EDGE COMPUTING: DATA SHARING AND INTELLIGENCE

Yeghisabet Alaverdyan[1,2], Suren Poghosyan[2] and Vahagn Poghosyans[2,3]

[1]EKENG CJSC, Yerevan, Armenia
[2]Institute for Informatics and Automation Problems of NAS RA, Yerevan, Armenia
[3]Synopsys Armenia

ABSTRACT

"The paper introduces certain timely and secure computing approaches that affect data intelligence related to methods and tools for real-time information processing. Timely solutions are achieved using local premises rather than supporting centralized servers or clouds. Computing within the network partly occurs near the physical endpoints, and this is where the edge computing paradigm comes in to help. The proposed method of the cloud optimization suggests splitting and sharing data between network data centers and local computing power. Provision of distinct paths between edge and main cloud for each smart device is achieved using separate blockchain which register and store the logical links of hierarchical data. Methods for cryptographic key splitting between the edge and the main cloud, as well as strong authentication ensuring the shared data confidentiality, integrity, availability, and consistency are also given.

KEYWORDS

Edge computing, key splitting, authentication, blockchain, data sharing, data intelligence

1. INTRODUCTION

Information systems vary in their primary roles and goals. Especially teal-time information systems impose deadlines for producing control responses, as latency in their data generation, processing and transfer may cause impermissible failures. For such systems, provision of proper transactions depends not only on their logical correctness, but also on operability in the required time. Latency makes these systems useless. Few examples of timely computation environments are: urgent medical diagnosis; nuclear plants management; traffic control, real-time speech and image processing; video surveillance, etc.

Edge computing is especially beneficial to specialized and intelligent devices which are not designed to perform hard computational tasks. These devices respond to particular needs in a specific way.

Edge computing is preferred over cloud computing in remote locations, where there is limited connectivity to the centralized cloud. Operational management of resource-constrained devices requires the deployment of rather local and low-latency data centers utilizing technological solutions of edge computing in terms of appropriate infrastructure and data intelligence.

Traditional cloud technologies do not address timely requirements for data processing. The main cloud is commonly used to store and process data that are of a huge volume and are not time-driven. On the other hand, moving large amounts of sensitive data to the main cloud opens backdoors to cyber threats, such as: data leakage, loss or theft especially in shared environments; insecure interfaces or APIs, etc., such that promoting prosperous innovations in the field is of a greater importance.

The remainder of this paper is organized as follows. Section 2 presents a review of related works in the field. Section 3 briefly describes the edge infrastructure and edge computing paradigm. Section 4 presents data sharing and intelligence at the edge. A brief Summary of the paper is given in Section 5.

2. Related Works

Cloud computing provides IT resources on demand and permanently optimizes resource provisioning [1]. Anyway, timely requirements affect data intelligence related to methods and tools for integrating and classifying real-time information collected from IoT smart devices, and, as authors in [2,3] state, given the unpredictable nature of the devices, maintaining and securing fast and high-quality algorithmic results in the face of uncertainty is still a challenging problem. The authors propose to tackle this challenge by applying coding theoretic techniques to provide resiliency against noise.

Edge/Fog computing [4] emerges as a novel computing paradigm that harnesses resources in the proximity of the IoT devices so that, alongside with the cloud servers, provides services in a timely manner. In the work, the authors proposed a weighted cost model to minimize the execution time and energy consumption of IoT applications in a computing environment with ever-increasing growth of IoT devices, multiple fog/edge servers and cloud servers. The paper introduces a new application placement technique based on the Memetic Algorithm to cover concurrency of IoT applications. In order to address IoT applications’ heterogeneity, a lightweight pre-scheduling algorithm to maximize the number of parallel tasks for concurrent execution has been suggested.

The work [5] introduces a survey on multi-access edge computing architecture and proposes an elaboration of the cloud computing platform via the deployment of storage and computational resources at the edge thus reducing latency of edge devices and utilizing the core network more efficiently.

