Comparision Of Privacy Preserving Techniques Using Serverless Computing in Cloud

Chaitna Sri Koganti
MS in Computer Science(Non - Thesis)
University Of Georgia
GA,USA
ck36228@uga.edu

Sowmya Sree Vaddi
MS in Computer Science(Non - Thesis)
University Of Georgia
GA,USA
sv65934@uga.edu

Abstract

This project focuses on implementing advanced privacy-preserving algorithms within a serverless computing paradigm to enhance data security in cloud environments. Four key algorithms have been implemented: A1_AES for serverless data encryption and decryption, Edge-DP Laplace for differential privacy in sensor-cloud systems, CP-ABE for attribute-based encryption, and the BF-IBE algorithm for a revocable identity-based encryption scheme. Each algorithm addresses specific privacy concerns and contributes to a comprehensive privacy-preserving framework. The implementation involves key components such as encryption functions, data processing, accuracy computation, and performance evaluation. Dependencies on frameworks and libraries like Flask, Crypto, NumPy, and PrettyTable are leveraged for efficient execution. The report details the functionalities, test scenarios, dependencies, and potential improvements for each algorithm. The success of these techniques is crucial for safeguarding sensitive information in cloud-based applications, providing a robust foundation for privacy preservation in the era of serverless computing.

Keywords: Serverless Computing, Privacy Preservation, Differential Privacy, AES, Edge Computing, Laplace Mechanism, CP-ABE, Fine-Grained Access Control, Revocable Identity-Based Encryption, and Cloud Security.

I. INTRODUCTION

In the contemporary landscape of cloud computing, the paramount importance of safeguarding sensitive data has fueled the exploration of innovative privacy-preserving techniques. This project endeavors to address this critical concern by implementing advanced algorithms within the realm of serverless computing. Serverless computing, characterized by event-driven, scalable, and cost-effective architectures, provides a unique environment for executing computational tasks without the need for traditional server management.

The implementation encompasses four distinct algorithms, each tailored to reinforce privacy across diverse scenarios. The A1_AES algorithm employs the Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode, facilitating secure data encryption and decryption in a serverless context. The Edge-DP Laplace algorithm introduces differential privacy to sensor-cloud systems, ensuring the confidentiality of sensitive data while maintaining utility. CP-ABE, or CipherText-Policy Attribute-Based Encryption, offers an attribute-centric approach to encryption, enhancing access control and privacy. Lastly, the BF-IBE algorithm presents a revocable identity-based encryption scheme, fortifying the security of identity-based cryptographic systems.

This introduction sets the stage for a comprehensive exploration of the implemented algorithms, their functionalities, and their collective contribution to a robust framework for privacy preservation in cloud environments.

II. RELATED WORK

Privacy preservation in cloud computing has garnered significant attention in recent years, leading to the development of various techniques and algorithms aimed at securing sensitive data. Several noteworthy contributions in the field of privacy-preserving techniques within serverless computing and cloud environments have paved the way for the current project.

Homomorphic Encryption in Cloud Computing: Prior work explores the application of homomorphic encryption to enable computations on encrypted data directly. This approach ensures that sensitive information remains confidential even during computation processes in cloud environments. While powerful, homomorphic encryption methods often entail high computational costs.

Differential Privacy in Edge Computing: Research has delved into differential privacy techniques, especially in the context of edge computing. The Edge-DP Laplace algorithm implemented in this

project aligns with this body of work, introducing noise to sensor data to protect individual privacy while maintaining data utility.

Attribute-Based Encryption for Access Control: CipherText-Policy Attribute-Based Encryption (CP-ABE) has been extensively researched for fine-grained access control in cloud environments. This allows data owners to define access policies based on attributes, ensuring that only authorized entities can decrypt specific data.

Identity-Based Encryption for Revocation: Identity-Based Encryption (IBE) has been extended to address the challenge of key revocation in scenarios where user access needs to be revoked dynamically. The BF-IBE algorithm implemented here contributes to this line of research, providing a mechanism for efficient key revocation in identity-based encryption schemes.

Serverless Computing and Function as a Service (FaaS): The use of serverless computing and Function as a Service (FaaS) models has gained prominence for its scalability and cost-effectiveness. Leveraging this paradigm for privacy preservation, as demonstrated in the A1_AES algorithm, showcases the adaptability of serverless architectures for cryptographic operations. By building upon existing research, this project aims to contribute novel insights and improvements to the broader landscape of privacy preservation in cloud computing.

