Cloud-ID-Screen: Secure Fingerprint Data in the Cloud

Fahad Alsolami [1], Bayan Alzahrani [1], and Terrance Boult [2]

[1] Department of Information Technology, King Abdulaziz University , KSA,
Email: { fjalsolami, balzhrani0043}@ {kau,stu.kau}.edu.sa

[2] Department of Computer Science , University of Colorado at Colorado Springs, USA,
Email: tboult@uccs.edu

Abstract

*Biometric data plays an important role today as an identity authentication tool. However, designing an efficient and secure biometrics authentication scheme in the cloud environment remains a research challenge. To address the security issue of biometric data in the cloud, we propose Cloud-ID-Screen scheme. Cloud-ID-Screen utilizes low-cost cloud storage and scalable cloud computation to improve security and privacy, as well as process speed. Cloud-ID-Screen achieves this improvement by splitting fingerprint features into small subsets and distributing these small subsets into multiple clouds simultaneously, where no single cloud stores all subsets of a fingerprint. During the matching operation, Cloud-ID-Screen utilizes a parallel processing tool (*e.g. *Hadoop) to match each subset of the fingerprint features independent and in parallel to maintain security and improve process speed. Our experiments show that Cloud-ID-Screen achieves comparable accuracy in matching while being statistically significantly faster as compared to the baseline.*

1. Introduction

Cloud storage provides many benefits for organizations and individuals in terms of scalability, availability, and unbounded resources to leverage and process big data like biometric data, *e.g.* fingerprints. These features attract biometric systems to leverage the cloud computing/storage, not only for enhancing computation performance (*e.g.* speed up identification, verification and de-duplication processes of biometric system), but also for sharing biometric data among multiple individuals and organizations. In addition, the use of biometric data is growing rapidly in terms of multi-modalities uses, which require new systems for processing and storage [4]. For example, the U.S. Department of Homeland Security (DHS) is collecting biometric data (10 fingerprints and a photograph) from all passengers coming to the United State of America [5].

2018 IEEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA)
978-1-5386-2248-3/18/\$31.00 ©2018 IEEE

Despite the fact that cloud computation/storage provide a great variety of advantages, the security and privacy of data while in transit or during computation/storage is a main concern in the cloud [21] [23]. Biometric data itself is very sensitive data, and it is being targeted by different attacks such as intrinsic failures and adversary attacks [10] [13]. Moreover, doppelganger attacks and the biometric dilemma are also considered harmful to the security and privacy of a biometric system [10] [18]. However, there are many mechanisms used to protect biometric data from such attacks, whether in local or cloud storage. Template protection schemes have been proposed to address the security issue of biometrics technology [13]. These schemes are classified into four classes: salting [24], non-invertible transform [17], key-binding biometric cryptosystem [14] [16], and key-generating biometric cryptosystem [11] [25]. Even though these schemes provide a variety of solutions but still not resolve all the challenges. Some of these approaches store the biometric template in the unencrypted form, which makes this template vulnerable to different attacks [13]. In order to provide more privacy, other approaches use encryption method to encode the biometric data [6]. However, this encrypted data need to be decrypted first for the matching process, which exposes the system to unauthorized attempt [13] [19]. Moreover, there are some biometric schemes suffering from the performance and accuracy issue [8] [27], which may prevent many applications from using these approaches.

To address the security and privacy issues of biometric data, we propose the Cloud- ID-Screen scheme. Cloud-

ID-Screen is a system that is designed to store and match biometric data in multiple clouds/machines while maintaining accuracy, security, and privacy, as well as increasing the speed of processing. During the enrollment operation, Cloud-ID-Screen splits the fingerprint features (*i.e.* pair-table) into small subsets based on distances, then distributes these subsets of pair-tables into multiple clouds simultaneously. During the matching operation, Cloud-ID-Screen matches probe subsets of the fingerprint features against all gallery subsets of all fingerprint features in parallel, then returns the matching score. Cloud-ID-Screen utilizes low-cost cloud computations/storage and Hadoop (*e.g.* mapReduce) to distribute and process fingerprint features while enhancing security and privacy while speeding up the processing of the biometric data and operations.

The rest of this paper is organized as follows: in section 2, we briefly describe previous work related to the biometric data in the cloud environment. The objectives of Cloud-ID-Screen are given in section 3. Our proposed Cloud-ID-Screen algorithm is presented in section 4. In section 5, the description of the experimental design is given. The experimental evaluation and results are provided in section 6. Finally, the conclusion is drawn in section 7.

