Public Video Surveillance: Using the Fog to Increase Privacy

Martin Grambow, Jonathan Hasenburg, David Bermbach
TU Berlin & Einstein Center Digital Future
Mobile Cloud Computing Research Group
Berlin, Germany
{mg,jh,db}@mcc.tu-berlin.de

ABSTRACT

In public video surveillance, there is an inherent conflict between public safety goals and privacy needs of citizens. Generally, societies tend to decide on middleground solutions that sacrifice neither safety nor privacy goals completely. In this paper, we propose an alternative to existing approaches that rely on cloud-based video analysis. Our approach leverages the inherent geo-distribution of fog computing to preserve privacy of citizens while still supporting camera-based digital manhunts of law enforcement agencies.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**; • **Social and professional topics** → *Surveillance*; *Governmental surveillance*;

KEYWORDS

Fog Computing, Edge Computing, Public Video Surveillance, Face Recognition

ACM Reference Format:
Martin Grambow, Jonathan Hasenburg, David Bermbach. 2018. Public Video Surveillance: Using the Fog to Increase Privacy. In *5th Workshop on Middleware and Applications for the Internet of Things (M4IoT'18), December 10–11, 2018, Rennes, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3286719.3286722

1 INTRODUCTION

Video surveillance in public areas is the method of choice for dealing with increasing requirements in crime prevention, to support manhunts, as well as to generally increase the feeling of safety of citizens. Recent examples include the Xueliang (sharp eyes) project in China [4] or the city video surveillance system in Moscow [6].

Existing video surveillance approaches from both research and practice, to the best of our knowledge, all send the collected video data to private or public cloud backends for further analysis. This way, the full benefits of the cloud such as inexpensive and scalable compute power can be leveraged. Cloud-based video analysis, however, comes with its own set of problems, the most obvious one being bandwidth limitations. Also, it strongly impairs privacy: First, cameras do not distinguish between persons of interest (PoI) and innocent bystanders – all video data is indiscriminately sent to the cloud. Second, once data has reached the cloud it is typically stored indefinitely since storage is inexpensive. While we may trust current governments to sensibly handle this data, history has taught us that we can never know what future governments may look like. Third, video data from a number of geo-distributed cameras typically ends up in the same central data lake which facilitates correlation of data sources. Such correlation, however, supports the unfounded surveillance of private citizens including the creation of extensive movement profiles.

Overall, this implies an inherent tradeoff between the citizens' privacy and public safety goals. While some societies may opt for the extremes, namely public video surveillance as described above or no cameras at all, democratic consensus tends to choose a middleground solution. Existing middleground solutions use, for instance, secure multiparty computation [5] or embedded watermarks [9] to protect privacy in the presence of video surveillance. Other approaches use machine learning and 'smart' algorithms to detect relevant situations in video streams, for instance, violence in a train station or elderly people who have fallen in a nursing home [2]. All these approaches still rely on cloud-based video analysis.

In this paper, we propose an alternative (or complementary) approach that uses the inherent geo-distribution of fog computing [1] to provide privacy levels that do not exist in cloud-based scenarios while still offering the functionality of a camera-supported digital manhunt. As contributions, we present the design of such a system and report on the lessons we learned while prototypically implementing said system.

The remainder of this paper is structured as follows: We describe the design of our fog-based manhunt system and discuss how we deal with the privacy tradeoff in section 2. Afterwards, in section 3, we present our proof-of-concept implementation, before we discuss related work in section 4 and our lessons learned in 5. Finally, section 6 concludes with a summary and outlook.

