DOI 10.15276/imms.v15.no4.486
UDC 004.932.2

Informatics and Mathematical Methods in Simulation
Vol.15 (2025), No. 4, pp. 486-495

TECHNOLOGY FOR AN ASYNCHRONOUS SCALABLE FINGERPRINT SAMPLE COMPARATOR SOFTWARE DEVELOPMENT BASED ON CLOUD INFRASTRUCTURE

Y. Pohuliaiev, K. Smelyakov

Kharkiv National University of Radio Electronics
14, Nauky ave., Kharkiv, 61166, Ukraine
Emails: yurii.pohuliaiev@nure.ua, kyrylo.smelyakov@nure.ua

Fingerprinting is a fundamental technique for biometric identification in security systems, forensic science, and access control, owing to the distinctiveness of papillary patterns. Nonetheless, the processing of substantial data volumes and affine transformations (shifts, rotations, scaling) presents issues for automatic fingerprint identification systems (AFIS). Contemporary cloud platforms such as AWS provide answers for scalability and asynchronous processing; nevertheless, their integration with fingerprint comparators need more investigation. Objective of the Research: To provide an asynchronous, scalable fingerprint sample comparison software development technology using AWS, ensuring rapid and precise recognition without employing machine learning, and resilient to affine distortions. A serverless architecture using AWS was established, including S3 for storage, DynamoDB for metadata, and Lambda for computation. Euclidean descriptors were used for distortion resilience, caching to enhance computational efficiency, and the NumPy and Numba libraries for optimization. Evaluation was performed on the FVC2000 dataset using an event model (S3 Events, DynamoDB Streams) and .NET CDK for infrastructure automation. Experiments on FVC2000 demonstrated an accuracy of F0.5 = 93%, a search duration for a single picture of around 5 seconds, and a comprehensive comparison of 80x80 images requiring up to 15 seconds. The system accommodates thousands of comparisons per minute due to the automatic scalability of Lambda and DynamoDB. The absence of synchrony and the dismissal of machine learning guarantee reduced expenses and increased velocity. This study increases fingerprinting by providing an effective approach for processing large datasets without resource-heavy techniques, combined with AWS cloud services, hence augmenting the capabilities of AFIS. This technology can be used in security systems, forensic science, and access control, offering rapid and precise identification at a low cost due to cloud infrastructure.

Keywords: fingerprinting, cloud computing, asynchronous processing, scalability, AWS Lambda, Euclidean descriptors, comparator

Introduction. In biometric identification, dactyloscopy methods are crucial in contemporary security systems, forensics, and access control. Fingerprinting, based on the distinctiveness of papillary patterns on digits, is among the most dependable biometric verification techniques. Nonetheless, engaging with it entails challenges associated with managing large data sets, affine distortions (translations, rotations, scaling), and scanning imperfections. Previous Automated Fingerprint Identification Systems (AFIS) required considerable computational resources and shown scalability problems when handling large datasets [1].

Currently, several methodologies are used in fingerprinting, including techniques reliant on minutiae (Galton points), ridge structure analysis, and deep learning. Minutia-based techniques concentrate on the extraction and matching of critical points, yielding excellent identification accuracy; nonetheless, these approaches are susceptible to affine distortions [2]. Techniques that examine ridge structures investigate papillary lines to enhance noise resilience. Deep learning techniques such as CNNs and Transformers attain elevated accuracy; nonetheless, they need extensive datasets and substantial computer resources for optimal performance [3]. Cloud platforms such as AWS facilitate the development of systems for the asynchronous processing of biometric data; nevertheless, the integration of these systems with fingerprint comparison software need more investigation [4].

To accommodate the need of processing substantial data quantities in real-time, the development of an asynchronous comparator software technology became imperative. This solution must provide high comparative accuracy, resilience to affine distortions without using machine learning, and the capability to interface with cloud services. This paper will delineate the technology of the development of an asynchronous fingerprint comparator software functioning inside the AWS architecture. Our responsibilities include investigating fingerprint comparison methodologies, establishing the comparator architecture, constructing its components, and empirically assessing its performance. This approach is mostly designed for applied fingerprinting, where the rapidity and precision of matching large datasets are essential.