Authors in [6] rightly state that cloud and edge/fog computing are non-interchangeable technologies, and they cannot replace one another, but, when combined, they can contribute to promoting hierarchical data infrastructure and layered data processing.

In [7], authors interpret fog/edge computing-based IoT to be the future infrastructure on IoT development and an important computing paradigm in realizing the intelligent cyber-physical world.

Authors of the paper [8] highlighted typical security issues in edge architecture. Besides, they explored security algorithms acceptable for governing IoT data privacy taking into account the resource-constraint nature of the devices. Particularly, elliptic curve based cryptographic solutions were selected as most appropriate.

Understanding and implementing edge computing implies consideration of traditional computing paradigms [9], exploration of which can orientate the developers in selecting and combining relevant approaches. Besides the well-known methodologies in computing, trusty systems' development requires embedding confidential computing [10, 11] which preserves the data confidentiality not only in the rest and in transfer, but also in use. Shielding the sensitive data, also the whole set of applied programming techniques from the rest of the entire system, makes the data being processed to become accessible solely to authorized programming codes.

3. Edge Computing Infrastructure

Edge computing architecture is still three-tier network. The very basic level in this hierarchy outlines the edge level network, where network devices and sensors reside. The second layer deploys the edge data centers which perform the operational management of individual groups of edge devices. Classification of devices prior to inclusion to some group supervised by an individual mini data center should be performed according to their mission and level of intelligence. At this level, data collected from sensors will be analysed and categorized before sending to the main cloud for further processing. The main cloud server is on the top of the hierarchy.

The segmented architecture of the entire network enables splitting the overall workload while leaving some portion of data processing at the basic layer of the network, closer to the devices. Note that edge computing shouldn't be confused with autonomous computations and data processing embedded in edge devices. Figure 1 illustrates the edge computing infrastructure.

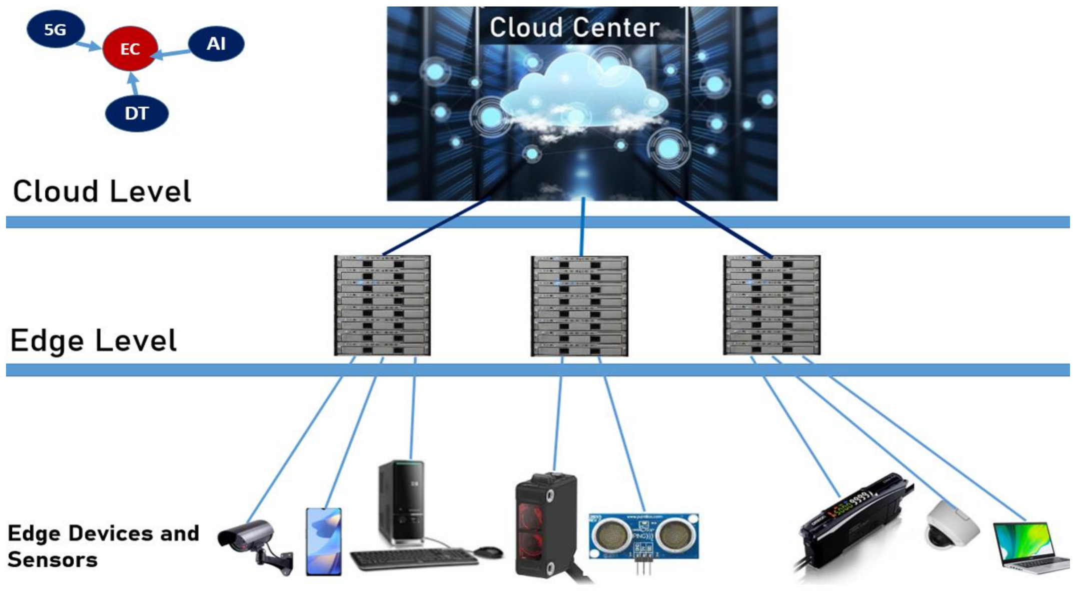

Figure 1. Edge computing architecture.