2. Related Work

The use of biometric data is growing rapidly in recent years and needs a new system for processing and storage. This rapid growth of biometric data, not only for its multi-modalities use, but also for its computation process in terms of de-duplication (N:N), identification (1:N), or verification (1:1). To manage this growth and high computations of biometric data, researchers designed cloud schemes for processing and storing biometric data in the cloud. First, Kohlwey *et al.* [22] discussed the next generation of biometric systems including the ability to store records, to implement business logic, and to support verification, identification, de-duplication, parallelism, flexibility, and redundancy. Then they discussed the underlying components such as Hadoop Distributed File System (HDFS) [2], Hadoop Map/Reduce [2] [12], HBase [3], and ZooKeeper [7]. Second, Shelly and Raghava [20] used Hadoop [2] for matching iris recognition. They used eight machines and conducted two iris recognition experiments as a sequential execution and a parallel execution. They found out that the parallel execution experiment achieves 66% improvement over the sequential execution experiment [20]. In sum, these two schemes can speed up the process of the biometric data computation. However, they do not address the basic security concerns of biometrics systems in cloud computing [20] [22].

3. Cloud-ID-Screen Objectives

In this section, we explore the Cloud-ID-Screen objectives in accuracy, security and privacy, as well as increased processing speed.

3.1. Accuracy Objective

A study of previous matcher algorithms, such as the NIST Bozorth fingerprint matcher [26] and Forest-Finger matching algorithm [9], was conducted to design a new algorithm, Cloud-ID-Screen, that maintains a comparable accuracy in matching. By studying both matcher algorithms, we found out that both matcher algorithms are constructing the intra-fingerprint pair-table from fingerprint features (*i.e.* minutiae points) based on distance, and the algorithms rely on distance while matching between probe and gallery images (*i.e.* pair-tables). In Cloud-ID-Screen algorithm, we split the fingerprint features into small subsets based on distance to maintain a comparable accuracy in matching between probe subsets of the pair-table and gallery subsets of the pair-table. In addition, we evaluate our algorithm by using well-known standard tools in a biometric system such as FAR (False Accept Rate) and GAR (Genuine Accept Rate) to ensure a meaningful measure of comparable matching accuracy is achieved for the Cloud- ID-Screen.

3.2. Security and Privacy Objectives

In the enrollment operation, Cloud-ID-Screen distributes multiple gallery subsets over multiple clouds/machines where no single cloud/machine stores the whole gallery subsets of the pair-table that belongs to one identity. Moreover, each subset of the pair-table is a part of the whole gallery pair-table features, and it is not given full information about the fingerprint features. Cloud- ID-Screen does not store the minutiae points of a fingerprint nor the fingerprint images which makes it difficult for an attacker to break the security of Cloud-ID-Screen. Even if an attacker holds an identical pair-table of fingerprint features to match it against all galleries to construct the match-table, our subsets pair-tables are distributed over multiple clouds, which makes it difficult for an attacker to know which subsets of pair-table should be matched. Moreover, insider attacks are less harmful for Cloud-ID-Screen in the sense that each cloud/machine only stores one subset of the gallery pair-table of fingerprint features, which is meaningless for an attacker to get. In sum, Cloud-ID-Screen provides protection to the fingerprint features in the cloud by preventing a meaningful search in the database by storing the fingerprint features of each person over multiple clouds/machines.

In the matching operation, Cloud-ID-Screen matches a probe subset of the pair-table against gallery pair-table subsets independently from other clouds/machines. Cloud-ID-Screen then collects match-tables from all clouds/machines in order to construct the consistent matching table, trees, and forests that can compute the final matching score. Cloud-ID-Screen does not collect all gallery pair-table subsets in order to perform matching. This independent matching ensures security and privacy for fingerprint data in the cloud.