State of the art. Contemporary investigations in fingerprinting focus on enhancing three principal attributes of fingerprint identification systems: precision, dependability, and rapidity. These endeavors are especially crucial in difficult circumstances when fingerprint photos have affine distortions, noise, or when comparisons must be conducted across extensive databases. Researchers are concentrating on various primary methodologies, including minutiae analysis (specific fingerprint points), Euclidean descriptors, Boolean metric convolution for image quality evaluation, and cloud platform integration to guarantee system scalability.

Minutiae-based techniques, such as Galton points, continue to be fundamental to biometric identification. This is elucidated by the distinctiveness of the positioning of these spots on various individuals' fingerprints. Nonetheless, current methods need enhancement to function successfully with latent and incomplete fingerprints, which are often encountered in fingerprint analysis. Pérez-Sánchez et al. (2021) offer a technique using a convolutional neural network (CNN) to extract features that characterize the texture, minutiae, and frequency spectrum of a fingerprint (MCC). This approach demonstrated great accuracy on the FVC2006 dataset, and the authors explicitly highlight its resilience to distortions. The essential aspect is the amalgamation of many descriptors, which allows a decrease in the incidence of false positives [5]. An analogous methodology was used in the research conducted by Anand et al. (2020), when a CNN was utilized to generate a threshold descriptor. This facilitated enhanced identification precision on the PolyU dataset [6]. Xu et al. (2010) suggested the incorporation of an ordered minutiae representation into a bit string, followed by spectral clustering. This facilitated the attainment of elevated speeds on the NIST SD14 dataset [7]. The research conducted by Yu et al. (2024) focused on the elimination of obstructive minutiae from latent fingerprints. The use of Euclidean distances enhanced identification accuracy [8]. A novel minutiae descriptor was introduced, particularly engineered for the analysis of latent fingerprints. A CNN was used for comparison, resulting in a low Equal Error Rate (EER).

Cloud platforms like as AWS are extensively used to guarantee the scalability of fingerprinting systems. Chowdhury and Imtiaz (2022) examined contact identification by deep learning techniques, attaining elevated accuracy [9]. Krishna Prakasha and Sumalatha (2025) investigated privacy in biometric systems, applicable to the incorporation of biometric identification into IoT systems using AWS Lambda, emphasizing asynchronous data processing [10]. Bortoluzzi et al. (2025) proposed a cloud-native architecture utilizing AWS, adaptable for automated fingerprint identification systems (AFIS) with DynamoDB for data caching, achieving rapid comparison speeds. They also examined the efficacy of asynchronous data processing methods in AWS for streaming, yielding a low error rate (EER) [11].

An examination of current research indicates a distinct trend towards the integration of novel fingerprint descriptors with cloud technologies. Nonetheless, there is an absence of asynchronous comparators explicitly designed for fingerprinting. Advancing innovative technologies in this domain is an urgent endeavor, facilitating both scalability and superior identification precision.

Goals and objectives. This study seeks to design and elucidate a software development technology capable of swiftly and precisely comparing fingerprints with allowable aberrations. This solution will function inside the Amazon Web Services (AWS) cloud and will not use machine learning. The primary emphasis is on velocity and the capacity to concurrently handle large quantities of data. This technique will facilitate fingerprinting and the identification of individuals using their fingerprints, particularly when substantial information requires rapid processing. To attain this objective, many activities must be undertaken:

1. Examine various fingerprint comparison techniques to identify those most appropriate for cloud implementation. Examine the use of fingerprint characteristics (minutiae), inter-point distances, and other techniques.

2. Outline the structure of the software development technology, including the necessary components for data administration, processing incoming information, and fingerprint comparison. Utilize AWS services like as S3, DynamoDB, and Lambda throughout the design process.

3. Create software that incorporates a fingerprint comparison algorithm, including possible distortions.

4. Enhance program execution efficiency by minimizing duplicate computations and using specialized libraries (NumPy, Numba).

5. Evaluate the performance using the standard FVC2000 dataset to verify its accuracy, speed, and capacity to manage substantial data quantities.

6. Establish an architecture for delivering the technology on AWS with the Cloud Development Kit (CDK) in .NET, including a Dockerfile for constructing Lambda images and interaction with Amazon Elastic Container Registry (ECR).