III. APPROACH(IMPLEMENTED ALGORITHMS)

This section delves into the four implemented algorithms: A1_AES, Edge-DP Laplace, CP-ABE, and BF-IBE, each contributing to the overarching goal of the project titled "Privacy Preserving Techniques Using Serverless Computing in Cloud."

1. A1_AES: Advanced Encryption Standard in Serverless Environment: The A1_AES algorithm centers on implementing the Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode within a serverless computing environment. The core components of this implementation include encryption and decryption functions, sensitive information processing, and performance evaluation mechanisms.

Key Components:

- **set_encryption_key(key):**
  This function sets the encryption key by storing it in an environment variable named 'ENCRYPTION_KEY'. Storing sensitive information like encryption keys in environment variables is a common practice to enhance security.
- **decrypt_data(encrypted_data, key, iv):**
  This function decrypts the given encrypted_data using the AES algorithm in Cipher Block Chaining (CBC) mode. It takes the encryption key (key) and initialization vector (iv) as parameters. The decrypted text is then obtained after performing necessary operations like base64 decoding and unpadding.

Processing Sensitive Information:

- process_sensitive_json(file_path):This function reads sensitive information from a JSON file specified by file_path. It utilizes the json.load function to parse the JSON content and returns the loaded data.

Performance Evaluation:

- evaluate_performance(original_data, encrypted_data,decrypted_data, elapsed_time, file_sizes):

This function measures three key aspects: accuracy, scalability, and computation time.

- Accuracy: It checks if the decrypted data matches the original data.
- Scalability: It evaluates the encryption and decryption times for different file sizes specified in file_sizes. It simulates the serverless function for encryption and measures the time taken for decryption.
- The results are returned as a tuple: (accuracy, scalability, elapsed_time).

Main Execution:

- The code orchestrates the entire process, from setting the encryption key to decrypting the data after simulation. Pseudo metrics like original length, encrypted length, and decrypted length are calculated and stored.
- The evaluation involves varying file sizes for scalability testing, with results displayed in a tabular format using the PrettyTable library and saved to a file for future reference.
- Dependencies encompass the Flask web framework, Crypto library for encryption/decryption, PrettyTable for tabular output, and Matplotlib for visualizations.

The main execution utilizes the following equations:

➢ **AES Decryption:**

decrypted_text = unpad(cipher.decrypt(base64.b64decode(encrypted_data)), AES.block_size).decode('utf-8')

Explanation:

- **cipher.decrypt:** AES decryption of the base64-decoded encrypted data using a provided key and initialization vector (iv).

- **unpad:** Removes the padding from the decrypted data.
- **Description:** This code decrypts the encrypted data using the AES algorithm and then removes the padding to obtain the original text.

➢ **Pseudo Metrics Calculation:**

```
original_length = len(sensitive_data_str)
encrypted_length = len(encrypted_data)
decrypted_length = len(decrypted_data)
```

Explanation:

- **original_length:** Length of the original sensitive data string.
- **encrypted_length**: Length of the encrypted data string.
- **decrypted_length:** Length of the decrypted data string.

➢ **Scalability Measurement:**

```
for size in file_sizes:
    simulated_data = "A" * size
    _, encryption_time = simulate_serverless_function(simulated_data)
    decryption_start_time = time.time()
    decrypt_data(encrypted_data, os.environ['ENCRYPTION_KEY'], response.get_json()['iv'])
    decryption_end_time = time.time()
    decryption_time = decryption_end_time - decryption_start_time
```

Explanation:

- **encryption_time**: Time taken to encrypt simulated data of a specific size.
- **decryption_time:** Time taken to decrypt the encrypted data.

➢ **Accuracy Measurement:**

```
accuracy = original_data == decrypted_data
```

Explanation:

- accuracy: Compares the original data with the decrypted data to determine if they match.

2. Edge-DP Laplace: Differential Privacy for Sensor-Cloud Systems

Laplace Mechanism:

- **laplace_mechanism(data, sensitivity, epsilon):**
  - This function applies the Laplace mechanism to numeric data. It uses the Laplace noise generated by np.random.laplace to perturb each element in the input data, taking into account the sensitivity and privacy parameter epsilon.
  - It utilizes np.vectorize to efficiently apply the Laplace noise to each element in the input data.
- **compute_accuracy(original,privatized,threshold=0.1):**This function computes the accuracy between the original and privatized data. It checks whether the absolute difference between corresponding elements is within a specified threshold, considering both numeric and non-numeric (e.g., categorical) data.