2 SYSTEM DESIGN

Our overall goal is to provide another approach for public video surveillance that enables law enforcement agencies to actively search for PoIs while preserving the privacy of non-involved citizens. For this purpose, we leverage the geo-distribution of fog computing and analyze the video streams on the edge instead of processing them in the cloud which avoids unnecessary collection of video data in a central data lake. Only relevant video snippets containing PoIs, which have to be explicitly specified and approved beforehand, are sent to the cloud backend. In this section, we give an overview of our system design, starting with a high-level overview in section 2.1 before describing authorization schemes (section 2.2)

M4IoT'18, December 10–11, 2018, Rennes, France

ACM ISBN 978-1-4503-6118-7/18/12…$15.00
https://doi.org/10.1145/3286719.3286722

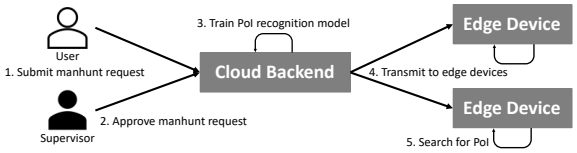

Figure 1: The Manhunt Creation Process

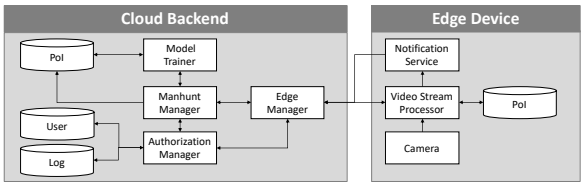

Figure 2: While the cloud management handles the manhunt management, the video stream processing is done on the edge devices exclusively

as well as face detection (section 2.3) and notification mechanisms (section 2.4).

2.1 Architecture and Manhunt Process

In this section, we will first outline the general process of initiating a manhunt before describing our overall system architecture and the distribution of system components over cloud and edge.

Figure 1 summarizes and illustrates the complete manhunt process: When a law enforcement agency needs to find a PoI, first, an officer with user rights submits a manhunt request. Such a manhunt request contains biometrical data about the PoI and also the area where the PoI shall be searched for, i.e., which edge devices should participate in the search. As required by the four-eyes principle, the submitted request must then be approved by someone with supervisor rights – this could be a higher ranking officer from another department or it could require a court order. Finally, once the request has been approved, a PoI recognition model is trained in the cloud backend and transmitted to the specified edge devices. These edge devices then actively search for PoIs in the video stream from their local camera and send any video snippets containing a PoI back to the cloud backend.

Figure 2 shows the overall architecture of our system: The central component of our cloud backend is the Manhunt Manager which controls the manhunt workflow and also provides a web frontend to system users. In this workflow, the Manhunt Manager continuously interacts with the Authorization Manager to assert proper authentication, authorization, and non-repudiation in all substantial workflow steps. To provide these guarantees, the Authorization Manager builds on an immutable event log (audit trail) and a user database; we will describe this in more detail in section 2.2. When the Authorization Manager has signaled that a manhunt request has been approved, the Model Trainer component takes over: It stores the photo set submitted along with the request in the PoI database and then uses the photo set to train a facial recognition model that can be distributed to edge devices. Once the model has been created, the Manhunt Manager again verifies that the requested manhunt has been correctly approved and then uses the Edge Manager to distribute the model to all relevant edge devices.

Connected edge devices wait for new manhunt requests and receive the facial recognition model containing the biometric data of the respective PoI. This model is stored in the local PoI database and is used within the Video Stream Processor. This component continuously monitors the live video stream from the connected camera and tries to match faces in the video stream to the models stored in the PoI database. If the Video Stream Processor detects and identifies a PoI, it creates a video snippet containing only the subsequence where the PoI was visible, and reports an alert to the Notification Service. The Notification Service then forwards the alert including the video footage and all available meta data to the cloud backend where they can be accessed by authorized users through the Manhunt Manager's web frontend.

While not implemented in our proof-of-concept prototype, we envision that the video stream should be buffered for a short period of time on the edge device (e.g., for 24 to 72 hours) so that new manhunt requests can also search in the near past. This also asserts that video data remains accessible locally for a short period of time in case an incident that creates a need for such data. While we explicitly envision access to this buffer to be strictly local (e.g., via a portable drive), the details of this buffering feature are not really a technical design decision but rather need to be considered for the tradeoff between privacy and public safety goals.

