**AMADEUS: Scalable, Privacy-Preserving Live Video Analytics**

**Sandeep D’souza**  
Carnegie Mellon University  
**Victor Bahl**  
Microsoft Research  
**Lixiang Ao**  
UC San Diego  
**Landon P. Cox**  
Microsoft Research

**Abstract**

Smart-city applications ranging from traffic management to public-safety alerts rely on live analytics of video from surveillance cameras in public spaces. However, a growing number of government regulations stipulate how data collected from these cameras must be handled in order to protect citizens’ privacy. This paper describes AMADEUS, which balances privacy and utility by redacting video in near realtime for smart-city applications. Our main insight is that whitelisting objects, or blocking by default, is crucial for scalable, privacy-preserving video analytics. In the context of modern object detectors, we prove that whitelisting reduces the risk of an object-detection error leading to a privacy violation, and helps AMADEUS scale to a large and diverse set of applications. In particular, AMADEUS utilizes whitelisting to generate composable encrypted object-specific live streams, which simultaneously meet the requirements of multiple applications in a privacy-preserving fashion, while reducing the compute and streaming-bandwidth requirements at the edge. Experiments with our AMADEUS prototype show that compared to blacklisting objects, whitelisting yields significantly better privacy (up to ~28x) and bandwidth savings (up to ~5.5x). Additionally, our experiments also indicate that the composable live streams generated by AMADEUS are usable by real-world applications with minimum utility loss.

1 Introduction

Surveillance cameras are a ubiquitous presence in public spaces. To take advantage of these cameras, researchers have proposed several video-analytics frameworks [7, 36, 44] that can ingest video data and perform object detection [63, 66] and tracking [9, 34] on behalf of smart-city applications like traffic management [22], pedestrian detection, and public-safety alerts. Cameras are an appealing data source for smart-city initiatives because they can support a wide-range of applications and installing cameras is relatively easy. In comparison, dedicated sensing infrastructure, such as in-road induction loops or RFID tags and readers, support a far smaller set of use-cases and are more difficult to deploy.

However, as we have learned from our own experience building pilot video-analytics systems for the last several years, the convenience of surveillance cameras comes at a significant cost: erosion of citizens’ privacy. Cameras collect data indiscriminately, and as a result they capture information that is both inessential to applications’ purposes and prone to abuse. An increasing cause of worry is applications going beyond their mandate or publicly-specified objective. Consider a smart-city scenario, where a video stream originating at a camera may be processed by third-party entities, each with their own mandate. For example, consumer A is mandated to analyze vehicles to measure traffic congestion, and consumer B is mandated to count pedestrians to decide an appropriate pedestrian-crossing duration. However, consumer A may exceed their mandate by performing facial recognition on the video stream, without the administrator’s knowledge.

Governments around the world have reacted to the privacy threat posed by surveillance technologies by passing regulations like the European Union’s General Data Protection Regulations (GDPR) and cities’ bans on face recognition [15, 31]. Preserving the utility of video-based smart-city applications while remaining compliant with privacy-protecting regulations will require technical solutions for controlling how applications extract information from video data.

Smartphones and PCs must also control access to video data, but they typically support only coarse-grained permissions, e.g., allowing full access to a device’s camera or no access at all. Fortunately, prior work has explored applying the principle of least privilege to video using fine-grained access-control mechanisms. These systems rely on computer vision to detect objects in realtime and then use the detected objects to transform raw video data before sharing it with an application. One approach to transforming video data is blacklisting or sharing-by-default, which redacts objects that a video consumer is not allowed to view (e.g., placing a black box over human faces) [6, 11, 41, 42, 62, 77], and then use the detected objects to transform raw video data before sharing it with an application. Another approach is whitelisting or blocking-by-default, which shares only the objects that a video consumer is explicitly allowed to view (e.g., blacking out everything except for detected cars and trucks) [41, 42, 62, 77].

These techniques provide a solid foundation for bringing smart-city infrastructure into compliance with privacy regulations, but prior work is insufficient on its own. In particular, prior work was designed for smartphones and PCs, in which a trusted kernel can transform video data for a small number of local applications (often just one). In contrast, surveillance-
camera data circulates through a distributed system consisting of resource-limited edge devices and a potentially diverse and large set of remote video-consuming applications.

In this paper we present the design, implementation, and evaluation of a privacy-preserving framework for surveillance-camera videos called AMADEUS. The three biggest challenges faced by AMADEUS are edge devices’ compute limitations, edge devices’ bandwidth limitations, and smart-cities’ large and diverse set of video consumers. The key observation underlying our design is that a block-by-default approach is crucial for preserving privacy in the face of these challenges.

To limit the size of AMADEUS’s trusted computing base, AMADEUS redacts videos on the edge using off-the-shelf object-detection models. However, performing object detection on resource-limited edge devices in near realtime often means using weaker models. Our experiments (Section 2.1) indicate that weaker models are more prone to false negatives (i.e., missing a present object) than false positives (i.e., misclassifying a present object), and these errors can lead to privacy violations. Thus, AMADEUS blocks video content by default, or whitelists objects, to reduce the likelihood of a model error causing a privacy violation.

Blocking by default (whitelisting) also helps AMADEUS scale to a large and diverse set of applications. Modern object-detection models like YOLO [63] can detect tens of object categories, such as people, cars, and traffic lights. Under share-by-default (blacklisting), AMADEUS would have to create a uniquely-redacted stream for each combination of categories. Creating all of these streams would require a prohibitive amount of computation and bandwidth for an edge node. Note that even when the number of permission combinations for active near realtime applications is small, the diversity of authorized video consumers may grow over time. For example, at the time a camera captures a video, no active application may need to view bicycles. However, at a later time, a city’s road-planning commission may wish to characterize bicycle traffic using archived videos.

As a result, AMADEUS creates a separate live stream for each object category (along with a residual background stream), and each stream contains nothing but its associated category (e.g., the bicycle stream only shows bicycles). This approach helps AMADEUS scale in three ways. First, the maximum number of streams that an edge server needs to generate is equal to the number of detected object categories (not the number of category combinations). Second, because most streams are mostly (or entirely) blacked out, individual streams compress well with minimum effort. And finally, by breaking object streams into short segments and encrypting the segments with unique keys, AMADEUS can decouple serving encrypted video data from making authorization decisions (i.e., distributing decryption keys). This allows video transfers to be handled by conventional, scalable web technologies like content distribution networks (CDNs) without expanding the trusted computing base or undermining privacy. Applications authorized to view multiple objects can simply retrieve the appropriate keys and encrypted video segments, and overlay the decrypted videos to compose a coherent, multi-object stream.

AMADEUS also provides object-specific encrypted live analytics which enable applications to extract insights without re-processing the video streams. This metadata also indicates the presence of low-confidence relevant objects withheld from the consumer, which can be used to recover lost utility.

**Threat Model:** We assume that applications are trusted to perform any operations on object streams they are authorized to view. For instance, an application authorized to view an object stream corresponding to the class faces, can perform any operation on faces (such as facial recognition and age/gender/emotion detection). Therefore, AMADEUS authorizes applications’ purpose at the granularity of the detection classes supported by the object detector. Providing more fine-grained authorization requires feature extraction [57], which reduces flexibility, and is beyond the scope of this work.

This paper makes the following contributions:

- We identify block-by-default or whitelisting as crucial for scalable, privacy-preserving live video analytics.
- We prove that whitelisting can guarantee lower privacy leakage than blacklisting, in the context of modern object-detection techniques.
- We describe the design of a new privacy-preserving video-analytics framework called AMADEUS that transforms video in near realtime and decouples video authorization from video transfer, using composable streams.
- Using our AMADEUS prototype we show that blocking-by-default balances privacy and utility and significantly reduces the bandwidth required to simultaneously serve multiple applications with different objectives.

## 2 Privacy and Object Detection

We now illustrate the benefits of whitelisting over blacklisting in the context of preserving privacy, by experimentally analyzing the performance of state-of-the-art object detectors.

