprint arXiv:2006.09628* (2020).

[41] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. 2013. P3: Toward Privacy-Preserving Photo Sharing.. In *USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 515–528.

[42] Nisarg Raval, Ashwin Machanavajjhala, and Landon P Cox. 2017. Protecting visual secrets using adversarial nets. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 1329–1332.

[43] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[44] Natacha Ruchaud and Jean-Luc Dugelay. 2016. Privacy protecting, intelligibility preserving video surveillance. In *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 1–6.

[45] Oleksii Shamuratov, Yuriy Ryshkovets, Nataliya Shakhovska, and Ihor Kohut. 2019. The methods for Contour Analysis of Images. In *IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*. IEEE, 41–45.

[46] Kimia Tajik, Akshith Gunasekaran, Rhea Dutta, Brandon Ellis, Rakesh B. Bobba, Mike Rosulek, Charles V. Wright, and Wu-chi Feng. 2019. Balancing Image Privacy and Usability with Thumbnail-Preserving Encryption. In *Network and Distributed System Security Symposium*. The Internet Society.

[47] Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. In *IEEE International Conference on Computer Vision*. IEEE, 839–846.

[48] Yongkang Wong, Shaokang Chen, Sandra Mau, Conrad Sanderson, and Brian C. Lovell. 2011. Patch-based Probabilistic Image Quality Assessment for Face Selection and Improved Video-based Face Recognition. In *IEEE Biometrics Workshop, Computer Vision and Pattern Recognition Workshops*. IEEE.

[49] Changhai Xu and Benjamin Kuipers. 2011. Object detection using principal contour fragments. In *Canadian Conference on Computer and Robot Vision*. IEEE, 363–370.

[50] Mengwei Xu, Xiwen Zhang, Yunxin Liu, Gang Huang, Xuanzhe Liu, and Felix Xiaozhu Lin. 2020. Approximate query service on autonomous iot cameras. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 191–205.

[51] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation.. In *IEEE International Conference on Computer Vision*. IEEE, 2868–2876.

[52] Hyunwoo Yu, Jaemin Lim, Kiyeon Kim, and Suk-Bok Lee. 2018. Pinto: Enabling Video Privacy for Commodity IoT Cameras. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1089–1101.

[53] Ning Zhang, Manohar Paluri, Yaniv Taigman, Rob Fergus, and Lubomir Bourdev. 2015. Beyond Frontal Faces: Improving Person Recognition Using Multiple Cues. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 4804–4813.

[54] Wei Zhang, Sen-Ching S Cheung, and Minghua Chen. 2005. Hiding privacy information in video surveillance system. In *IEEE International Conference on Image Processing 2005*.

[55] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. HiDDeN: Hiding Data With Deep Networks. In *European conference on computer vision*. Springer, 682–697.

[56] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision*. IEEE, 2242–2251.

[57] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. Automating visual privacy protection using a smart led. In *Proceedings of the Annual International Conference on Mobile Computing and Networking*. ACM, 329–342.