2022 IEEE 6th International Conference on Fog and Edge Computing (ICFEC) | 978-1-6654-9524-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICFEC54809.2022.00018

Specification and Operation of Privacy Models for Data Streams on the Edge

Boris Sedlak
Distributed Systems Group
Vienna University of Technology
Vienna, Austria

Ilir Murturi
Distributed Systems Group
Vienna University of Technology
Vienna, Austria

Schahram Dustdar
Distributed Systems Group
Vienna University of Technology
Vienna, Austria

***Abstract*—The growing number of Internet of Things (IoT) devices generates massive amounts of diverse data, including personal or confidential information (i.e., sensory, images, etc.) that is not intended for public view. Traditionally, predefined privacy policies are usually enforced in resource-rich environments such as the cloud to protect sensitive information from being released. However, the massive amount of data streams, heterogeneous devices, and networks involved affects latency, and the possibility of having data intercepted grows as it travels away from the data source. Therefore, such data streams must be transformed on the IoT device or within available devices (i.e., edge devices) in its vicinity to ensure privacy. In this paper, we present a privacy-enforcing framework that transforms data streams on edge networks. We treat privacy close to the data source, using powerful edge devices to perform various operations to ensure privacy. Whenever an IoT device captures personal or confidential data, an edge gateway in the device's vicinity analyzes and transforms data streams according to a predefined set of rules. How and when data is modified is defined precisely by a set of triggers and transformations - a privacy model - that directly represents a stakeholder's privacy policies. Our work answered how to represent such privacy policies in a model and enforce transformations on the edge.**

***Index Terms*—Edge Computing, Privacy Models, Data Stream Transformations, Data Anonymization**

I. Introduction

In recent years, the number of Internet of Things (IoT) devices has widely increased while each device continuously generates massive amounts of various data. Several examples exist; for instance, a security camera in a smart factory that produces a constant stream of high-quality images or a sensor network in a car that opportunistically provides live information about the road conditions. Traditionally, data produced in such scenarios is processed or aggregated at a central cloud server. Research literature shows that data often can be significantly large (e.g., image, audio, video streams, 3D content, etc.) and usually requires to be processed with low latency [1]. Nevertheless, the massive amount of data streams, heterogeneous devices, and networks involved causes high traffic and affects the overall latency [2].

One prominent approach that has emerged in recent years suggests utilizing distributed computation entities (i.e., perceived as edge devices) in proximity to end users, respectively at the edge of the networks [3]. Edge devices available in vicinity to end users can be leveraged to process various data or workloads instead, and as such the edge emerges as a central architectural entity. As a result, the key theme is that these processing elements in proximity to end users first aim to avoid user-perceived latency, reduce the need to transfer data to the cloud, and improve privacy by analyzing released information by users. The latter plays a core role to protect and secure user data that is released without users' consciousness or information that can violate privacy policy requirements defined by a stakeholder (e.g., company, school, etc.). In this context, edge devices and, in general, edge architectures play an important role because they support network affairs and improve overall privacy protection. For instance, edge-based infrastructures provide a seamless opportunity for accelerating the development of data-centric tasks such as crowd-based platforms [4], [5]. In crowd-based tasks, edge devices can act as an intermediary entity to protect released information and enforce various privacy strategies to make sure that sensitive data (i.e., user information, sensory data streams, etc.) is not released. Therefore, we need an edge-based mechanism to prevent sensitive information to be released and to ensure for contributed data that third parties cannot identify individuals without their consent.

Edge-based infrastructures are heterogeneous environments with various devices featuring different capabilities; dedicated edge servers, network routers, telecommunication stations, or simple edge gateways that aggregate sensor data in healthcare or smart city environments [6]. Because of this heterogeneous nature of edge-based infrastructures, we require a standard format of how data has to be transformed to ensure privacy - more precisely, one specification serves for all possible types of edge devices. Specifically, we refer to transformations as modifications of data or its metadata, an operation that arbitrarily combines or discards information. Transformations are derived from privacy policies, a set of rules on how to transform data to ensure privacy on the data once it has passed through the edge device. Especially, we would eliminate the need to implement a policy multiple times for different types of edge devices by providing a common runtime environment on these devices where privacy transformations are executed. For instance, an IoT device captures personal or confidential data which is forwarded to an arbitrary device on the network's edge where the data stream is to be transformed. We might choose any edge device here since the uniform environment will analyze and transform the data stream according to the

978-1-6654-9524-0/22/$31.00 ©2022 IEEE
DOI 10.1109/ICFEC54809.2022.00018

predefined set of privacy policies.