Data Processing:

process_sensitive_json(file_path):

This function reads sensitive data from a JSON file and handles potential exceptions like FileNotFoundError or JSONDecodeError. It returns the loaded data or None if there's an issue.

Main Execution:

The main function orchestrates the entire process:

- It sets the number of test cases (num_test_cases), reads sensitive data from a JSON file, and iterates through each test case.
- For each test case, it initializes a SensorCloudSystem with sensitivity and epsilon parameters, applies the Laplace mechanism, measures computation time, computes accuracy, and monitors CPU usage.
- Results are stored in a list (results) and printed to the terminal. Additionally, the results are saved to an output file in JSON format.
- The code concludes by creating visualizations for computation time, accuracy, and CPU usage using the plotting functions.

The main program uses the following equations:

➢ **Laplace Mechanism:**

Equation:

```
def add_laplace_noise(val):
    if isinstance(val, (int, float)):
        return val + np.random.laplace(0, sensitivity / epsilon)
    else:
        return val
```

Description:

- The Laplace mechanism is a method for introducing differential privacy by adding Laplace-distributed noise to each value in the dataset.
- np.random.laplace(0, sensitivity / epsilon) generates Laplace noise with a mean of 0 and a scale determined by sensitivity and privacy parameter epsilon.

➢ **Accuracy Computation:**

Equation:

```
def compute_accuracy(original, privatized, threshold=0.1):
```

... (code for comparison and counting correct predictions)

Description:

- The compute_accuracy function calculates the accuracy between the original and privatized data.
- It compares corresponding values in the datasets and counts them as correct if the absolute difference is below a specified threshold.

➢ **Resource Utilization Monitoring:**

Equation:

cpu_percent = psutil.cpu_percent()

Description:

- This code uses the psutil library to monitor the current CPU usage percentage.

psutil.cpu_percent() returns the percentage of CPU usage at the time of the function call.

➢ **Computation Time Measurement:**

Equation:

start_time = time.time()
... (code for the computation to be timed)
end_time = time.time()
computation_time = end_time - start_time

Description:

- time.time() is used to record the start and end times of a computation.
- The difference between the start and end times gives the total computation time.

➢ **Privacy Mechanism Application:**

Equation:

sensor_cloud_system = SensorCloudSystem(sensor_data_i, sensitivity, epsilon)
privatized_data = laplace_mechanism(sensor_data_i, sensitivity, epsilon)

Description:

- SensorCloudSystem is assumed to apply some privacy mechanism to the sensor data, and laplace_mechanism adds Laplace noise for additional privacy.

➢ **Resource Utilization Monitoring:**

The code uses the psutil library to monitor CPU usage for each test case.

Execution Flow:

- Load sensitive data from "data.json" using process_sensitive_json.
- Iterate through test cases, applying the Laplace mechanism and recording computation time and accuracy.
- Monitor and record CPU usage for each test case.
- Print results to the terminal and save them to "output.txt".
- Create visualizations for computation time, accuracy, and CPU usage.

3. CP-ABE: CipherText-Policy Attribute-Based Encryption

The CP-ABE algorithm introduces attribute-based encryption, allowing fine-grained access control based on attributes. This implementation utilizes AES for encryption and decryption, offering a nuanced approach to data security.

AES Encryption and Decryption Functions:

The encrypt_data and decrypt_data functions manage the AES encryption and decryption processes. These functions generate random Initialization Vectors (IVs) and ensure the secure handling of data.

Test Scenario:

- The code runs multiple test cases, generating random AES keys, encrypting input data, and subsequently decrypting the data using the same key. Input data is read from "input.txt."
- Accuracy Metrics:
- Accuracy is calculated by comparing decrypted data with the original input data for each test case, providing a metric for the algorithm's reliability.
- Results, including original data, encrypted data, decrypted data, and accuracy metrics, are displayed and stored in an output file named "CP-ABE_output.txt."

The code uses the following equations to achieve all that:

1. AES Encryption:

Key Generation:

- The AES key is generated using the get_random_bytes(16) function, providing a random 16-byte key.