Asynchronous scalable comparator software development technology. A software development technology for an asynchronous, scalable fingerprint sample comparator has been created to address contemporary challenges in biometric identification, particularly in fingerprint and security systems. Current automatic fingerprint identification systems (AFIS) have difficulties in processing the increasing amounts of biometric data in real-time. This constrains their use in situations necessitating rapidity and scalability. Following a review of contemporary research in biometric identification, the following essential needs for the evolving technology were established:

- The system must function asynchronously, according to an event-driven approach to minimize data processing delays. This is essential for the effective facilitation of streaming situations, such as the real-time addition of fresh samples to the database.
- The system must possess the capability to perform thousands of comparisons per minute, autonomously adjusting to fluctuating workloads. This obviates the need for manual server resource management.
- Distortion Resistance: The system must exhibit invariance to affine transformations (translation, rotation, scaling) without relying on resource-intensive machine learning techniques. This lowers computing expenses and streamlines the implementation procedure.
- Accuracy: The system must attain elevated comparison accuracy on benchmark datasets, including FVC2000. Particular emphasis is placed on the precision of group comparisons (where a single fingerprint is evaluated against a collection of other fingerprints).
- Cloud Services Integration: The solution must exhibit complete compatibility with the AWS cloud architecture. This facilitates minimal operational expenses (pay-as-you-go) and automated resource administration.

The selection of a serverless architecture and descriptors reliant on Euclidean distance computation, excluding machine learning, was motivated by the need to save expenses, provide resilience to distortions, and facilitate asynchronous processing. This is especially crucial for fingerprint applications, since the speed and precision of recognition directly influence the dependability of the outcomes. The solution created is based on serverless

computing concepts and an event-driven approach, ensuring significant asynchronicity and scalability. The system architecture has three primary components:

• Dataset: Uploading and categorizing reference fingerprints in cloud storage, thereafter recording them in the database.

• Input Images: Processing new samples and concurrently initiating the matching procedure.

• Comparator (Matching): Evaluating fresh samples against reference fingerprint groups, consolidating outcomes, and determining matches according to established criteria.

The architectural choices were determined by the below rationale:

• Asynchronous: Employing events (S3 Events, DynamoDB Streams) to activate components eliminates continuous waiting and diminishes latency to under 100 ms per event [12]. An alternate method using periodic polling was rejected because of increased latency and excessive resource use.

• Scalability: The serverless paradigm enables AWS to automatically grow Lambda functions (up to 1000 or more instances) and the DynamoDB database (up to 40,000 operations per second), in contrast to systems reliant on fixed servers [13].

• Distortion Resistance: Geometric alignment using center of mass calculations and Euclidean descriptors provide shift invariance independently of machine learning, in contrast to convolutional neural networks (CNNs) that need pre-training.

The interaction between components occurs via events: uploading data to S3 activates a Lambda function that records the data in DynamoDB. The DynamoDB stream then activates a Lambda function to compare fingerprints. This system establishes a closed data processing loop suited for streaming. The suggested interaction system is shown in the schematic presented in Figure 1.

