PUF-based Authentication Scheme for Edge Data Centers in Collaborative Edge Computing

Seema G. Aarella*, Saraju P. Mohanty*, Elias Kougianos† and Deepak Puthal‡
*Department of Computer Science and Engineering, University of North Texas, USA
†Department of Electrical Engineering, University of North Texas, USA
‡Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE
Email: seema.aarella@unt.edu, saraju.mohanty@unt.edu,
elias.kougianos@unt.edu, deepak.puthal@ku.ac.ae

***Abstract*—Fog computing gained importance when the required infrastructure became more decentralized. The difference between Fog and Edge computing is that the former has nodes between the cloud and edge devices. In a way both fog and edge computing bring data processing closer to end-user devices. Edge Data Centers (EDCs) are employed to bring the processing power closer to the end-user to address latency in real-time applications, bandwidth, battery life constraints, as well as data privacy and safety. Edge Data centers are employed in collaborative edge computing environments across network edges, and they deliver uninterrupted processing through load balancing, where an overloaded Edge Data Center can off-load its task to the next available EDC. Authenticating EDCs is one of the challenges in collaborative edge computing environment where load balancing is applied. Physical Unclonable Functions (PUFs) are lightweight innovative primitives that are used for authentication and secure key storage. This work proposes a novel authentication scheme for EDCs using virtual Physical Unclonable Functions as a secure option.**

***Index Terms*—Edge Computing, Edge Data Centers, Collaborative Edge Computing, Load Balancing, Secure Authentication, Physical Unclonable Functions, Virtual PUFs**

I. INTRODUCTION

Computing at the cloud involves bigdata storage and processing. The processing environments that involve real-time and low latency and try to move the processing infrastructure closer to the edge. The Edge Data Centers which are deployed at different geographical locations collaborate with each other in getting the task done through load balancing schemes [1]. Some of the areas of applications that involve Edge Data centers are Content Delivery, Gaming, 5G and 4G infrastructure, Telemedicine, Smart Grid, Smart Home, Traffic Management and Autonomous vehicles. In the Internet-of-Things (IoT) architecture, everything is connected through the Internet, right from the end-user devices to the could, Fig. 1.

Fog computing brings the cloud resources closer to these underlying networks, enabling computation, storage and other services between the end nodes [2]. Hence the Fog nodes should be capable to authenticate the end users as well [3]. Some of the existing security mechanisms are attribute-based access control, group-signature based authentication, public-key based and homomorphic encryption [4].

Authenticating the EDCs in a collaborative Edge computing environment requires a secure and light weight option. Though many applications and research suggests using algorithms based on hash functions, Elliptic Curve Cryptosystem (ECC), and other encryption and decryption methods. However, the communication cost and the maintenance cost for these techniques also needs to be considered to call them as light weight options. The use of AES and RSA for generating security keys is also a secure solution to prevent man-in-the-middle attack, however it is not lightweight as the computation cost becomes high [5].

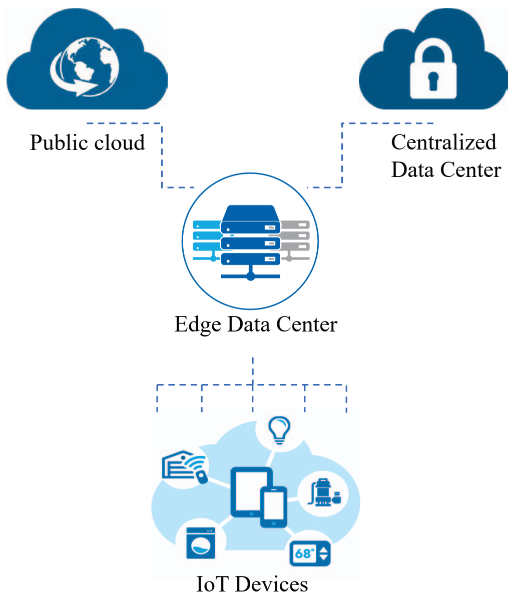


Figure 1: An Overview of Edge Data Center.

Physical Unclonable Functions (PUFs) use the device's physical variations from which unique keys can be extracted [6]. The function of PUF based authentication scheme is shown in Fig. 2. In general, a PUF is used in two steps: one is *enrollment*, where a number of CRPs (Challenge Response Pairs) are generated from a PUF and stored. The second step is *verification or authentication*. Given to the robustness of the PUF authentication scheme, uniqueness and unclonable properties of the PUF and considering that it takes less computational power and eliminates the need to store the security keys, in this research we will look at PUF as a lightweight authentication scheme for EDCs.