Edge computing stands at the intersection of 5G, artificial intelligence (AI) and digital transformation (DT).

3.1. Edge Computing Paradigm

In edge computing, data are collected, processed and analysed hierarchically at different layers on the network according to a predefined role distribution scheme, still near the source where they are generated. Thus, besides data collection, the edge level may implement timely intelligent tasks, such as:

- data aggregation, classification and validation,
- peer-to-peer communication among regrouped devices (if needed),
- visualization,
- data filtering,
- distribution, dissemination and replication,
- data mapping and reducing,
- caching, chaining, backup,
- data and cryptographic key splitting, etc.

Edge architecture eliminates the necessity of allocating individual bandwidth for each device on the main cloud. This significantly optimizes the overall computing power available on the entire network by executing some tasks at the edge level. Introducing the additional layer will solve a series of issues by adopting the so-called *data minimization* principle, according to which the amount of data being sent to the main cloud should be minimized while keeping the whole chain of the data under control. The main cloud still plays its own role in big data analysis, business logic maintenance and data warehousing.

4. DATA INTELLIGENCE AT THE EDGE

In the edge architecture, mass of data generated and transferred by IoT devices are accumulated at the edge level, raw and unanalysed. As digital transformation becomes increasingly ubiquitous, more and more sophisticated solutions (appreciable) are being engineered and embedded to reinforce edge computing.

The paper focuses on operating deliberative AI agents and introduces a selective approach for intelligent supervision of real-time AI systems regrouped according to their intellectual mission. For this scenario, relevant expert systems each designated to respond to specific queries is recommended to be deployed on the main cloud site, where historical experience in terms of the knowledgebase is stored and gets updated, also the overall data analytics is conducted.

Generally, data intelligence refers to analytical tools and methods used to not only derive information from the collected data, but also to support machine intelligence, such as: deduction and semantic reasoning.

In this sense, edge computing, when implemented for timely management of self-organized collaborative computing agents, can deploy mini data centers for information accumulation, reasoning and exchange.

For example, semantic graph models for agent machinery and the system abstract architecture may assume development of mathematical structures of the following shape [12]:

$$MS = \{E, S, Node, KBP, WU, Ac, Ag, Fb\},$$

Where

- *E* is the set of environmental states
- *S* is the set of states of the agent
- *Node* is a vertex on the semantic graph. The nodes assume accumulating knowledge, history and evolving operational capabilities of the agents
- *KBP* is the knowledge base pool which collects and assimilates the derived information
- *WU* is the so-called wisdom unit which decides on the best strategy of utilizing the knowledge while interacting with other agents relevant to the scope of interaction

- *Ac* is the set of actions which are selected in accordance with the current decisions made
- *Ag* is the agent which in its strict mathematical formulation is a chain of mappings
- *Fb* is the feedback which is activated upon completion of every execution of actions.

Here, computation is performed over relevant pieces of information having two qualities: value and weight. The deductive logic is constructed on the strict logical arguments reinforced with the specified rules of inferences over premises and conclusions. Activation within the network depends on the weights. Reasoning based on zero, first, second and even higher order logic [13] assumes involvement of non-supervised machine learning methods and techniques implementing facts, relations and rules.

