Blockchain Enabled IoT Edge Computing: Addressing Privacy, Security and other Challenges

Pankaj Mendki
Talentica Software Pvt Ltd.
Pune, India 411045
pankajm@talentica.com

ABSTRACT

The earlier work related to this paper described blockchain enabled IoT edge computing architecture, where any compute resource owner can join the ecosystem and lend out the compute resources required for edge computing/analytics. This paper lists possible set of challenges implementing the above solution. The primary challenges are ensuring data privacy for the input data files and ensuring computation integrity when executing the computation remotely. The paper talks about current state research and possible practical implementation to address the privacy and integrity challenges using hardware assisted trusted executing environment primarily Intel SGX. The paper proposes using unidirectional payment channels to address payment related challenges.

CCS Concepts

• **Computing methodologies → Distributed computing methodologies**

Keywords

IoT; Edge computing; Blockchain, Trusted execution environment (TEE), Payment channel.

1. INTRODUCTION

Number of internet connected devices is exponentially rising every year. Devices like drones, connected vehicles, wearable gadgets, smart robots etc. are producing immense amount of data at the source. Edge computing is now becoming the de-facto architecture to process the data generated by these internet of thing (IoT) devices. Edge computing adoption is accelerated by the growing maturity of the edge computing platforms and evolving technologies like 5G and Wi-Fi 6 [1].

In the last decade, blockchain technology has born as electronic cash system and evolved via multiple forms like crypto-currencies, smart contract and decentralized applications. One of the possible decentralized applications is blockchain enabled IoT edge computing proposed in this work [2]. The current paper is extension to this work. The structure of the paper is as follows.

Section 2 explains the background about the previous work, the proposed architecture for blockchain enabled IoT edge computing solution and implementation challenges. Section 3 describes issues related to data privacy, it also explains possible solution to address these challenges with emphasis on trusted execution environment based approach. Section 4 explains how to ensure the computational integrity and validate the results. The next section highlights challenges related to payment; in the next section packaging and deployment challenges are explained. This is followed by the conclusion section.

ICBCT'20, March 12–14, 2020, Hilo, HI, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7767-6/20/03…$15.00
https://doi.org/10.1145/3390566.3391689

2. BACKGROUND

2.1 IoT Edge Computing

Typical IoT based applications have sensors producing data and sending to the nearby gateway. Gateway devices then send data to the cloud server for further processing or storage. Fog and Edge computing architectures [3] help reducing the compute and storage resource overloading at cloud infrastructure by exploiting the resources close to the data source. Edge computing helps to reduce the data transfer cost and latency and mitigate the scalability challenge for the central cloud based architecture. Edge computing is becoming an integral part of IoT solution, most of the leading IoT cloud platforms [4, 5] are also providing edge computing as their offerings.

2.2 Blockchain

Blockchain was introduced as Bitcoin [6] cryptocurrency – a peer to peer electronic cash system. Blockchain is a form of distributed ledger where the copy of ledger resides with all the participants; the next entry to the ledger is appended when majority of nodes are in the agreement with the new state. Some of the blockchain implementations evolved to provide the mechanism of smart contracts [7] – these are the programs which can be executed on every blockchain nodes resulting in the state change of the blockchain, if the changes are successfully validated and agreed upon by the majority of the nodes.

Blockchain implementation could be public and permissionless, where any new node can join the blockchain network to read and propose a new block. Other flavor of blockchain could be private and permissioned which restricts the membership to blockchain network and also restricts the permission to propose a block. There are multiple open source frameworks [8, 9] that let you create private and permissioned blockchain, write smart contract in higher lever programming languages and build blockchain applications.

One of the possible applications is blockchain enabled IoT edge computing proposed in [2]. The following section describes high level architecture for the solution.