2.2 Authorization of Manhunts

Our design puts a strong focus on the protection of privacy and tries to reduce opportunities for system abuse. Specifically, our system design guarantees non-repudiation of executed actions, ensures that manhunts are properly authorized, and all video data remains inaccessible unless a PoI is contained in the respective video frames. To achieve these goals, we use three mechanisms: a role-based access control model, a four-eyes principle, and an audit trail system based on an immutable event log.

Our role model consists of a user and a supervisor role: supervisors can manage accounts and approve user actions; users can upload biometric information of PoIs to the cloud backend and request new manhunts. Of course, this needs to be combined with proper authentication measures.

To avoid situations where a single person could start an unfounded manhunt, we rely on a four-eyes process which guarantees the participation of two different authorized users for initiating a manhunt. For typical scenarios, we envision that the user who initially creates the manhunt request will be a law enforcement officer while the supervisor who approves the request should be a court ordering the manhunt. Beyond the manhunt process, we also rely on the four-eyes principle for actions such as management of user accounts, edge devices, and all access to existing manhunt data and notification results.

In most scenarios, the four-eyes principle should already suffice. To provide additional protection, however, for cases where a user and a supervisor have either been corrupted or cooperate, our system design uses an audit trail to ensure non-repudiation for all actions within the system. Although, malicious cooperation cannot be avoided this way, the log asserts that such actions can at least

be detected so that disciplinary or legal measures outside of our system can be initiated.

2.3 Face Detection

As already outlined in section 2.1, we leverage the compute power of the cloud to train a machine learning-based PoI recognition model based on photos of the respective PoI. An alternative approach might be to directly use the biometrical data stored in today's passports (and, thus, in government databases). The pretrained PoI recognition model is then distributed to edge devices while the cloud backend can continue to improve the model whenever new information becomes available. Furthermore, edge devices typically suffer from resource constraints so that analysis of the video stream should be light in resource needs – this is easily possible with a pretrained model.

There is one caveat though in this approach: The time to analyse the video stream scales with the number of faces in the video stream and the number of PoIs, i.e., in crowded places, edge devices need more compute power or can only catch up with their analysis during offpeak hours. This implies that there is a tradeoff between timeliness of analysis results and the cost of providing the necessary compute infrastructure – this tradeoff should be solved as required by the respective use case. Furthermore, there is also a large potential for optimization. For instance, two manhunt requests could have different priorities or the video analysis could start by tracing moving objects in a first analysis step and then running face analysis on such objects only once in a second step. This would help to avoid continuously reanalysing a person looking straight at the camera for a long time and should, hence, reduce the analysis load.

2.4 Notification Mechanisms

If an edge device recognizes a PoI in the video stream, multiple actions are possible. In our design, the Notification Service transmits a video snippet of the corresponding image sequence to the cloud backend. There, these snippets can only be accessed by authorized users who are responsible for taking further action such as storing the video snippet as evidence or notifying nearby police officers. Moreover, the reported footage can also be added to the manhunt picture set to further improve the PoI detection model if the recognition is deemed correct. Our notification mechanism, however, could easily be extended. Instead of (only) sending the notification to the cloud backend, the system might also notify nearby officers directly, stop train service on a particular subway line, or initiate some other action based on a rule engine.

3 PROOF OF CONCEPT IMPLEMENTATION

To validate our system design and to demonstrate its practical applicability, we implemented a proof of concept prototype. For this, we emulated a public video surveillance environment utilizing AWS cloud services as cloud backend and two Raspberry Pis with attached cameras as connected edge devices.

In the cloud backend, most of our functionality is implemented as AWS Lambda functions[1] in combination with AWS API Gateway[2] for access to all services. For the PoI recognition model, we used the open source library OpenFace[3] which is trained using the AWS Batch[4] service. All data and the web frontend are stored either in AWS DynamoDB[5] tables or AWS S3[6] buckets.

For authentication and authorization of users, we implemented a role-based access control model using Lambda functions, JSON Web Tokens[7], and the external service provider Auth0[8] for the assignment of tokens. Although this requires trust in external services, we believe that this suffices for a proof of concept prototype. For reasons of simplicity, we explicitly decided not to use an existing authorization system as our prototype only requires two non-overlapping roles.

