Pushing Edge Computing one Step Further: Resilient and Privacy-Preserving Processing on Personal Devices

Ludovic Javet
Inria Saclay
U. Versailles-St-Q.-en-Yvelines
U. Paris-Saclay
ludovic.javet@inria.fr

Nicolas Anciaux
Inria Saclay
U. Versailles-St-Q.-en-Yvelines
U. Paris-Saclay
nicolas.anciaux@inria.fr

Luc Bouganim
Inria Saclay
U. Versailles-St-Q.-en-Yvelines
U. Paris-Saclay
luc.bouganim@inria.fr

Léo Lamoureux
Inria Saclay
U. Versailles-St-Q.-en-Yvelines
U. Paris-Saclay
leo.salagnac-lamoureux@inria.fr

Philippe Pucheral
Inria Saclay
U. Versailles-St-Q.-en-Yvelines
U. Paris-Saclay
philippe.pucheral@inria.fr

ABSTRACT

Can we push Edge computing one step further? This demonstration paper proposes an answer to this question by leveraging the generalization of Trusted Execution Environments at the very edge of the network to enable resilient and privacy-preserving computation on personal devices. Based on preliminary published results, we show that this can drastically change the way distributed processing over personal data is conceived and achieved. The platform presented here demonstrates the pertinence of the approach through execution scenarios integrating heterogeneous secure personal devices.

1 INTRODUCTION

With forecasts at 29.3 billion connected devices by 2023 [1], the current reliable, server-based, and infrastructure-centric data management approach exhibits its limits in terms of efficiency, privacy, and energy consumption. While Edge computing aims at letting the data close to the sources and leverage in situ computing resources [22], data is usually gathered again on cloud or cloudlet servers [19] whenever multi-sources computation is required (e.g., to perform statistics or machine learning).

A game changer is the generalization of Trusted Execution Environments (TEEs) [18] at the extreme edge of the network: Intel SGX [10] on PC and tablets, ARM's TrustZone [17] on smartphones (Figure 1.a) and Trusted Platform Module (TPM) on smart objects (Figure 1.b). The security features of TEEs enable generic and scalable computations, processing cleartext data without any tradeoff between privacy and accuracy. Hence, large-scale privacy-preserving computations are within reach in a more generic and flexible way than with homomorphic cryptography [8], differential privacy [12], or secure multiparty computation protocols [6].

In [14], we advocate that the convergence between TEEs and Opportunistic Networks (OppNets) [16] can be leveraged to organize fully decentralized and secure query computations among data scattered on multiple personal devices, without resorting to any central authority. We call this new data management paradigm *Edgelet computing* and propose dedicated execution strategies that guarantee three fundamental properties, namely *Resiliency* – a query completes before a given deadline according to a given fault presumption rate –, *Validity* – the query result is equivalent to the one obtained in a centralized context – and *Crowd Liability* – the liability of the processing is equally distributed among all query participants –. The shift of responsibility from a unique data controller (in the GDPR sense) to the crowd is a salient feature of Edgelet computing. These properties provide tangible guarantees to both *Data Contributors* (individuals supplying the data) and *Data Processors* (people whose personal device performs the processing), but also to the beneficiaries of the query results.

This research work is being validated in the field through the DomYcile project [2]: 8,000 elderly people receiving home care in the French Yvelines district are equipped with a secure box (see Figure 1.b) where their medical records are stored and processed (local queries today, clustering algorithms crossing the data of several patients tomorrow); currently, the boxes are not connected to the Internet for subscription cost and acceptability reasons, but are connected opportunistically by caregivers during their visits.

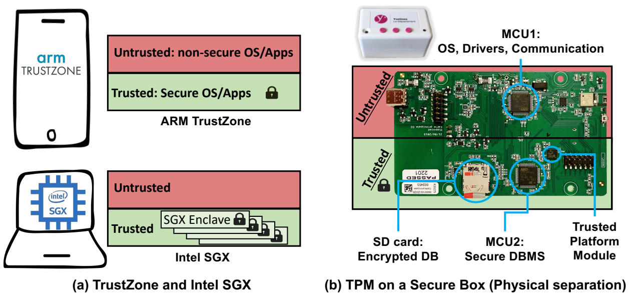


Figure 1: Instances of Trusted Execution Environments.