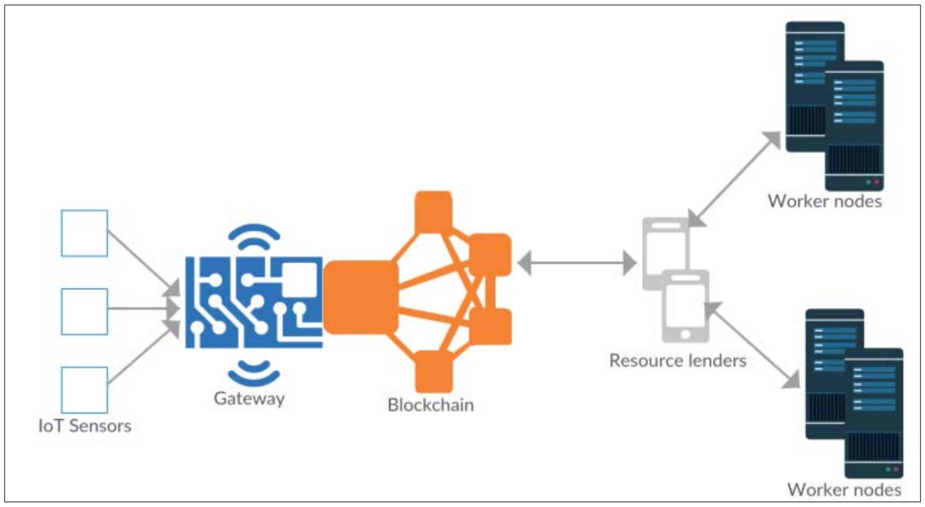

Figure 1. Architecture for edge computing over blockchain [2]

2.3 Existing Architecture

In a typical IoT application, data captured by IoT sensors is sent to the connected gateway. Gateway devices generally have long range connectivity for sending the data to remote serves for downstream processing.

With edge computing architecture, some of the data processing like aggregation or analytics can be processed closer to the gateway. Gateway devices may not have enough resources required for the edge processing. In the proposed solution, the edge processing can be executed by leasing the required compute resources; blockchain can facilitate this mechanism.

The component architecture for the proposed solution is shown in Figure 1. The major components involved are:

2.3.1 Blockchain Network

This is a permissioned blockchain network. All the nodes have end-points where external client applications can submit transaction request. The network validates the incoming transaction and keeps record of those transactions in the distributed ledger.

2.3.2 Client or Requester

Gateway device acts as a requester, where it creates a transaction for processing a compute intensive task (a.k.a a job); the input required for the processing is time series data received from the IoT sensors.

2.3.3 Decentralized Storage

The required input data files will be uploaded to decentralized storage like InterPlanetary File System (IPFS)[10]. Code executables and the required artifacts can also be uploaded to this.

2.3.4 Resource Lender

Any user can join as a resource lender. Resource lenders would lend out their compute resources to execute jobs.

2.3.5 Worker Node

Resource lender owns worker nodes where the computation task or job is actually executed. Once the job is executed successfully on a worker node, resource owner submits the output as a transaction to the blockchain network.

2.4 Usecase

The proposed architecture can be deployed in the scenario where IoT sensors are producing high volume of data and the processing demand at the edge can’t be fulfilled by the resources available at the edge. This could be applicable for surveillance cameras where the captured video stream is to be processed to analyze some threat scenario or to analyze the video content and produce a searchable indexed repository of the video content [11].

2.5 Challenges

The possible challenges involved in the implementation of the above solutions are

1. Preserve code and data privacy when running it on external workers
2. Ensure that the intended code is actually being executed and the output of the execution is shared by the worker/resource lender without tampering it.
3. Ensuring proper reward system to incentivize resource lenders
4. Packaging and distributing to ensure execution in the sandboxed environment.

These challenges and their possible solutions are discussed in the subsequent sections.

3. PRIVACY

Resource lender owns workers nodes which execute the computation task over the given input data for that task. The edge analytics processing could vary from simple aggregation to complex machine learning inference that uses some machine learning model and predicts the output for the given job input data. In either case, the code that is executed, the input data and the machine learning model if any should be protected. This work [12] shows that it is possible to extract sensitive data like social security number (SSN) from trained neural network model. Also the input data for every task is likely to expose sensitive information.