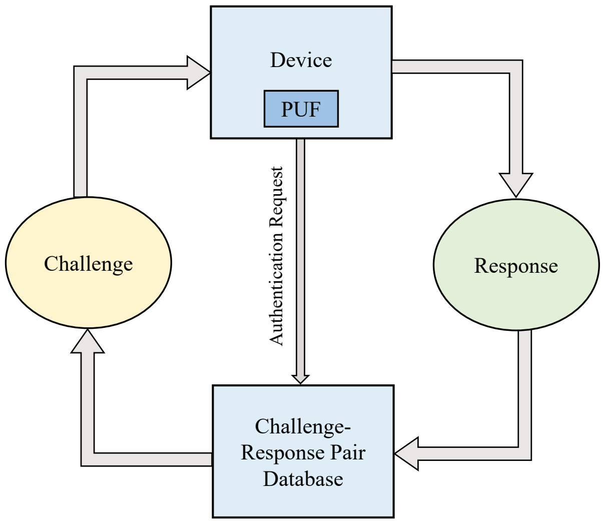


Figure 2: Process of PUF based Challenge and Response.

The rest of the paper is organized in the following manner. Section II summarizes the novel contributions of the current paper. A brief discussion of the related prior works in presented in Section III. Section IV has details of the proposed PUF based approach. Section V presents the experiments. The conclusions and future research directions are in Section VI.

II. Novel contributions of the current paper

A. The Proposed Solution

An authentication scheme needs to be lightweight, require low computational power, be robust, does not require large memory to store keys and should be able to protect against some of the external attacks like man-in-the-middle attack, replay attack, etc. Hence, we propose the following:

- A PUF-based authentication scheme for Load Balancing, where Virtual XORArbiter PUFs will be used by the cloud server to authenticate the EDCs
- A scheme for the EDCs to authenticate each other during load balancing
- XORArbiter PUFs can also be used to authenticate the user devices connected in the fog environment.

B. Novel contributions

The current paper which at developing a robust, secure and lightweight authentication scheme for EDCs using PUFs.

- Virtual XORArbiter PUF implementation and generation
- Verification of Edge Data Centers
- Using Virtual PUFs to authenticate EDCs
- EDCs authenticating each other during load balancing without cloud server
- A real-time testbed for the proposed security scheme

III. Related Prior Research

Some research works suggest using the Blockchain as a solution where public keys are stored in Ethereum and private keys are stored in the IoT devices [7]. A blockchain based authentication scheme for EDCs is presented in [8]. In contrast, we look at some of the earlier research that suggests PUF as the authentication scheme for Internet-of-Things (IoT) based applications, which includes the authentication of the IoT devices, like the Edge Data Centers (EDCs)), communication gateways, routers, modems, and the end-user devices involved as listed in Table I. One of the works proposes the use of a double PUF authentication model, to design an IP copyright anonymous authentication scheme [9]. Another research that is similar and uses multiple PUFs for authentication is suggested in [10]. This research proposes a lightweight PUF based authentication scheme for IoT devices based on multiple PUFs.A research based on an Elliptic Curve Qu-Vanstone (ECQV) certificate and DTLS standard protocol to establish authentication between resource-constrained IoT devices [11].

Table I: Comparative Table for State-of-the-Art Literature.

Research	Algorithm	Application
Long et al. [9]	Double PUF Authentication	IP protection in FPGA trade
Yoon et al. [10]	Multiple PUF Authentication	IoT device security
Ha et al. [11]	Elliptic curve cryptography based ECQV	Mutual authentication and key establishment between two resource constrained IoT devices
Zhang et al. [12]	PUF based Multi-Server Authentication	Cloud-Edge IoT using Blockchain
Current paper	XORArbiter PUF	Edge Data Center Authentication in Load Balancing

IV. The Proposed PUF based Scheme

A. XORArbiter PUF

The Arbiter PUF is classified as a delay-based PUF: the response is generated based on the timing difference in two functionally identical paths in an IC. The output of an Arbiter PUF is 1 bit. The electrical signals pass through multiple stages of multiplexers, determined by the challenge $C[i]$. The arbiter, which is a latch or flip-flop determines the fastest signal. The arbiter outputs '1' if the upper path is faster; otherwise it will output a '0'. An N-stage arbiter PUF can generate $2N$ hallenge Response Pairs (CRPs) [13]. The security of PUF is based on the difficulty in measuring the PUF internal parameters, reproducing and predicting the behavior. Despite the unpredictability of PUFs, they are not safe from Machine Learning attacks, also called software modelling attacks [14]. Hence, PUFs need some kind of postprocessing to introduce some nonlinearity to make the PUF stronger. In the case of an arbiter PUF whose response is linear, an XOR circuit is added at the end [15].