This paper presents a privacy-enforcing framework that transforms data streams on the edge. We enforce privacy close to the data source, using powerful edge devices to perform operations that would otherwise absorb resource-limited IoT devices whose software stacks are limited. We assume that a central trusted entity manages edge devices and the privacy policies for the entire data pipeline, ensuring exactly how and when privacy is enforced. As soon as data streams flow through an edge device that supports such transformation, we have complete control over the information flow in our system. Therefore, whenever an IoT device captures personal or confidential data, an edge gateway (i.e., edge device) in the IoT device's vicinity analyzes and transforms the data stream according to a predefined set of rules. Such rules build up privacy models, a representation of their privacy policies which consist of a set of triggers and transformations that describe how data must be modified before it can be released to stream subscribers.

The rest of the paper is structured as follows. Section II gives an overview of our approach along with a motivating example used throughout the paper. In Section III, we describe in detail the framework, the processes, and details of the proposed architecture. Related work is considered in Section IV. Finally, Section V concludes the paper and outlines future work directions.

II. Use Case & Motivation

The most intuitive examples of areas that can benefit from such applied privacy models are those within surveillance environments. Specifically, private information might be recorded unintentionally just because a person or object was present in the area by the time data was collected. We can imagine an office building where the meeting room has a camera for security reasons. It might not be desired to capture the content of a given presentation or laptop screen; otherwise, confidential information might be leaked by somebody who gets hold of the live stream or a recording. The exact rules and restrictions for privacy can be extracted from the company's policies that they have agreed on within their enterprise; those will be translated into a privacy model on how to transform data before it can be released to the public or recorded internally. Transforming written agreements directly into privacy models is itself an ongoing challenge as described in [7] and [8], so we stick to manual conversion here. Once a company has come up with a model representing their privacy policies, it is a rule that laptop screens are blurred out in every video frame. This has to be ensured before anybody can consume the video stream. Supposedly, a conventional camera installed within the office cannot perform this video transformation directly. This task will be performed in the device's vicinity on the network edge before directly streaming the video to recipients.

Edge gateways provide sufficient computational resources to perform stream transformations close to the data source; this is especially important with the emergence of GPU-accelerated single-board computers. For instance, NVIDIA Jetson [9], can be installed close to IoT devices to perform fast image recognition. In Figure 1 it is depicted how a captured image from an IoT device is transferred to an edge device in its vicinity. The edge device transforms the data by the rules defined in the deployed privacy model before being streamed to an audience or being recorded internally.

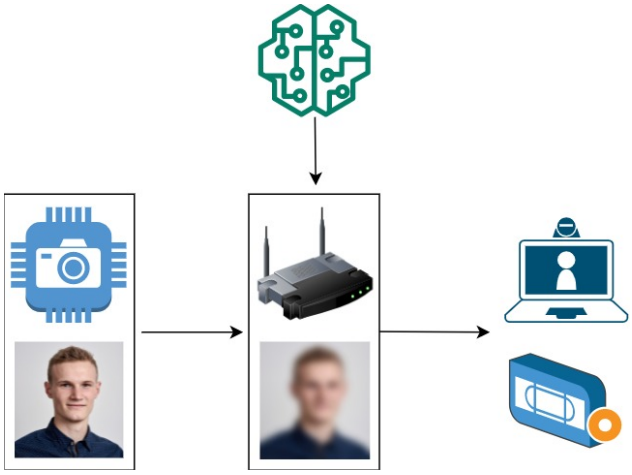

Fig. 1. Transforming streamed data according to a privacy model.