Using hardware-assisted trusted execution environments (TEEs), it should be possible to mitigate the above issues. In the last few years, hardware vendors have been actively working on improving the hardware-assisted TEEs [13, 14, 15]

3.1 Trusted Execution Environment

TEE provides a secure area in the main processor. Code and data can reside in this area. It ensures the security in terms of data privacy and integrity, even if the system software on the device is compromised. The popular TEEs are Intel Software Guard Extension (SGX) [13], ARM Trust Zone Technology [14] and AMD Memory Encryption Technology.

Assuming most of the worker nodes are likely to be desktop and/or laptops with idle computing resources, this paper primarily focuses on Intel SGX based approach.

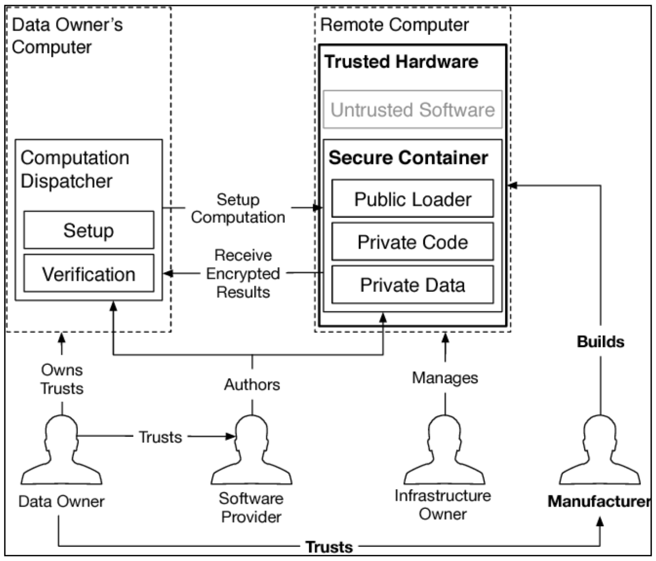

Figure 2. Trusted computing [16]

3.1.1 *Concepts*

The trust relationship for the Intel SGX based secure remote computation is shown in Figure 2. The data owner need not trust infrastructure owner of the remote computer – which is the resource lender, instead the data owner trusts the manufacturer of the hardware (Intel) on the remote computers.

Since the release of Intel SGX, it has attracted many researchers attention. There have been many application prototypes developed by researchers, some of those are in the field of machine learning [17-20], there are some applications proposed in the field of blockchain as well [21, 22]. Some development frameworks are also proposed based on Intel SGX [23-27, 31, 32]

3.1.2 *Privacy Enforcement*

Intel SGX application developer can develop an application using SDK provided by Intel. When the application is executed in the userspace, it creates an isolated memory region called enclave. Code and data within enclave are not accessible by operating systems and other processes (Refer Figure 3). When SGX enabled application is launched, it has two parts, a secured part and non-secured part, the secured part resides in the enclave. When a function within enclave code is invoked it can access enclave data, enclave data access is restricted only to the code inside that enclave, when the function returns, the code stays in the protected memory. SGX ensures the confidentiality and integrity of the enclave code and data.

Let's revisit the scenario where we need to protect privacy of machine learning model and input data for tasks. This data can be encrypted using symmetric encryption secret key and uploaded to the decentralized storage. On a remote worker machine, untrusted code can download the encrypted model and data. If the secret key is available in the enclave data, code running inside that enclave can decrypt the model and task input data. This ensures the confidentiality of the machine learning model and task input data. We need to setup a secure channel to share the secret key to enclave from outside – it could be a blockchain node or a dedicated server which can send the secret key to the enclave. For this enclave should undergo remote attestation, which is described in the next section.