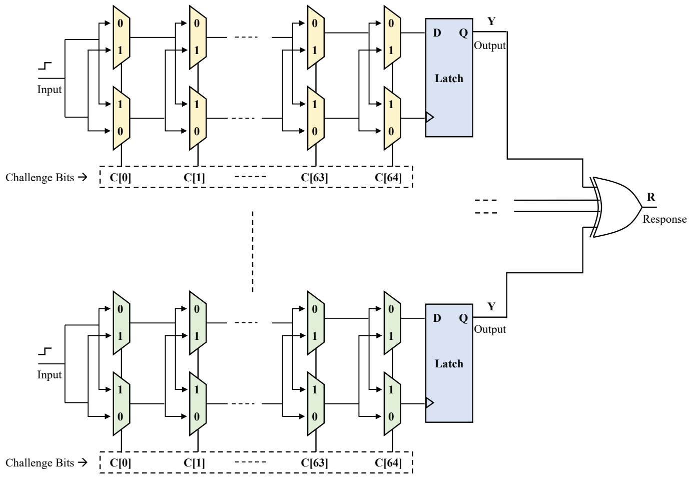


Figure 3: N Stage XORArbiter circuit with 64 Input Challenge bits and 1 bit Response.

An XORArbiter PUF has multiple bit input $C[i]$ and based on the delay a one bit output Y is generated. The circuit of an N stage XORArbiter PUF is shown in Fig. 3. The input controls the delay path by controlling the multiplexers. By adding an XOR at the end stage, a strong PUF can be created which cannot be remodeled.

B. *Proposed Authentication Scheme for EDCs*

The proposed EDC authentication scheme for load balancing is shown in Fig. 4. In the proposed scheme the cloud-based server initiates the authentication scheme by registering the Edge Data Centers, Edge Devices, and the End Devices. Whenever the participating devices request a data transfer, communication or processing the, cloud authenticates the EDC and the connected Edge devices and End devices. In the load balancing scenario where the task from one overloaded EDC needs to be offloaded to another available EDC, the EDC can initiate the authentication process itself and authenticate the EDC to which it wants to transfer the task.

C. *Implementation of Virtual PUFs*

To design and test the PUF based authentication scheme, we are considering the use of virtual PUFs. Virtual PUFs are generated using the software toolbox *pypuf*, which is used for simulation, testing and attacking PUFs. The PUF being used for the simulation is an XORArbiter PUF. The challenge length is chosen and a seed for reproducible results is provided. An N stage XORArbiter PUF circuit is simulated with 10 stages, a 64-bit challenge is given to each stage and the 1-bit response is received as the output. An XOR at the end of Arbiter makes the PUF stronger. Challenge-Response Pairs are stored in an SQLite3 database along with the EDCID which is unique.

Fig. 5 shows the Challenge-Response Pair(CRP) Database created and loaded with the EDCID, which is a unique ID for the Edge Data Centers, 64-bit Challenges that will be sent to the EDCs and 1-bit Responses that are expected for a given set of 64-bit Challenge. The virtual PUF is flexible enough to generate 128-bit challenges as well.

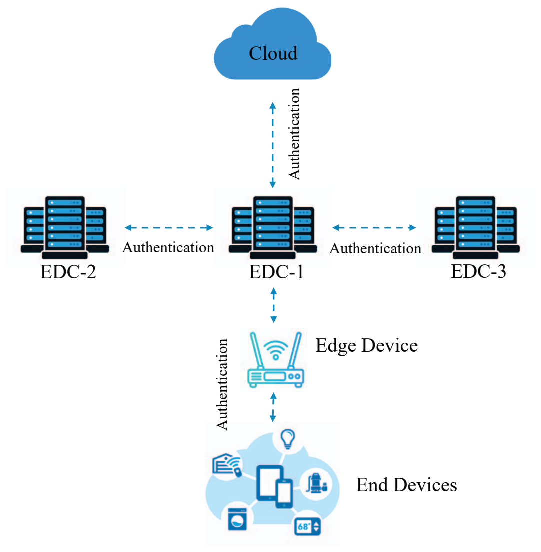
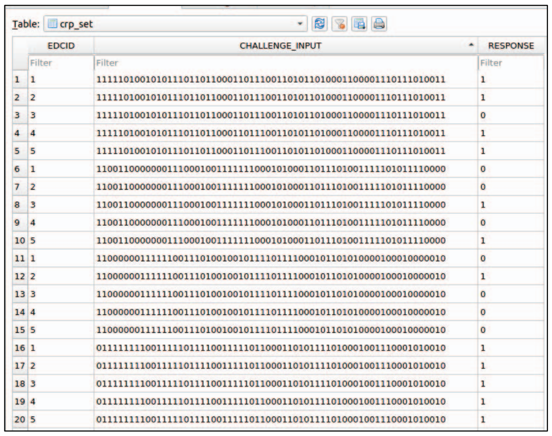


Figure 4: Authentication Scheme for Edge Data Centers.