Today, we aim at enlarging the use cases targeted by Edgelet computing, considering that the solutions proposed in [14] apply whenever decentralized computations satisfying Resiliency, Validity, and Crowd Liability need to be performed among TEE-enabled personal devices (edgelets) connected through "uncertain" communications. OppNets are an extreme case of uncertain communications in terms of latency and then fault presumption, but uncertainty is inherent in any decentralized computation over a crowd of personal devices, because devices can be disconnected at will, be temporarily out of reach, or simply fail. We sketch below two promising use cases as motivating examples:

Opportunistic polling: during events that welcome a large audience (e.g., a conference, a museum, a concert, a match, a protest), the participants using a TrustZone-based smartphone could contribute with their data (e.g., their centers of interest, nationality, age, etc.) to a global processing to improve the user experience in real time (i.e., adapting the services to the characteristics of the audience).

Data altruism: the Data Governance Act [3] fosters data subjects to give consent to process their personal data for purposes such as scientific research or public services improvement (e.g., a health survey organized by Santé Publique France). The Edgelet framework allows individuals to provide their health data with strong guarantees of confidentiality, through a secure box, a smartphone, or a PC, and to benefit from the results obtained.

The goal of this demo paper is twofold. First, it exemplifies the versatility of the approach by demonstrating the Edgelet computing mechanisms running on different TEE-enabled devices (from PC with SGX up to smart objects with TPM). Second, it presents the internals of Edgelet computing and let the audience play with important parameters related to resiliency and data privacy and observe the outcome by themselves. A video of the demonstration is available online[1].

2 EDGELET COMPUTING FRAMEWORK

A lot of research work has been produced on the different areas encompassed by Edgelet computing: *Computing architectures*: from cloud, fog, edge computing to mobile edge [13], mobile ad-hoc cloud computing [25] and cross-device federated machine learning [7]; *Private/Confidential computing*: using homomorphic encryption [8], secure multiparty computation [6], differential privacy techniques [12] or TEEs (in the cloud [20], or at the edges [9, 15]). To the best of our knowledge, none of the aforementioned papers and surveys address the challenge raised by Edgelet computing: *how to perform complex and scalable distributed queries on data scattered on multiple personal devices while ensuring fault and disconnection tolerance?*

In this section, we first recall from [14] the basics of Edgelet execution strategies that can scale to a large number of participants while providing privacy guarantees. Then, we present a resilient execution strategy that handles device failures as well as temporary disconnections, and produces accurate results.

2.1 Design of Edgelet Computation Plans

Contrary to participatory sensing or sensor networks which focus on stream queries over elementary data, Edgelet computing aims at handling rich data (e.g., healthcare folders, spending habits) and complex processing (e.g., machine learning, data mining) over edgelets data seen as a horizontal partitioning of a shared database (i.e., edgelets data conform to a common schema). We represent these computations in the form of a Query Execution Plan (QEP), that is a directed graph where vertices materialize the operators to be computed and edges represent the dataflow among them. A secure assignment of these operators is then essential to avoid any targeted attacks, but is not detailed here for the sake of brevity.

The simplest form of a QEP is a tree with *Data Contributors* at the leaves (one per edgelet contributing to the query with its data) sending the requested data to a *Snapshot Builder* operator, the

[1]https://project.inria.fr/edgeletdemo

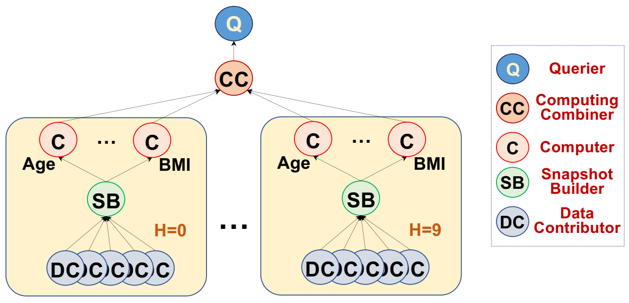


Figure 2: Vertically and Horizontally partitioned QEP.

role of which is to collect a representative snapshot of cardinality C (e.g., select tuples from $C = 2000$ patients over 65 years old). The Snapshot Builder then sends the representative snapshot to a *Computer* operator which, in turn, computes the final result and eventually delivers it to the *Querier*.