Object detection is key to generating whitelisted and blacklisted video streams. Therefore, the object detector utilized at video-ingest time plays an important role in deciding:

1. the amount of private information that will be disclosed to the consumer, or *privacy loss*, due to private objects not being redacted (blacklisting), or being accidentally disclosed (whitelisting); and
2. the amount of useful information that will be withheld from the consumer, or *utility loss*, due to relevant objects not being disclosed (whitelisting), or being accidentally redacted (blacklisting)

Convolutional Neural Networks (CNNs) [47] are the basis for most state-of-the-art techniques for both object-detection [29,50,63] and image-classification [33] tasks.

Object-detection techniques in the family of Region-based CNNs (RCNNs) [28,29,66] utilize a two-stage approach. In
The relative prevalence of false positives and false negatives decides whether whitelisting or blacklisting can lead to lower privacy loss. This is usually measured using precision and recall, which are defined as follows:

- **Precision** measures how accurate a detector is, i.e., the proportion of detections that are correct. Precision can be calculated by: \( \text{precision} = \frac{TP}{TP + FP} \).
- **Recall** measures the fraction of ground truth objects that are both correctly detected and labeled by the detector. Recall can be calculated by: \( \text{recall} = \frac{TP}{TP + FN} \).

Increasing precision involves decreasing the number of false positives (FP), and to increase recall the number of false negatives (FN) need to decrease. Note that most object detectors are trained to maximize mean-average precision [49].

Figure 2 plots the precision and recall of three object-detection techniques: Faster-RCNN [66], YOLOv3 and Tiny-YOLOv3 [65], on three videos generated from the training (bdd-training), validation (bdd-validation), and tracking (bdd-tracking) data from the Berkeley DeepDrive Dataset (BDD) [81]. These videos collectively contain 1,100 hours of labeled video data, consisting of 100,000 video sequences recorded at different times of the day, weather conditions, and scenarios [81]. Thus, this dataset is representative of many real-world conditions and contains labeled objects corresponding to 10 classes (a subset of the COCO dataset [49]).

Among the object-detection techniques we benchmark, Faster-RCNN and YOLOv3 are relatively-heavy full-scale models, while Tiny-YOLOv3 is a lightweight approximation of the full-scale YOLOv3 model. We do not re-train these three models and use their default weights obtained by training on the COCO dataset [49], which has 80 object classes. This gives us an un-biased measurement of their performance on unseen data. We observe that, while all three detectors yield very high precision (~0.93 – 0.96), the recall is significantly lower (~0.07 – 0.6). For all three videos also observe that the most-complex model, i.e., Faster-RCNN yields the highest recall, followed by YOLOv3 and Tiny-YOLOv3.

**Observation 1:** Figure 2 indicates that state-of-the-art object detectors tend to have significantly higher precision than recall, i.e., false negatives are significantly more common than false positives for state-of-the-art object-detection techniques. This implies that object detectors tend to miss more objects in a scene than mis-classify detected objects. As indicated by Figure 2, this problem is more acute for low-cost detec-
tors like Tiny-YOLOv3, which have significantly lower recall (~3x), and are often used in resource-scarce settings.

We hypothesize that decreasing false positives is an easier problem to solve, as it involves correctly classifying detected objects. This is supported by the results in Figure 2, which indicate that light-weight detectors like Tiny-YOLOv3 have similar precision to computationally-heavy detectors like YOLOv3 and Faster-RCNN. On the other hand reducing false negatives is tougher, as it involves both detecting the missed objects as well as classifying them.

2.2 Whitelisting: Privacy-Loss Guarantees

While precision and recall are useful metrics to capture the prevalence of detection errors, as indicated in Figure 1, not all such errors contribute to either privacy or utility loss.

Observation 2: A false positive $FP_{S→R}$ mis-labeling a sensitive class $S ∈ β$, as a relevant class $R ∈ ω$ causes privacy loss for whitelisting. Similarly, a false positive $FP_{R→S}$ mis-labeling a relevant class $R ∈ ω$, as a sensitive class $S ∈ β$ causes utility loss for blacklisting. On the other hand, a false negative $FN_{S}$ not detecting a sensitive class $S ∈ β$ causes privacy loss for blacklisting. Alternatively, a false negative $FN_{R}$ not detecting a relevant class $R ∈ ω$ causes utility loss for whitelisting. We formally define privacy and utility loss:

- **Privacy Loss**, $P$ is the fraction of objects belonging to sensitive classes $S ∈ β$ disclosed to the consumer.
- **Utility Loss**, $U$ is the fraction of objects belonging to relevant classes $R ∈ ω$ withheld from the consumer.

Using the above definitions, we can mathematically calculate privacy and utility loss, $P$ and $U$, for both whitelisting (WL) and blacklisting (BL), using the following equations:

$$P_{WL} = \frac{\sum_{S \in \beta, R \in \omega} FP_{S \rightarrow R}}{\sum_{S \in \beta} TP_{S} + FN_{S}}$$

$$U_{WL} = \frac{\sum_{R \in \omega} FN_{R}}{\sum_{R \in \omega} TP_{R} + FN_{R}}, \quad U_{BL} = \frac{\sum_{S \in \beta} FP_{R \rightarrow S}}{\sum_{R \in \omega} TP_{R} + FN_{R}}$$

(1)

Combining Observations 1 and 2 we can conclude that:

1. False positives are rare as modern object detectors have high precision. When using whitelisting, only a subset of false positives mis-labeling a sensitive class $S ∈ \beta$ as a relevant class $R ∈ \omega$ cause privacy loss.
2. False negatives are common as modern object detectors have relatively low recall. Although, recall can be increased by training on specialized data, it is significantly more challenging than increasing precision. When using blacklisting, all false negatives not detecting a sensitive class $S ∈ \beta$ cause privacy loss.

We now state the Privacy-Loss guarantee theorem.

**Theorem 1:** Consider objects $Θ$ belonging to sensitive classes $S ∈ \beta$, which if disclosed lead to privacy loss. Given an object-detection technique with higher precision than recall over all objects $Θ$ belonging to classes $S ∈ \beta$, then whitelisting can guarantee lower privacy loss than blacklisting.

![Figure 3: Vehicle-Counting Privacy-Loss Measurements on the Berkeley DeepDrive Dataset [81]](image)

**Proof:** Using the definitions of precision and recall from Section 2.1, we can formulate the following inequality:

$$\text{precision} > \text{recall} \implies \frac{\sum_{S \in \beta} TP_{S}}{\sum_{S \in \beta} TP_{S} + FP_{S}} > \frac{\sum_{S \in \beta} TP_{S} + FN_{S}}{\sum_{S \in \beta} TP_{S} + FN_{S}}$$

The above inequality implies that the number of false negatives, $\sum_{S \in \beta} FN_{S}$ exceeds the number of false positives, $\sum_{S \in \beta} FP_{S}$, over all objects belonging to classes $S ∈ \beta$. We can also reformulate the above inequality as the following:

$$\frac{\sum_{S \in \beta} FP_{S}}{\sum_{S \in \beta} TP_{S} + FN_{S}} < \frac{\sum_{S \in \beta} FN_{S}}{\sum_{S \in \beta} TP_{S} + FN_{S}}$$

As the number of false positives mis-classifying a relevant class $R ∈ \omega$ as a sensitive class $S ∈ \beta$ is a subset of the total false positives over all objects belonging to classes $S ∈ \beta$, i.e., $\sum_{S \in \beta, R \in \omega} FP_{S \rightarrow R} ≤ \sum_{S \in \beta} FP_{S}$, the following holds:

$$\sum_{S \in \beta, R \in \omega} FP_{S \rightarrow R} ≤ \sum_{S \in \beta} FN_{S} \implies P_{WL} < P_{BL}$$

where, $P_{WL}$ and $P_{BL}$ is the privacy loss caused when using whitelisting and blacklisting respectively. This follows from the definitions of privacy loss in Equation 1.

**Key Takeaway:** Our observations coupled with the Privacy-Loss guarantee theorem indicate that when using modern object-detection methodologies at ingest time, whitelisting can yield significantly lower privacy loss than blacklisting. Figure 3 compares the privacy loss between whitelisting and blacklisting, for the vehicle-counting application described in Figure 1. We observe that whitelisting yields significantly lower percentage privacy loss (~0.05-1.4%), as compared to blacklisting (~46-93%). On the flip side, whitelisting also suffers potentially higher utility loss (~38-87%) as compared to blacklisting (~0.02-0.09%).