For example, to support declarative statements having either of two outcomes; *true* or *false*, zero-order (propositional) logic may be constructed as follows:

Algebraic structures

group = 'grp'   # (G, $\circ$), a non-empty set with a single operation
# Define facts and properties of the algebraic group
fact(group, closure)   # means: $\forall a, b \in G \rightarrow a \circ b \in G$
fact(group, associative) # means: $\forall a, b, c \in G \rightarrow a \circ (b \circ c)=(a \circ b) \circ c$
fact(group, unit)   # means: $\forall a \in G \rightarrow a \circ e = e \circ a = a$
fact(group, inverse)   # means: $\forall a \in G \rightarrow a \circ a^{-1}= e$*, etc.*

First-order logic (a.k.a first order predicate logic) is an extension of the propositional logic and involves quantifiers; Boolean representations to recognize variables and constants, and relations which cannot be classified as *true* or *false*. Notions used in first-order logic constructs are:

- ✓ variables (x, y, z, …),
- ✓ constant magnitudes (SIZE, FILE_PATH, etc.),
- ✓ predicates (isEven, isPrime, ...),
- ✓ functions (fit, linReg, ...),
- ✓ operators and connectives (logical, bitwise, ….), etc.

A nice example of the first order formula, given a nonempty domain *D* and a nonempty range *R*, is the definition of functions as a specific type of binary relations involving an existential quantifier.

$$\forall x \in D \rightarrow \exists! y \in R: y = f(x).$$

First order logic is widely used to construct number-theoretical statements over atomic expressions of type $x = y$, $x + y=z$ and $x \times y=z$, $x, y, z \in N$, equipped with the propositional operations $\wedge$, $\neg$, $\vee$, $\rightarrow$ and the quantifiers $\forall x$ and $\exists x$.

With second-order logic we can make machine more intelligent: having variables *x, y, z,…* from the first order logic, we involve properties *X, Y, Z,…* and quantifiers ∀X…, ∃X… to express binary relations among pairs of numbers. Having done this, second and higher order logic may be constructed over generalized identities using quantifiers of existential and universal type, such as: ∀, ∃, ∃!

For example, the followings are absolutely closed second-order formulas [13],

$$\forall X_1 .... X_m \; \forall x_1 ... x_n \; (\omega_1 = \omega_2),$$

$$\forall X_1 \ldots X_k \exists X_{k+1} \ldots X_m \; \forall x_1 \ldots x_n \; (\omega_1 = \omega_2),$$

where ω_1, ω_2 are words written in functional variables $X_1 \ldots X_m$, and in the objective variables $x_1 \ldots x_n$, are called $\forall$ $(\forall)$ identity and $\forall\exists$ $(\forall)$ identity, respectively.

Second-order quantification, commonly used in machine intelligence, can be successfully applied in solving reachability on large decision trees. For example, *DerivedNode(a, b)* denoting that *a* is a child node of *b* cannot be implemented solely in the first-order logic.

Knowledge representation in AI agents combines objects, relations, and functions via a symbolized reasoning. Here, instructions are partitioned into well-defined subjects and predicates with a strict restriction: predicates refer to single subjects.
F.e., given an original sentence: *The vehicle should decrease the speed,* the corresponding predicate logic will be of the form: *decrease (vehicle, speed).*
Here, *vehicle* is subject, *decrease* is a predicate and *speed* is the value defined by the predicate.

Reinforcement learning can be applied to train deductive agents situated in an environment to simulate collaboration and information full exchange among peers, as follows:

```
env = env ()
wisdom=wisdom () # suggests the best strategy to react
skill=skill () # inherit skills
for agent in env. agent_iter ():
    success_story=[]      #empty logical sentence
    observation, info = env. snapshot ()
    action = policy (agent, observation, wisdom)
    info_ dissemination (other_agents, info, gossiping)
    success= env. step (action, success_story)
    if success:
      reward=true
      skill. increment ()
      wisdom. accumulate (action, success-story)
```

Mini data centers endowed with local strategy patterns each operating separate groups of intelligent devices with similar intelligence will react to stimuli faster and according to specific needs. Urgent medical diagnosis, traffic control, borders' security and other timely computations utilizing machine intelligence can certainly benefit from the edge computing, meanwhile historical data and relevant expert knowledge will reside on the traditional cloud being queried on demand.

This way, edge computing promotes deployment of specialized data centers to operate classified IoT devices rather than embedding unified clouds for serving the needs of heterogeneous sources of information.