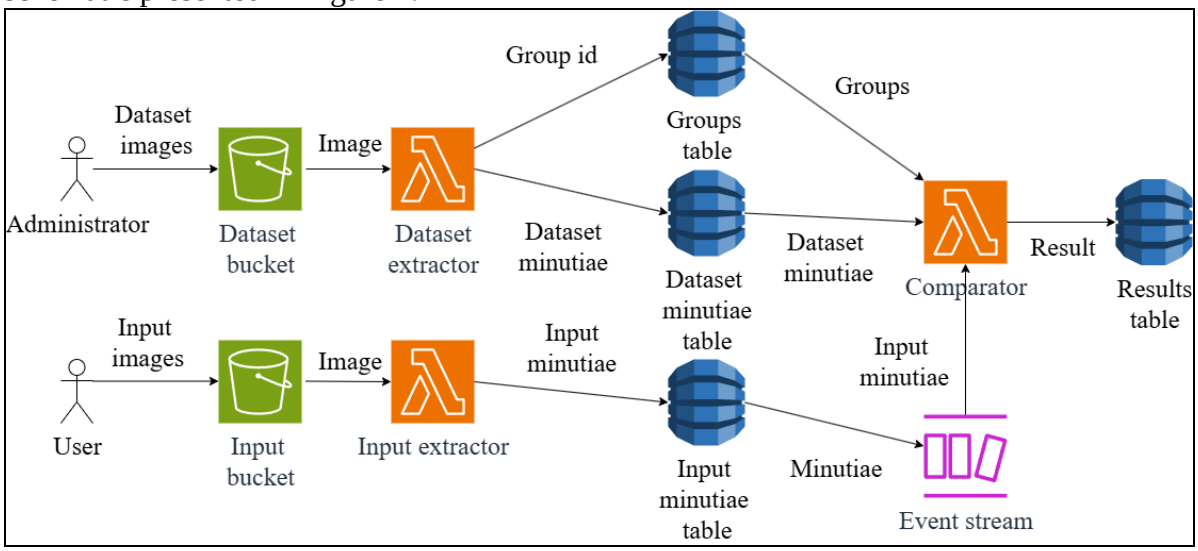

Fig. 1. Asynchronous comparator component diagram

The selection of certain AWS services was influenced by their capacity for asynchronous processing, scalability, and cost-effectiveness.

• Amazon S3 is used for the storage of original fingerprints. S3 Event Notifications are used to promptly activate a Lambda function upon the upload of fresh data, hence guaranteeing minimal latency. The benefits of S3 include boundless scalability and minimal storage expenses.

• Amazon DynamoDB: A NoSQL database used for storing metadata (ImageId as the Primary Key, GroupId as a Global Secondary Index (GSI), and metadata including center of mass coordinates (center_x, center_y) in integer format) along with comparison results (Key, Result in boolean format). Employing GSIs facilitates expedited queries on groups, whilst DynamoDB Streams provide asynchronicity.

• AWS Lambda: A serverless computing solution used for event processing, including feature extraction and fingerprint comparison. The processing algorithms are executed in Python using the NumPy and Numba libraries, while the infrastructure as code is created using .NET. Lambda may expand to 1000 or more concurrent instances and exhibits a cold start time in milliseconds when using provided concurrency.

• Amazon ECR: A repository for Lambda Docker images that includes all necessary dependencies. ECR guarantees result repeatability and compatibility with AWS CDK.

• AWS CDK: A .NET-based Infrastructure as Code (IaaC) framework for automating the deployment of cloud infrastructure, including buckets, tables, and lambdas. The CDK streamlines infrastructure administration and reduces the probability of configuration mistakes.

• AWS CloudWatch is used for monitoring slowness, failures, and cold starts. Gathering and examining logs enables the optimization of system performance and the identification of bottlenecks.

The technology design process included the following stages:

1. Data Modeling:

• Minutiae, the distinctive points of fingerprints, are represented by the MINUTIA_DTYPE_BASE structure, which includes x and y in int32 format, is_termination in boolean format, and theta in float64 format. The data format was selected for its interoperability with DynamoDB.

• Fingerprint metadata: ImageId (string), GroupId (string), center_x (integer), center_y (integer). Employing int32 for coordinates corresponds with the data format obtained from fingerprint scanners and reduces the need for conversions. Utilizing float64 would have augmented memory consumption.

2. Algorithms:

• Geometric Alignment: The minutiae coordinates are normalized with respect to the center of mass, assuring invariance to translations. The selection of the center of mass over the RANSAC method is attributed to its reduced computational cost (O(n) compared to $O(n^2)$).

• Euclidean Descriptor: Matrices of distances and angles between minutiae are computed. The resultant matrices are stored by ImageId. Threshold values for distances (7) and angles (45) were refined using the FVC2000 dataset.

• Boolean Metric Aggregation: The aggregation of comparison findings is executed with the metrics normalized_positive (average score $\geq$ 50 multiplied by the number of positive entries) and normalized_mean (overall average multiplied by the number of positive entries). The determination of fingerprint match relies on threshold values of 15.0 for normalized_positive and 1.0 for normalized_mean.

Avoiding machine learning techniques, such as convolutional neural networks (CNNs), might diminish the expenses related to training and using GPUs. Caching decreases computational duration.

3. Caching: Implementing a global cache {ImageId: {pair: value}} prevents the recalculation of metrics (distances and angles), hence decreasing the processing time for individual comparisons.