The Data Processors (here, the Snapshot Builder and the Computer) can be decomposed into sub-operators assigned to different edgelets (i.e., TEE-enabled devices) for performance or privacy concerns. Indeed, although highly difficult to conduct and requiring physical instrumentation, side-channel attacks compromising data confidentiality cannot be totally ignored [24], placing the TEE in a "sealed glass" mode [23] (the integrity guarantee is preserved, but not the confidentiality). This decomposition can help minimizing the amount of data exposed at each edgelet by horizontally partitioning the dataset. This can also preclude the concomitant exposure, in the same edgelet, of data items that become sensitive when combined (e.g., a quasi-identifier) by vertically partitioning the dataset. This can finally help minimizing the workload (e.g., when energy consumption matters) or exploit the inherent Edgelet computing parallelism. A *Computing Combiner* operator must then be added in the QEP to combine the outputs of all sub-operators.

Figure 2 presents the QEP for an example query using horizontal partitioning (Data Contributors are assigned to Snapshot Builders, e.g., by hashing their public key) and vertical partitioning (each Computer manages a single statistic, e.g., Age, BMI).

2.2 Edgelet Query Execution

During the execution of a QEP, each edgelet acting as a Data Processor is a potential Single Point of Failure, either because it really fails or because of a temporary disconnection.

To solve this problem, we explored in [14] two resiliency strategies, namely *Backup* and *Overcollection*, which both enable the execution of a query while respecting the resiliency, validity and crowd liability properties introduced earlier. These two strategies differ on complexity, performance and on the way privacy is enforced, making each strategy best adapted to different contexts (see [14] for a complete taxonomy). In this demonstration, we focus on the Overcollection strategy which is best adapted to any use case where performance matters and approximate results are acceptable (e.g., statistics, machine learning processes).

We illustrate the Overcollection principle by considering a query over a sample dataset (e.g., C individuals with age > 65) where Data Processors (i.e., Snapshot Builder and Computers) execute distributive operators. Instead of executing the operators on single edgelets, we distribute (using hashing) its execution over $n + m$ edgelets where each one processes a partition of the original dataset, with n the minimum number of partitions to be collected and m the overcollection parameter (see Figure 3).

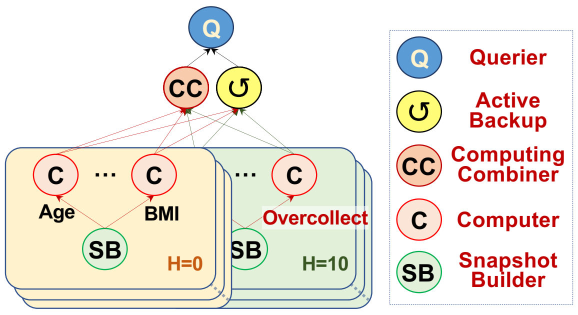


Figure 3: Overcollection for the QEP of Figure 2.

The validity is then trivially preserved as long as (1) each of the $n + m$ partitions is representative and has a cardinality C/n, and (2) less than m partitions are lost. Note that we need to add to the Computing Combiner an *Active Backup*, i.e., a replicated instance running in parallel with it, in order to handle its potential failure.