	EDCID	CHALLENGE_INPUT	RESPONSE
	Filter	Filter	Filter
1	1	1111101001010111011011000110111001101011010001100001110111010011	1
2	2	1111101001010111011011000110111001101011010001100001110111010011	1
3	3	1111101001010111011011000110111001101011010001100001110111010011	0
4	4	1111101001010111011011000110111001101011010001100001110111010011	1
5	5	1111101001010111011011000110111001101011010001100001110111010011	1
6	1	1100110000000111000100111111100010100011011101001111101011110000	0
7	2	1100110000000111000100111111100010100011011101001111101011110000	0
8	3	1100110000000111000100111111100010100011011101001111101011110000	1
9	4	1100110000000111000100111111100010100011011101001111101011110000	0
10	5	1100110000000111000100111111100010100011011101001111101011110000	1
11	1	1100000011111100111010010010111101111000101101010000100010000010	0
12	2	1100000011111100111010010010111101111000101101010000100010000010	1
13	3	1100000011111100111010010010111101111000101101010000100010000010	0
14	4	1100000011111100111010010010111101111000101101010000100010000010	0
15	5	1100000011111100111010010010111101111000101101010000100010000010	0
16	1	0111111110011111011110011111011000110101111010001001110001010010	1
17	2	0111111110011111011110011111011000110101111010001001110001010010	1
18	3	0111111110011111011110011111011000110101111010001001110001010010	1
19	4	0111111110011111011110011111011000110101111010001001110001010010	1
20	5	0111111110011111011110011111011000110101111010001001110001010010	1

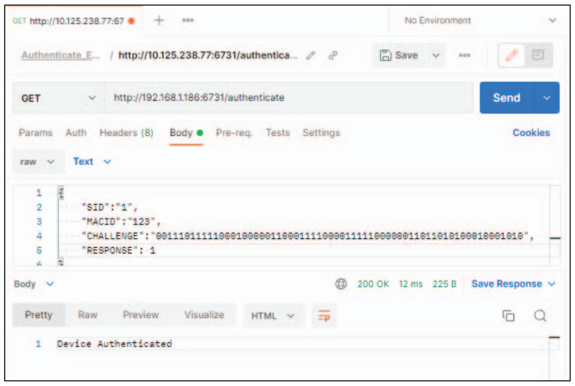
Figure 5: CRP database with 64-bit Challenges and 1-bit Responses.

D. *Verification and Authentication Scheme*

The EDC authentication scheme consists of two steps. In step 1, the EDC device sending a request to the server is verified. In step 2, upon device verification completion, the EDC is authenticated based on the PUF challenge-Response. The client side authentication request process is explained in Algorithm 1. When the server receives an authentication request from the client it will verify the device using the digital signature and authenticates the device if the PUF response to the given challenge matches the one in the CRP database. The authentication scheme is given in algorithm 2.

E. *EDC-1 Authenticating EDC-2 without Cloud*

In a collaborative edge computing environment, the EDCs requesting the service of other EDCs must go through cloud



```
GET http://192.168.1.186:6731/authenticate
```

```
{
    "SID":"1",
    "MACID":"123",
    "CHALLENGE":"0011101111100010000011000111100001111100000011011010100010001010",
    "RESPONSE": 1
}
```

```
200 OK   12 ms   225 B
```

```
Device Authenticated
```

Figure 6: Test of CRP where Edge Device is Authenticated.

server for authentication process. This computing paradigm is bound to longer latency periods. Hence new research works are trying to address this issue by offloading the cloud. Edge computing has similar processing capabilities like the cloud which can be utilized for offloading [16]. In dynamic load balancing the overloaded EDC must look for another EDC with less load and availability to offload its tasks. The EDC sends a load request to the neighboring EDCs, and they respond back by sending their current load status. The requesting EDC then selects the EDC with minimum load and begins the authentication and data transfer process [17].