3.3. Speed Improvement Objective

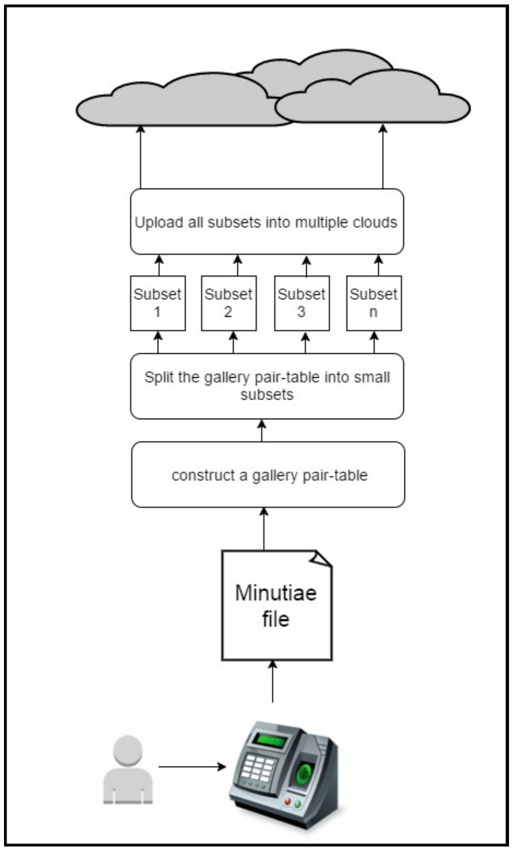
In this section, we examine the objective of increasing the Cloud-ID-Screens processing speed over the industry state-of-the-art speed. Cloud-ID-Screen utilizes Hadoop mapReduce [2] [26] to provide an increase in speed. The Hadoop mapReduce strategy is congruent with Cloud-ID-Screen algorithms because the mapper maps/matches probe subsets of the pair-table (fingerprint features) against gallery subsets of the pair-table in parallel, and constructs the match-table in each cloud/machine independently. Then the mapper gives the result of mapping/matching (match-tables) to the reducer to merge the match-tables from each cloud/machine. Once the reducer merges match-tables from related identities, it creates consistent match-tables in order to construct the trees. Finally, the reducer merges trees that have enough features to be matched and drops the other trees. As a result, the reducer constructs forests of trees in order to compute the maximum matching score. This mapping and reducing in parallel improves the speed up of Cloud-ID-Screen scheme.

4. Design of Cloud-ID-Screen Algorithm

In this section, we show the architecture of Cloud-ID-Screen for both the enrollment and matching operations. First we show how we split gallery fingerprint features into small subsets and save them as gallery pair-table. Second, we split the probe fingerprint features into small subsets and save them as a probe pair-table. Finally, we show how to match the probe pair-table against all gallery pair-tables and return the matching score.

4.1. Enrollment Operation

During the enrollment operation, Cloud-ID-Screen takes a gallery fingerprint image and creates the minutiae points in order to construct the pair-table based on the NIST Bozorth fingerprint matcher [26] and Forest-Finger matching algorithm [9]. Then, Cloud-ID-Screen splits the gallery pair-table of each fingerprint feature into smaller subsets based on distance where the distances are different in each subset. Cloud-ID-Screen splits the gallery pair-table without overlap between subsets to maintain security and privacy. Once all of these small subsets of the gallery pair-table are created, Cloud-ID-Screen distributes these gallery subsets of the pair-table into multiple clouds/machines where each cloud/machine stores one gallery subset of the pair-table of each fingerprint features.



Upload all subsets into multiple clouds
Subset 1
Subset 2
Subset 3
Subset n
Split the gallery pair-table into small subsets
construct a gallery pair-table
Minutiae file

Figure 1. Enrollment operation of Cloud-ID-Screen.

Cloud-ID-Screen scheme supports three modes of subset-splitting sizes for splitting each pair-table into smaller subsets: Eight-Subsets, Sixteen-Subsets, and Thirty-Two-Subsets. Figure 1 shows the enrollment operation of the Cloud-ID-Screen.

In the enrollment operation of Eight-Subsets mode, Cloud-ID-Screen splits the gallery pair-table (fingerprint features) into eight small pair-tables based on distance. The gallery pair-table consists of relative distance between two minutiae points and three relative angles [9] [26]. The data are stored in the gallery pair-table based on the relative distance ordered from the smallest to largest distance. Table 1 shows how the Cloud-ID-Screen splits the pair-table into small pair-tables based on distance for each mode.