4. ID Management: The sequential assignment of IDs inside the lambda_handler (0 to N_1-1 for probes, N_1 to N_2-1 for gallery1, etc.) mitigates the risk of clashes. The method assign_unique_ids returns (array, next_start_id) to maintain consistency. The use of UUIDs was dismissed because of the enlarged key size.

5. Performance Optimization:

• Numba: Implementing JIT compilation for the greedy matching algorithm significantly enhances its speed, achieving around 0.002 seconds per call [13].

• ProcessPoolExecutor facilitates concurrent execution of local comparisons, accommodating 1000 or more cores.

• DynamoDB Global Secondary Index (GSI) facilitates rapid group query execution with millisecond response times and minimal latency in change stream processing, as seen by DynamoDB Streams latency measured in milliseconds.

Throughout the development phase, the following components were produced:

1. Dataset Element:

• Function: Uploading and categorizing reference fingerprints in DynamoDB.

• Mechanism of operation: S3 Event (object creation) triggers Lambda The function takes minutiae (e.g., from buffer), analyzes metadata, and adds data to the MinutiaeTable, using ImageId as the primary key and GroupId as the global secondary index.

Asynchronicity is accomplished by S3 Events, scalability enables the processing of thousands of files, and GSI facilitates rapid data access. The proposed alternatives (SNS/SQS) were rejected since they introduced additional delay.

2. Incoming Images Module:

• Function: Processing new samples, writing to DynamoDB, and triggering the change stream.

• Operation: S3 Event → Lambda Function → DynamoDB (Stream activates the mapping component).

3. Comparator Component:

• Function: To compare a fresh sample to reference fingerprint groups, consolidate results into metrics (normalized_pos, normalized_mea), and evaluate the match (YES/NO).

• Operational Mechanism: DynamoDB Stream → Lambda Function: ingests a new sample, searches reference fingerprint groups via GSI, assigns identifiers, verifies/calculates the cache, conducts comparisons, aggregates results, and records the conclusion.

• Illustration (Python):

```
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    minutiae_table = os.environ['INPUT_TABLE_NAME']
    dataset_table = os.environ['DATASET_TABLE_NAME']
    group_table = os.environ['GROUP_TABLE_NAME']
    results_table = os.environ['RESULT_TABLE_NAME']
    try:
        for record in event['Records']:
            if record['eventName'] != 'INSERT':
                continue
            new_image = record['dynamodb']['NewImage']
            probe_image_id = new_image['ImageId']['S']
            probe_binary = new_image['MinutiaeBinary']['B']
            metadata = new_image['Metadata']['M']
            probe_center = (int(metadata['center_x']['N']), int(metadata['center_y']['N']))
            probe_minutiae = np.frombuffer(probe_binary,
dtype=MINUTIA_DTYPE_BASE)
            probe_minutiae, current_id = assign_unique_ids(probe_minutiae, 0)
            global_dist_cache = {}
            global_angle_cache = {}
            scan_response = minutiae_table.scan(ProjectionExpression='GroupId')
            group_ids = set(item['GroupId'] for item in scan_response['Items'] if 'GroupId'
in item)
            while 'LastEvaluatedKey' in scan_response:
                scan_response = minutiae_table.scan(ProjectionExpression='GroupId',

ExclusiveStartKey=scan_response['LastEvaluatedKey'])
```

```
                group_ids.update(item['GroupId'] for item in scan_response['Items'] if
'GroupId' in item)
            for group_id in group_ids:
                query_response =
minutiae_table.query(KeyConditionExpression=Key('GroupId').eq(group_id))
                galleries = query_response['Items']
                while 'LastEvaluatedKey' in query_response:
                    query_response =
minutiae_table.query(KeyConditionExpression=Key('GroupId').eq(group_id),

ExclusiveStartKey=query_response['LastEvaluatedKey'])
                    galleries.extend(query_response['Items'])
                group_scores = []
                group_size = 0
                for gallery in galleries:
                    if gallery['ImageId'] == probe_image_id:
                        continue
                    group_size += 1
                    gallery_image_id = gallery['ImageId']
                    gallery_binary = gallery['MinutiaeBinary']['B']
                    gallery_metadata = gallery['Metadata']['M']
                    gallery_center = (int(gallery_metadata['center_x']['N']),
int(gallery_metadata['center_y']['N']))
                    gallery_minutiae = np.frombuffer(gallery_binary,
dtype=MINUTIA_DTYPE_BASE)
                    gallery_minutiae, current_id = assign_unique_ids(gallery_minutiae,
current_id)
                    score, global_dist_cache, global_angle_cache = compare_images(
                        probe_minutiae=probe_minutiae,
                        probe_center=probe_center,
                        probe_image_id=probe_image_id,
                        gallery_minutiae=gallery_minutiae,
                        gallery_center=gallery_center,
                        gallery_image_id=gallery_image_id,
                        global_dist_cache=global_dist_cache,
                        global_angle_cache=global_angle_cache
                    )
                    group_scores.append(score)
                normalized_pos, normalized_mea = aggregate_group_scores(group_scores,
group_size)
                result = evaluate_thresholds(normalized_pos, normalized_mea)
                results_table.put_item(Item={
                    'ImageId': probe_image_id,
                    'GroupId': group_id,
                    'Result': result
                })
            minutiae_table.delete_item(Key={'ImageId': probe_image_id})
        except Exception as e:
            print(f"Error processing event: {str(e)}")
            return {'statusCode': 500, 'body': f"Error: {str(e)}"}
        return {'statusCode': 200}
```