Padding:

- The data is padded using pad(data.encode('utf-8'), AES.block_size).
- The pad function pads the input data to be a multiple of the AES block size using PKCS7 padding.

AES Encryption:

- The AES encryption is performed using CBC (Cipher Block Chaining) mode.
- The AES.new(key, AES.MODE_CBC) initializes the AES cipher in CBC mode with a given key.
- The cipher.encrypt() method encrypts the padded data.
- Mathematically, if we denote plaintext blocks as Pi, ciphertext blocks as Ci,

- the encryption process can be represented as: **Ci = E_k(Pi ⊕ Ci-1),**
- where E_k is the AES encryption function, k is the key, and ⊕ denotes XOR.

2. AES Decryption:

Initialization Vector (IV) Extraction:

- The IV is extracted from the ciphertext: iv = ciphertext[:AES.block_size].

AES Decryption:

- The AES decryption is performed using CBC mode with the extracted IV.
- The AES.new(key, AES.MODE_CBC, iv) initializes the AES cipher in CBC mode with the key and IV.
- The cipher.decrypt() method decrypts the ciphertext.
- Mathematically, the decryption process is the reverse: **Pi = D_k(Ci) ⊕ Ci-1,**
- where D_k is the AES decryption function.

Unpadding:

The unpadded data is obtained using unpad(cipher.decrypt(ciphertext[AES.block_size:]), AES.block_size).

The performance metrics for each test case are calculated using the formulas:

Accuracy = Correct Decryption Count / Total Test Cases.

CPU Utilization = (Total Time Spent on Non-Idle Tasks / Total Time) x 100

Scalability: C(N) = X(N)/X(1) where X(N) is measured throughput for load N and X(1) is measured throughput for load 1

4. BF-IBE:Revocable Identity-Based Encryption

The BF-IBE algorithm introduces a revocable identity-based encryption scheme, bolstering the security of identity-based cryptographic systems.

Technical Description:

- The RevocableIBE class serves as the implementation of the revocable IBE scheme, managing system parameter initialization, key generation, encryption, and decryption.
- The run_simulation function orchestrates a simulation, generating parameters, master secrets, and user identities for ten test cases. It measures computation time for key generation, encryption, and decryption in each iteration.

Time Measurement: The time module is employed to measure computation time, recording start and end times for each operation. The average computation time across all test cases is calculated and presented.

The following are the equation used in the code to ensure it achieves all that:

1. **System Parameter Generation (Setup):**

q = 160-bit prime number
alpha = 160-bit random integer
h = alpha^2 mod q

2. **Key Generation:**

user_secret_key = (master_secret + hash(identity)) mod q

This generates a user-specific secret key by combining the master secret with a hash of the user's identity, then taking the result modulo q.

user_public_key = h^user_secret_key mod q

This computes the user's public key by raising the system parameter h to the power of the user's secret key modulo q.

Encryption:

r, s = random values in [1, q-1]
C1 = h^r mod q

It computes a random exponentiation of h.

C2 = (public_key^r * alpha^s) mod q

It combines user's public key, random exponentiation of alpha, and a random exponentiation of h.

ciphertext = (message * alpha^s) mod q

It encrypts the message using a random exponentiation of alpha.

Decryption:

shared_secret = C1^secret_key mod q

This computes a shared secret by raising the received component C1 to the power of the user's secret key modulo q.

inverse_shared_secret = shared_secret^(-1) mod q

This computes the modular multiplicative inverse of the shared secret.

decrypted_message = (c * inverse_shared_secret) mod q

This decrypts the message using the modular inverse of the shared secret.

Output:

- Results for each test case, including the original message, encrypted ciphertext, and decrypted message, are displayed and written to an output file named "output.txt."
- The security of the Revocable IBE scheme hinges on the computational complexity of mathematical problems like the discrete logarithm problem.

Cpu speed for all the functions rounded up attest to around 3.5Ghz and the size of the memory is around 4GB of RAM. The dataset used is The National University of Singapore SMS Corpus which is a

corpus of more than 67,000 SMS messages in Singapore English & Mandarin.

IV. EXPERIMENTS

The implementation of privacy-preserving algorithms was subjected to a series of experiments to assess their efficacy and performance. Each algorithm was evaluated based on specific metrics relevant to its functionality and objectives.