AS shown in Table 1, Cloud-ID-Screen splits the gallery pair-table into eight subsets, where gallery subset-1 stores the small distance of the gallery pair-table while gallery subset-8 stores the largest distance of the gallery pair-table. In the other words, gallery pair-tables store pairs with increasing distance. Once all eight subsets of the pair-tables are created, Cloud-ID-Screen uploads these small eight subsets into the cloud storage where each cloud/machine stores only one subset of the pair-tables to maintain security and

Cloud-Id -Screen mode	Subsets	Distances	Clouds /Machines
8 Subsets	Subset-1	0-2000	Cloud-1 /Machine-1
	Subset-2	2000-4000	Cloud-2 /Machine-1
	Subset-8	Greater than 14000	Cloud-8 /Machine-1
16 Subsets	Subset-1	0-800	Cloud-1 /Machine-1
	Subset-2	800-1600	Cloud-1 /Machine-2
	Subset-16	Greater than 12000	Cloud-8 /Machine-2
32 Subsets	Subset-1	0-400	Cloud-1 /Machine-1
	Subset-2	400-800	Cloud-1 /Machine-2
	Subset-32	Greater than 12400	Cloud-8 /Machine-4

Table 1. Cloud-ID-Screen splits the pair-table based on distance into smaller subsets: Eight-Subsets, Sixteen-Subsets, and Thirty-Two-Subsets. In Eight-Subsets mode, the subset-1 starts from 0 to 2000, subset-2 starts from 2000 to 4000 and so on and so forth until the last subset where subset-8 greater than 14000. While in Sixteen-Subsets mode, the subset-1 starts from 0 to 800, subset-2 starts from 800 to 1600 and so on and so forth until the last subset where subset-16 greater than 12000. Similarly in Thirty-Two-Subsets mode, the subset-1 starts from 0 to 400, subset-2 starts from 400 to 800 and so on and so forth until the last subset where subset-32 greater than 12400.

privacy. In addition, Cloud-ID-Screen follows the same technique as in Eight-Subsets Mode but for Sixteen and Thirty-Two Subsets.

4.2. Matching Operation

During the matching operation, Cloud-ID-Screen follows the same steps as in the enrollment operation in order to create the probe pair-tables. Once all small subsets of the probe pair-tables are created, Cloud-ID-Screen matches these probe subsets of the pair-table against gallery subsets of the pair-tables that already stored in the clouds. Figure 2 and and Algorithm 1 shows the matching operation of Cloud-ID-Screen.

In the matching operation of Eight-Subsets mode, Cloud-ID-Screen splits the pair-table into eight smaller subsets in order to create the probe pair-tables for Eight-Subsets mode. Once all eight subsets of the probe pair-tables are

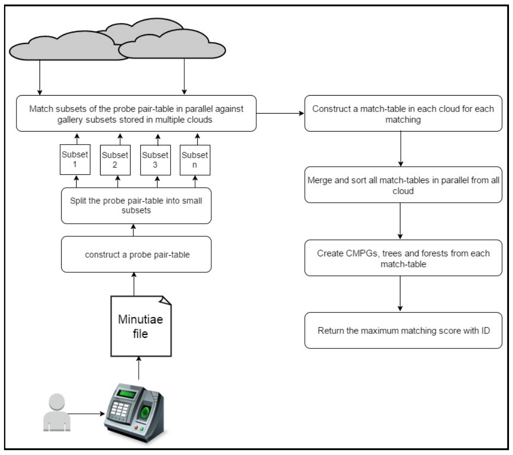

Match subsets of the probe pair-table in parallel against gallery subsets stored in multiple clouds
Construct a match-table in each cloud for each matching
Subset 1
Subset 2
Subset 3
Subset n
Merge and sort all match-tables in parallel from all cloud
Split the probe pair-table into small subsets
construct a probe pair-table
Create CMPGs, trees and forests from each match-table
Return the maximum matching score with ID
Minutiae file

Figure 2. Matching operation of Cloud-ID-Screen.

created, Cloud-ID-Screen performs matching process (mapping and reducing) between probe subsets against gallery subsets in order to compute the matching score.

In the mapping stage, Cloud-ID-Screen matches these probe subsets of the pair-tables against gallery subsets of the pair-tables that are already stored in the clouds. Cloud-ID-Screen matches probe subsets against gallery subsets where distances are similar. In cloud-1/machine-1, probe subset-1 matches against all gallery subset-1, and so on for all subsets in all clouds/machines. All theses matches happen in parallel in order to construct the match-table for all fingerprint features in each cloud/machine. Then, in Cloud-1/machine-1, Cloud-ID-Screen constructs match-table-1, where match-table-1 is a result of matching between probe subset-1 against gallery subset-1 and so on for other probe/gallery subsets in all clouds/machines.