**Observation 3:** Privacy loss is permanent, utility loss can be recovered. For instance, when an object belonging to a sensitive class $S$ is accidentally disclosed or not redacted, the privacy loss caused cannot be reversed. On the other hand, if the detection of a low-confidence object belonging to a relevant class $R$ is withheld from the consumer, it can be recovered by post-processing the frame with a more accurate object detector. This can be achieved by providing consumers with
Figure 4: Per-second frame-rate (FPS) for different object detection techniques combined with object tracking metadata indicating low-confidence relevant objects. Consumers can request post processing to recover lost utility. Therefore, we bias towards privacy and utilize whitelisting over blacklisting for creating privacy-preserving video streams. Although our implementation supports both blacklisting and whitelisting, MADEUS favors whitelisting. In subsequent sections, we highlight the added benefits whitelisting introduces in terms of both administering and distributing privacy-preserving video streams to multiple consumers.

3 AMADEUS Design

We now describe the motivation for MADEUS’s design, with a focus on the trade-offs of using whitelisting. We assume that MADEUS performs object detection using a surveillance camera’s onboard computation [20] or a directly connected edge server [18]. In either case, MADEUS’s trusted computing base (TCB) extends to all software and hardware with access to plaintext video. In the future, it may be possible to reduce the size of MADEUS’s TCB by performing object detection and video encoding in a secure execution environment, such as ARM TrustZone or Intel SGX, but this is currently infeasible. Regardless, because edge platforms are generally more resource constrained than high-end server machines, we explore some lightweight techniques to detect and disclose/redact objects in video streams. Subsequently, we describe how MADEUS simultaneously delivers video to multiple applications, while minimizing privacy leakage and reducing the bandwidth required by the edge site. Lastly, we also discuss the benefits of whitelisting from the standpoint of application authorization.

3.1 Combining Detection with Tracking

Most object detectors are computationally-intensive and require powerful hardware accelerators like GPUs to support inference frame rates required for real-time streaming and analytics. Such powerful accelerators may often not be available on resource-constrained edge platforms. A large body of work has looked at using neural-network specialization [36, 53, 71] and approximation [13, 30] to support fast inference.

This section explores utilizing existing pre-trained object-detection models coupled with low-cost object trackers. Given that objects remain in the field of view of a camera for at least a few frames, we perform inference every n frames, and track the detected objects between two consecutive inference steps.

In addition to tracking, other traditional techniques like frame differencing can also be used by MADEUS to detect objects arriving between frames [36]. However, we focus on tracking as it also uniquely identifies object between frames, which can be utilized to service more sophisticated queries.

Figure 4 compares the inference frame rates of: (i) state-of-the-art object detectors: Faster-RCNN (rnn) [66] and YOLOv3 (yolo) [65], (ii) a low-cost object detector: Tiny-YOLOv3 (tiny-yolo) [65], and (iii) performing object detection every n frames using Faster-RCNN (rnn-skip-n) or YOLOv3 (yolo-skip-n) followed by object tracking (using the SORT [9] framework and OpenCV-based trackers [5]), for n = 5 and n = 10. These frame rates were obtained on a machine with an Nvidia GTX 1070 GPU [19]. We consider four videos each with an HD resolution of 1280x720.

On average rnn yields the lowest frame rate (~2.7 FPS), and tiny-yolo the fastest frame rate (~42 FPS). Additionally, yolo-skip-10 can deliver a similar or faster frame rate than tiny-yolo (~36-46 FPS). Note that while rnn, yolo and tiny-yolo yield frame rates independent of the video content, the frame rate for rnn-skip-n and yolo-skip-n is video dependent, as the tracking speed depends on the number of objects in a frame.

Figure 5 compares the precision and recall of yolo-skip-n for n = 5 and n = 10 and tiny-yolo. As the videos we consider are not labeled, we use labels generated by YOLOv3 (yolo) as the ground truth. This yields a good relative comparison, as all the techniques we compare are based on YOLOv3. We observe that yolo-skip-n for both n = 5 and n = 10, yields better precision and significantly higher recall as compared to tiny-yolo. This makes it a viable alternative for resource-constrained edge platforms, as it can deliver higher precision and recall than tiny-yolo, which translates to both lower privacy and utility loss, while delivering a similar or higher frame rate.
3.2 Composable Streaming

Bandwidth is an important constraint when streaming from an edge device [67, 68, 72]. If all consumers are homogeneous in terms of their viewing privileges and objectives, a single whitelisted or blacklisted stream can also be distributed through a multicast. In this scenario the bandwidth demand at the edge is constant regardless of the number of consumers, and is always less than or equal to the bandwidth required to stream the video with no privacy-preserving transformations.

However, a video stream will often have multiple consumers with different objectives, situated at different locations. Consider a smart-city scenario, where we have multiple consumer applications (or human viewers) utilizing the same camera feed. Table 1 showcases five smart-city applications, each responsible for analyzing different types of objects in the video stream, along with the object classes that need to be disclosed or redacted based on whether whitelisting or blacklisting is used. In this scenario, a naive approach involves creating consumer-specific video streams to satisfy each consumer’s requirements while also preserving privacy. This can cause the bandwidth requirement at the edge to rapidly increase as multiple consumers are added.

Consider the use of blacklisting in the smart-city scenario. Figure 6 illustrates the five blacklisted video streams created to satisfy each application. Each of these streams contain a common background, which in this case is streamed five times from the edge. In the adverse setting, when no objects are detected as belonging to any of the blacklisted classes, each consumer application gets the same frame in its entirety.

If we use whitelisting to preserve privacy, and create a unique stream for each application, then some object classes may be in two or more streams. For example, as shown in Figure 6, objects belonging to the class bicycle are used by both the bicycle safety and two-wheeler counting applications. Thus, these objects are transmitted twice from the edge. However, this is still more efficient than blacklisting, as the background is blacked out and not disclosed to the viewers.

Observation 4: Video frames are typically encoded and transmitted in a compressed format such as Motion JPEG [35] or H264 [80]. These encoding formats encode information in the frequency domain. Hence, blacked-out regions do not contain any information or contribute to streaming bandwidth.

Streaming Objectives: Consider a pixel $p_{i,j}^t$ of a frame $t$ as the smallest unit of information in a video stream with resolution $(W,H)$, where $0 < i < W, 0 < j < H$ and $t, W,H \in \mathbb{Z}^+$. By using an object detector, if a pixel lies inside the bounding box for an object, we can classify it as: (i) belonging to that object class, else (ii) as belonging to the background. Therefore, to distribute the video stream in a privacy-preserving and bandwidth-efficient manner we need to do the following:

- Each pixel $p_{i,j}^t$ in a frame $t$ of the video stream contains useful information. Therefore, to communicate all the useful information while minimizing bandwidth, each pixel in the frame must be encoded only once.

Figure 6: Multi-consumer Smart-City Scenario

- Each pixel $p_{i,j}^t$ in the frame $t$ must only be accessible to consumers who are authorized to view it, i.e., if it belongs to a whitelisted class $R \in \omega_k$ or does not belong to a blacklisted class $S \in \beta_k$ for the consumer $C_k$.

An Ideal Solution to achieve both objectives is as follows:

1. Assign each pixel $p_{i,j}^t$ in frame $t$ to a set $\Gamma_c$, where $c \in \{\Omega, background\}$, and $\Omega$ represents all the object classes which an object detector can detect.
2. Encrypt each pixel set $\Gamma_c$ with a unique key $\kappa_c$.
3. Encode all the encrypted pixels $\rho_{i,j}^t$ as a single frame.

Using this ideal scheme, we could in theory distribute this single encrypted video stream to multiple consumers. However, because each set of pixels $\Gamma_c$ is encrypted with its own key $\kappa_c$, only consumers with the correct key will be able to decrypt and decode the corresponding pixels.

Key Properties: This scheme effectively converts the video-stream distribution problem into one of key management. Combining this with the knowledge of a consumer’s whitelisted classes, we can authenticate each consumer and only provide them the set of keys they require to view their relevant sets of pixels, thus: (i) preserving privacy as each viewer can only see objects belonging to their whitelisted classes, (ii) preserving bandwidth as each pixel is only transmitted once from the edge, and (iii) reducing computational costs as each pixel is only encrypted and encoded once.