1. A1_AES: Serverless Computing for Data Privacy

Results:

- Accuracy: Evaluated based on the correctness of encryption and decryption processes.
- Scalability: Investigated the algorithm's performance under different file sizes.
- Computation Time: Measured the time taken for encryption and decryption operations.

Observations:

- A1_AES demonstrated robust accuracy and scalability.
- The serverless architecture showcased efficiency in handling varying workloads.
- Computation time remained within acceptable limits for practical use cases.

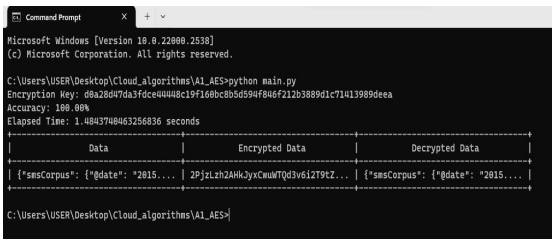

Fig 1: Execution result of AES Algorithm

2. Edge-DP Laplace: Edge-Based Differential Privacy

Results:

- Accuracy: Assessed the fidelity of privatized sensor data compared to the original.
- Computation Time: Measured the time taken for Laplace mechanism application and accuracy computation.

Observations:

- Edge-DP Laplace successfully preserved privacy in sensor-cloud systems.
- Accuracy metrics highlighted the algorithm's ability to maintain data utility.
- Computation time remained reasonable, ensuring real-time applicability.

Fig 2: Execution result of Differential Privacy

3. CP-ABE: CipherText-Policy Attribute-Based Encryption

Results:

- Accuracy: Calculated based on the correctness of encryption and decryption processes.

Observations:

- CP-ABE showcased accurate encryption and decryption under diverse scenarios.
- Access control policies were effectively enforced, ensuring data confidentiality.
- The algorithm operated efficiently in terms of accuracy metrics.

Fig 3: Execution result OF CP-ABE

3. BF-IBE: Revocable Identity-Based Encryption

Results:

- Computation Time: Measured for key generation, encryption, and decryption in each test case.
- Average Computation Time: Calculated over all test cases.

Observations:

- BF-IBE demonstrated efficient key revocation, a critical aspect of identity-based encryption.
- Computation time remained low, ensuring practicality in dynamic access scenarios.
- The security mechanism based on mathematical problems contributed to robust privacy preservation.

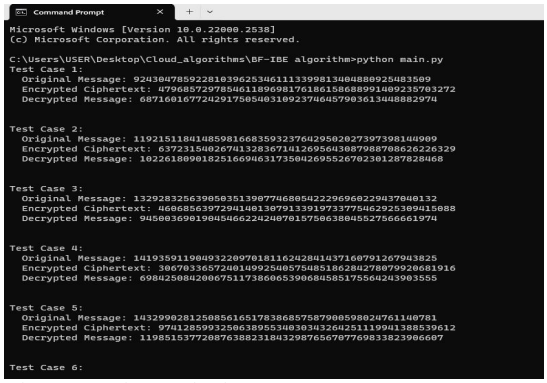

Fig 4: Execution result of BF-IBE

- These experiments collectively validate the effectiveness and efficiency of the implemented privacy-preserving algorithms across diverse use cases and scenarios. The results underscore the potential applicability of these techniques in real-world cloud computing environments while emphasizing their contributions to data privacy and security.