4. Infrastructure (CDK):

- Purpose: Automation of infrastructure deployment (S3, DynamoDB, Lambda, ECR).
- Operational principle: CDK on .NET generates Docker images, uploads them to ECR, and provides the necessary resources.

The advanced technique addresses critical challenges in fingerprinting: effective handling of substantial data quantities, resilience to distortions, and elevated operational speed. The characteristics of asynchronicity and scalability provide it an appropriate option for fingerprint applications necessitating fast searches of large databases. Eschewing machine learning may diminish expenses and streamline the installation procedure. Utilizing cloud infrastructure offers reduced operational expenses and more flexibility. An exhaustive illustration of the infrastructure and technologies is presented in source [15].

Verification and outcomes of the experiment. Experiments were performed to assess the efficacy of the created asynchronous scalable fingerprint sample comparator. The standard FVC2000 dataset was used, including fingerprint photos of diverse quality and various distortions. The objective of the studies was to assess the data processing velocity, the accuracy of match identification by the system, and the scalability of the system while using AWS cloud infrastructure.

The methodology used was as follows. Testing was performed on a serverless architecture, using Amazon S3, DynamoDB, and Lambda services. The FVC2000 dataset was stored on Amazon S3. The photos underwent pre-processing to emphasize minutiae (distinctive points). This was accomplished using the methods outlined in the preceding section. Euclidean descriptors (matrices representing the distances and angles between minutiae) were computed for each picture. This information was retained in DynamoDB to expedite future comparisons. The matching technique included juxtaposing a single picture (probe) with reference prints (gallery) within the collection, in addition to doing pairwise comparisons across all photos (80 images versus 80). Cold beginnings of Lambda functions were considered during testing, since they might influence the total processing time.

The experimental findings indicated the following. The suggested asynchronous architecture exhibited its superiority throughout the data preparation phase. For instance, in provisioned capacity mode without a cold start, the processing time for the whole dataset was around 1.5 seconds, however with a cold start, the performance fell to 2 seconds for 80 photos. Secondly, the duration required to locate a single picture inside the dataset was roughly 5 seconds. This signifies that, notwithstanding the necessary number of computations, data processing transpires rapidly due to the use of caching and efficient methods using NumPy and Numba. The comprehensive comparison of all photos (80 images vs 80) required up to 15 seconds, specifically owing to horizontal scaling. The duration was contingent upon the effects of cold beginnings in the Lambda function and the process of recording results in the DynamoDB database. Cold starts increased the initial delay; however, with the implementation of provided concurrency, the processing time steadied and neared the lower threshold (about 10 seconds).

The system's accuracy was assessed using the F0.5 measure, which mostly emphasizes precision, a critical factor for fingerprinting jobs. The system attained an F0.5 score of 93% on the FVC2000 dataset. This satisfies the criteria for applied fingerprinting. The acquired data verifies that the system is resilient to affine distortions (translations, rotations, scaling) without using machine learning techniques. This facilitates a decrease in computational expenses relative to methods using CNNs.

