Data Security in Multiparty Edge Computing Environments

Alireza Jolfaei
School of Science, Engineering and Information Technology
Federation University Australia
Mt Helen, Victoria 3350, Australia
Email: a.jolfaei@federation.edu.au

Krishna Kant
Department of Computer and Information Sciences
Temple University
Philadelphia, Pennsylvania 19122, USA
Email: kkant@temple.edu

***Abstract*—The paper considers data security issues in edge computing scenarios that involve data streams coming out from individual devices to edge controllers (ECs). We assume multiple edge controllers each possibly belonging to a different party and controlling some devices. In this environment, there are issues in regards to the scalability of key management and computation limitations at the edge of the network. To address these issues, we suggest the formation of groups in the device layer, where a group leader is assigned to communicate with group devices and the EC. We propose a lightweight permutation mechanism for preserving the confidentiality of sensory data.**

***Keywords*—Edge Computing; Data Security; IoT; Multitenancy; Scalability; Access Control**

I. Introduction

Edge computing is a rapidly developing field [1] with numerous real-time analytics applications in supporting smart cyber-physical systems, such as traffic management, surveillance, patient care, smart manufacturing, and food quality monitoring [2]. As shown in Fig. 1, the typical model of the edge computing consists of a 3-layer architecture as follows:

(a) Bottom layer: IoT devices such as cameras and other sensors that continuously stream data for the monitored area.
(b) Middle layer: A set of edge controllers(ECs), each of which receives data streams from the devices in its vicinity and is responsible for temporary data storage, and real-time data analytics (possibly in collaboration with other ECs), to derive intelligence.
(c) Top-layer: a cloud that provides longer-term data storage and deep analytics.

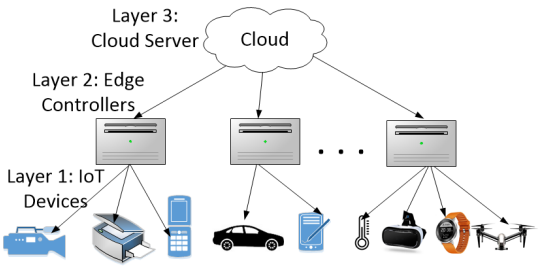

Fig. 1. Hierarchical architecture of edge computing.

A large cyber-physical system requires multiple ECs, each managing many IoT devices and interacting with other ECs for collaborative data access and analytics. There are several important applications of such an infrastructure, of which three are explained in some detail in section II. Two prominent examples of such an infrastructure are the intelligent transportation system (ITS) where a large number of cameras and other sensors provide a real-time view of the traffic in a city, and intelligent health care (IHC) where a large facility provides care to a large number of patients or seniors. Each EC generally covers a single physical region and the devices in that region will generally associate with that EC. However, a more dynamic association is also possible. Mobile devices generally use limited range wireless communications and are thus likely to associate with different ECs as they move.

Multi-tenancy is likely to be an essential aspect of this infrastructure, with various deployment and hence ownership arrangements. One such arrangement is where a vendor deploys the edge computing infrastructure in an entire area such as a city neighborhood for ITS or a hospital ward (or even the entire hospital in a large network of hospitals). In this case, both the devices and the EC will belong to the same party. The devices would likely be bound to their EC in this case and communicate freely – although protection against external intruders or compromised devices is still an issue. Another likely ownership scenario is where the vendors play at different layers. For example, the devices may be installed and managed by one party, whereas the ECs are installed and managed by another. (Of course, multiple vendors may be involved at each layer, e.g., multiple clumps of devices installed by different parties, and multiple groups of ECs installed by different parties).

Regardless of the ownership model, it is necessary for ECs to collaborate among themselves and obtain relevant data from the devices to provide the above services. Thus access control across party lines becomes an important aspect of the system and raises many questions concerning how it is handled. In this paper, we discuss a mechanism to provide access control in this environment. Also, there are both security and privacy concerns that must be addressed to ensure a robust system. While a comprehensive solution to these issues is quite challenging, this paper presents a lightweight mechanism for even the low capability devices to support real-time encryption of the data stream and protect their privacy.