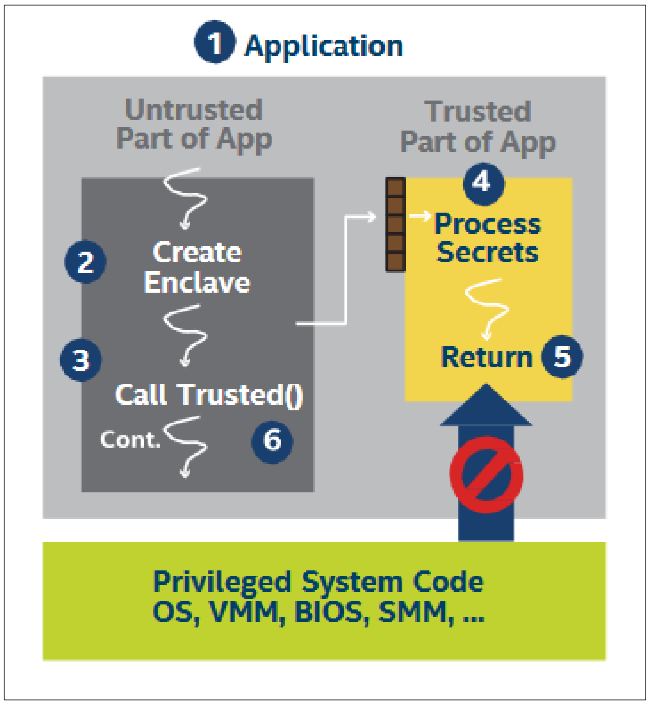

Figure 3. SGX application runtime execution [29]

4. RESULTS VERIFICATION

Another challenge is to ensure that worker nodes are actually executing the code and sharing the same results generated by the code. The remote attestation service can provide the identity of the software being attested and assessment of the software tampering [28]. The process is described below.

4.1 Remote Attestation

Enclave's author/developer adds a self-signed certificate, also called as enclave signature - SIGSTRUCT. The enclave signature contains information that allows the Intel SGX architecture to detect whether any portion of the enclave file has been tampered with. When the enclave is loaded, CPU calculate 256-bit hash of enclave code and data, called MRENCLAVE. This is matched against the value in the enclave signature SIGSTRUCT.

Attestation is mechanism to demonstrate remote entities that enclave is instantiated in the correct manner. In our case, attestation will ensure remote worker node is executing the correct code. The process works as follows:

Remote party, in our case could be blockchain node or a server component, sends an attestation challenge to the application running on the worker node. It in return receives an attestation report (quote) generated by the enclave. Remote party forwards this attestation report to Intel Authentication Service (IAS), which verifies it using Enhanced Privacy Identification (*EPID*) group signature scheme [28]. Attestation result confirms whether the enclave has loaded with the desired code or not. The attestation report can also contain manifest or custom data, it could be the ephemeral public key generated by the enclave. The remote party can share secrets, keys or credentials encrypting them using the enclave's public key and then share it with the enclave.

Data loaded inside enclave is volatile, before enclave exits, this data are persisted by encrypting using CPU/enclave specific sealing key (Refer Figure 4).

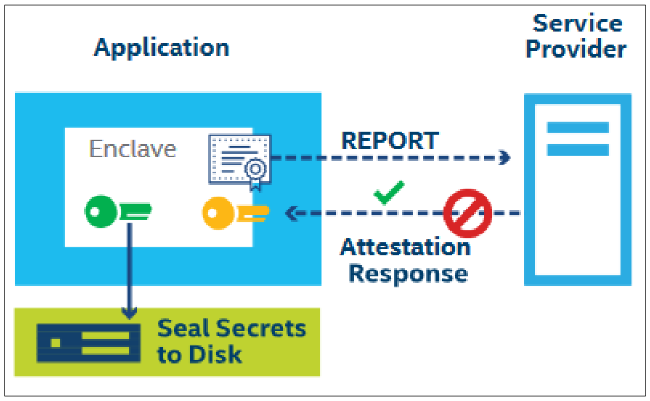

Figure 4. SGX remote attestation and sealing [29]

Using the shared secret, results produced by enclave code execution are encrypted and can be shared over blockchain and decentralized storage.

Attestation ensures valid code is getting executed in the enclave; it also enables sharing remote secret which can be used to encrypt/decrypt the input data and output results.

5. REWARD SYSTEM

Resource lender should get paid when the valid results are submitted and accepted by the requester. It is most likely that a requester is going to create multiple jobs and the resource lender will pick many of those jobs for execution. In the current system, every job is producing a set of blockchain transactions even if these transactions are happening between the same set of parties. Using unidirectional payment channels these transactions can be batched and executed offchain.