Although not a natural fit, we can also use the above-mentioned approach to generate a blacklisted stream, by providing a consumer $C_k$ all the keys $\kappa_c$, for all classes $c \in \beta_k$.

Video-Streaming Realities: Unlike the proposed scheme, video-streaming techniques first perform encoding, and then encrypt the encoded data [39, 73, 80], as performing encryption on raw frames destroys redundant information in a video frame, which is used to efficiently compress and encode the video. Therefore, we propose a practical scheme called AMADEUS which performs encoding followed by encryption.

AMADEUS creates $N + 1$ encrypted video streams for each object class $c \in \{\Omega, background\}$, where $\Omega$ represents all the object classes which an object detector can detect, and $N$ is the number of object classes in $\Omega$. We call these object streams. Each object stream contains all the pixels $p_{i,j}^t \in \Gamma_c$ corresponding to objects detected as belonging to a particular class $c$, and is encrypted with an object-class-specific key $\kappa_c$. $N$ can be dynamically changed based on the object classes we need to detect, based on application white/blacklisting requirements. AMADEUS also creates a residual background
MADEUS: Composable Encrypted Streaming

stream, which contains all the pixels not assigned to any object class. This background stream is encrypted with a special key $\kappa_{background}$. Figure 7 provides an overview of MADEUS considering the multi-application smart-city example.

Composable Streams: Each pixel is assigned to only one of the $N + 1$ created object streams. As shown in Figure 7, all the areas of an object stream not assigned any pixels are blacked out and do not contain any information. Therefore, post decryption, we can combine all the $N + 1$ streams, using simple pixel addition, to reconstruct the original video stream.

Compute + Bandwidth Efficiency: MADEUS still satisfies the key desirable properties provided by the ideal scheme. For instance, each object stream is encrypted with a different key. Therefore, while every user can consume all the object streams through a Content-Distribution Network (CDN), they are only able to decrypt the streams which they are authorized to view (whitelisting). Additionally, each pixel in a frame is assigned to only one object stream. Therefore, as each pixel is encoded, encrypted and transmitted to the CDN only once, MADEUS can provide privacy-preserving video streaming in a compute and bandwidth-efficient manner. MADEUS can also directly live-stream video to multiple consumers. However, in the absence of a CDN each consumer creates a direct streaming connection to the edge, which can cause bandwidth requirements at the edge to increase as new consumers are added. While, MADEUS is bandwidth efficient at the edge, it can also save downstream bandwidth at the video consumer, as a consumer can now only consume relevant object streams instead of the entire video stream.

Access Control: MADEUS can also generate new encryption keys at periodic intervals to ensure better security, while allowing consumer viewing permissions to change over time. An additional benefit of MADEUS is the fact that despite having a subset of all the encryption keys $\kappa$, every consumer can still stream and locally store the encrypted video stream in its entirety. This opens up the possibility of a consumer obtaining permission to process a previously inaccessible object stream from the administrator in the future. A good example of this can be surveillance video. We can envision a situation where security personnel can view all aspects of a video frame except human faces. However, if a crime is committed, they can request access to the encryption key $\kappa_{faces}$ required to view faces in a particular time duration, in order to analyze the archived footage. This process is comparable to a legal search warrant, where a judge needs to grant permission to perform an investigative search.

3.3 Administrative Benefits of Whitelisting

We now describe the benefits of whitelisting from the stand-point of administering privacy-preserving live streams to multiple consumer applications.

Blacklisting requires objects belonging to sensitive classes to be redacted from the consumer’s video stream. Therefore, the video-stream administrator has to anticipate: (i) which object classes are sensitive and can disclose privacy, and (ii) whether redacting sensitive objects impacts the performance of a consumer application. Therefore, in this administrator-centric permissions model, an administrator needs to balance both user privacy and application utility.

Whitelisting requires objects belonging to relevant classes to be disclosed to a consumer. Therefore, consumers are asked to specify what object classes they require to meet their stated objective. These requests can then be approved or declined by the video-stream administrator. This is similar to the permissions model used in smartphones [45], where applications must request the smartphone user to grant access to a certain set of capabilities. Thus, whitelisting features a consumer-centric permission model. This is significantly easier to administer as it shift the onus of specifying utility requirements on the consumer, while the administrator checks if application requirements are invasive of privacy.

However, MADEUS authorizes applications at the granularity of the object classes supported by the object detector. Therefore, an administrator must presume that if an object stream corresponding to an object class is disclosed to an application, then it can perform any operation on those objects.

MADEUS can be implemented using a proprietary format. However, to ensure compatibility with real-world systems, our implementation uses existing streaming formats. This introduces some overheads. We describe this implementation and benchmark these streaming formats in subsequent sections. We believe that the key ideas of MADEUS are also applicable to other content streaming domains, including text and audio.

4 MADEUS Implementation

Our realization of the MADEUS privacy-preserving live streaming and analytics pipeline, as visualized in Figure 8, builds on existing video-streaming technologies. This makes it compatible with existing applications and CDNs. We now describe each of MADEUS’ component modules.

4.1 Policy Engine

The policy engine exposes both a consumer and administrator-facing REST API [27].
Administrator API: An administrator is allowed to create/remove consumer profiles, as well as specify object classes that a consumer is allowed to view (whitelisting), or not allowed to view (blacklisting).

Consumer API: The consumer API allows consumers to request to view a set of object streams, by specifying a list of object classes. The request is denied if any of the requested classes are: (i) not in the whitelist, or (ii) in the blacklist

Key Management: The policy engine is also responsible for generating, revoking and distributing the encryption keys used to protect the object streams.

4.2 Object Detector

The object-detection module is responsible for ingesting a raw video feed, and detecting the objects in each frame. Currently, we utilize Keras-based [14] implementations of Faster-RCNN [66], YOLOv3 [65] and Tiny-YOLOv3 [65].

As described in Section 3.1, the object-detection module also supports running detection only once every $n$ frames, followed by object tracking. We utilize the SORT [9] framework to create and keep track of multiple objects, in conjunction with object trackers [34,51] available in OpenCV [21].

4.3 Privacy Preserver

The privacy preserver is responsible for implementing the MADEUS streaming mechanism, and generates composable encrypted object streams. These streams are created based on the object-detection data received from the object detector.

Instead of generating streams for all the object classes which a detector can detect, the privacy-preserver relies on the policy engine to dynamically specify which object classes consumers (or administrators for blacklisting) are interested in. All pixels not detected as belonging to any class specified by the policy engine are encoded into the background stream.

Nested Objects: When using bounding boxes to disclose/redact objects, there is a possibility of some objects being nested within or overlapping with other objects. For example, if the object detector used can detect both faces and persons, the bounding boxes for the faces will most likely overlap or be nested within the bounding box corresponding to the person. As described in Section 3.2, each pixel is assigned to only one of the object streams. Therefore, to prevent privacy loss, we assign overlapping pixels to the smaller bounding box. In the described scenario, this ensures that faces get assigned to their own object stream, and consequently redacted from the person stream.

Live Streaming: The privacy preserver supports two types of outputs for each object stream: (i) raw frames to be locally consumed by applications on the edge, and (ii) H264-encoded [80] HTTP Live Streams (HLS) [39] for consumption by external consumers. HLS encodes a video as files of a fixed-configurable duration, and uses HTTP to transport them. This file-based nature of HLS makes it especially useful from a content-distribution standpoint. Each object stream is encrypted using AES-128 [25], using the 128-bit per-object-stream encryption keys generated by the policy engine.

Metadata Streaming: MADEUS also streams encrypted metadata corresponding to every object stream. This per-frame metadata contains a list of detected objects corresponding to the object class, their bounding boxes and confidence scores. This is useful as: (i) metadata providing live analytics may be sufficient for many applications, and (ii) low-confidence object detections may be withheld from the user to prevent privacy loss. An application can then use metadata to detect the presence of such objects. If required, an application can recover utility, by requesting that the frame be re-analyzed by a specialized object detector.