The remainder of this paper is organized as follows. Section II discusses three prominent examples of the edge infrastructure to highlight the issues. Section III discusses the need for distributed caching of data and access control in multiparty environments. Section IV discusses the security and privacy issues and provides a preliminary solution to the problem. Section V describes the related works. Finally, Section VI concludes the paper.

II. Examples of Edge Computing Scenarios

In the following, we describe three prominent examples of the edge computing system. In each case, we can assume that the devices generate a data stream to which the ECs subscribe either statically or dynamically. The subscription depends on the specified access rights of the ECs and the need which is driven by the higher level services that the ECs are trying to provide. The services may cover a wide range both in importance and response time needs, such as safety, anomaly detection, and performance improvement.

Intelligent Transportation System (ITS): Fig. 2 shows an ITS communication model that includes a group of vehicles driving on the road and a group of road-side units (RSUs) to which the vehicles are subscribed. The RSUs here play the roles of ECs. A traffic management system (a cloud server) collects data from vehicles across transport regions such as an area of the city. In this application, we have both fixed and mobile IoT devices. The fixed devices are cameras installed on the roadside for monitoring the overall traffic, whereas the mobile devices are installed in the car. The fixed devices may be subscribed permanently to the ECs in that area whereas the mobile devices are subscribed to the closest EC c.

In every time step t, each vehicle d collects some data $\boldsymbol{s_t}$ from its status and neighborhood environment and sends the data stream to the EC through a wireless connection. The communications are typically based on the IEEE 1609 [3] and IEEE 802.11 standards [4], specifically designed for vehicular communications. The sensed features could include two types of information: (a) vehicle's parameters such as speed, direction, and location, and (b) the environment sensed by the vehicles, such as specific features recognized in the video stream produced by the vehicular cameras.

We assume that the ECs are not necessarily tied to the base station with which the device establishes a direct wireless connection, and instead can be reached by any device through the available wireless/wired infrastructure. The ECs may need to coordinate to monitor certain features. Specific applications include tracking of suspicious or abnormally behaving vehicles, and routine tracking of vehicles belonging to an organization (for example, city buses, ambulances, delivery vehicles, and refuse collection trucks). The non-tracking applications may include monitoring congestion, safety parameters, and alerting vehicles. The privacy issues in this environment include protection of detailed routes and times, and security concerns are important both for safety and performance (for example, congestion caused by malicious information).

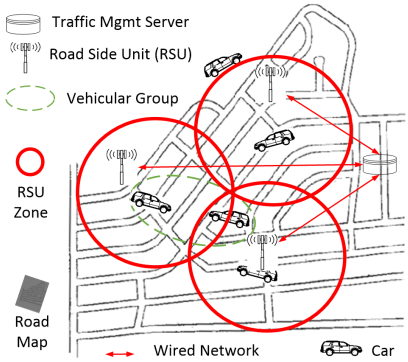

Fig. 2. ITS communication model.

Intelligent Health Care (IHC): In smart healthcare applications, doctors need to monitor the health and well-being of the patients either within a single health care facility or across a region of interest. The area may be sufficiently big to require multiple ECs with which the individual devices associate. Each patient may be monitored by multiple devices, some of them worn by the patients, and thus these devices physically move with the patient. These include monitors like blood pressure, pulse, and heart rate. Other devices such as cameras may be mounted in the patient rooms, and thus monitor different patients at different times. At each time instance t, an IoT device d senses the patient data $\boldsymbol{s_t}$, and the corresponding data and feature streams are sent to an EC c that aggregates the to determine health care intervention needs such as a patient who may have fallen, become incapacitated, or otherwise shows some abnormalities in the parameters. The ECs need to collaborate to track the patients. The privacy and access control issues are inherent in this environment; the security issues may arise due to potentially compromised devices.

Intelligent Robotic Manufacturing (IRM): A smart manufacturing environment consists of several major assembly stages, somewhat similar to the traditional assembly lines, but with two significant differences. First, the stages are not necessarily connected sequentially in the form of an assembly line but could form a graphs, potentially with cycles and waiting queues. Second, each step itself requires a coordination of multiple robots to execute the step. It is reasonable to assume that each step is supervised by an EC, which receives data streams both from individual robots and from cameras in the area watching the robots. The coordination among robots requires real-time analytics by the local EC, and the routing of the part to the appropriate stage requires coordination among them. In this example, there are no privacy or even access control issues; however, it is still important to protect the entire system from compromised robots and compromised information exchange.