4.1. Data Splitting and Sharing

Data splitting is the act of partitioning available data into portions according to some well-defined criteria. Generally, splitting is done to perform pre-processing prior to model construction, and may contribute to:

- ✓ parallel computations
- ✓ segmentation and pattern recognition

- ✓ cross-validation, where one portion of the data is used to run a specific computation and the other to evaluate the results
- ✓ secret splitting and threshold scheme construction, like: Shamir, Blakely, or other algebra based
- ✓ hierarchical computing.

Edge computing paradigm can benefit from this methodology by applying unsupervised learning for classifying the data being stored and processed on different layers on the entire network.
Graph partitioning algorithms and equivalence relations from discrete mathematics can stand for excellent basics for implementing data splitting and resource regrouping at the edge. Spectral clustering and Louvain Clustering are other mechanisms for splitting and evaluating the data models.

The following should be ensured prior splitting the data into disjoint segments:

a) data distribution is always the same, or
b) data distribution changes over time.

In the simplest implementation, edge data splitting paradigm assumes:

a) leaving and processing timely data at the edge level, and
b) transferring the historical data, also skills (successful or unsuccessful stories) gained through intelligent actions to the main cloud. Here, both successful and unsuccessful stories will be assimilated by the expert system and contribute to the overall wisdom.

Data sharing is the process of making the same data accessible to a number of users or applications based on the one-to-many principle of association.

Note that data within a device are under the sole control and are of a required physical and cyber security. In this sense, the edge level must ensure the confidentiality and integrity of data being shared or exchanged in order to exclude exposing data to unauthorized access.

In edge computing, some portions of data are shared with the main cloud for a deeper and probably latent analysis. The edge level may even split and share cryptographic keys, tokens and other cyber identifiers in order to keep trace of the shared data between computational and storage levels. The hierarchical nature of the edge computing paradigm points out a series of sensitive aspects that should be addressed. Particularly, while splitting is done within a specific edge data point and under the local control, sharing data parts results in transferring them, and this is when solely GDPR compliant solutions have to be embedded in order to ensure zero-trust based identification and strong authentication of the shares. The volume of intelligible data that needs to be shared should be minimized. Here, the data minimization refers to encrypting data thus making their content inaccessible for non-authorized access. As a result, the amount of available and intelligible data consequently decreases and resides in the scope of confidential computing.

4.2. Edge Data Privacy and Identity Management

Privacy at the edge is of a great importance and implies preserving data confidentiality, integrity and availability [14]. For this purpose, we suggest delegating most of the cryptographic operations to edge servers while embedding pre-image resistant and collision-free cryptographic hash functions.

A hash function $H: S \rightarrow S'$ is a mapping from a set of arbitrary cardinality (S) to a single value from a set of a fixed (fewer) cardinality (S'). Cryptographic one-way hash function H, with the domain S; the range S'; s ∈ S, and y ∈ S', is of a polynomial time computation for all s ∈ S, while finding $s: H(s) = y$, is an NP-complete problem. This feature provides pre-image resistance of the function. If $Prob\_of\_finding\ (M, M'): H(M) = H(M') \rightarrow 0$, then the mapping H is also collision-free (Russell 1992).

Besides, the following mechanisms are suggested to be embedded at the edge:

- ✓ Hardware based tamper-proof storage (HSMs) for secure cryptographic key generation
- ✓ Honey Encryption combined with Advanced Encryption Standard algorithm for data encryption at rest and at transfer.

Identity management ensures automatic identification and authentication of edge devices and relevant edge computing resources. For this purpose, the following mechanisms should be embedded at the edge level.