5 Evaluation

We evaluate MADEUS using real-world video streams and applications, and investigate the (i) frame rate at which MADEUS can stream privacy-preserving video, (ii) privacy and utility loss that whitelisting and blacklisting yield, (iii) bandwidth required by MADEUS to stream privacy-preserving video from the edge, and (iv) MADEUS’s usability measured from the perspective of real-world applications.

Some of our key results are as follows:

1. MADEUS’ whitelisting approach can yield up to ~5000x lower privacy loss as compared to blacklisting.
2. MADEUS reduces the bandwidth required to stream video from the edge by up to ~5.5x as compared to the naive blacklisting approach.
3. MADEUS is usable by real-world applications and yields up to ~28x lower privacy loss.

We deploy MADEUS on a mid-range machine with an 8-core Intel i7-7700 processor [16], 16 GB of memory and an Nvidia GTX 1070 [19] GPU, which is comparable to the Nvidia Xavier edge platform [4]. We perform experiments using 12 videos described in Table 2, which include static traffic cameras, urban scenes captured by moving cameras, and news-reel footage. The videos in our dataset each contain multiple object classes (including vehicles, people and bicycles), and were recorded at 25 or 30 FPS.

Applications: We utilize two real-world applications: (i) a background-subtraction-based vehicle counting application [55] and (ii) a face-detection application which also predicts...
gender and age for each detected face [54]. While we utilize both the vehicle-counting and face-detection application to measure utility loss (the number of vehicles/faces which were not counted), we also utilize the face-detection application in an adversarial setting to measure privacy loss (the number of faces that were detected when faces were not shown or redacted from the consumer).

5.1 End-to-End System Measurements

AMADEUS is deployed at the edge, and hence it needs to deliver a relatively-high video frame rate while meeting edge resource constraints. We now benchmark the resources used by AMADEUS and the end-to-end frame rate it can deliver.

End-to-end performance involves the entire pipeline including: object detection, privacy-preserving video whitelisting/blacklisting, and video encoding and encryption. Therefore, it is a function of the following variables:

1) **Video Resolution**, which determines the number of pixels to process. Therefore, we consider two video streams: jackson-hole-town with a full-HD (1920x1080) resolution, and abc-action-news with an HD (1280x720) resolution.

2) **Object-detection complexity** which influences system requirements. Therefore, we consider (i) a state-of-the-art object detector: YOLOv3 (yolo) [65], (ii) a low-cost object detector: Tiny-YOLOv3 (tiny-yolo) [65], and (iii) performing object detection using YOLOv3 every \( n = 10 \) frames (yolo-skip-10) followed by object tracking using the SORT [9] framework and an OpenCV-based KCF tracker [34].

3) **System configuration** limits the frame rate which can be achieved. Therefore, we compare performance between running object detection on the GPU and the CPU.

4) **Video encoding and encryption** complexity, which depends on the number of output object streams that have to be created. Therefore, we vary the number of object streams required to service multiple consumer applications.

Figures 9 and 10 plot AMADEUS’s output frame rate, and CPU usage respectively, for two videos, jackson-hole-town and abc-action-news, while varying the (i) object detectors, (ii) object-detection deployment device (GPU or CPU), and (iii) the number of output object streams. In terms of object-stream ordering we start with the highest-occurring object class in the video, and subsequently add streams in order of their occurrence. Note that in each case, all the pixels not belonging to a requested stream are also encoded into the background stream. Therefore, the number of pixels encoded and encrypted in each case is constant. The key insights garnered from Figures 9 and 10 are:

**Platform:** As expected, running object detection on the GPU yields significantly higher frame rates than the CPU – up to ~5x for yolo and ~2x for yolo-skip-10 and yolo-tiny.

**Object Detection:** We observe that using the low-cost tiny-yolo yields the highest end-to-end frame rate – up to 2x higher than both yolo and yolo-skip-10 on the GPU. Additionally, for tiny-yolo, the frame rate falls with an increase in object streams. Thus, indicating that video encoding is the bottleneck in this case. On the other hand, for large models like yolo running on the GPU, the end-to-end frame rate can remain constant as we add additional object streams. In this case, the object-detection model is the bottleneck.

**Object Tracking:** While object detection can run on the GPU, the object tracker we use only runs on the CPU. Figure 4 in Section 3, indicated that yolo-skip-10 (detection + tracking) can yield similar or higher frame rates than tiny-yolo. If we consider the added CPU workload of encoding and encryption, this observation still holds for the HD video abc-action-news. On the contrary, for the full-HD jackson-hole-town, yolo-skip-10 surprisingly yields even lower frame rates than yolo – up to ~0.33x lower. We believe that this is due to: (i) a higher number of objects in the jackson-hole-town video leading to a large number of trackers, and (ii) the higher video resolution leading to higher CPU usage for encoding and encryption.

Table 2: Evaluation Video Data Description

| Video            | Camera | Description                        |
|------------------|--------|-----------------------------------|
| chicago-urban    | moving | Chicago urban scenes              |
| mumbai-urban     | moving | Mumbai urban scenes               |
| new-york-urban   | moving | New York City urban scenes        |
| abc-action-news  | moving | news feed about traffic cameras   |
| abc-7-news       | moving | news feed about traffic rules     |
| cbs-la-news      | moving | news feed about traffic collisions|
| jackson-hole-town| static  | Jackson Hole town-square camera   |
| jackson-hole-restaurant | static  | surveillance camera outside a restaurant |
| m4-motorway      | static  | traffic camera on the M4 highway (UK) |
| seattle-urban-1  | static  | residential area in Seattle       |
| seattle-urban-2  | static  | university intersection in Seattle |
| seattle-urban-3  | static  | university intersection in Seattle |

Figure 9: AMADEUS measured end-to-end frame-rate

Figure 10: AMADEUS measured CPU utilization
which leaves fewer CPU resources for object tracking.

Impact of Object Streams: Adding object streams caused the frame rate to fall, CPU usage to rise (Figure 10), and memory requirements to increase linearly. In the worst case, the total memory used never exceeded ~45%.

If we consider using yolo-skip-10 for object detection, MADEUS can stream privacy-preserving HD video at a usable rate of ~14-16 FPS using the CPU. Thus, MADEUS is usable on a platform without a GPU. Using the mid-tier GPU we were able to stream HD video at ~20-35 FPS.

5.2 Privacy and Utility Loss

Preserving privacy while allowing applications to extract utility is MADEUS’ key objective. As described in Section 2.2, object detection errors namely, false positives and false negatives, play a key role in influencing the amount of privacy and utility loss. Therefore, we now measure the privacy and utility loss yielded by using whitelisting and blacklisting, while utilizing different object detectors.

We compare Tiny-YOLOv3 (tiny-yolo) [65], and object detection every $n$ frames using Faster-RCNN (rcnn-skip-$n$) or YOLOv3 (yolo-skip-$n$) followed by object tracking for $n = 5$ and $n = 10$. We perform measurements over 10 unlabeled videos described in Table 2, and utilize labels generated by YOLOv3 [65] as the ground truth for tiny-yolo and yolo-skip-$n$, and Faster-RCNN [66] as the ground truth for rcnn-skip-$n$.

Precision and Recall: As described in Section 2.1, precision and recall measure the prevalence of false positives and false negatives, i.e., higher precision translates to fewer false positives, and higher recall translates to fewer false negatives. For our test videos and object detectors, we observe that precision is high for all combinations, with an average precision of 0.91. If we consider individual videos, the precision is similar for techniques based on the same class of object detector. However, we observe an average recall of 0.65, but that recall steadily decreases as object-detection becomes lower cost. Consider object-detection performed only once every $n$ frames followed by tracking. For both rcnn-skip-$n$ and yolo-skip-$n$ we observe that recall decreases as we go from $n = 10$ to $n = 5$. We also observe that tiny-yolo has the lowest recall (as low as 0.03). Figure 11 plots the overall precision and recall of object detectors for 4 of the 10 videos.