Many data-intensive queries of interest are distributive by nature and can accommodate Overcollection. This is also the case of general interest Machine Learning (ML) algorithms. However, ML algorithms are often iterative and require to exchange partial results computed over different data partitions. A strict validity between the different iterations would require that a valid snapshot is processed at each iteration, and that the same valid snapshot (same partitions) is kept during the subsequent iterations, which is not easily ensured due to various message loss at each iteration. Fortunately, strict validity is not a prerequisite for these algorithms, and resampling at each iteration sometimes even produces better accuracy (as in Mini-batch K-Means [21]).

The method for executing an iterative algorithm $\mathcal{A}$ (e.g., K-Means [11]) in the Edgelet context is the following: each edgelet implementing a Computer iterates on (1) a local convergence phase where it computes $\mathcal{A}$ on its local partition, improving its local knowledge (e.g., its centroids), and broadcasts this knowledge to all others Computers, and (2) a synchronization phase where it receives the knowledge of the other Computers it has heard of and integrates them in its own knowledge (e.g., the barycenter for each centroid). Right before the query deadline, the knowledge is sent to the Computing Combiner which combines all received knowledges and sends the final result to the Querier. We enforce the progression of the algorithm on all edgelets thanks to a *Heartbeat*, that is each iteration is cadenced by a clock, whatever the local state of the processing, i.e., the Computers move to the next iteration even if few or no messages were received (failures or disconnections).

3 DEMONSTRATION

3.1 Demonstration Platform

The first objective of this demonstration is to present the computational mechanisms of the Edgelet framework using TEE-enabled devices ranging from high-end device (PC) to low-end device (home box). Therefore, we deploy the following hardware:

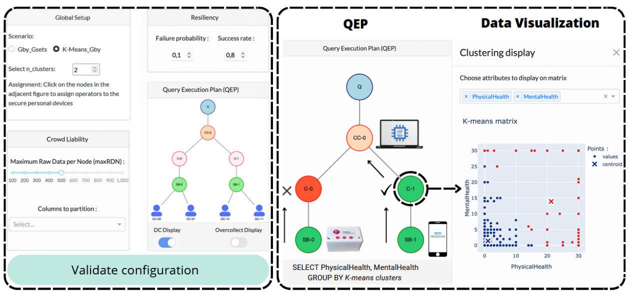
- **PC (Intel SGX)**. A laptop with an Intel Core i5-9400H 2.5GHz 4 Cores with SGX 1-FLC runs secure enclaves in an Ubuntu Linux 20.04 environment. The code is written on top of Open Enclave [5], an SDK for developing enclave applications in C/C++. Open Enclave provides support for Intel SGX as well as preview support for ARM TrustZone, thus aiming to generalize the development of enclave applications across TEEs.
- **Home boxes (TPM)**. These secure boxes (see Figure 1.b, identical to those deployed in the field by the DomYcile project [2], incorporate a STM32F417 microcontroller. The latter is connected to a μ-SD card hosting the owner's raw data and to a tamper-proof Trusted Platform Module (TPM). This TPM secures the cryptographic keys and guarantees, during the secure boot, that the embedded code has not been tampered with.

The second objective is to present the internal aspects of the solution, we thus develop the following software: (1) a GUI implemented in Dash Python that allows interactive configuration and visualization of Edgelet queries; (2) an Edgelet manager that orchestrates executions and communications between simulated and real edgelets (PC and home boxes); (3) a web client accessible to attendees' smartphones via a QR code allowing them to visualize the processed data and interact in real-time with the execution.

3.2 Demonstration Scenario

In this demonstration, we take the motivating example presented in the introduction. Santé Publique France (Querier) wants to perform a set of queries on population health data to improve the quality of its services. Some individuals are equipped with a PC, others with a secure home box, but all are interconnected by uncertain communications. The scenario consists of two interactive parts related to the configuration and execution of Edgelet computations. In the first part, the attendees will understand the impact of privacy and resiliency parameters on the QEPs. In the second part, they will follow the execution in real time and visualize the results obtained. These two parts are represented on Figure 4.