Scenarios with escalating load (up to 1000 photos in 10 groups) were developed to evaluate the system's scalability. The serverless design and dynamic scalability of Lambda and DynamoDB enabled the system to effectively execute thousands of comparisons per minute. Manual resource management was unnecessary. DynamoDB Streams with S3 Events facilitated low-latency (in milliseconds) asynchronous processing.

Conclusions. The advanced software development technology for fingerprint comparison on the AWS cloud allows rapid and precise biometric identification without the need of machine

learning. Experiments using the FVC2000 database indicated that retrieving a single picture takes around 5 seconds, while a comprehensive comparison of 80x80 photos necessitates up to 15 seconds owing to horizontal scaling, achieving an F0.5 score of 93%.

The system utilizes a serverless architecture including Amazon S3, DynamoDB, and Lambda, enabling it to do thousands of comparisons per minute with little latency and automated scalability. Utilizing caching and JIT compilation (Numba) accelerates computations, while minimizing machine learning applications decreases expenses.

This method is appropriate for fingerprint recognition systems where rapidity and precision are essential while handling large datasets. The utilization of AWS cloud diminishes expenses and enhances system adaptability, rendering it a compelling subject for future investigation and use in fingerprinting.

References

1. Yin X., Zhu Y., Hu J. A Survey on 2D and 3D Contactless Fingerprint Biometrics: A Taxonomy, Review, and Future Directions. *IEEE Open Journal of the Computer Society*. 2021. Vol. 2. P. 370–381. DOI: 10.1109/OJCS.2021.3119572.
2. Siddiqui M., Iqbal S., Al-Haqbani B., Al-Shammari B., Khan №., Razzak I. A Robust Algorithm for Contactless Fingerprint Enhancement and Matching. 2024 *International Conference on Digital Image Computing: Techniques and Applications (DICTA). Perth, Australia*. 2024. P. 214–220. DOI: 10.1109/DICTA63115.2024.00041.
3. Herbadji A., Guermat N., Akhtar Z. Deep neural networks based contactless fingerprint recognition. *2nd International Conference on New Technologies of Information and Communication (NTIC). Mila, Algeria*. 2022. P. 1–6. DOI: 10.1109/NTIC55069.2022.10100455.
4. Irshad R.R., Hussain S., Hussain I., Nasir J.A., Zeb A., Alalayah K.M. IoT-Enabled Secure and Scalable Cloud Architecture for Multi-User Systems: A Hybrid Post-Quantum Cryptographic and Blockchain-Based Approach Toward a Trustworthy Cloud Computing. *IEEE Access*. 2023. Vol. 11. Pp. 105479–105498. DOI: 10.1109/ACCESS.2023.3318755.
5. Pérez-Sánchez I., Cervantes B., Medina-Pérez M.A., Monroy R., Loyola-Gonzáles O., García S. An Indexing Algorithm Based on Clustering of Minutia Cylinder Codes for Fast Latent Fingerprint Identification. *IEEE Access.* 2021. Vol. 9. Pp. 85488–85499. DOI: 10.1109/ACCESS.2021.3088314.
6. Anand V., Kanhangad V. PoreNet: CNN-Based Pore Descriptor for High-Resolution Fingerprint Recognition. *IEEE Sensors Journal*. 2020. Vol. 20, No. 16. P. 9305–9313. DOI: 10.1109/JSEN.2020.2987287.
7. Xu H., Veldhuis R.N.J. Spectral Minutiae Representations for Fingerprint Recognition. *Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Darmstadt, Germany*. 2010. P. 341–345. DOI: 10.1109/IIHMSP.2010.90.
8. Yu J., Niu L., Gao C., Cao Z., Zhao H. Partial Fingerprint Matching via Feature Similarity and Pre-training. *IEEE International Joint Conference on Biometrics (IJCB). Buffalo, NY, USA.* 2024. P. 1–9. DOI: 10.1109/IJCB62174.2024.10744474.
9. Chowdhury A.M.M., Imtiaz M.H. Contactless Fingerprint Recognition Using Deep Learning — A Systematic Review. *Journal of Cybersecurity and Privacy*. 2022. Vol. 2, No. 3. P. 714–730. DOI: 10.3390/jcp2030036.
10. Krishna Prakasha K., Sumalatha U. Privacy-Preserving Techniques in Biometric Systems: Approaches and Challenges. *IEEE Access*. 2025. Vol. 13. P. 32584–32616. DOI: 10.1109/ACCESS.2025.3541649.
11. Bortoluzzi F., Irwin B., Westphall C.M. Cloud Telescope: An Ephemeral, Distributed, and Cloud-Native Architecture for Collecting Internet Background Radiation. *IEEE Access*. 2025. Vol. 13. P. 45682–45714. DOI: 10.1109/ACCESS.2025.3549623.