In this paper we propose a scheme for authenticating the EDCs without cloud to reduce the latency and improve the security of the EDC. The PUF CRP dataset is locally stored in the EDC which often requires to offload. That way the EDC can send the challenges to the EDC that it needs to authenticate and begin offloading. In this scheme say EDC-1 has to offload its tasks to EDC-2. It sends a request to EDC-2, which will respond back with the payload encrypted with EDC-2's P_u (Public Key). EDC-1 decrypts the payload with its P_r (Private Key), once the EDC-2 is verified. It sends the 64 bit PUF Challenge, C_1, and receives the Response R_2 from EDC-2. If the response matches with the response in the Database the EDC-2 is authenticated and data transfer is initiated, otherwise the communication is closed, as explained in algorithm 3.

```
Algorithm 1: Algorithm for EDC Sending Authentication Request to Server
Input  : EDC (Client) Create Request String,
         Compute Hash, Create Digital Signature
Output: Send request payload to Server
Create authentication request string ;
Select random Challenge-Response Pair ;
Compute Hash ;
Create Private Key ;
Generate Digital Signature using the Hash and Private Key ;
Append the Digital signature to random CRP and create payload ;
Send request to the server ;
```

```
Algorithm 2: Algorithm for EDC Sending Authentication Request to Server
Input  : Receive Client Request with Payload
Output: Verify and Authenticate Client EDC
Client request recieved ;
Get MacID;
if $MacID_c = MacID_s$ then
    EDC is Identified;
else
    EDC is NOT Identified ;
    Close Connection ;
Get Digital Signature ;
Verify Digital Signature ;
Get PUF Response based on Challenge and EDCID ;
Verify Response ;
if $R_c = R_s$ then
    EDC is Authenticated ;
else
    EDC is NOT Authenticated ;
    Close Connection ;
/* Server Verifies the EDC based on
   MACID and EDIC, if verified
   Authenticates the EDC based on PUF
   Response to the Challenge          */
```

V. Experimental Results

A. Experimental Setup

The EDC authentication scheme is implemented using three Raspberry Pi4 boards, each set up as Server, Client1 (EDC-1) and Client2 (EDC-2). The PUF challenge Response Pairs generated from the pypuf package are stored in an SQLite3 database. The CRPs are unique for each EDC, identified by EDCID. For each EDC a set of 10 CRPs are generated and stored. The database consists of 100 CRPs for 10 different EDCs. The challenges are 64-bit and the responses are 1-bit. The database is considered as the CRP cluster, which serves the EDCs participating in the load balancing. The implementation is done in two steps, first the server authenticating the EDC-1 and then EDC-1 authenticating EDC-2 during Load balancing. The hardware setup is shown in Fig. 7.

B. Validation and Analysis

The CRP database is saved on the server; it sends the challenge to the EDC that is requesting authentication, and the EDC sends back the response to the server. When the response matches the response in the CRP database, the server will authenticate the EDC. If the response does not match then the EDC will not be authenticated. Request-payload consists of the EDC ID which is a Unique ID of every EDC, the Mac ID of the EDC-1, computed hash using the SHA256 algorithm, and

Algorithm 3: Algorithm for EDC-1 authenticating EDC-2

```
Input  : Receive EDC-2 Request with Payload
Output: Verify and Authenticate EDC-2
1  EDC-2 request recieved ;
2  Get MacID;
3  if $MacID_2 = MacID_1$ then
4      EDC-2 is Identified;
5  else
6      EDC-2 is NOT Identified ;
7      Close Connection ;
8  Get Digital Signature ;
9  Verify Digital Signature ;
10 Get PUF Response based on Challenge and EDCID ;
11 Verify Response ;
12 if $R_2 = R_1$ then
13     EDC-2 is Authenticated ;
14     Close Connection ;
15 else
16     EDC-2 is NOT Authenticated ;
   /* EDC-1 Verifies and Authenticates
      EDC-2 based on Digital Signature
      and PUF CRP data.                */
```

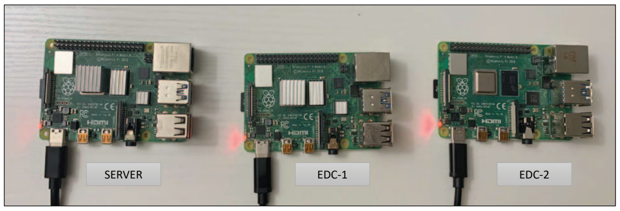
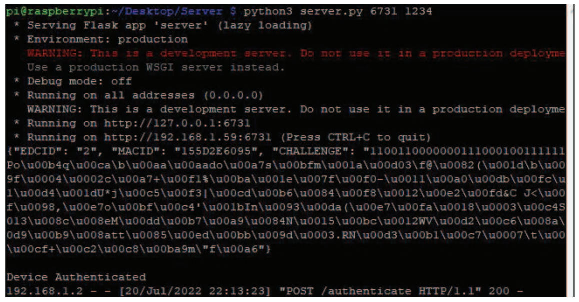


Figure 7: Hardware setup for Implementing Authentication Scheme.

the private key generated from the KDC (Key Data Center). A digital signature is created using the computed hash and the private key.