Part 1: QEP Configuration. The attendees are first invited to select a query among the 2 proposed ones: (i) a Grouping Sets query [4] which allows multiple Group-By clauses to be evaluated within a single SQL query (e.g., to cross multiple statistics over the same data sample); or (ii) a K-Means [11] followed by a Group By on the resulting clusters (e.g., to identify which characteristics most influence the dependency level of an elderly person). Then, following their intuition, the attendees can try to improve the privacy of the QEP of the selected query to reduce data exposure in case of TEEs compromise. To do so, they can adjust the horizontal and vertical partitioning parameters presented in Section 2.1, by specifying the maximum number of raw data per edgelet and selecting the attribute pairs to be separated. Finally, the attendees can vary the failure probability value of the scenario and observe automatic changes in the execution plan to keep it resilient.



Scenario: Gby_Gsets [✓ K-Means_Gby]

Select n_clusters: 2

Assignment: Click on the nodes in the adjacent figure to assign operators to the secure personal devices

Failure probability : 0,1

Success rate : 0,8

Maximum Raw Data per Node (maxRDN) : 500

Columns to partition : Select...

Choose attributes to display on matrix: PhysicalHealth, MentalHealth

SELECT PhysicalHealth, MentalHealth
GROUP BY *K-means clusters*

Part 1: Configuration of a QEP **Part 2: Execution of an Edgelet Computation**

Figure 4: Edgelet computing demonstration platform.

Part 2: Execution of an Edgelet computation. After the attendees have configured the query as desired, we proceed to its real-time execution on the heterogeneous personal devices available for the demonstration (concrete edgelets), the rest of the operators being associated to a configurable number of simulated edgelets to attest scalability. The demonstration platform allows the attendees to visualize, step by step, the query execution. Initially, we launch the collection phase where Snapshot Builders receive contributions from thousands of simulated Data Contributors and build representative snapshots. Next, the Edgelet platform redistributes the data and launches the computation phase with the corresponding Computers. At each step, the user can interact with the execution using their own smartphone to analyze the input and output data. At the end, the aggregated data is transmitted to the Computing Combiners for the combination phase and the query is completed. In case of failures or disconnections, the attendees will be able to directly identify the concerned edgelets on the QEP and understand the impacts on the execution. For example, we can intentionally power off some concrete devices to generate a failure at will. In order to verify the results, the attendees can take the same dataset used with the distributed edgelets and run the processing centrally on the demonstration platform.

3.3 Demonstration Results

The demonstration shows the internals of the Edgelet computing framework applied to the fully decentralized context and illustrates its usage through execution scenarios on multiple personal devices. It helps to answer the following questions:

Does Edgelet computing concretely make sense? The practical implementation on high-end and low-end personal devices demonstrates both its applicability and versatility. It shows that large-scale general-purpose computations can be performed over devices while providing high-security guarantees. This opens up important opportunities in terms of personal data management.

Can any form of computation be handled? Edgelet computing leverages the TEE security to perform computations on clear-text data (once decrypted locally). It can then combine computation generality – demonstrated by our demonstration queries – and scalability – demonstrated by the number of simulated edgelets –, contrary to homomorphic encryption, secure multiparty computation, or local differential privacy solutions. Note that the Overcollection strategy only applies if the processing is distributive; otherwise, the Backup strategy can be used at the price of a higher complexity and lower performance (see [14] for details).

Is privacy protected whatever the attack? While highly difficult to implement, side-channel attacks on TEEs could compromise the confidentiality of the manipulated data. Edgelet computing counter-measures are horizontal and vertical partitioning. Through the demonstration, attendees will be able to visualize the distribution of data among the edgelets and measure the respective benefit of both types of partitioning. They will also understand that only the results of the computations, i.e. the aggregated data, are sent (encrypted) to the successor operators.

Can a query always proceed despite the failures? Providing fault tolerance in a distributed context where messages are sent among weakly connected personal devices is a real challenge, either because they are down or because they are temporarily unavailable (e.g., offline smartphones). The demonstration shows that the Overcollection strategy can answer this: attendees will be allowed to vary the failure context (e.g., disconnection probability) and see the impact on the overcollection degree as well as the effects on the results accuracy with respect to the number of heartbeats.