In the reducing stage, the mappers give the reducers all of the match-tables that were already constructed in mapping stage at different clouds/machines. Then, Cloud-ID-Screen merges match-tables from related identities across neighboring clouds/machines.Therefore, Cloud-ID-Screen constructs consistent minutiae pair group tables (CMPG) from the given match-tables in order to create link between rows to construct trees and form forest.

Finally, The Cloud-ID-Screen computes the match score and return only the maximum matching score along with its ID. In addition, Cloud-ID-Screen follows the same technique as in Eight-Subsets Mode but for Sixteen and Thirty-Two Subsets.

5. Experimental Design

The main goal of this experiment is to evaluate the accuracy and increased speed of the Cloud-ID-Screen. The experiments compare two systems: the baseline (Forest-Finger matching algorithm [9]) and Cloud-ID-Screen. We

```
Data: Probe fingerprint image $p_i$ where i=1,2,...,n
Result: matching score
for ( each probe fingerprint image $p_i$ ) {
    extract minutiae points $m_i$ from fingerprint image $g_i$ ;
    compute minutiae file $mf_i$ from minutiae points $m_i$ ;
    each minutiae point in the minutiae file $mf_i$ has its location (x and y) , $\theta$ angle and quality ;
    construct the pair-table $t_i$ from the minutiae file $mf_i$ ;
    each pair of minutiae points (k and j) in the pair-table $t_i$ has its relative distance and angles ( $beta1$, $beta2$ and $\theta$) ;
    split the probe pair-table $t_i$ into subsets $r_i$ based on distance;
for ( all subsets $r_i$ of the probe pair-table $t_i$ ) {
    map/match each probe subset $r_i$ in parallel against gallery subsets $r_i$ stored in multiple clouds/machines ;
for ( each mapping/matching in each cloud/machine $s_i$ ) {
    construct match-tables $mt_i$ for each subset $r_i$ in each cloud/machine $s_i$ ;
    each row in the match-table $mt_i$ contains two probe minutia points, two gallery minutia points and $\Delta(\theta$(Probe), $\theta$(Gallery)) ;
    merge all match-tables $mt_i$ from related identites across different clouds/ machines $s_i$ ;
for ( each match-table $mt_i$ ) {
    for ( each row in match-table $mt_i$ ) {
        find inconsistent entry (row) to create a new Consistent Minutiae Pair Group CMPGs;
        inconsistent entry (row) is computed where prope minutiae point connot match more than one gallery minutiae point ;
    for ( each row in CMPGs ) {
        find all pairs that create link between rows to construct trees $st_i$ ;
form forests $ft_i$ from trees $st_i$ to calculate matching score $ms_i$ ;
for ( all matching scores $ms_i$ ) {
    return only the maximum matching score along with probe image ID and gallery image ID ;
```

Algorithm 1: Algorithm of matching operation of Cloud-ID-Screen.

conducted the experiment using Hadoop mapReduce [2] [12]. In the experiment, we constructed the pair-tables from fingerprint images and saved them in text files (Cloud-ID-Screen ((Eight-Subsets) pair-tables, Cloud-ID-Screen (Sixteen-Subsets) pair-tables, and Cloud-ID-Screen (Thirty-Two-Subsets) pair-tables), for both gallery and probe fingerprint images. During the enrollment operation, we uploaded the gallery pair-tables to Amazon S3. We used eight cloud storage centers of Amazon S3 public cloud storage (N. Virginia, Oregon, N. California, Ireland, Singapore, Tokyo, Sydney, and Sao Paulo). During the matching operation, we used Amazon Elastic MapReduce (EMR) [1] to match the probe pair-table of fingerprint features against all gallery pair-table in parallel. Then, EMR reduces the results of matching and returned the matching score with the fingerprint ID. we applied our matching algorithm to a widely known dataset ($FVC2002Db2\_a$) [15]. This dataset has 8 impressions for 100 subjects, which means there are 800 fingerprint images. The results of these experiments are the average of at least twenty runs. We implemented the experiment in C and Python. We wrote the mapper and reducer in C and ran it on Amazon EMR. For Internet connection, we used Comcast home Internet, with a measured upload bandwidth of 11.65 Mbps and a download bandwidth of 37.12 Mbps.