For the audit trail, we used a blockchain-like data structure where log entries store SHA512 hashes of preceding entries in combination with public/private key cryptography to sign each entry. For communication between cloud backend and edge devices, we used the AWS IoT[9] framework which offers an as-a-service implementation of MQTT[10].

We have made our prototype including all installation scripts and a more detailed documentation available as open source on GitHub[11].

4 RELATED WORK

There are existing approaches in both fog computing in general as well as privacy control for video surveillance. For instance, as a general approach, Satyanarayanan et al. [7] propose to move compute-intensive cognitive assistance tasks closer to the end user by utilizing cloudlets. Their focus, though, is on performance gains.

Regarding privacy control, Erkin et al. [5] use secure multiparty computation and implement a comparison protocol for encrypted images to preserve privacy of uninvolved citizens. Chattopadhyay et al. [3] propose to increase the privacy through invertible cryptographic obscuration. In their approach, faces are encrypted with a hardware chip directly in the camera. Winkler and Rinner [8] extend this approach with a digital signature to guarantee integrity of the video stream. Bretthauer and Krempel [2] propose a system that detects accidents in nursing homes and only forwards the video stream to authorized employees when such an incident occurs. Zhang et al. [9] propose a system for video surveillance in the work place. Their approach recognizes personnel in the video stream and digitally removes them while retaining all non-authorized personnel. This way, privacy of employees in a work place can be preserved while still monitoring for unauthorized access. Our approach leveraging the geo-distribution of fog computing is an alternative approach which addresses a slightly different use case but which could also complement some of the other approaches, e.g., the one by Zhang et al. [9].

[1] aws.amazon.com/lambda
[2] aws.amazon.com/api-gateway
[3] cmusatyalab.github.io/openface/
[4] aws.amazon.com/batch
[5] aws.amazon.com/dynamodb
[6] aws.amazon.com/s3
[7] jwt.io
[8] auth0.com
[9] aws.amazon.com/iot
[10] mqtt.org
[11] github.com/OpenFogStack/PublicVideoSurveillance

5 LESSONS LEARNED AND DISCUSSION

As we have seen, it is indeed possible to find a middle ground solution between public safety goals of a society and privacy of citizens when using the geo-distribution of fog computing as the key enabler. Our proposed system design, which makes use of the fog computing paradigm effectively, shows that even small and comparably cheap edge devices with cameras are able to record and analyze video streams in a sufficiently good resolution to enable face detection on the edge. This way, the privacy of inhabitants is ensured as much as possible while still supporting camera-based manhunt scenarios. During our implementation process, we also encountered a number of challenges and problems:

First, the face detection algorithm requires as much footage as possible from the PoI, as it is not possible to perform a reliable recognition of subjects without a sufficiently high number of images in a acceptable resolution. For a PoI where only a single photo may exist, this is critical. We are, however, aware of ongoing research work that aims to reduce training data sets.

Second, there is no standard tooling support and methods for the roll-out process to edge devices, since fog computing is a relatively new paradigm and there are not many applications yet. This makes it a labor-intensive task to deploy a first version of a system to edge devices. We can only recommend to roll-out an intelligent update mechanism along with that first software version – otherwise, every single software update will again result in a manual and time-intensive deployment process. This also needs to account for anything that happens in the cloud with regards to new software versions, updated interfaces, etc.

Third, researchers tend to consider failure a corner case that rarely happens. This is obviously driven by the observable stability in today's cloud services. In the context of edge devices, however, this basic assumption should be reconsidered – we ran into a variety of small problems and one out of three of our Raspberry Pi edge devices suddenly overheated (while running a high but still normal processing load) that it permanently failed. While this is only a small sample size, it tells us to expect temporal (e.g., due to network problems) or permanent outages of edge devices: It should be obvious that a low budget device will usually not offer the same level of stability as a standard server.

Finally – a minor thing – while MQTT does not limit the message size, the AWS IoT service does, i.e., it is not possible to use video snippets as payload. While this was straightforward to work around (upload the video to AWS S3 and only send the S3 key via AWS IoT), this may not always be as easy to circumvent in case of larger IoT data payloads.