5.1 Payment Channels

To initiate a payment channel, a blockchain transaction to create secured deposit is required. Another blockchain transaction is required to release the deposited fund when both the involved parties agree up the settlement state [30]. The intermediate payment transfers are not recorded over blockchain. The secure deposit will safeguard interest of the resource lender to enter into the payment channel. For every job result submitted by the resource lender, a micro-payment is to be released from the secured deposit. The payment will not be immediately transferred but the state changes are recorded offchain. Settlement of the payment happens when both the parties agree upon the current state – this can be implemented by multi-sig protocol. If the closure is not instantiated by anyone of the party, the channel can get timed and auto-settled as per the last valid state.

Using micro-payment over the payment channel ensures protecting interest of both parties; resource lender and requester. Smart contract can enforce that, when the results submitted by the resource lender are valid (based on signature), micro-payments should be released. The maximum risk for the resource lender is equal to the cost of a single micro-payment.

6. PACKAGING AND DEPLOYMENT

Application to be executed on remote workers should be executed in a sandboxed environment to ensure the deterministic output independent of the underlying OS. Container based environment like docker, kubernetes can be used with minimal overhead as compared to the native deployment [33].

Docker support for SGX is not fully available although there are SGX enabled docker images available. This work [34] compares the SGX support of multiple container technologies. There are challenges with respect to enclave page cache (EPC), remote attestation and ensuring reduced performance overhead. As per [36], Kubernetes cluster can support SGX. Microsoft Azure confidential computing platform has announced kubernetes based deployment [35].

7. CONCLUSION

Blockchain enabled IoT edge computing implementation poses privacy and integrity as primary challenges. Intel SGX based solution seems practically implementable to address those challenges. Using SGX based approach poses new challenges like packaging and distribution over container environment in terms of security and performance overhead.

The token based reward system can be implemented using unidirectional payment channels.

8. REFERENCES

[1] Predictions 2020: Edge computing expected to take a big leap in 2020. https://go.forrester.com/ blogs/predictions-2020-edge-computing/. (Accessed on 01/28/2020).

[2] Pankaj Mendki. Blockchain enabled iot edge computing. In *Proceedings of the 2019 International Conference on Blockchain Technology*, ICBCT 2019, page 66–69, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362689. . URL https://doi.org/10.1145/3320154.3320166

[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450315197. . URL https://doi.org/10.1145/2342509.2342513.

[4] Intelligent Edge – Future of Cloud Computing | Microsoft Azure. Retrieved from https://azure.microsoft.com/en-us/overview/future-of-cloud/ (Accessed on 01/28/2020).

[5] AWS IoT Greengrass - Amazon Web Services. Retrieved from https://aws.amazon.com/greengrass/ (Accessed on 01/28/2020).

[6] Satoshi Nakamoto. bitcoin.pdf. https://bitcoin.org/bitcoin.pdf. (Accessed on 01/28/2020).

[7] Ethereum project. https://www.ethereum.org/ (Accessed on 01/28/2020).

[8] Hyperledger – Open Source Blockchain Technologies. Retrieved from https://www.hyperledger.org (Accessed on 01/28/2020).

[9] Parity Substrate: Build Your Own Blockchain | Parity Technologies. Retrieved from https://www.parity.io/substrate/ (Accessed on 01/28/2020).

[10] IPFS Powers the Distributed Web. Retrieved from https://ipfs.io/ (Accessed on 01/28/2020).

[11] 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. ISSN 1536-1268.

[12] Nicholas Carlini, Chang Liu, Ulfar Erlingsson, Jernej ´ Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. 2018. URL http://arxiv.org/abs/1802.08232.

[13] Intel® Software Guard Extensions | Intel® Software. Retrieved from https://software.intel.com/en-us/sgx (Accessed on 01/28/2020).

[14] TrustZone – Arm Developer. Retrieved from https://developer.arm.com/ip-products/security-ip/trustzone/ (Accessed on 01/28/2020).