5.1. Baseline Setup

We used Forest-Finger matching algorithm [9] as our baseline. During the enrollment operation, we constructed the minutiae points from fingerprint images in order to construct the pair-tables. Then, we saved these gallery pair-tables in files.

During the matching operation, we constructed the minutiae points from fingerprint images in order to construct the probe pair-tables. Then, we saved these probe pair-tables in files. We then used the Linux pipe command line to match the probe and gallery fingerprint images along with the score of matching. Finally, we obtained the result of matching as the ID of probe and gallery fingerprint images along with the score of matching.

5.2. Cloud-ID-Screen Setup

During the enrollment operation, we split each gallery pair-table (fingerprint features) into smaller pair-tables based on distance start from subset-1 to subset-n (where n=8 in Eight-Subset mode, n=16, n=32). Then, we created a gallery n-subsets of pair-table where each subset is saved in text file format to be congruent with Hadoop file system. Then, we uploaded these gallery n-subsets of pair-table into Amazon S3 where each cloud/machine stores one subsets. During the matching operation, we split each probe pair-tables (fingerprint features) into n-subsets of pair-tables based on distance. We used the Amazon EMR [1] for matching. The mapper matched probe subset-1 against gallery subset-1 in cloud-1/machine-1 and so on for all subsets in all cloud/machines. Then, the mapper gave the reducer all of the match-table that were already collected from all of the clouds/machines, one match-table for each ID. The reducer collected all of the match-tables together to construct one match-table per ID. Finally, the reducer constructed the consistent matching table, trees, and forests of trees (we follow Forest-Finger matching algorithm [9]) in order to compute the matching score along with the matching gallery ID.

6. Experimental Evaluation

In this experiment, we seek to show that if we split the pair-table of the fingerprint features into small subsets and distribute them over multiple clouds/machines, we can increase the matching speed while maintaining comparable accuracy. We will evaluate to determine if the Cloud-ID-Screen achieves its goal in faster processing and security. Firstly, we analyze if the Cloud-ID-Screen achieves its goal in accuracy by testing GAR and FAR, then comparing it to state-of-the-art baseline [9]. Secondly, we analyze if Cloud-ID-Screen achieves its goal in increased speed by measuring the total time, including mapping and reducing time, for the matching operation and comparing it to state-of-the-art baseline [9]. Finally, we draw our conclusion if our experiment results support our hypothesis claim by using formal statistics (T-Test P-Value).

6.1. Accuracy Evaluation

In the accuracy evaluation, we evaluated Cloud-ID-Screen against the state-of-the-art baseline (Forest-Finger matching algorithm [9]). We ran our experiment after splitting the pair-table, then we evaluated the FAR (False Accept Rate) and GAR (Genuine Accept Rate) on the data before we measure the performance, to make sure that we have a comparable accuracy with baseline [9]. In this experiment, we found that the Cloud-ID-Screen received promising results in Eight-Subsets and Sixteen-Subsets while it received comparable results in Thirty-Two-Subsets, Table 2 and Figure 3 show the details of this results.

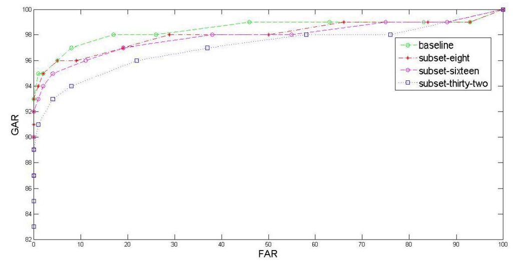


Figure 3. ROC curve comparing Forest-Fingers algorithm to Cloud-ID Screen algorithm (subset-eight, subset-sixteen, and subset-thirty-two).

These promising results for Cloud-ID-Screen show that splitting the pair-table of the fingerprint features into small pair-tables would give a promising accuracy in Eight-Subset and Sixteen-Subset comparing to the baseline [9]. However, Cloud-ID-Screen has low accuracy in Thirty-Two-Subsets and this because we split the pair-table into thirty-two tables, which reduces the match features that depend on the threshold in match algorithm. In the other words, any pair-table with less features than the threshold would not be matched.