- ✓ IoT Device Identity Lifecycle Management
- ✓ Authentication, in order to ensure validity of entities within the system. This is the first step to perform any computing operation before moving to the next steps
- ✓ Authorization, that always comes after the authentication and allows solely verified entities to get access to resources
- ✓ Distributed identity management architecture, equipped with 5G network, in order to assign unique identities by the software defined identity management systems. Here, local nodes will share parameters with the local Software Defined Network servers to acquire identity. Local SDN devices synchronize their identity databases with the main cloud SDN device. All Global SDN devices will also synchronize their databases. Thus, the whole hierarchy in the edge architecture will be strongly identified.

Whenever appropriate, timestamp may be embedded.

A timestamp is a mapping $T:(A \times L \times G) \rightarrow S$, where

- ✓ A is an alphabet,
- ✓ L is the set of literals,
- ✓ G is the global TSA (Time Stamp Authority, the global secure time stamp server) data. The TSA will register the current transaction date and time which, when necessary, can be verified with the main cloud to resist to replay attacks.
- ✓ S presents the resultant timestamp string.

In cases, when embedding PKI (Public Key Infrastructure) is expensive, self-certified cryptography may be utilized to implement the registration and at least two-factor authentication of network entities. Meanwhile, authentication is verified each time before edge and main cloud levels interact. Internally generated certificates can be bound to identities and construct the identity and certificate management mechanism based on blockchain. Blockchain identity management systems are commonly used to address identity issues such as data insecurity and fraudulent identities. The distributed ledger, blockchain, will store:

- ✓ complete paths (individual or joint, depending on the implementation) for each device having data shared between the edge and main cloud,

✓ labelling of the classified data at their cascading aggregation, validation, classification and then filtering phases, in order to trace the data hierarchical chain from the edge to the main cloud.

Due to resource-constraint nature of IoT devices, we propose constructing of the edge-to-main cloud blockchain based on secure computations on elliptic curves. Unlike the simplified curves used in traditional data encryption and decryption, the selection of unique points here is done on an original curve in order to meet higher security requirements.

An elliptic curve E over the real numbers R is defined by a Weierstrass equation,

$$E: y^2+a_1xy+a_3xy+a_3y = x^3+a_2x^2+a_4x+a^5$$

with coefficients $a_1, a_2, a_3, a_4, a_3 \in N$, and $\Delta \neq 0$ discriminant.

The set of points on the curve is:

$$E(L) = \{(x, y) \in R\times R: y^2+a_1xy+a_3xy+a_3y = x^3+a_2x^2+a_4x+a^5= 0)\} \cup \{0\}$$

with the point of infinity (the 0 point).

It is well known that blockchain is a digitally distributed, decentralized, public ledger that exists across a network. No single entity controls the blockchain network; anyone can join at any time. The above premises provided, identification of IoT devices within a group and on the main cloud (when necessary) is achieved as described below.

Initially, prior regrouping the devices, the ledger is an empty list. The list gets incrementally updated with every new device joining the specific group and presents a dynamic list of unique identifiers obtained in the following steps:

a) For every device in a group, a unique point on the elliptic curve is selected and recorded at the edge secure server zone. AES 256 is applied to encrypt the points according to data anonymization principle: *encrypted data is neither usable nor decrypted* as they are not used in any of further transactions. Utilizing elliptic curves is motivated by an NP-completeness of guessing the point coordinates even if the curve is made public.
b) The encrypted point is hashed with the timestamp H_1 *(Point, Timestamp)* using SHA256.
c) H_1 *(Point, Timestamp)* is hashed with the previous content in the public ledger (this step is skipped with the first record).
   The resultant H_2 *(Point, Timestamp, History)* is the unique ID for a device joining the group.
d) H_2 is recorded in the edge ledger.
e) The ledger is transmitted to the main cloud server to a distinct location.

Distinct paths for each blockchain provided, anyway, access control mechanisms that meet the edge computing security, privacy and data diversity requirements should be upgraded from traditional access control schemes. For this purpose, involving Bloom filter integrated with identity management and lightweight secret key agreement protocols based on self-authenticated public key may serve as a good basis for innovating edge/cloud access control.