Privacy Loss: Privacy loss depends on what classes are defined as sensitive. Consider a car-counting application. To successfully count cars, an application must be able to view all objects belonging to the class car. Therefore, whitelisting discloses only objects belonging to class car. On the other hand, blacklisting redacts objects belonging to sensitive classes. In this case, let the sensitive class be person. Therefore, while using whitelisting, all false positives classifying a person as a car, would cause privacy loss. On the other hand, for blacklisting, all false negatives (mis-detections) corresponding to the class person, would lead to a person not being redacted from the video stream. Figure 12 plots the potential privacy loss yielded by different object-detection techniques for 4 of the 10 videos we consider. Observe that whitelisting yields significantly lower privacy loss as compared to blacklisting – up to ~5000x fewer private objects disclosed in the worst case, and ~72x fewer in the average case.

Utility Loss: Figure 13 plots the utility loss for different object detectors for 4 of the 10 streams. Whitelisting on average yields ~34% utility loss compared to blacklisting’s ~0.6%.

We conclude that, while whitelisting on average yields ~72x lower privacy loss, blacklisting yields ~57x lower utility loss. However, while privacy loss is permanent, utility loss may be recoverable by a post-processing query. If we compare the low-cost tiny-yolo against yolo-skip-10 (detection every 10 frames followed by tracking), we observe that tiny-yolo has both significantly higher privacy loss for blacklisting (up to ~5x) and utility loss for whitelisting (up to ~8x). Therefore, even though Figure 9 indicates that both tiny-yolo and yolo-skip-10 can deliver similar frame rates, yolo-skip-10 is significantly better at preserving both privacy and utility.
5.3 Bandwidth Measurements

A Madeus distributes privacy-preserving video to multiple consumers in a bandwidth-efficient manner. Therefore, we benchmark MADEUS compared to the naive whitelisting and blacklisting approaches. For both naive approaches, a custom video stream is created for each application. We perform experiments measuring bandwidth as a function of: (i) the number of consumer applications, and (ii) the number of object streams. The first experiment compares the bandwidth requirement of MADEUS versus the naive approaches. As object detection plays a key role in determining which object stream each pixel in a frame is assigned to, the second experiment measures the impact of object-detection techniques on bandwidth, as we increase the number of object streams.

Bandwidth usage is also a function of frame rate, which depends on system configuration and object-detection techniques. Therefore, simply measuring the bandwidth used at the edge would not yield a fair comparison. Instead, we measure the total file size of the HLS-encoded video divided by the length of the video, and use it as a proxy for bandwidth.

Multi-application Scenario: We consider the five application smart-city scenario described in Table 1 in Section 3.2. Figure 14 plots the output bandwidth (in Mbps) at the edge, for two different video streams, as we increase the number of consumer applications based on their ordering in Table 1. Observe that for a single application, both the naive approaches yield lower bandwidth – up to 3.6x and 1.4x lower for whitelisting and blacklisting respectively. In this scenario both whitelisting and blacklisting are publishing a single stream with information removed, as compared to MADEUS which is re-encoding all the data in the original stream, as separate object streams. However, as the number of applications increase, the naive blacklisting approach suffers significant increases in bandwidth requirement (~1.1-5.5x higher). This is because the naive blacklisting approach creates a separate stream per consumer, and each of these streams have the background pixels duplicated across them. On the other hand, as the number of applications increase, as compared to MADEUS, the naive whitelisting approach yields lower bandwidth. This is because the whitelisted streams do not carry any background pixels. However, when we add the last application Surveillance, which requires an un-redacted stream, the naive whitelisting approach now needs to create a separate stream with the entire video. This causes the bandwidth requirements to rise up to 1.44x higher than MADEUS.

The key advantage of MADEUS is that each pixel is encoded, encrypted and transmitted only once from the edge. However, using existing streaming technologies can increase overhead and MADEUS can require up to 1.79x the bandwidth required for encoding the raw video stream. Additionally, as the number of object streams increase to support more applications, we see that MADEUS’ bandwidth requirements increase slightly. We believe that this is due to the overhead of per-stream headers and the breaking of the video stream into chunks by HLS. However, MADEUS is still more efficient than the naive approaches, and the overhead can be reduced by creating a proprietary streaming standard.

We also measure input bandwidth at each application and the results are plotted in Figure 15. Observe that MADEUS can reduce input bandwidth at the application, as applications need not consume object streams not relevant to them.

Impact of Object Detection: Figure 16 plots the bandwidth requirements for MADEUS as the number of object streams increase, for different object-detection techniques: yolo, tiny-yolo, and yolo-skip-10. Note that each scenario also has a residual background stream. Due to the overhead of HLS, bandwidth requirements increase as we add new object streams. Observe that while using tiny-yolo or yolo-skip-10, the bandwidth requirements can be about ~10-15% lower than yolo. This is because yolo can detect more objects, due to lower false negatives (Figure 11). This leads to more objects being moved to the object streams from the background.

We can conclude that while distributing video to multiple consumers, MADEUS is bandwidth efficient at the edge, as when we add new applications there is minimal increase in bandwidth requirements. For instance, when adding the fifth application Surveillance, no additional bandwidth is required at the edge for MADEUS. On the other hand the naive whitelisting and blacklisting approaches see ~2.5x and ~3.5x jump in bandwidth requirements respectively. This disparity between MADEUS and the naive approaches can only get worse as more applications or object streams are added.

5.4 MADEUS Usability Measurements

The potential privacy and utility loss calculated by equation 1 described in Section 2.2 provide a good idea of the impact of using blacklisted or whitelisted streams. However,
while all objects may be perceivable by human viewers or custom detectors, they may not be perceivable by an application. Additionally, there is also a possibility of usability degradation (utility loss) due to (i) modification of the visual nature of a video stream due to whitelisting/blacklisting, and/or (ii) model mismatch. Therefore, we measure privacy and utility loss by considering two real-world applications which utilize the composable video streams generated by MADEUS:

1) Vehicle Counting: This is an open-source background-subtraction-based vehicle-counting application [55]. It assumes a static camera, and uses background subtraction [84] to detect moving objects. The application also takes as input, regions of the video frame, and counts all objects moving into those regions as vehicles. For this application to operate as intended, we whitelist all objects belonging to the classes car, truck and bus. In the blacklisting mode, we blacklist all objects belonging to the class person.

2) Face Detection: This is an open-source face-detection application [54] which also predicts gender and age for each detected face [76]. For this application to operate as intended, we whitelist all objects belonging to the classes person.

Both the vehicle-counting and face-detection applications are used to measure utility loss, i.e., the number of vehicles/faces not detected. We also utilize the face-detection application as an adversary to measure privacy loss, i.e., the number of faces that were detected when faces were not supposed to be disclosed. Since the videos we use are unlabeled, the detections on the un-redacted video is used as the baseline.

Usability Experiments: We compare three different scenarios against the baseline to measure utility loss for the vehicle-counting application where it consumes: (i) whitelisted video, (ii) a blacklisted video, and (iii) only the metadata. Note that, as described in Section 4.3, MADEUS also provides a per-object-class metadata stream. This metadata stream contains the bounding boxes for all objects belonging to the class along with the detection confidence. By using the metadata stream corresponding to object classes car, truck and bus, the vehicle-counting application can directly count vehicles without the need to detect them using background subtraction. However, the face-detection app also tries to extract application-specific attributes like gender and age, which may not always exist in the metadata, and hence, we measure utility loss only for the whitelisting case.

Privacy Experiments: To measure privacy loss we consider the face-detection app as an adversary and run it on the whitelisted and blacklisted streams that we created for the vehicle-counting application, where: (i) the whitelisted stream is only supposed to disclose objects belonging to the classes car, truck and bus, and (ii) the blacklisted stream redacts all objects belonging to class person. In an ideal world, the face-detection app should not be able to see a face of any person in these streams. However, some faces may show up due to detection/mis-detection errors. Therefore, we measure the fraction of faces detected in these privacy-preserving video streams.
Figure 19: Fraction of faces detected by the face-detection app, comparing whitelisting (car, truck, bus disclosed) and blacklisting (person redacted) streams, as compared to the raw video.

Figure 19 plots the fraction of faces (privacy loss) detected by the face-detection app, comparing whitelisting (car, truck and bus disclosed) and blacklisting (person redacted), for different object-detection techniques. In most cases we observe that blacklisting can lead to higher privacy leakage than whitelisting – up to ~28x higher. However, for video abc-7-news using yolo-skip-10, we observe that it is whitelisting that leads to a slightly higher privacy loss (~1.2x).