Privacy models can describe all sorts of stream transformations; some of them might seem unrealistic from a state-of-the-art perspective because they can hardly be supported with current technologies. For instance, detecting who is speaking and removing a certain audio layer from the video stream, but doing the same for images by detecting an individual in a video frame seems well possible. Machine Learning (ML) models have already shown that they can successfully identify celebrities after being trained on a sufficient number of images [10], so this can be the case for a company's employees as well. Suppose an ML model is regularly extended with new faces or voices that have been learned. In that case, the respective privacy model might as well need to be updated, and changes propagated to edge gateways. To maintain the interoperability of heterogeneous edge gateways, these devices have to support a consistent environment that allows to frequently (re-)deploy models and perform the described transformations on handled data streams. In case that the model is extended with novel transformations that are not yet supported by the edge device's environment, the environment has to be upgraded equivalently on all edge devices in the network. Hence, the environment that supports the transformations can be extended arbitrarily with new transformations by a company or an agency as they see fit. Still, the structure of the model itself should not change.

IoT devices are not limited to generating audio and video data, neither should the enforcement of privacy policies on edge devices be. In the context of a smart city, cameras and sensor networks can capture all kinds of information of vehicle movement within the road and parking network, like their exact speed, location, and car type. Edge gateways in these infrastructures can be located in a traffic junction's vicinity or in

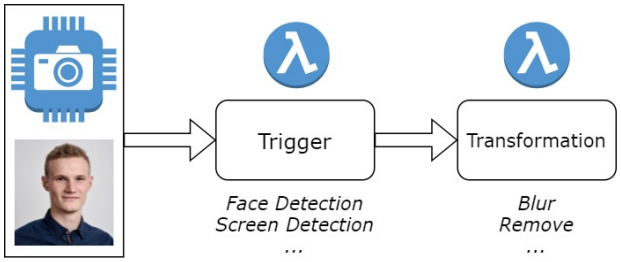

Fig. 2. Defining triggers and transformations for the privacy model.

a parking garage, where sensor data is aggregated before being transferred to a central monitoring unit. To avoid any misuse of the collected data, like tracking an individual's movement, private information (e.g., car plate) is either not captured or removed by a transformation imposed by a deployed privacy model. However, it might still be possible to track this car's movement. Supposed there is a car type that is not common in the area, as an outstanding sports car. The mere appearance of this car in any traffic junction or parking area allows for precise tracking of this car because the lack of equivalent cars makes it easy to track down this one car. For static data, it is well reported how to release a tuple only if there are sufficient similar values in the data set. Specifically, this is commonly known as k-anonymity [11]. However, it is possible to perform this anonymization on streams as well [12] by maintaining a history of processed tuples and releasing a tuple only if there have been sufficient similar tuples in the recent past.

Several different examples exist where edge gateways can help protect privacy on streamed data. Therefore, some of these examples might only emerge soon, even though the gateway could already provide the most common stream analysis methods and transformations. In this sense, we advocate that the edge gateway capabilities must be easily extendable according to stakeholder needs [13]. For instance, a precise anonymization technique (i.e., blurring a face) is just an extension that an edge gateway can support. A policy manager can then combine these transformations into a privacy model. This calls for a modular separation of the transformation functions from the edge gateway, allowing developers to define new transformations and analysis methods that fit a company's demands without modifying the gateway's source code.

III. The Framework

There are three significant parts of the proposed framework that need to be addressed—first, the structure of the privacy model and how rules and transformations are expressed. Afterwards, one should note how these models should be executed on the streamed data. And third, the structure of the edge network. Specifically, one should note how data is streamed between an IoT device and an edge gateway. Furthermore, an interesting point is how the output data can be consumed after assuring that all privacy policies have been respected.

In a streaming scenario, the most natural representation of how a privacy model can be expressed is a flow diagram, expressing how data is step-by-step analyzed to find certain patterns in the stream and then transforming the data in case a condition was fulfilled. Figure 2 shows how we define triggers and transformations on an image; such cause-and-effect rules can as well be chained so that one rule only applies once another has come true. The privacy model builds up to a set of such rules that are expressed in an acyclic graph, where the next step can always be determined by following the flow towards its end. The streamed data type already restricts what triggers and transformations are possibly defined in the privacy model, e.g., image recognition with a subsequent replacement does not make any sense if the stream tuples contain car plates paired with locations. A model compiler is executed on the edge gateway before applying a privacy model to assure that it matches a valid grammar.