Once an authentication request is received, the server verifies the digital signature, checks if EDC ID and Mac ID match the IDs in the server database by decrypting the digital signature with the Public Key. Once the Digital signature is verified the PUF challenge and Response are verified and if the data matches the EDC-1 is authenticated. Fig. 8 shows the server authenticating EDC-1.

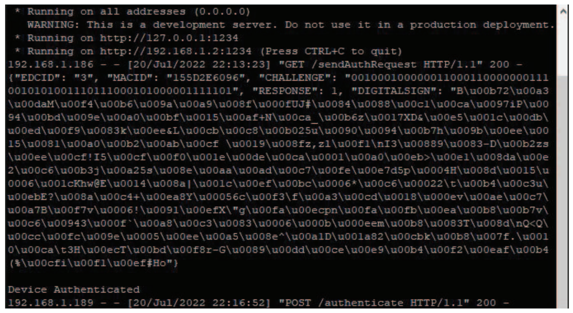
During load balancing the EDC must transfer tasks to the next available EDC with lower load. The load balancing scheme is not discussed in this paper. If EDC-1 has found an EDC-2 to transfer the load we go ahead with the authentication scheme, where EDC-1 will authenticate EDC-2 without involving the server. The CRP database in server holds the challenges and response pairs for each EDC in the network based on the EDC ID. These CRP sets are unique to each EDC. Assuming one EDC might have to connect and authenticate



```
pi@raspberrypi:~/Desktop/Server $ python3 server.py 6731 1234
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployme
 * Running on http://127.0.0.1:6731
 * Running on http://192.168.1.59:6731 (Press CTRL+C to quit)
{"EDCID": "2", "MACID": "155D2E6095", "CHALLENGE": "110011000000001110001001111111
Po\u00b4q\u00ca\b\u00aa\u00aado\u00a7s\u00bfm\u001a\u00d03\f@\u0082(\u001d\b\u00
9f\u0004\u0002c\u00a7+\u00f1%\u00ba\u001e\u007f\u00f0-\u0011\u00a0\u00db\u00fc\u
1\u00d4\u001dU*j\u00c5\u00f3|\u00cd\u00b6\u0084\u00f8\u0012\u00e2\u00fd&C J<\u00
f\u0098,\u00e7o\u00bf\u00c4'\u001bIn\u0093\u00da(\u00e7\u00fa\u0018\u0003\u00c4S
013\u008c\u008eM\u00dd\u00b7\u00a9\u0084N\u0015\u00bc\u0012WV\u00d2\u00c6\u008a\
0d9\u00b9\u008att\u0085\u00ed\u00bb\u009d\u0003.RN\u00d3\u00b1\u00c7\u0007\t\u00
\u00cf+\u00c2\u00c8\u00ba9m\"f\u00a6"}

Device Authenticated
192.168.1.2 - - [20/Jul/2022 22:13:23] "POST /authenticate HTTP/1.1" 200 -
```