III. Multiparty Access Control

The key characteristic of the edge computing environment is the real-time data streams that must be handled by ECs for collaborative intelligence extraction. We assume that the data stream produced by each device, henceforth named Device Data Stream (DDS), is associated with a secondary stream that we name Device Feature Stream (DFS). The stream

represents the "features" of the DDS and can be regarded as the application level metadata for the DDS. The DFS may be generated by the device itself or by the EC receiving the data stream via standard feature detection/extraction techniques. For simplicity, we assume that the DFS is generated by the device itself in all cases. The sophistication of feature detection could vary in general, but we will assume a basic uniformly applicable capability across all streams. In this paper, we are not concerned with the feature detection details and simply assume that each frame (or a small sequence of frames) results in an initial feature vector that is further processed for certain pre-defined feature extraction/selection (for example, faces, cars, and road in the street monitoring scenario). We also assume that the two streams are properly cross-indexed so that one could easily identify the features for data frames and vice versa.

We assume that both the DDS and the corresponding DFS are divided into some suitable chunk sizes, which typically will contain one or more contiguous frames from the video stream. Such chunking not only simplifies the storage management but is also crucial for access control and encryption, since in this environment both the security and access control are likely to be context dependent and time-dependent. For example, the access may be granted to the part of the stream (that is, a consecutive set of chunks) if the stream contains interesting features that must be correlated with features in other parties video streams.

In order to specify the access rules more precisely, suppose that we have numbered all the streams as $s_1, s_2, ...$, each with a feature set $f_{mi}, i = 1, 2, ...$ Also suppose that we have a function $\eta(m)$, which is the index of the EC that is currently subscribed to this stream. Then an access rule can be specified in the following form: $B(T, e_1, ..., e_m) \rightarrow (EC_{\eta(i)}, s_i, d)$, which says that if the Boolean expression B holds at time instant t, $EC_{(\eta(i)}$ gets access to the ith stream until time $t + d$. Here the Boolean expression is over the occurrence of an external trigger T and individual events $e_1, ..., e_M$ for some $M \geq 1$. Here an event $e_m, m \in M$ is defined as the triplet (t_m, s_m, f_m), meaning that some stream s_m was found to report some feature f_m at time $t-t_m$ (that is, t_m seconds in the past from current time t). (Per our convention, the stream s_m would be currently subscribed to $EC_{\eta(s_m)}$). The intent is to allow $EC_{\eta(i)}$ to access stream s_i for a time period d if (a) an external trigger is received (such as a request from authorized persons to trace an object or person or a query from the clients of the system), and/or (b) some "interesting" events are observed in the (recent) past relative to some streams. This specification is very general and allows arbitrary times in the past and arbitrary features from an arbitrary number of streams. In practice, however, the rules are likely to be much simpler and involve a few events in the recent past. For example, if an EC wants to track a car, it enables collaborative access to a few ECs depending on the position of the car, locations of other ECs, and possible prediction of the car path. Nevertheless, any solutions offered must be general.

Given a large number of rules with overlapping time-periods and common features, it is important to determine efficiently whether a query is allowed and if so, how to perform it optimally. In particular, to evaluate the net effect of multiple rules on a query, we need to consider the following three aspects:

1) Temporal overlaps: If some ECs have access to stream s_1 for the period $(t_1, t_1 + d_1)$, and to s_2 for the period $(t_2, t_2 + d_2)$, the joint access is available only during the intersection of these two periods. While this is easy to compute, we need to track what ECs have access to which streams during a given time interval.
2) Chain Reactions: Overlapping access given to EC_1 and EC_2 could, in turn, enable another access, for example, access to EC_3. Accordingly, overlapping accesses for EC_3 and EC_4 may grant an access to EC_5. The difficulty is that such implications can change as accesses are granted or revoked and must be evaluated constantly.
3) Growing Rule Base: If an EC is given access to the chunks of some stream during the time interval $(t, t+d)$, this access stays the time advances, and more chunks arrive. Thus, we need to process the rules continuously to determine if additional rules are enabled, and add those rules to the rule base. This can be handled relatively easily except if we allow changes in underlying rules themselves.