Some triggers and transformations do not require a state, like static analysis of images containing a specific pattern. These operations can easily be expressed as stateless lambda functions that are chained together in the aforementioned acyclic graph, passing the result on to the next step before eventually returning the result to the stream subscribers. However, in some cases, we would need to keep a state similar to the example with z-anonymity in Section II. In these cases, the environment has to supply temporary storage that lambda functions can address for such matters. AWS lambdas [14], and NVIDIA Deep Streams [15] are examples of how such functions can be combined by dragging and dropping them through a UI. This approach supposedly does not require any programming skills from the responsible defining the privacy model. The model graph itself is defined in a central cloud application from where it is deployed to all edge gateways. After compilation, the model is active and incoming data streams are analyzed and transformed according to the defined criteria. Lambda functions and the privacy model are maintained in distinct modules, separated from the stream processing. This allows the gateway to continuously transform a data stream without resetting active connections whenever a new privacy model or new lambda functions are received. Updates of the privacy model and the lambda functions must be as lightweight as possible to not impact the device and network performance, maintaining a stable stream latency.

Edge gateways must not only handle in- and outgoing data streams but also decode stream data and re-encode it after performing the transformations imposed by the privacy model. WebRTC [16] is primarily a protocol for continuous streaming of videos and other data between two peers. However, one peer can be extended with routing functionalities to receive and forward data streams between multiple clients; this role is called a Selective Forwarding Unit (SFU) and is assumed by the edge gateway. Figure 3 contains all the major parts of the architecture:

1) A cloud application where a policy manager can define new lambda functions and policy models, which communicates with the edge gateways through their exposed REST endpoint for configuration.
2) Multiple IoT devices that stream data to the edge gateway

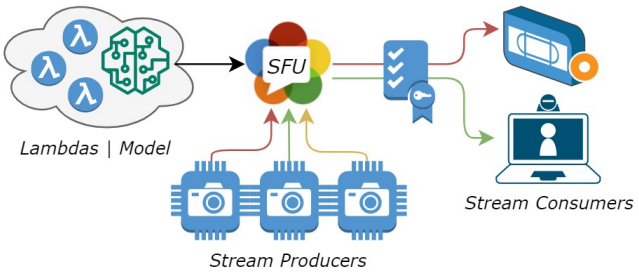

Fig. 3. Conceptual framework.

through a supported protocol, including but not limited to WebRTC. If there were a serial connection to the gateway, it could use that for transfer.

3) Different types of stream consumers which can be consumer devices or even recorders. Regardless of their purpose, they must all support the WebRTC protocol.
4) The core of the topology is the SFU, which is deployed on the edge gateway. It receives new models and functions through a REST endpoint without interrupting the ongoing stream connection. Incoming stream packages are decoded and transformed according to an active privacy model before streaming them to connected peers.

The SFU establishes connections to producer and consumer devices through a dedicated Session Description Protocol (SDP), which contains information about the session and the data type to be transferred. Once both peers have agreed on the content of the stream, they establish a peer-to-peer connection. WebRTC provides more than one channel to communicate between peers through the given connection [16]: a media stream that itself consists of two encoded tracks - audio and video - that do not need to have a relation (i.e., originate from the same video file), and the data channel, a bidirectional connection that can exchange any data encoded in text or byte arrays. In this framework, data is always transferred and analyzed in frames; this is the same for audio, video, or other streamed data. This means that privacy violations that would only emerge from analyzing multiple consecutive frames at once cannot be detected. The SFU waits for incoming frames on each channel, which are then analyzed and transformed according to the privacy model. Depending on the type of data that arrives, the SFU's environment has to provide different analysis methods. For instance, if we want to stream video frames over the SFU, we need to analyze incoming frames with a suitable environment that supports video operations, like OpenCV [17]. Through our privacy model we can then express a function that detects privacy violating patterns in the video frame (trigger). Once we detect e.g. a face in the video frame, the respective area is blurred (transformation) and the frame is routed to connected consumer devices. Extending the SFU with other environments makes it possible to analyze and transform data frames in various ways.