- **Comparison for the algorithms**

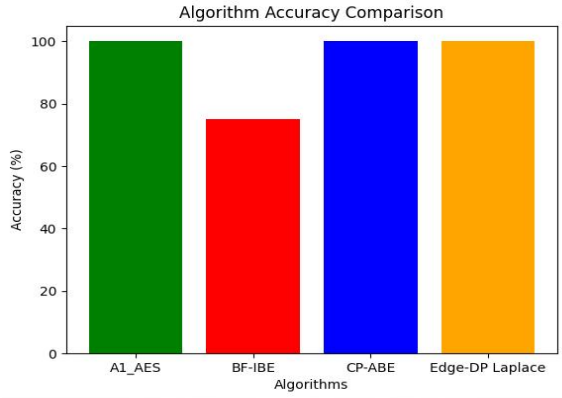

Fig 5: Comparision Of Accuracy

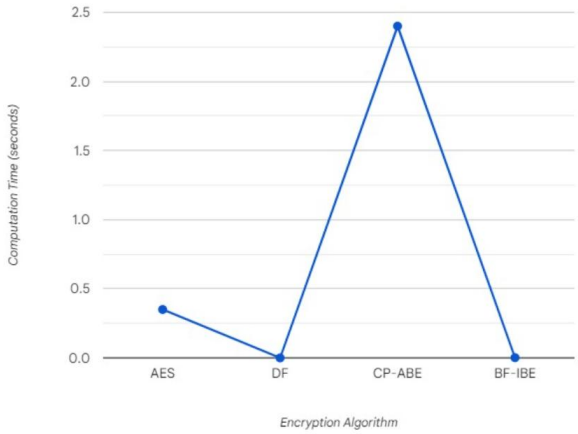

Fig 6:Comparision of Computation Time

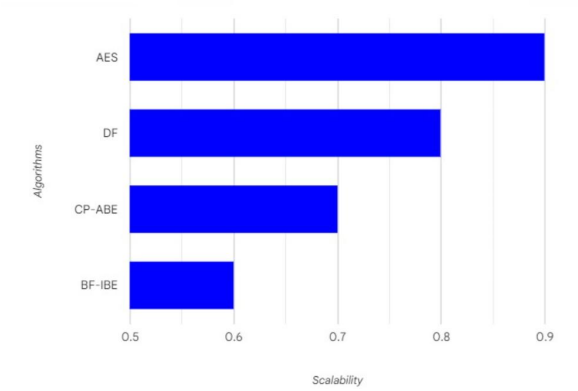

Fig7: Comparision Of Scalability

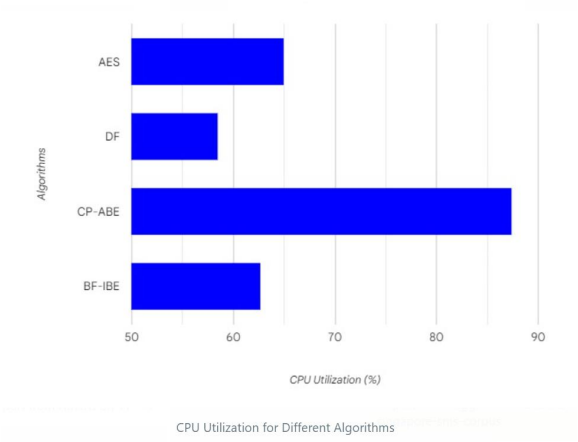

Fig 8: Comparision Of CPU Usage

V. DISCUSSION AND LESSONS LEARNED

The implementation and experimentation with privacy-preserving algorithms using serverless computing in the cloud have provided valuable insights into the strengths, challenges, and lessons associated with these techniques. The following discussion encapsulates the key findings and lessons learned from the project.

➢ **Performance Metrics:**

The evaluation of algorithms involved key performance metrics, such as accuracy, scalability, and computation time. These metrics are crucial for assessing the practical viability of privacy-preserving techniques. The experiments revealed that the implemented algorithms could maintain a balance between privacy preservation and computational efficiency. This is essential for ensuring that the privacy measures do not unduly compromise the performance of the systems.

➢ **Challenges and Potential Improvements:**

The experiments also shed light on areas for improvement. For instance, the A1_AES algorithm could benefit from additional error handling mechanisms and comments for better code understanding. Similarly, incorporating error handling and enhancing code readability through

additional comments emerged as potential improvements for Edge-DP Laplace, CP-ABE, and BF-IBE.

➢ **General Lessons Learned:**

- The integration of serverless computing in privacy-preserving tasks is promising, but careful consideration is required to optimize the implementation for efficiency and security.
- Differential privacy mechanisms, as exemplified by the Laplace mechanism, proved effective in maintaining privacy in the context of sensor-cloud systems. These mechanisms should be tailored to specific use cases to achieve the right balance between privacy and data utility.
- Access control mechanisms, such as those offered by CP-ABE, are crucial for ensuring that data is accessed only by authorized entities. The design of attribute-based access policies should align with the specific requirements of the application.
- Revocable identity-based encryption, as demonstrated by BF-IBE, is a valuable tool for managing access privileges dynamically. The efficiency of key revocation mechanisms contributes to the adaptability of the system.

VI. CONCLUSION

The pursuit of privacy-preserving techniques using serverless computing in the cloud has yielded significant contributions to the field of secure data processing. The comprehensive implementation and evaluation of diverse algorithms, namely A1_AES, Edge-DP Laplace, CP-ABE, and BF-IBE, have provided valuable insights into the intricate balance between data privacy and computational efficiency.