Here we propose a simple approach that assumes that the time is discretized at a sufficiently coarse grain both for chunking and access specification. Then, we can build a data structure that tracks all access right for each time slot. Then, as we move to successive time slots, most accesses can be simply carried over to the new slot, and any additional ones that are enabled updated. Such an approach is reasonable if the length of history to be maintained is limited; that is when the queries do not refer to any accesses beyond a certain point in the past. In our past research, we have explored multiparty access models in the context of databases, which can be exploited for more general situations where data from multiple parties is "composed" using specific composition functions, but this is beyond the scope of this paper [5].

IV. Security and Privacy Issues

There are several security concerns in this environment. Since at least some of the information (for example, driving behavior in ITS or health information in IHC) is sensitive, the information must be protected during transit between device and EC and between ECs. Given the real-time streaming of large amounts of data from cameras, standard encryption mechanisms, such as AES, may be difficult to implement at the device level. Thus, it is necessary to devise lightweight algorithms for encryption. A related issue is that of remote storage of DDS or DFS. For this, each chunk needs to be encrypted before storage at a remote node, and yet nodes that have access to this chunk should be able to access it. This again requires a lightweight encryption scheme. Also managing keys individually for each device could become a burden and a source of weakness. Notice that if each device can subscribe

to any EC, we need enough keys for each pairing of a device and EC. We also need a group keying mechanism so that the ECs that have access to the chunk can freely share it without repeated encryption/decryption. Each device and node also needs an authentication mechanism to avoid impersonation; however, since authentication is likely done very infrequently, traditional mechanisms may be adequate for it.

The data collected from various IoT devices are spatially and temporally correlated in nature. For example, individual drivers do not necessarily want to be identified when supplying their data since such data can be misused in numerous ways. For example, if a car provides its ID and speed, it can be easily identified whenever it violates the speed limit. Similarly, knowing the identity of the patient for all data transfers may be undesirable. The goal of IoT devices is to preserve their anonymity as much as possible.

The security and privacy concerns apply to both streams, and there are several ways to handle them together for encryption. A simple possibility is to encrypt both streams using the same key and in the same way (using the structured encryption described below) so that an EC can retrieve either both the feature and corresponding data, or neither. If separate keys are used, it is reasonable to share DFS key if DDS key is shared (since DFS can always be derived from DDS); however, DFS key may be shared without sharing the DDS key. In any case, with encryption, the DDS/DFS can be stored not only in the currently subscribed EC of the devices but also in any other EC.

In general, there is the question of a device being captured and replaced by a malicious device that appears normal. We do not address with this scenario since the only way to detect such a device is by its behavior that differs from that of an authentic device. There is also the possibility of an EC behaving maliciously and not respecting the access rules. ECs are expected to be deployed by reputable companies, and it is reasonable to expect they would not deliberately subvert access restrictions, or provide inadequate/incorrect data. Thus we assume that the ECs are Curious-But-Honest (CBH), and authentication/encryption is largely adequate to handle security issues.

A. A Potential Security Approach

In this section, we consider a specific situation where the devices can form groups and choose a leader that collects data from other devices and communicates with the ECs. The group formation is done both to reduce key management overhead and to obfuscate device identities. In the ITS context, the group leader is one of the cars for the mobile (car-based) devices and one of the fixed devices (e.g., a camera) among the devices installed in an area. In the IHC context, we could have a leader per group of contiguous rooms which covers both fixed and mobile (patient-based) devices.

There are a large variety of effective approaches for group formation as well as group leader selection [6], whereby different approaches typically focus on different performance metrics. For our application, and important performance metric is the size of the group – a group should be large enough to obfuscate device identities well but small enough from the latency and leader bandwidth perspective. The chosen leader should be a device that is less restricted in computing power, storage, energy, and transmission bandwidth. A deployment friendly to our scheme may even choose to deploy devices such that a few of them are powerful enough to serve as leaders, while others are chosen as less powerful as an offsetting mechanism.