The bottleneck of this architecture is supposed to be the edge gateway, more precisely decoding and encoding the packages and running the model's functions on the packages. As the chain of triggers and transformations grows that was explained for Figure 2, the latency of the stream grows equally. This requires monitoring to give feedback on how long the chain of transformations and the en-/decoding process takes. One mechanism that can be implemented to cope with increasing or unstable latency is an adaptive bitrate, regulating the video quality according to the network quality and the transformations' time. In other words, if the network quality is low, then the video bitrate is decreased by the SFU and the IoT devices, respectively.

All data transferred is encrypted using WebRTC [18]. Essentially, WebRTC communicates through Transport Layer Security (TLS), which is used to secure HTTPS connections; therefore, any stream data sent over a media stream or data channel is as secure as any other data sent or received by a web browser. Since WebRTC is a peer-to-peer connection between two agents, the data never passes through web or application servers. This drastically reduces opportunities to have the data intercepted.

IV. Related Work

Anonymizing streaming data is one measure to protect the privacy of data contributors. Authors in [19], propose data collection schemes for IoT sensors that do not give evidence of the data source. This can be seen as an extension because it obfuscates the connection of an IoT sensor and the measured data; nevertheless, removing private information from the data is still an ongoing challenge. In [20], the authors present an edge-based system that removes private attributes from sensor data. They propose an ML model that identifies which attributes to erase in the transformation step. However, they did not extend their approach to continuous data streams and privacy models representing a company's policies.

The authors in [8] discuss how role-based access control schemes can be expressed through privacy models. They implemented an edge-based system where data is consumed over a message broker after transformation. Their privacy models focused on a variety of access control restrictions combined with token-based authorization. More generally, in [21], the authors discuss privacy considerations on edge, including stream processing and anonymization techniques that can be applied, like the aforementioned z-anonymity. They also discuss how federated learning protects privacy by maintaining training data in the edge level, a thought that [22] pursue for image recognition with deep learning. Furthermore, the authors assess the advantages of edge networks for ML and how a model can be trained and extended on the go. Still, the low latency in edge computing is also advantageous for other feedback mechanisms. Especially important in the context of this paper, the authors in [23] present how video streams can be transformed on edge. If the latency of the received stream drops due to increased processing on the edge device, they scale up to several stream transformation workers. Maintaining a low and stable latency by such means is definitely a

useful extension for stream transformations scenarios. Still, it requires an unavoidable overhead in communication and the means to scale edge devices horizontally.

The above-mentioned research works show that the edge has found a lot of attention for various tasks, including enforcing privacy policies. Up to now, research literature shows that transformations on edge were dedicated entirely to specific and selected operations (e.g., image removal). Nevertheless, it remains a step towards a general model of how to specify policies and enforce a variety of transformations expressed by the introduced framework. As mentioned above, several research works exist on specifying privacy requirements for enterprises. However, these works focus on business processes and do not consider low levels aspects such as removing confidential attributes in a continuous data stream.

V. Conclusion & Future Work

We have presented a privacy-enforcing framework that describes how data streams are transformed on edge networks close to the IoT device's data source. The two primary components of this architecture were how to specify policies in a privacy model through chained lambda functions and enforce these rules on streamed data. We assume that the data is contributed and received through an SFU, responsible for routing the data and, most importantly, decoding stream packages, transforming its content, and re-encoding it. All data communicated through WebRTC is secured with TLS; therefore, the data flowing through the IoT device, over the SFU, to the stream consumer is encrypted. Due to the absence of a central cloud server we have a small number of peers handling the stream, which drastically reduces the chance of having the data intercepted. Additionally, as soon as the stream leaves the SFU, the potential damage of having data intercepted is significantly lower since the information that is considered confidential has been removed.

This paper is only a small step towards a general framework for specification and operation of privacy models for data streams on the edge. Regarding future work, we first plan to implement the proposed framework as a proof of concept in a suitable programming environment which supports the desired features. Afterwards, we will evaluate this prototype in terms of streaming latency and performance to see to what extent it is applicable in a desired usage scenario. By doing so, we narrow down problems that the framework might present, whereby it can be improved gradually.