	Baseline	8-Subsets	16-Subsets	32-Subsets
GAR	93	93	92	89
FAR	0	0	0	0

Table 2. The comparison of Forest-Fingers algorithm to our Cloud-ID-Screen algorithm in term of Genuine Accept Rate and False Accept Rate.

6.2. Increased Speed Evaluation

In this evaluation, we compared Cloud-ID-Screen against the state-of-the-art baseline [9]. We ran identification (1:N) experiment in the cloud using Amazon Elastic MapReduce (EMR) [1]. Cloud-ID-Screen achieved increased speed in all subsets as compared to the baseline. We observed that the speed of Cloud-ID-Screen in total times for matching operation is increased is 307.04% in Eight-Subsets, 303.28% in Sixteen-Subsets, and 453.80% in Thirty-Two-Subsets. For formal testing, the null hypothesis Ho is that the time for the baseline is less than or equal to Cloud-ID-Screen, *i.e.* the baseline is better. Table 3 shows the p-values from the t-test rejects the null hypothesis over 20 run , using a one-sided two-sample unequal variance (heteroscedastic) t-test. We can reject the null hypothesis for the matching operation in a parallel experiment. We conclude that Cloud-ID-Screen outperforms the baseline in all subsets in total time of matching operation because Cloud-ID-Screen matched all subsets of pair-tables in parallel, then reduced one matching score.

6.3. Security and Privacy Evaluations

In the security and privacy evaluation, we evaluated Cloud-ID-Screen against the state-of-the-art baseline [9]. We observe the parallel experiment in the enrollment and matching operations. In the enrollment operation, we notice Cloud-ID-Screen does not store all subsets of a pair-table in one cloud/machine. Moreover, Cloud-ID-Screen does not store fingerprint images or minutiae points, which can be compromised. However, Cloud-ID-Screen stores parts of a pair-table in each cloud/machine. This distribution grants additional security to fingerprint data in the cloud during the enrollment operation. Moreover, the security level increases with the number of subsets. In other words, if we increase the number of subsets, the security level will increase. Hence, the Thirty-Two-Subset has the highest secu-

	Baseline	Cloud-Id-Screen (8-Subsets)			Cloud-Id-Screen (16-Subsets)			Cloud-Id-Screen (32-Subsets)		
Matching Time	Total Time	Mapping Time	Reducing Time	Total Time	Mapping Time	Reducing Time	Total Time	Mapping Time	Reducing Time	Total Time
Matching time(Ave.)	109.21	0.84	25.99	26.83	0.79	26.28	27.08	0.79	18.84	19.72
Matching time (Std.)	16.81	0.09	0.15	0.22	0.12	0.23	0.33	0.12	0.11	0.21
T-test P-value		2.99179E-15			3.13589E-15			6.53222E-16		

Table 3. The comparison between two algorithms (Cloud-ID-Screen and Forest-Fingers algorithm [9]). The p-value for the test for all subsets are statistically significant, rejecting the null-hypothesis and supporting the claim.

rity level since the individual subset only stores a small part of data. Therefore, the individual subset has a meaningless information about the fingerprint features.

During the matching operation, Cloud-ID-Screen does not collect all subsets of a pair-table from all clouds/machines in order to perform matching. Cloud-ID-Screen matches each subset of a pair-table independently and in parallel at each cloud/machine, then collects the results (match-tables) in order to compute the matching score. Cloud-ID-Screen provided security and privacy to the fingerprint features in the cloud. In addition, Cloud-ID-Screen provides more privacy/security when using the Forest-Finger matching algorithm that matches slap fingerprint images. Hence, the Forest-Finger matching algorithm is designed for unsegmented slap fingerprint to provide privacy/security to the fingerprint data while preventing searching with a latent print [9].

6.4. Scalability of Cloud-ID-Screen

Cloud-ID-Screen provides scalability for the matching process. Cloud-ID-Screen can match 10,000 * N people in parallel and then merge the final maximum matching score. This matching in parallel for multiple people up to N, gives the Cloud-ID-Screen more scalability by adding more machines and matching more people in parallel, then merging the final maximum matching score.

7. Conclusion

The conclusion we can draw from this paper is that the Cloud-ID-Screen achieved promising accuracy in case of the Eight-Subset and the Sixteen-Subset but had comparable accuracy in Thirty-Two-Subset. In terms of increased speed, Cloud-ID-Screen outperforms the baseline in all subsets in total time of matching operation. Therefore, we can conclude that if we increase the number of subsets, the security and performance level will increase at the expense of the accuracy level. In other words, with increasing the number of the subsets , the accuracy level will decrease.