We can conclude that, while MADEUS preserves privacy, its composable streams can also be easily utilized by unmodified applications, with minimal utility loss as compared to using an un-redacted video. Additionally, the availability of per-object class metadata streams can enable many commonly found simple object-counting applications.

6 Related Work

We now briefly describe the relevant prior work.

Object Detection: CNNs [47] are the basis of most state-of-the-art techniques for object-detection [50, 63, 66]. A more complex and computationally-intensive task is that of image segmentation [32, 83], which involves assigning each pixel in the image a class label. MADEUS can also use segmentation to create more accurate whitelisted/blacklisted streams with lower privacy leakage. However, segmentation is resource-intensive and not suited for execution on edge platforms.

Fast inference is key to live analytics. Therefore, prior work has looked at techniques to speed up inference including: (i) neural-network specialization [36, 53, 71], (ii) approximation [13, 30], and (iii) cascading a series of classifiers [10, 48, 74, 78]. Recent work [44] has also looked at exploiting the presence of the same objects across multiple frames to speed up inference on video streams. All these techniques are complementary to MADEUS, and can be plugged in to generate privacy-preserving video streams.

Video Analytics: Recent video-analytics systems [36–38, 43, 44, 82] have focused on multiple aspects including, indexing video at ingest time to reduce query latency [36], scheduling query execution across clusters [37, 38] increasing accuracy [43], and video/query-specific detector specialization [44]. MADEUS is complementary to these systems and provides privacy-preserving live streaming and analytics.

Visual Privacy: Multiple prior works utilize the blacklisting approach to redact or distort sensitive information from videos in real time based on: (i) specification [79], (ii) detected context [6, 11], and (iii) visual markers [69]. Video-sharing platforms like YouTube also provide tools for creators to automatically blur out sensitive objects [40]. On the other hand, some prior work [41, 62, 77] has also used the whitelisting approach based on the principle of least privilege, to restrict access to objects in a video stream based on user requirements in the context of augmented reality [41, 77] and smartphone applications [62]. Video-conferencing systems like Skype [17] also provide an option to automatically blur out all the background except the person’s face. Our work proves that whitelisting is crucial for scalable, privacy-preserving video analytics. Unlike prior work and commercial products which deal with homogeneous consumers, MADEUS also provides a whitelisting-inspired bandwidth-efficient scheme to distribute privacy-preserving video streams to multiple consumers with different requirements. Our scheme for encrypting each object-stream with a different key builds on the P3 photo-sharing mechanism [60], which encrypts all sensitive information in a photo with a single key.

Recent work has also looked at obfuscating video streams to reduce privacy leakage [12, 56, 61]. These approaches require significant effort to train an obfuscation mechanism, and also require applications to be modified or re-trained to work with the obfuscated video. This obfuscated video may also be unsuitable for human viewing. MADEUS works with off-the-shelf object detectors and provides human-viewable video streams that are usable by unmodified applications.

Feature Extraction: In [57], the authors propose a collaborative framework for private feature extraction at the edge (smartphone) based on a verifiable feature-extractor module provided by the data consumer (cloud). This framework allows the data producer to control which private features are used by the data consumer, and is useful for performing fine-grained data sharing. In contrast, to be usable by a large set of unmodified applications, MADEUS performs data transformation using an object detector which extracts features at the scale of individual objects.

Object-Detection Attacks: Recent work has also looked at scenarios where malicious actors can induce object-detection errors by masquerading as other entities [58, 70]. In response, techniques have been proposed to train detectors/classifiers to be robust against such attacks [52, 75]. These techniques can be used to train the object-detection model used by MADEUS, to mitigate the risks of such attacks.

Privacy-Preserving Data Analytics: Researchers have also looked at techniques based on differential privacy [24, 46, 59] and secure multi-party computation [23]. These cryptographic approaches are more suited to structured data and/or require applications to be modified to extract utility.
7 Conclusion

Machine-learning-based techniques only offer probabilistic guarantees. Therefore, perfect privacy cannot be guaranteed while using object detection to create whitelisted or blacklisted video streams. This paper has made the case that a whitelisting, or blocking-by-default, approach to redaction is crucial for scalable, privacy-preserving live video analytics. We have designed, implemented, and evaluated a system called AMADEUS that embodies this approach. To minimize the size of its trusted computing base, AMADEUS uses resource-constrained edge nodes to perform object detection on video streams. For each object category, AMADEUS then generates an encrypted, per-category live stream and distributes these encrypted videos to applications. Applications retrieve decryption keys for their authorized categories, and can compose decrypted videos to create a coherent multi-object stream. Experiments with our AMADEUS prototype show that (i) compared to blacklisting, whitelisting yields significantly better privacy (up to 28x) and bandwidth savings (up to 5.5x), and (ii) AMADEUS is usable by unmodified real-world applications with negligible utility loss.

While the key ideas proposed in this paper are relevant for video streaming, we believe that they are also applicable for other types of content streaming, including audio and text.

References

[1] FFmpeg project. https://ffmpeg.org/about.html.

[2] NATS streaming server. https://github.com/nats-io/nats-streaming-server.

[3] NGINX HTTP server. https://www.nginx.com/resources/wiki/.

[4] NVIDIA Jetson AGX Xavier. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit.

[5] OpenCV tracking API. https://docs.opencv.org/master/d9/df8/group__tracking.html.

[6] Paarjaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. I-pic: A platform for privacy-compliant image capture. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’16, pages 235–248, New York, NY, USA, 2016. ACM.

[7] G. Ananthanarayanan, P. Bahl, P. Bodik, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. Computer, 50(10):58–67, Oct 2017.

[8] Roman Arutyunyan. NGINX-based media streaming server. http://nginx-rtmp.blogspot.com.

[9] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. CoRR, abs/1602.00763, 2016.

[10] Zhaowei Cai, Mohammad J. Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. CoRR, abs/1507.05348, 2015.

[11] J. Chaudhari, S. S. Cheung, and M. V. Venkatesh. Privacy protection for life-log video. In 2007 IEEE Workshop on Signal Processing Applications for Public Security and Forensics, pages 1–5, April 2007.

[12] Jiawei Chen, Janusz Konrad, and Prakash Ishwar. Vgan-based image representation learning for privacy-preserving facial expression recognition. CoRR, abs/1803.07100, 2018.

[13] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. CoRR, abs/1504.04788, 2015.

[14] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[15] Kate Conger, Richard Fausset, and Serge F. Kovaleski. San francisco bans facial recognition technology. New York Times, March 2019.

[16] Intel Corporation. Intel® Core™ i7-7700 Processor. https://www.intel.com/content/www/us/en/products/processors/core/i7-processors/i7-7700.html.

[17] Microsoft Corporation. Introducing background blur in skype. https://blogs.skype.com/news/2019/02/06/introducing-background-blur-in-skype/.

[18] Microsoft Corporation. What is azure data box edge? https://docs.microsoft.com/en-us/azure/databox-online/data-box-edge-overview.

[19] NVIDIA Corporation. Geforce GTX 1070. https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-1070/specifications.

[20] Qualcomm Corporation. A smart camera for the intelligent edge. https://azure.github.io/Vision-AI-DevKit-Pages/.

[21] I. Culjak, D. Abram, T. Prihan, H. Dzapo, and M. Cifrek. A brief introduction to openCV. pages 1725–1730, May 2012.
[22] Sandeep Dsouza, Heiko Koehler, Akhilesh Joshi, Satyam Vaghani, and Ragunathan (Raj) Rajkumar. Quartz: Time-as-a-Service for Coordination in Geo-Distributed Systems. In 2019 IEEE/ACM Symposium on Edge Computing, SEC 2019, 2019.

[23] Wenliang Du and Mikhail J Atallah. Secure multi-party computation problems and their applications: a review and open problems. In Proceedings of the 2001 workshop on New security paradigms, pages 13–22, 2001.

[24] Cynthia Dwork. Differential privacy. Encyclopedia of Cryptography and Security, pages 338–340, 2011.