REFERENCES

[1] 2020. Cisco Annual Internet Report (2018–2023) White Paper. https://tinyurl.com/cisco-internet-report
[2] 2020. DomYcile Project. https://tinyurl.com/domycile, Salon E-Tonomy: https://tinyurl.com/e-tonomy.
[3] 2020. Proposal for a Regulation of the European Parliament and of the Concil on European data governance (Data Governance Act). https://tinyurl.com/data-governance-act
[4] 2023. Group By Grouping Sets — Snowflake Documentation. https://tinyurl.com/grouping-sets
[5] 2023. Open Enclave SDK. https://openenclave.io/sdk/
[6] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel N. Kho, and Jennie Rogers. 2017. SMCQL: Secure Query Processing for Private Data Networks. *Proc. VLDB Endow.* 10, 6 (2017), 673–684.
[7] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *MLSys 2019, Stanford, CA, USA.*
[8] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J. Wu. 2013. Private Database Queries Using Somewhat Homomorphic Encryption. In *ACNS 2013, Banff, AB, Canada*, Vol. 7954. 102–118.
[9] Luc Bouganim, Julien Loudet, and Iulian Sandu Popa. 2022. Highly distributed and privacy-preserving queries on personal data management systems. *The VLDB Journal* (2022), 1–31.
[10] Robin Carpentier, Floris Thiant, Iulian Sandu Popa, Nicolas Anciaux, and Luc Bouganim. 2022. An Extensive and Secure Personal Data Management System Using SGX. In *EDBT 2022, Edinburgh, UK.* 570–573.
[11] Inderjit S. Dhillon and Dharmendra S. Modha. 1999. A Data-Clustering Algorithm on Distributed Memory Multiprocessors. In *SIGKDD 1999, San Diego, CA, USA*, Vol. 1759. 245–260.
[12] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy*, Vol. 4052. 1–12.
[13] Ana Juan Ferrer, Joan Manuel Marquès, and Josep Jorba. 2019. Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad hoc, and Edge Computing. *ACM Comput. Surv.* 51, 6 (2019), 111:1–111:36.
[14] Ludovic Javet, Nicolas Anciaux, Luc Bouganim, and Philippe Pucheral. 2022. Edgelet Computing: Pushing Query Processing and Liability at the Extreme Edge of the Network. In *CCGrid 2022, Taormina, Italy.* 160–169.
[15] Riad Ladjel, Nicolas Anciaux, Philippe Pucheral, and Guillaume Scerri. 2019. Trustworthy Distributed Computations on Personal Data Using Trusted Execution Environments. In *TrustCom 2019.* Rotorua, New Zealand, 381–388.
[16] Luciana Pelusi, Andrea Passarella, and Marco Conti. 2006. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine* 44, 11 (2006), 134–141.
[17] Sandro Pinto and Nuno Santos. 2019. Demystifying Arm TrustZone: A Comprehensive Survey. *ACM Comput. Surv.* 51, 6 (2019), 130:1–130:36.
[18] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted Execution Environment: What It is, and What It is Not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. 57–64.
[19] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, and Nigel Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* 8, 4 (2009), 14–23.
[20] Felix Schuster, Manuel Costa, Cedric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy Data Analytics in the Cloud Using SGX. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA.* 38–54.
[21] D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010.* Raleigh, North Carolina, USA, 1177–1178.
[22] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.
[23] Florian Tramer, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. 2017. Sealed-Glass Proofs: Using Transparent Enclaves to Prove and Sell Knowledge. In *EuroS&P 2017, Paris, France.* 19–34.
[24] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A. Gunter. 2017. Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX. In *CCS 2017, Dallas, TX, USA.* 2421–2434.
[25] Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran Razzak, and Sghaier Guizani. 2016. Mobile ad hoc cloud: A survey. *Wirel. Commun. Mob. Comput.* 16, 16 (2016), 2572–2589.