References

[1] amazon emr amazon web services. *Amazon Web Services, Inc.*, [Online]. Available: http://aws.amazon.com/elasticmapreduce/.

[2] apache hadoop. *Hadoop.apache.org*, [Online]. Available: http://hadoop.apache.org/.

[3] apache hbase. *Hbase.apache.org*, [Online]. Available: http://hbase.apache.org/.

[4] data catalog. *Unique Identification Authority of India*, [Online]. Available: https://data.uidai.gov.

[5] new biometric technology improves security and facilitates us entry process for international travelers. *Homeland Security*, [Online]. Available: https://www.dhs.gov.

[6] griaule biometric framework. *Griaule Biometrics*, [Online]. Available: http://www.griaulebiometrics.com/en-us/biometric-framework.

[7] apache zookeeper. *Zookeeper.apache.org*, [Online]. Available: http://zookeeper.apache.org/.

[8] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-preserving fingercode authentication. In *Proceedings of the 12th ACM Workshop on Multimedia and Security*, MM&#38;Sec '10, pages 231–240, New York, NY, USA, 2010. ACM.

[9] A. Bendale and T. E. Boult. id-privacy in large scale biometric systems. In *2010 IEEE International Workshop on Information Forensics and Security, WIFS 2010, Seattle, WA, USA, December 12-15, 2010*, pages 1–6, 2010.

[10] T. E. Boult, W. J. Scheirer, and R. Woodworth. Revocable fingerprint biotokens: Accuracy and security analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[11] Y.-J. Chang, W. Zhang, and T. Chen. Biometrics-based cryptographic key generation. In *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, volume 3, pages 2203–2206 Vol.3, June 2004.

[12] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[13] A. K. Jain, K. Nandakumar, and A. Nagar. Biometric template security. *EURASIP J. Adv. Signal Process*, 2008:113:1–113:17, Jan. 2008.

[14] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, CCS '99, pages 28–36, New York, NY, USA, 1999. ACM.

[15] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[16] K. Nandakumar, A. Nagar, and A. K. Jain. *Hardening Fingerprint Fuzzy Vault Using Password*, pages 927–937. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[17] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle. Generating cancelable fingerprint templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(4):561–572, Apr. 2007.

[18] W. J. Scheirer, B. Bishop, and T. E. Boult. Beyond pki: The biocryptographic key infrastructure. In *The IEEE International Workshop on Information Forensics and Security (WIFS)*, December 2010.

[19] W. J. Scheirer and T. E. Boult. Cracking fuzzy vaults and biometric encryption. In *2007 Biometrics Symposium*, pages 1–6, Sept 2007.

[20] Shelly and N. S. Raghava. Iris recognition on hadoop: A biometrics system implementation on cloud computing. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 482–485, Sept 2011.

[21] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1 – 11, 2011.

[22] A. Sussman, J. Trost, A. Maurer, and E. Kohlwey. Leveraging the cloud for big data biometrics: Meeting the performance requirements of the next generation biometric systems. *2011 IEEE World Congress on Services (SERVICES 2011)*, 00:597–601, 2011.

[23] H. Takabi, J. B. D. Joshi, and G. J. Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security Privacy*, 8(6):24–31, Nov 2010.

[24] A. B. J. Teoh, A. B. J. Teoh, A. Goh, and D. C. L. Ngo. Random Multispace Quantization as an Analytic Mechanism for BioHashing of Biometric and Random Identity Inputs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1892–1901, 2006.

[25] C. Vielhauer, R. Steinmetz, and A. Mayerhofer. Biometric hash based on statistical features of online signatures. In *Object recognition supported by user interaction for service robots*, volume 1, pages 123–126 vol.1, 2002.

[26] C. I. Watson, M. D. Garris, E. Tabassi, C. L. Wilson, R. M. McCabe, S. Janet, and K. Ko. User's guide to non-export controlled distribution of nist biometric image software. Technical report, NIST, 2004.

[27] Z. Wu, L. Tian, P. Li, T. Wu, M. Jiang, and C. Wu. Generating stable biometric keys for flexible cloud computing authentication using finger vein. *Information Sciences*, 2016.