[15] David Kaplan, Jeremy Powell, and Tom Woller. Amd memory encryption. 2016.

[16] Victor Costan and Srinivas Devadas. Intel SGX Explained. *IACR Cryptology ePrint Archive*, 2016:86, 2016

[17] Wenting Zheng, Ankur Dave, Jethro G Beekman, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *14th* USENIX *Symposium on Networked Systems Design and Implementation NSDI 17)*, pages 283–298, 2017.

[18] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N Asokan. The circle game: Scalable private membership test using trusted hardware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 31–44, 2017.

[19] Felix Schuster, Manuel Costa, C´edric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. VC3: Trustworthy data analytics in the cloud using sgx. In *2015 IEEE Symposium on Security and Privacy*, pages 38–54. IEEE, 2015.

[20] Olga Ohrimenko, Felix Schuster, C´edric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 619–636, 2016.

[21] Fan Zhang, Ittay Eyal, Robert Escriva, Ari Juels, and Robbert Van Renesse. {REM}: Resource-efficient mining for blockchains. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1427–1444, 2017.

[22] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*, page 270–282, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. URL https://doi.org/10.1145/ 2976749.2978326.

[23] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, Mark Stillwell, David Goltzsche, David M. Eyers, Rudiger Kapitza, Peter R. Pietzuch, and Christof Fetzer. Scone: Secure linux containers with Intel SGX. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI* 16), pages 689–703, 2016.

[24] Andrew Baumann, Marcus Peinado, and Galen Hunt. Shielding applications from an untrusted cloud with HAVEN. *ACM Trans. Comput. Syst*., 33(3), August 2015. ISSN 0734-2071. URL https://doi.org/10.1145/ 2799647.

[25] Chia-Che Tsai, Donald E. Porter, and Mona Vij. Graphene-SGX: A practical library OS for unmodified applications on SGX. In *Proceedings of the 2017 USENIX* Conference *on USENIX Annual Technical Conference, USENIX ATC '17*, page 645–658, USA, 2017. USENIX Association. ISBN 9781931971386.

[26] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. Ryoan: A distributed sandbox for untrusted computation on secret data. *ACM Trans. Comput. Syst*., 35(4), December 2018. ISSN 0734-2071. URL https://doi.org/10.1145/3231594.

[27] MesaTEE: A Framework for Universal Secure Computing. Retrieved from https://mesatee.org (Accessed on 01/28/2020).

[28] Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel software guard extensions: EPID provisioning and attestation services. *White Paper*, 1:1–10, 2016.

[29] intel-sgx-product-brief-2019.pdf. Retrieved from https://int5-software.intel.com/sites/default/files/managed/c3/8b/intel-sgx-product-brief-2019.pdf (Accessed on 01/28/2020).

[30] Payment channels - Bitcoin Wiki. Retrieved from https://en.bitcoin.it/wiki/Payment_channels#Spillman-style_payment_channels (Accessed on 01/28/2020).

[31] Open Enclave SDK. Retrieved from https://openenclave.io/sdk/ (Accessed on 01/28/2020).

[32] Home · enarx/enarx Wiki. Retrieved from https://github.com/enarx/enarx/wiki (Accessed on 01/28/2020).

[33] Pankaj Mendki. Docker container based analytics at iot edge video analytics usecase. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4. IEEE, 2018.

[34] S´ebastien Vaucher, Rafael Pires, Pascal Felber, Marcelo Pasin, Valerio Schiavoni, and Christof Fetzer. Sgxaware container orchestration for heterogeneous clusters. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 730– 741. IEEE, 2018.

[35] Bringing confidential computing to Kubernetes | Azure Blog and Updates | Microsoft Azure. Retrieved from https://azure.microsoft.com/en-us/blog/bringing-confidential-computing-to-kubernetes/ (Accessed on 01/28/2020).

[36] Dave Tian, Joseph I Choi, Grant Hernandez, Patrick Traynor, and Kevin RB Butler. A practical intel sgx setting for linux containers in the cloud. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, pages 255–266, 2019.