4.3. Cryptographic Key Management at the Edge

Key management at the edge administers the full lifecycle of hierarchic cryptographic keys.
It is strongly recommended to limit

- ✓ the amount of information protected by a given key,
- ✓ the amount of exposure if a single key is compromised,

Note, that edge-level encryption secures the data collected from edge devices such that no level in computing receives the raw version of the payload directly. For effective and secure key management, HSM and related secure zone at the edge level should be embedded. This will ensure supporting sensitive operations, like:

- ✓ key generation, key secure storage and key encryption,
- ✓ key distribution, backup,
- ✓ verifiable secret splitting and sharing, etc.

In order to support two-level cryptographic key sharing, the edge-level HSM internally performs:

- ✓ cryptographic key generation,
- ✓ cryptographic key encryption applied AES-256,
- ✓ cryptographic key verifiable secret splitting. One portion of the encrypted key remains at the edge secure zone, while the second portion is transferred to the main cloud
- ✓ each time a transaction is activated on the main cloud site, firstly a verification of the validity of the key shares is performed.

Most of existing secret sharing schemes of proven security (like in Shamir's threshold scheme) are based on the assumption that all participating users are legitimate. This approach is prone to sophisticated attacks: the attacker can impersonate a legitimate party without being detected. Schemes of proven security can be found in [15,16] where the impersonation attack is advised to be blocked based on modification of Shamir's threshold scheme, or based on plane parametric curves with one-parameter representation for a master key, respectively. Other solutions are arising from Latin squares [17].

We propose constructing a verifiable secret sharing scheme based on abstract structures from non-associative algebras. The theory of quasigroups is a permanently evolving scientific direction. Quasigroups are based on the Latin square property and stand for a generalization of groups without the associative law or identity element [18].

The attractiveness of quasigroups in construction of verifiable secret splitting and sharing schemes is in their easily programmable nature due to utilization of solely logical operations. A quasigroup with its parastrophs (Q, ·, \, /) is a set closed under three different binary operations, referred to as multiplication (·), left division (\) and right division (/) satisfying the conditions:

1. $x \cdot (x \backslash y) = y$
2. $(y / x) \cdot x = y$
3. $x \backslash (x \cdot y) = y$
4. $(y \cdot x) / x = y$
5. $x / (y \backslash x) = y$
6. $(x / y) \backslash x = y$

The verifiable secret splitting is performed as follows:

1. the order *n* of the quasigroup (the number of its elements) dictates the number of shares
2. edge computing paradigm suggests having (2-out-of 2^n) shares satisfying above properties for the secret quasigroup
3. the two shares are encrypted and distributed between the network layers
4. when combined, the two shares are decrypted within the edge level HSM, where shares are authenticated.

The proposed scheme eliminates the risk of impersonation attacks against both the edge and the main cloud levels: the shares are verified at the edge level HSM by a polynomial time computation (meanwhile for the non-legitimate party this computation will lead to a numerical explosion with a large order of the secret quasigroup), and the main cloud site is secured by confidential computing.

Another significant factor motivating the usage of quasigroups is that generalized identities of higher order logics can be effectively constructed on quasigroups without having any significant impact on algorithmic complexity.

5. CONCLUSIONS

The paper presents a series of methodologies which can be efficiently implemented in edge computing. By combining relevant computing paradigms and secure data processing, the proposed approaches can reinforce timely and intelligent computing within the cloud-edge hierarchy, where data chaining and logical linkage is achieved by embedding blockchain through verifiable secret sharing.

REFERENCES

[1] Ramanathan R. (2019) “Towards optimal resource provisioning for Hadoop-MapReduce jobs using scaleout strategy and its performance analysis in private cloud environment”, *Cluster Computing*, Springer, Vol. 22, No. 6, pp 14061–14071.

[2] Cao K., Liu Y., Meng G., & Sun Q. (2020) “An Overview on Edge Computing Research”. *IEEE access*, Vol. 8, pp 85714-85728.