Figure 8: Server Authenticating EDC-1.



```
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:1234
 * Running on http://192.168.1.2:1234 (Press CTRL+C to quit)
192.168.1.186 - - [20/Jul/2022 22:13:23] "GET /sendAuthRequest HTTP/1.1" 200 -
{"EDCID": "3", "MACID": "155D2E6096", "CHALLENGE": "0010001000000011000110000000111
001010100111011100010100000011111101", "RESPONSE": 1, "DIGITALSIGN": "B\u00b72\u00a3
\u00daM\u00f4\u00b6\u009a\u00a9\u008f\u000fUJ#\u0084\u0088\u00c1\u00ca\u0097iP\u00
94\u00bd\u009e\u00a0\u00bf\u0015\u00af+N\u00ca_\u00b6z\u0017XD&\u00e5\u001c\u00db\
u00ed\u00f9\u0083k\u00ee&L\u00cb\u00c8\u00b025u\u0090\u0094\u00b7h\u009b\u00ee\u00
15\u0081\u00a0\u00b2\u00ab\u00cf \u0019\u008fz,z1\u00f1\nI3\u00889\u0083-D\u00b2zs
\u00ee\u00cf!I5\u00cf\u00f0\u001e\u00de\u00ca\u0001\u00a0\u00eb>\u00e1\u008da\u00e
2\u00c6\u00b3j\u00a25s\u008e\u00aa\u00ad\u00c7\u00fe\u00e7d5p\u0004H\u008d\u0015\u
0006\u001cKhw@E\u0014\u008a|\u001c\u00ef\u00bc\u0006*\u00c6\u00022\t\u00b4\u00c3u\
u00ebE?\u008a\u00c4+\u00ea8Y\u00056c\u00f3\f\u00a3\u00cd\u0018\u000ev\u00ae\u00c7\
u00a7B\u00f7v\u0006!\u0091\u00efX\"g\u00fa\u00ecpn\u00fa\u00fb\u00ea\u00b8\u00b7v\
u00c6\u00943\u000f`\u00a8\u00c3\u0083\u0006\u000b\u000eem\u00b8\u0083T\u008d\nQ<Q\
u00cc\u00fc\u009e\u0005\u00ee\u00a5\u008e^\u00a1D\u001a82\u00cbk\u00b8\u007f.\u001
0\u00ca\t3H\u00ecT\u00bd\u00f8r-G\u0089\u00dd\u00ce\u00e9\u00b4\u00f2\u00eaf\u00b4
(%\u00cfi\u00f1\u00ef#Ho"}

Device Authenticated
192.168.1.189 - - [20/Jul/2022 22:16:52] "POST /authenticate HTTP/1.1" 200 -
```

Figure 9: EDC-1 Verifying and Authenticating EDC-2.

a neighboring EDC, we have stored the EDC-1 and EDC-2 CRP data in EDC-1 and EDC-2. At any point of time these EDCs will be able to authenticate each other.

Figure 9 shows the EDC-1 authenticating the EDC-2 by verifying the Digital signature and CRP data.

Load testing is done on the authentication process, by sending 100 requests to the server and the response times are observed. Fig. 10 shows the Response Time percentile of the server in handling 100 requests.

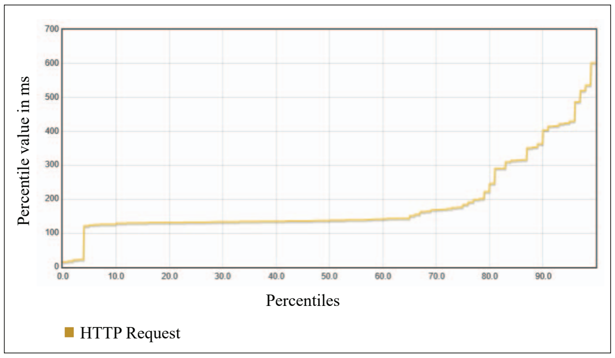


Figure 10: Server Response Time Percentiles for Authentication Requests.

Fig. 11 shows the number of requests effectively processed in less than 500ms. The requests that were not processed is relatively very low. There are not requests that have taken longer than 1,500ms.

From Table II we can see that the Virtual PUF based authentication system can achieve a uniqueness up to 46.84%.

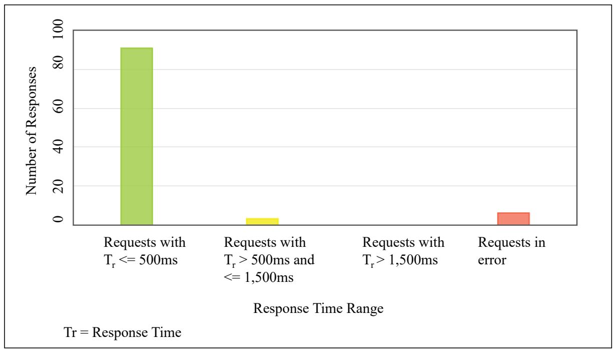


Figure 11: Server Response Time Overview.

Table II: Comparative Table for State-of-the-Art Literature.

Research	Algorithm	Hamming Distance	Randomness	Authentication Time
Long et al. [9]	Double PUF Authentication	46.84%	48.64%	NA
Zhang et al. [12]	PUF based Multi-Server Authentication	NA	NA	3302.9 ms
Current paper	XORArbiter PUF	44.86%	48.47%	< 1500 ms

Uniqueness is based on the average Hamming distance calculated for the generated PUF challenges and a randomness of 48.47% is achieved. This XORArbiter PUF based authentication scheme has much lower total authentication time, which includes receiving a request and authenticating the requesting device.