We now discuss the obfuscation mechanism. Consider a set of n IoT devices that are subscribed to a specific leader. In any time slot t, each device i collects a data chunk, which we assume contains K_i consecutive data stream segments. It sends this data to the leader. After receiving data from all n devices, the leader prepares a "matrix" of data segments $\boldsymbol{D_t}$ as follows:

$$\boldsymbol{D_t} = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1K_1} \\ d_{21} & d_{22} & \dots & d_{2K_2} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nK_n} \end{bmatrix}, \tag{1}$$

where d_{ij} denotes the jth segment of the chunk sent by the $i$the device. Note that we allow each device to have a different number of data segments in a chunk; this is essential with heterogeneous devices if we want to have an identical chunk size in the system to facilitate storage management. This matrix is then encrypted and sent to the EC to which this leader is subscribed. This forms the DDS going into the EC.

A similar aggregation is required for the DFS. We assume that the device i extracts k_i features from the data chunk and send them to the leader. The leader prepares a feature "matrix" $\boldsymbol{F_t}$ after receiving features from all n devices as follows:

$$\boldsymbol{F_t} = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1k_i} \\ f_{21} & f_{22} & \dots & f_{2k_i} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \dots & f_{nk_n} \end{bmatrix}, \tag{2}$$

where f_{ij} denotes the j-th sensed feature of the data chunk sent by the ith device. Note that k_i need not be same as K_i. The leader encrypts the matrix using the same or different key (as discussed earlier) and sends it to the EC.

In [7], we have proposed a lightweight, permutation-only encryption scheme for use in transmitting smart-grid protection messages that are normally not encrypted due to high latency associated with normal encryption mechanisms. In this paper, we propose a simple variant of such an algorithm for encrypting the DDS/DFS matrices in Equations 1 and 2, which we henceforth call as messages.

To permute a given message, we first construct a sequence of pseudo-random numbers by concatenating three pseudo-random sequences generated from a linear congruential pseudo-random number generator, defined by the following recurrence relation:

$$z_{j+1} = (h \cdot z_j + q) \mod 2^{n_0}, \tag{3}$$

Algorithm 1 Permutation-only encryption

procedure PERMUTATION($\boldsymbol{F_t}$,KEY)
  for $j \leftarrow 0, N$ **do**
    $z'_{j+1} \leftarrow (h_1 \cdot z'_j + q_1) \mod 2^{n_0}$
    $z''_{j+1} \leftarrow (h_2 \cdot z''_j + q_2) \mod 2^{n_0}$
    $z'''_{j+1} \leftarrow (h_3 \cdot z'''_j + q_3) \mod 2^{n_0}$
    $z_{3j} \leftarrow z'_{j+1}, \quad z_{3j+1} \leftarrow z''_{j+1}, \quad z_{3j+2} \leftarrow z'''_{j+1}$
  end for
  $X \leftarrow sort\ \{z_w\}_{w=0}^{K-1}$
  For $j \leftarrow 0, K-1$ **do** $c(j) \leftarrow e(x_j)$
end procedure

where the seed z_0 is arbitrarily chosen, n_0 is the period of the generator, $h \mod 4 = 1$, and $q \mod 2 = 1$. Here the index j corresponds to the jth random number produced. Let $K = \sum_{i=1}^{n} K_i$. Obviously, $0 \leq j \leq \lceil \frac{K-1}{3} \rceil$, since we want K random numbers from three streams.

If h and q are selected appropriately, for example, $h \in \{2891336453, 29943829, 32310901\}$ and $q \in \{3, 5, 7\}$, the generated sequences pass formal statistical tests [8]. To avoid repetition, differing seeds are used to initiate linear congruential generators. The pseudo-random sequence is then sorted in a descending order. This provides a unique index order for relocating each *field* in the message. For example, if Z_j (jth random number ends up in position m after sorting, it means that the j field in the message will be relocated to the mth position. Note that the contents of each field remain unmodified and in the clear. The entire scheme is described in Algorithm 1.