Acknowledgment

Research has received funding from the EU's Horizon 2020 Research and Innovation Programme under grant agreement No. 871525. EU web site for Fog Protect: https://fogprotect.eu/

References

[1] I. Murturi, C. Jia, B. Kerbl, M. Wimmer, S. Dustdar, and C. Tsigkanos, "On provisioning procedural geometry workloads on edge architectures," in *Proceedings of the 17th International Conference on Web Information Systems and Technologies, WEBIST 2021*, pp. 354–359.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[4] O. Alvear, C. T. Calafate, J.-C. Cano, and P. Manzoni, "Crowdsensing in smart cities: Overview, platforms, and environment sensing issues," *Sensors*, vol. 18, no. 2, p. 460, 2018.

[5] B. Rexha and I. Murturi, "Applying efficient crowdsourcing techniques for increasing quality and transparency of election processes," *Electronic Government, an International Journal*, vol. 15, no. 1, pp. 107–128, 2019.

[6] I. Murturi and S. Dustdar, "A decentralized approach for resource discovery using metadata replication in edge networks," *IEEE Transactions on Services Computing*, 2021.

[7] A. K. Massey, J. Eisenstein, A. I. Antón, and P. P. Swire, "Automated text mining for requirements analysis of policy documents," in *2013 21st IEEE International Requirements Engineering Conference (RE)*, Jul. 2013, pp. 4–13, iSSN: 2332-6441.

[8] P. Baniya, G. Bajaj, J. Lee, A. Bastani, C. Francis, and M. Agumbe Suresh, "Towards Policy-aware Edge Computing Architectures," in *2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020, pp. 3464–3469.

[9] Nvidia, "Embedded Systems for Next-Gen Autonomous Machines." [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

[10] N. Kasim, N. Rahman, Z. Ibrahim, and N. N. Abu Mangshor, "Celebrity Face Recognition using Deep Learning," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, pp. 476–481, Nov. 2018.

[11] M. Khavkin and M. Last, "Preserving Differential Privacy and Utility of Non-stationary Data Streams," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 29–34, iSSN: 2375-9259.

[12] N. Jha, T. Favale, L. Vassio, M. Trevisan, and M. Mellia, "Z-anonymity: Zero-delay anonymization for data streams," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 3996–4005.

[13] S. Dustdar and I. Murturi, "Towards distributed edge-based systems," in *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, 2020, pp. 1–9.

[14] A. Sabbioni, L. Rosa, A. Bujari, L. Foschini, and A. Corradi, "A Shared Memory Approach for Function Chaining in Serverless Platforms," in *2021 IEEE Symposium on Computers and Communications (ISCC)*, Sep. 2021, pp. 1–6, iSSN: 2642-7389.

[15] Nvidia, "DeepStream SDK." [Online]. Available: https://developer.nvidia.com/deepstream-sdk

[16] WebRTC, "Real-time communication for the web." [Online]. Available: https://webrtc.org/

[17] OpenCV, "OpenCV." [Online]. Available: https://opencv.org/

[18] Mozilla, "Using WebRTC data channels - Web APIs." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Using_data_channels

[19] A. Wang, J. Shen, C. Wang, H. Yang, and D. Liu, "Anonymous data collection scheme for cloud-aided mobile edge networks," *Digital Communications and Networks*, vol. 6, no. 2, pp. 223–228, May 2020.

[20] O. Hajihassani, O. Ardakanian, and H. Khazaei, "Anonymizing sensor data on the edge: a representation learning and transformation approach," *ACM Transactions on Internet of Things*, vol. 3, no. 1, pp. 1–26, 2021.

[21] C. Tsigkanos, C. Avasalcai, and S. Dustdar, "Architectural considerations for privacy on the edge," *IEEE Internet Computing*, vol. 23, no. 4, pp. 76–83, 2019.

[22] Y. Mao, J. Feng, F. Xu, and S. Zhong, "A Privacy-Preserving Deep Learning Approach for Face Recognition with Edge Computing," *HotEdge*, 2018.

[23] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: latency-aware video analytics on edge computing platform," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, ser. SEC '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1–13.