[25] Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, and James F. Dray Jr. Advanced encryption standard. Federal Inf. Process. Stds. (NIST FIPS), 197, November 2001.

[26] Miro Enev, Jaeyeon Jung, Liefeng Bo, Xiaofeng Ren, and Tadayoshi Kohno. Sensorsift: Balancing sensor data privacy and utility in automated face understanding. In Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC ’12, pages 149–158, New York, NY, USA, 2012. ACM.

[27] Roy T Fielding and Richard N Taylor. Architectural styles and the design of network-based software architectures, volume 7. University of California, Irvine Doctoral dissertation, 2000.

[28] Ross B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015.

[29] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR, abs/1311.2524, 2013.

[30] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. CoRR, abs/1506.02626, 2015.

[31] Caroline Haskins. A second u.s. city has banned facial recognition. Vice, July 2019.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. CoRR, abs/1703.06870, 2017.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[34] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. CoRR, abs/1404.7584, 2014.

[35] Peter Hoddie, Brian Yamabe, Mitchel Wienstock, Jean-Michel Berthoud, Marshal Johnson, Nick Moss, Jay Cuccurese, Rod Sheffield, Michel Rynderman, Randy Ubillos, Michael Piper, Raymond Picard, Louis Shay, and Scott Bronson. Motion JPEG format. https://staticky.com/mirrors/ftp.apple.com/developer/Development_Kits/QuickTime/Programming_Stuff/Documentation/QuickTime-JPEGSpec.pdf, 1996.

[36] Kevin Hsieh, Ganesh Ananthanarayan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthias Phil- pose, Phillip B. Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pages 269–286, Carlsbad, CA, October 2018. USENIX Association.

[37] Chien-Chun Hung, Ganesh Ananthanarayan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. Videoedge: Processing camera streams using hierarchical clusters. In 2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018, pages 115–131, 2018.

[38] Chien-Chun Hung, Ganesh Ananthanarayan, Leana Golubchik, Minlan Yu, and Mingyang Zhang. Wide-area analytics with multiple resources. In Proceedings of the Thirteenth EuroSys Conference, EuroSys ’18, pages 12:1–12:16, New York, NY, USA, 2018. ACM.

[39] Apple Inc. HTTP live streaming. https://developer.apple.com/documentation/http_live_streaming.

[40] Google Inc. Blur moving objects in your video with the new custom blurring tool on youtube. https://youtube-creators.googleblog.com/2016/02/blur-moving-objects-in-your-video-with.html.

[41] Suman Jana, David Molnar, Alexander Moshchuk, Alan Dunn, Benjamin Livshits, Helen J. Wang, and Eyal Ofek. Enabling fine-grained permissions for augmented reality applications with recognizers. In Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13), pages 415–430, Washington, D.C., 2013. USENIX.

[42] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. A scanner darkly: Protecting user privacy from perceptual applications. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP ’13, pages 349–363, Washington, DC, USA, 2013. IEEE Computer Society.

[43] Junchen Jiang, Ganesh Ananthanarayan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable adaptation of video analytics. In Proceedings of the
Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Optimizing deep cnn-based queries over video streams at scale. CoRR, abs/1703.02529, 2017.

Patrick Gage Kelley, Sunny Consolvo, Lorrie Faith Cranor, Jaeyeon Jung, Norman Sadeh, and David Wetherall. A conundrum of permissions: Installing applications on an android smartphone. In Jim Blyth, Sven Dietrich, and L. Jean Camp, editors, Financial Cryptography and Data Security, pages 68–79, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Krishnaram Kenthapadi and Thanh T. L. Tran. Pripearl: A framework for privacy-preserving analytics and reporting at linkedin. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM ’18, pages 2183–2191, New York, NY, USA, 2018. ACM.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1097–1105. 2012.

R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In Proceedings. International Conference on Image Processing, volume 1, pages I–I, Sep. 2002.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. CoRR, abs/1405.0312, 2014.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015.

Alan Lukezic, Tomáš Vojír, Luka Cehovin, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. CoRR, abs/1611.08461, 2016.

Alessander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083, 2017.

Ala Mhalla, Thierry Chateau, Houda Maâmatou, Sami Gazzah, and Najoua Essoukri Ben Amara. SMC faster R-CNN: toward a scene-specialized multi-object detector. CoRR, abs/1706.10217, 2017.

Andrey Nikishaev. Simple demo of using PyQt, OpenCV and Haar Cascades for face detection in real-time. https://github.com/creotiv/object_detection_projects/tree/master/face_detection_with_opencv_and_pyqt.

Andrey Nikishaev. Traffic counting example based on OpenCV object detection with background subtraction. https://github.com/creotiv/object_detection_projects/tree/master/opencv_traffic_counting.

Seong Joon Oh, Mario Fritz, and Bernt Schiele. Adversarial image perturbation for privacy protection - A game theory perspective. CoRR, abs/1703.09471, 2017.

Seyed Ali Ossia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R. Rabiee. Deep private-feature extraction. IEEE Transactions on Knowledge and Data Engineering, 32:54–66, 2020.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia conference on computer and communications security, pages 506–519, 2017.

Do Le Quoc, Martin Beck, Pramod Bhatioti, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. Privapprox: Privacy-preserving stream analytics. In 2017 USENIX Annual Technical Conference (USENIX ATC 17), pages 659–672, Santa Clara, CA, July 2017. USENIX Association.

Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. P3: Toward privacy-preserving photo sharing. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), pages 515–528, Lombard, IL, 2013. USENIX.

Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. Olympus: Sensor privacy through utility aware obfuscation. PoPETs, 2019(1):5–25, 2019.

Nisarg Raval, Animesh Srivastava, Ali Razeen, Kiron Lebeck, Ashwin Machanavajjhala, and Lanodn P. Cox. What you mark is what apps see. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’16, pages 249–261, New York, NY, USA, 2016. ACM.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015.
[64] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. CoRR, abs/1612.08242, 2016.

[65] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. CoRR, abs/1804.02767, 2018.

[66] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[67] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos. Edge analytics in the internet of things. IEEE Pervasive Computing, 14(2):24–31, Apr 2015.

[68] Mahadev Satyanarayanan, Victor Bahl, Ramon Caceres, and Nigel Davies. The case for VM-based cloudlets in mobile computing. IEEE Pervasive Computing, November 2009.

[69] J. Schiff, M. Meingast, D. K. Mulligan, S. Sastry, and K. Goldberg. Respectful cameras: detecting visual markers in real-time to address privacy concerns. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 971–978, Oct 2007.

[70] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, page 1528–1540, New York, NY, USA, 2016. Association for Computing Machinery.

[71] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast video classification via adaptive cascading of deep models. CoRR, abs/1611.06453, 2016.

[72] Weisong Shi, George Pallis, and Zhiwei Xu. Edge computing [scanning the issue]. Proceedings of the IEEE, 107(8):1474–1481, 2019.

[73] I. Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. IEEE MultiMedia, 18(4):62–67, April 2011.

[74] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In 2013 IEEE Conference on Computer Vision and Pattern Recognition, pages 3476–3483, June 2013.

[75] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204, 2017.

[76] Yusuke Uchida. Keras implementation of a CNN network for age and gender estimation. https://github.com/yu4u/age-gender-estimation.

[77] J. Vilk, D. Molnar, B. Livshits, E. Ofek, C. Rossbach, A. Moshchuk, H. J. Wang, and R. Gal. Surroundweb: Mitigating privacy concerns in a 3d web browser. In 2015 IEEE Symposium on Security and Privacy, pages 431–446, May 2015.

[78] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, pages I–I, Dec 2001.

[79] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. A scalable and privacy-aware iot service for live video analytics. In Proceedings of the 8th ACM on Multimedia Systems Conference, MMSys’17, pages 38–49, New York, NY, USA, 2017. ACM.

[80] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology, 13(7):560–576, July 2003.

[81] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. CoRR, abs/1805.04687, 2018.

[82] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Victor Bahl, and Michael Freedman. Live video analytics at scale with approximation and delay-tolerance. In USENIX NSDI, March 2017.

[83] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. CoRR, abs/1704.08545, 2017.

[84] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern recognition letters, 27(7):773–780, 2006.