B. Algorithm Performance

The recurrence relations used in Algorithm 1 can be computed very fast with minimal memory to preserve the state. This allows the simulation of multiple independent streams. Also, there is no repetition in the permutation sequence, since the period of each linear congruential generator is 2^{n_0}, which by design is much larger than $K-1$. In [7] and [9], we showed that this generator passes all of the 22 statistical tests specified by NIST-SP800.

The key is randomized with a salt value to avoid using the same permutation mapping repeatedly. Since the security margin of the proposed permutation scheme is $\lceil \log_2(K-1) \rceil$ chosen queries, the key is salted every $\lceil \log_2(K-1) \rceil - 1$ message communications. The salting must be done by both the transmit and receive side so that the two sides remain synchronized. The salting helps to resist certain types of cryptanalysis, such as plaintext attacks and precomputed rainbow table attacks [10]. For example, for $K = 128$, the key is salted every ten transmissions using a counter, which is 32 bits in most microprocessors. With a video-frame rate of 30/sec and frame level chunks (usually a chunk contains several frames), rekeying is needed once in 44 years. The salting effectively avoids an EC to attack the encryption scheme and determine the ID of the devices using chosen-ciphertext attacks, which is the most powerful form of the attack.

TABLE I
ALGORITHM PERFORMANCE

Algorithm	KB/s
AES-128-CCM	70
AES-128-EAX	70
AES-128-GCM	41
Chaskey	145
ChaCha20-Poly1305	94
MD5	147
Proposed method	424

The computational complexity of the proposed method is the sum of complexities of the pseudo-random number generation and the radix sorting procedure. Our algorithm was tested on a 48-MHz ARM Cortex-M0 microcontroller along with other popular schemes such as AES-128-CCM, AES-128-EAX, AES-128-GCM, Chaskey, MD5, and ChaCha20-Poly1305 on the same platform [11]. The detailed performance is discussed in [7] and is repeated in Table I. It is clear that our method is much faster than all of the competitors and is suitable for real-time use.

C. Discussion of the Algorithm

Although the above algorithm can be used by individual devices, the reason to use it on matrices is to obfuscate the device IDs. Also, while we could just as well apply the algorithm for shuffling individual bits, we decided to use it to shuffle fields so that it is possible to retrieve individual fields within the encrypted message if the index (or location) of the field after shuffling is provided. Even though all the fields are available to the EC in the clear, it cannot "guess" which devices provided them without such an index – assuming that the message is sufficiently long.

The scheme effectively provides a lightweight mechanism to protect the data/feature stream in transit and privacy-preserving querying on the encrypted contents. Since an EC is only interested in some of the fields, the leader can allow it to extract those without revealing the identity of the device, where the field came from. The same thing is true concerning storing the encrypted data in other ECs. Whenever another party (that is, an EC or a cloud) wishes to query a certain feature, the leader (or the EC that has full access to the data) quickly computes the pointer to the feature in the encrypted domain and exchanges this information with the other party to recover the intended ciphertext. Also, any party can obtain statistical information (for example, the count of the number of devices that report the same or similar features) without any decryption of the message.

The encrypted operation capabilities also introduce some vulnerabilities due to fields being in the clear. First, external intruders can observe the communications, but this is easily fixed by doing bit-level encryption of the entire matrix using the same algorithm, by potentially using just one key per EC. Second, the ECs can monitor the intelligible information in various fields for subtle markers of each device and thereby predict the most likely device producing each field. For example, the richness of the colors, image defects, time gaps between same or similar features in different fields, could be analyzed together to give away the device ID. This issue might be resolvable by slight perturbation of the field data, but we do not discuss it further.

The proposed mechanism essentially assumes that an EC cannot be trusted with keeping the device features confidential, but the leader device can be trusted to properly aggregate, encrypt, and send the message to the ECs. This may be somewhat questionable particularly when the leader device is chosen randomly among those that happen to be in the vicinity (for example, cars around a given car that associate with a given RSU). It is possible to address this by randomly choosing another leader as well so that occasionally the same matrix is relayed by different leaders. A simple comparison could then detect any discrepancies. Identifying which device is problematic in case of a mismatch requires more work, and could be done by mandating full identity release to either the EC or another checker party.