VI. Conclusion

Dynamic Load Balancing in a Collaborative Edge Computing environment needs a very secure authentication system as the EDCs will be transferring data to other EDCs for processing. Latency and bandwidth are the main concerns when an authentication scheme involves the cloud. To design a secure authentication system with low latency and high security for dynamic load balancing the proposed PUF based EDC authentication scheme proves to be lightweight, highly secure and with low latency, as the cloud is not involved for an EDC authenticating another EDC using the CRP clusters. An XORArbiter PUF is a strong PUF that adds more non-linearity to the response, thus making it safe against Machine Learning attacks and Power-side channel attacks. The load testing is done on the Server handling 100 requests. From the results it is seen that the authentication is faster, and the server can handle multiple requests and process them withing 0.5 second.

In future we intend to present comprehensive integrated cybersecurity framework for collaborative edge computing in the context of smart villages [18]. The smart villages applications need to have cybersecurity solutions with a very minimal overhead and energy-efficient which is motivated by the paradigm called Security-by-Design (SbD) [19].

References

[1] D. Puthal, M. Obaidat, P. Nanda, M. Prasad, S. Mohanty, and A. Zomaya, "Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing," *IEEE Communications Magazine*, vol. 56, pp. 60–65, 05 2018.

[2] M. Aazam and E.-N. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *Proc. International Conference on Future Internet of Things and Cloud*, 2014, pp. 464–470.

[3] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and Privacy in Fog Computing: Challenges," *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017.

[4] K. Sha, T. A. Yang, W. Wei, and S. Davari, "A survey of edge computing-based designs for IoT security," *Digital Communications and Networks*, vol. 6, no. 2, pp. 195–202, 2020.

[5] M. Yahuza, Y. Idris, A. Wahid, M. Alhaji Musa, and A. Garba, "A Lightweight Authentication Technique For Secure Communication Of Edge/Fog Data-Centers," *Science Proceedings Series*, vol. 2, pp. 76–81, 04 2020.

[6] V. P. Yanambaka, S. P. Mohanty, and E. Kougianos, "Making Use of Manufacturing Process Variations: A Dopingless Transistor Based-PUF for Hardware-Assisted Security," *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 2, pp. 285–294, 2018.

[7] T. M. Fernández-Caramés and P. Fraga-Lamas, "A Review on the Use of Blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32 979–33 001, 2018.

[8] D. Puthal, E. Damiani, and S. P. Mohanty, "Secure and Scalable Collaborative Edge Computing using Decision Tree," in *Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022, pp. 247–252.

[9] J. Long, W. Liang, K.-C. Li, D. Zhang, M. Tang, and H. Luo, "PUF-Based Anonymous Authentication Scheme for Hardware Devices and IPs in Edge Computing Environment," *IEEE Access*, vol. 7, pp. 124 785–124 796, 2019.

[10] S. Yoon, B. Kim, and Y. Kang, "Multiple PUF-based lightweight authentication method in the IoT," in *Proc. International Conference on Information and Communication Technology Convergence (ICTC)*, 2021, pp. 1198–1200.

[11] D. A. Ha, K. T. Nguyen, and J. K. Zao, "Efficient Authentication of Resource-Constrained IoT Devices Based on ECQV Implicit Certificates and Datagram Transport Layer Security Protocol," in *Proc. Seventh Symposium on Information and Communication Technology*, 2016, p. 173–179.

[12] Y. Zhang, B. Li, B. Liu, Y. Hu, and H. Zheng, "A Privacy-Aware PUFs-Based Multiserver Authentication Protocol in Cloud-Edge IoT Systems Using Blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 958–13 974, 2021.

[13] Z. He, W. Chen, L. Zhang, G. Chi, Q. Gao, and L. Harn, "A Highly Reliable Arbiter PUF With Improved Uniqueness in FPGA Implementation Using Bit-Self-Test," *IEEE Access*, vol. 8, pp. 181 751–181 762, 2020.

[14] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Proc. 44th Annual Design Automation Conference*, 2007, p. 9–14.

[15] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[17] M. Albin, N. E. I Deepu, and M. Maya, "Intelligent Edge Security with Dynamic Task Offloading in Fog Environment," in *Proc. International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 367–372.

[18] D. Puthal, S. P. Mohanty, S. Wilson, and U. Choppali, "Collaborative Edge Computing for Smart Villages [Energy and Security]," *IEEE Consumer Electronics Magazine*, vol. 10, no. 3, pp. 68–71, May 2021.

[19] S. P. Mohanty, "Secure IoT by Design, Keynote, 4th IFIP International Internet of Things Conference (IFIP-IoT), 2021, Amsterdam, Netherlands," 2021, last Accessed on 12 Nov 2022. [Online]. Available: http://www.smohanty.org/Presentations/2021/Mohanty_IFIP-IoT_2021_Keynote_SbD-IoT.pdf