Besides these challenges, we also found a couple of possible optimizations for our design and proof of concept prototype. As face detection is an expensive algorithm, it should not be run on every single video frame. Instead, as already outlined above, it should be triggered based on a new person entering the surveillance area which should then be traced based on trajectories instead of re-analysing the same face over and over again. Other optimizations include compression before upload to the cloud or cropping irrelevant frame areas. Thus reducing the amount of data that is sent to the cloud is a necessary requirement for deploying the system in more bandwidth-constrained situations, e.g., in rural areas or during music festivals where mobile networks tend to be overloaded.

6 CONCLUSION

Video surveillance in public areas is the method of choice for dealing with increasing requirements in crime prevention, to support manhunts, as well as to generally increase the feeling of safety of citizens. Typically, though, this is done in a way that threatens privacy of citizens. Existing research approaches partly address this, e.g., through obfuscation of faces, but still collect all video data in a central data lake. This, however, significantly simplifies the correlation of data sets and, thus, unnecessarily endangers privacy of citizens.

In this paper, we proposed an alternative but complementary approach that leverages the inherent geo-distribution of fog computing to preserve privacy while still supporting law enforcement agencies through a digital manhunt feature. In our approach, law enforcement agencies can create manhunt requests for a specific PoI. Using photos of the PoI, a facial recognition model is pretrained in the cloud and then distributed to edge devices with attached cameras. These edge devices then monitor the local video feed using the pretrained model to detect the PoI. Only when the PoI is detected, the edge device uploads the respective video snippet to the cloud and notifies the original manhunt requester. To avoid misuse, our system follows the four-eyes principle and uses an audit trail system to implement non-repudiation. To evaluate our system design, we presented our proof of concept prototype using AWS and Raspberry Pis as edge devices. In lab setups, we found that even these comparatively cheap edge devices can provide a reliable recognition of PoIs while improving privacy in comparison to state-of-the-art cloud solutions.

ACKNOWLEDGMENTS

We would like to thank Jun-Zhe Lai, Florian Muchow, Tobias Pfandzelter, Calvin Schmidt, Florian Stanek, and Patrick Willmann who prototypically implemented the proposed system design within the scope of a master's course.

REFERENCES

[1] David Bermbach, Frank Pallas, David Garcia Perez, Pierluigi Plebani, Maya Anderson, Ronen Kat, and Stefan Tai. 2018. A Research Perspective on Fog Computing. In *Proc. of ISYCC*. Springer.
[2] Sebastian Bretthauer and Erik Krempel. 2014. Videomonitoring zur Sturzdetektion und Alarmierung–Eine technische und rechtliche Analyse. *Proc. of IRIS*.
[3] Ankur Chattopadhyay and Terrance E Boult. 2007. Privacycam: a privacy preserving camera using uclinux on the blackfin dsp. In *Proc. of CVPR*. IEEE.
[4] E-Hualu. 2017. Xueliang Project Solution. Retrieved June 27, 2017 from http://www.ehualu.com/en/Article/index/id/841/aid/12508137
[5] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. 2009. Privacy-preserving face recognition. In *Proc. of PETS*. Springer.
[6] Zac Johnson. 2017. How Closed Circuit TV and Artificial Intelligence Are Making Moscow Safer. Retrieved November 30, 2017 from https://tech.co/how-cctv-ai-are-making-moscow-safer-2017-11
[7] Mahadev Satyanarayanan, Zhuo Chen, Kiryong Ha, Wenlu Hu, Wolfgang Richter, and Padmanabhan Pillai. 2014. Cloudlets: at the leading edge of mobile-cloud convergence. In *Proc. of MobiCASE*. IEEE.
[8] Thomas Winkler and Bernhard Rinner. 2010. Trustcam: Security and privacy-protection for an embedded smart camera based on trusted computing. In *Proc. of AVSS*. IEEE.
[9] Wei Zhang, Sen-Ching S Cheung, and Minghua Chen. 2005. Hiding privacy information in video surveillance system. In *Proc. of ICIP*. IEEE.