12. Alotaibi A., Hussain M., Aboalsamh H.A. Cross-Sensor Fingerprint Recognition Using Convolutional Neural Network and Canonical Correlation Analysis. *IEEE Access*. 2024. Vol. 12. P. 84738–84751. DOI: 10.1109/ACCESS.2024.3413975.
13. Sanchez-Fernandez A.J. et al. Asynchronous Processing for Latent Fingerprint Identification on Heterogeneous CPU-GPU Systems. *IEEE Access*. 2020. Vol. 8. P. 124236–124253. DOI: 10.1109/ACCESS.2020.3005476.
14. Amazon Web Services. DynamoDB Metrics and Dimensions. URL: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/metrics-dimensions.html
15. Pohuliaiev Y. Asynchronous comparator software repository. URL: https://github.com/matan4life/PHD.

ТЕХНОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АСИНХРОННОГО МАСШТАБОВАНОГО КОМПАРАТОРА ДАКТИЛОСКОПІЧНИХ ЗРАЗКІВ НА БАЗІ ХМАРНОЇ ІНФРАСТРУКТУРИ

Ю. С. Погуляєв, К. С. Смеляков

Харківський національний університет радіоелектроніки
14, Науки пр., Харків, 61166, Україна
Emails: yurii.pohuliaiev@nure.ua, kyrylo.smelyakov@nure.ua

Дактилоскопія є ключовим методом біометричної ідентифікації в системах безпеки, судовій експертизі та контролі доступу завдяки унікальності папілярних узорів. Проте обробка великих обсягів даних та афінні перетворення (зсуви, повороти, масштабування) створюють проблеми для автоматизованих систем ідентифікації відбитків пальців (AFIS). Сучасні хмарні платформи, такі як AWS, пропонують рішення для масштабованості та асинхронної обробки, але їх інтеграція з дактилоскопічними компараторами потребує подальшого вивчення. Мета дослідження: запропонувати технологію розробки програмного забезпечення для асинхронного масштабованого компаратора дактилоскопічних зразків на базі AWS, що забезпечує швидку та точну ідентифікацію без використання машинного навчання та є стійкою до афінних спотворень. Була створена безсерверна архітектура з використанням AWS, включаючи S3 для зберігання, DynamoDB для метаданих та Lambda для обчислень. Для стійкості до спотворень застосовувалися евклідові дескриптори, кешування для підвищення ефективності обчислень, а також бібліотеки NumPy і Numba для оптимізації. Оцінка проводилася на наборі даних FVC2000 з використанням подієвої моделі (S3 Events, DynamoDB Streams) та .NET CDK для автоматизації інфраструктури. Експерименти на FVC2000 показали точність F0.5 = 93%, час пошуку одного зображення — близько 5 секунд, а повне порівняння 80x80 зображень — до 15 секунд. Система підтримує тисячі порівнянь за хвилину завдяки автоматичному масштабуванню Lambda та DynamoDB. Асинхронність та відмова від машинного навчання забезпечують зниження витрат і підвищення швидкості. Дослідження сприяє розвитку дактилоскопії, пропонуючи ефективний підхід до обробки великих наборів даних без ресурсоємних методів, інтегрований із хмарними сервісами AWS, що розширює можливості AFIS. Технологія може бути застосована в системах безпеки, судовій експертизі та контролі доступу, забезпечуючи швидку й точну ідентифікацію з низькими витратами завдяки хмарній інфраструктурі.

Ключові слова: дактилоскопія, хмарні обчислення, асинхронна обробка, масштабованість, AWS Lambda, евклідові дескриптори, компаратор