[3] Lee K., Lam M., Pedarsani R., Papailiopoulos D., & Ramchandran K. (2018) “Speeding up distributed machine learning using codes”, *IEEE Transactions on Information Theory*, Vol. 64, No. 3, pp 1514–1529,

[4] Goudarzi M., Wu H., Palaniswami M., & Buyya R. (2021) “An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments”, *IEEE Transactions on Mobile Computing*, Vol. 20, No. 4, pp. 1298 – 1311.

[5] Taleb T., Samdanis K., Mada B., Flinck H., Dutta S., & Sabella D., (2017) “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration”, *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 3, pp 1657–1681.

[6] Ferdinand N. & Draper S. C. (2018) “Hierarchical coded computation”, *IEEE International Symposium on Information Theory (ISIT)*, pp 1620–1624.

[7] Lin J., Yu W., Zhang N., Yang X., Zhang H., & Zhao W. (2017). “A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications”. *IEEE Internet of Things Journal*, Vol. 4, No. 5, 1125-1142.

[8] Jose D. V., & Vijayalakshmi A. (2018) “An Overview of Security in Internet of Things”, *Procedia Computer Science*, Elsevier, Vol. 143, pp 744-748.

[9] De Donno M., Tange K. P., & Dragoni N. (2019) “Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and fog”, *IEEE Access*, vol. 7, pp. 150936 – 150948.

[10] Baumann A., Peinado M., & Hunt G. (2014) “Shielding Applications from an Untrusted Cloud with Haven”, In: *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, Broomfield, CO: USENIX Association, pp.267–283.

[11] Alaverdyan Y., & Satimova E. (2019) “Fully Homomorphic Cipher Based on Finite Algebraic Structures”, *Earthline Journal of Mathematical Sciences*, Vol. 1, No. 1, pp 97-103.
[12] Alaverdyan Y. (2022) “Multi Agent Machinery in Construction of Cognitive Systems”. *Journal of NeuroQuantology*, Vol. 20, No. 8, pp 2445 -2452.
[13] Movsisyan Yu., & Gevorgyan A. (2021) “Invertible algebras satisfying associative identities with functional variables”, *Asian-European Journal of Mathematics*, 2021, Vol. 14, N 1, 2050155 (16 pages).
[14] Pang H. & Tan K.-L. (2004) “Authenticating query results in edge computing”, *Proceedings of 20th International Conference on Data Engineering*, pp 560–571.
[15] Chor B., Goldwasser S., Micali S. and Awerbuch B. (1985) “Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults”, FOCS85, pp. 383-395.
[16] Li B. (2021) “Verifiable Secret Sharing Scheme Based on the Plane Parametric Curve”. Applied Mathematics, 12, pp. 1021-1030.
[17] Cooper J., Donovan D., Seberry J. (1994) “Secret sharing schemes arising from Latin squares, Bulletin of the Institute of Combinatorics and its Applications, 12, pp. 33-43.
[18] Pflugfelder H., (1990) “Quasigroups and loops”, Heldermann Verlag, Sigma series in Pure Mathematics, 7, pp.28-59.

AUTHORS

Yeghisabet Alaverdyan. Head of Systems Integration Department at EKENG CJSC since 2018. Graduate of the National Polytechnique University of Armenia. PhD, Associate Professor. Author of more than 25 scientific papers published in local and international scientific journals. Research is in Mathematical Cryptography, AI and Cognitive systems modelling.

Suren Poghosyan. Lead of Scientific group at the IIAP NAS RA since 1988. Graduate of the Yerevan State University. Author of more than 25 scientific papers published in local and international scientific journals. PhD. Research is in Self-organized systems modelling.

Vahag Poghosyan. Research fellow at the IIAP AS RA and Senior Software developer at Synopsis Armenia. Graduate of the Yerevan State University. PhD. Author of more than 30 scientific papers published in local and international scientific journals. Research is in Self-organized systems modelling.