V. Related Work

IoT devices have limited computation power and energy resources. Hence, they may not be able to execute computationally expensive tasks. Indeed, IoT devices collect the sensed data and hand it to computationally more powerful devices, through which the sensed data will be further processed and analyzed. ECs can mitigate the power consumption of the IoT devices through the offloading of computation tasks. However, they are limited in computation power and hence are subject to the scalability problem.

For enhancing privacy, often the sensed information is perturbed either to make the source identification difficult or to make the output inexact [12], [13]. Previous works [14] focus on the application of a single masking mechanism, often a lightweight noise generation process that samples random noise values from a Laplace distribution and then aggregates it into the data.

Another family of solutions is structured encryption [15], which is a generalization of searchable symmetric encryption [16]. Structured encryption schemes encrypt data structures in such a way that they can be queried. Such schemes can be distinguished depending on the type of operations they support. This includes non-interactive and interactive schemes where the former require only a single message while the latter requires several rounds for queries and updates. The study of literature shows numerous searchable symmetric encryption constructions for various data structures, including two-dimensional arrays [17], graphs [18], and multi-maps [19]. However, the past searchable and structured encryption works are computationally expensive and henceforth are not suitable for edge computing application scenarios.

VI. Conclusions

In this paper, we described the security, privacy and access control issues in an edge computing environment and sketched an initial solution to the access control and security problem. For access control, we divide streams into chunks and enforce time/event based access to them. The security solution uses a lightweight permutation scheme which is suitable for encrypting real-time data streams generated by weak devices. The encryption is done by a leader device which allows device ID obfuscation, easier key management, and allows operation on encrypted data. In the future, we want to consider more elaborate access control mechanisms and test their scalability. We will also examine distributed data sharing under such constraints and the benefits of the approach. We will devise more general security mechanisms that better obfuscate different types of information that may be considered sensitive.

Acknowledgment

This research was supported by NSF grant CPS-1544904.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[2] A. Pal and K. Kant, "Iot-based sensing and communications infrastructure for the fresh food supply chain," *Computer*, vol. 51, no. 2, pp. 76–80, 2018.

[3] IEEE 1609 Working Group, "IEEE standard for wireless access in vehicular environments-security services for applications and management messages," *IEEE Std 1609.2*, 2016.

[4] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke, "The IEEE 802.11 universe," *IEEE Communications Magazine*, vol. 48, no. 1, pp. 62–70, 2010.

[5] M. Le, K. Kant, M. Athamnah, and S. Jajodia, "Minimum cost rule enforcement for cooperative database access," *Journal of Computer Security*, vol. 24, no. 3, pp. 379–403, 2016.

[6] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Computer communications*, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[7] A. Jolfaei and K. Kant, "A lightweight integrity protection scheme for low latency smart grid applications," *Computers and Security Journal (COSE)*, Oct 2018.

[8] M. Bellare, S. Goldwasser, and D. Micciancio, ""pseudo-random" number generation within cryptographic algorithms: The DDS case," in *Annual International Cryptology Conference*, 1997, pp. 277–291.

[9] A. Jolfaei and K. Kant, "A lightweight integrity protection scheme for fast communications in smart grid," in *14th International Conference on Security and Cryptography (SECRYPT)*, 2017, pp. 31–42.

[10] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Annual International Cryptology Conference*, 2003, pp. 617–630.

[11] J. Birr-Pixton, "Benchmarking modern authenticated encryption on €1 devices," https://jbp.io/2015/06/01/modern-authenticated-encryption-for-1-euro/, 2015.

[12] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[13] T. Wang and L. Liu, "Output privacy in data mining," *ACM Transactions on Database Systems (TODS)*, vol. 36, no. 1, p. 1, 2011.

[14] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and D. Megías, "Individual differential privacy: A utility-preserving formulation of differential privacy guarantees," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1418–1429, 2017.

[15] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 577–594.

[16] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[17] S. Kamara and L. Wei, "Garbled circuits via structured encryption," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 177–188.

[18] Y. Zhang, A. O'Neill, M. Sherr, and W. Zhou, "Privacy-preserving network provenance," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1550–1561, 2017.

[19] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 965–976.