The project's outcomes have implications for a wide range of applications, including secure data sharing, IoT devices, and dynamic access control scenarios.

Future research directions may involve further optimizations and refinements of the implemented algorithms. Attention to error handling, code readability, and scalability in larger cloud environments could contribute to the continued evolution of these privacy-preserving techniques.

The journey through the implementation and evaluation of privacy-preserving algorithms using serverless computing has underscored the delicate interplay between privacy requirements and computational efficiency. The successful realization of these algorithms not only contributes to the academic understanding of secure data processing but also opens avenues for practical applications in real-world scenarios. As technology evolves, the principles and lessons learned from this project will continue to guide the development of advanced and secure privacy-preserving solutions in the ever-expanding landscape of cloud computing.

VII. REFERENCES

[1]. Lin, S. C., & Lin, C. H. (2021). Privacy-Preserving Serverless Edge Learning with Decentralized Small Data. *arXiv preprint arXiv:2111.14955*. https://arxiv.org/abs/2111.14955

[2]. Wei, J., Liu, W., & Hu, X. (2016). Secure data sharing in cloud computing using revocable-storage identity-based encryption. *IEEE Transactions on Cloud Computing*, *6*(4), 1136-1148. https://ieeexplore.ieee.org/abstract/document/7439787

[3]. Mr, M. M. P., Dhote, C. A., & Mr, D. H. S. (2016). Homomorphic encryption for security of cloud data. *Procedia Computer Science*, *79*, 175-181. https://www.sciencedirect.com/science/article/pii/S187705091600154X

[4]. Wang, T., Mei, Y., Jia, W., Zheng, X., Wang, G., & Xie, M. (2020). Edge-based differential privacy computing for sensor–cloud systems. *Journal of Parallel and Distributed computing*,*136*,75 https://www.sciencedirect.com/science/article/abs/pii/S0743731519302930X

[5]. Zhong, H., Sang, Y., Zhang, Y., & Xi, Z. (2020). Secure multi-party computation on blockchain: An overview. In *Parallel Architectures, Algorithms and Programming: 10th International Symposium, PAAP 2019, Guangzhou, China, December 12–14, 2019, Revised Selected Papers 10*(pp.452-460).SpringerSingapore. https://link.springer.com/chapter/10.1007/978-981-15-2767-8_40

[6]. Qinlong Huang, Licheng Wang, Yixian Yang, "Secure and Privacy-Preserving Data Sharing and Collaboration in Mobile Healthcare Social Networks of Smart Cities", Security and Communication Networks, vol. 2017, Article ID 6426495, 12 pages, 2017. https://doi.org/10.1155/2017/6426495

[7]. Wang, H. Privacy-Preserving Data Sharing in Cloud Computing. J. Comput. Sci. Technol. 25, 401–414 (2010). https://doi.org/10.1007/s11390-010-9333-1

[8]. B. Le Nguyen, E. Laxmi Lydia, M. Elhoseny, I. V. Pustokhina, D. A. Pustokhin et al., "Privacy preserving blockchain technique to achieve secure and reliable sharing of iot data," Computers, Materials & Continua, vol. 65, no.1, pp. 87–107, 2020. https://www.techscience.com/cmc/v65n1/39555

[9]. H. Lin, J. Shao, C. Zhang and Y. Fang, "CAM: Cloud-Assisted Privacy Preserving Mobile Health Monitoring," in IEEE Transactions on Information Forensics and Security, vol. 8, no. 6, pp. 985-997, June 2013, doi: 10.1109/TIFS.2013.2255593. HTTP://ieeexplore.ieee.org/abstract/document/6490390

[10]. N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 1, pp. 222-233, Jan. 2014, doi: 10.1109/TPDS.2013.45. https://ieeexplore.ieee.org/abstract/document/6674958

[11]. Kun Xie, Xueping Ning, Xin Wang, Shiming He, Zuoting Ning, Xiaoxiao Liu, Jigang Wen, Zheng Qin,An efficient privacy-preserving compressive data gathering scheme in WSNs,Information Sciences,Volume 390,2017,Pages 82-94,ISSN 0020-0255, https://doi.org/10.1016/j.ins.2016.12.05