Engineering and Technology Journal
Journal homepage: engtechjournal.org

Key Generation from Multibiometric System Using Meerkat Algorithm

Duha D. Salman [a*], Raghad A. Azeez [b], Abdul Mohssen J. Hossen[c]

[a] Department of Computer Science, University of Technology, Baghdad, Iraq,:111820@uotechnology.edu.iq

[b] College of Education Ibn Rushd, University of Baghdad, Baghdad, Iraq, raghad.azeez@ircoedu.uobaghdad.edu.iq

[c] Department of Computer Science, University of Turath, Baghdad, Iraq, abdulmohsen.jaber@turath.edu.iq

*Corresponding author.

Submitted: 29/09/2019 Accepted: 01/12/2019 Published: 25/12/2020

K E Y W O R D S

Key generation, Meerkat Clan Algorithm, Multiibiometric, Ear, The outer edges of eye.

A B S T R A C T

Biometrics are short of revocability and privacy while cryptography cannot adjust the user's identity. By obtaining cryptographic keys using biometrics, one can obtain the features such as revocability, assurance about user's identity, and privacy. Multi-biometrical based cryptographic key generation approach has been proposed, subsequently, left and right eye and ear of a person are uncorrelated from one to other, and they are treated as two independent biometrics and combine them in our system. None-the-less, the encryption keys are produced with the use of an approach of swarm intelligence. Emergent collective intelligence in groups of simple autonomous agents is collectively termed as a swarm intelligence. The Meerkat Clan Key Generation Algorithm (MCKGA) is a method for the generation of a key stream for the encryption of the plaintext. This method will reduce and distribute the number of keys. Testing of system, it was found that the keys produced by the characteristics of the eye are better than the keys produced by the characteristics of the ear. The advantages of our approach comprise generation of strong and unique keys from users' biometric data using MCKGA and it is faster and accurate in terms of key generation.

How to cite this article: D. D. Salman, R. A. Azeez, and A. J. AH, "Key Generation from Multibiometric System using Meerkat Algorithm,," Engineering and Technology Journal, Vol. 38, Part B, No. 03, pp. 115-127, 2020.
DOI: https://doi.org/10.30684/etj.v38i3B.652

1. INTRODUCTION

Biometrics can be defined as the science of the establishment of an individual's identity according to the person's chemical, behavioral, or physical characteristics, [1]. As a result, to the distinctive characteristic of biometrical feature and the non-repudiation it supplies the biometry which is often utilized for the enhancement of the general security of the system which implements the system of authentication or the biometrical crypto-system [2, 3].

Biometrical authentication can be defined as the procedure of the validation of an individual's identity based on the behavioral or physiological qualities that they have [4], Physiological characteristics, like the fingerprints, irises or faces, indicate things which are individual to each person. Behavioral characteristics such as signature, key-stroke dynamics, and speech which refer to things which each person can do. Based on Biggio [5], generic modular biometrical system of authentication operates via the following stages. An individual who needs accessing some resources providing their identity. The sensor asks for the biometrical user sample. Features which have been obtained from the sample and a same score has computed between provided biometrical sample and the samples which has been stored in the data-base of the biometrical templates which correspond to the given identity of the user. The similarity of the score undergoes comparison against the threshold and the person is recognized as genuine or an impostor. Based on this decision, accessing the resources is given or denied.

Key generation and key binding Biometric crypto-systems combined with a high security level, given by non-repudiation and cryptography which are given by the biometry. Systems of key generation are the ones producing a stable cryptography key which is obtained from the biometrical data [6, 7]. Systems of key binding are the ones which bind an arbitrarily produced cryptography key to the biometrical template [8, 9]; the bound key is released to the application in the case of valid presenting of suitable biometrical template. Biometrical storage samples could present a risk to privacy of the users.

Meerkat Clan Algorithm (MCA) approach is presented for the generation of the key stream. The aim behind the use of an MCA method is that a key stream is selected according to the distribution of characters in the plaintext. Which would give the ability to encode characters in the key stream which occur in the plaintext [10].

2. RELATED WORK

The authors in [11] creates every time various keys based on the snapshot captured from the scanner. Identical biometric cryptography key can be obtained from the fingerprints captured with several quality of image from distinct scanners. The application to data storage and Biometric-based cryptographic key generation have been presented in [12]. They have inspected how to create a comparable feature vector for the same user but in various periods. The authors also used the Reed-Solomon encoding and decoding strategies as encryption and decryption operation in order to get a powerful data storage security. In [13] a biometric-based cryptography key generation have been generated using fingerprint information of a person. With regard to the security views, this method is safe and further flexible in the process of the key generation. User's personal data could be encrypted using this key. The fingerprints characteristic (Reference Points) have been produced two-way area (directional element) and probability distribution. Fingerprint data privacy and security are produced with voidable template in [14]. Also, the authors suggest a way with which key can be canceled thus the restriction of irrevocability attribute of biometric habit could be addressed. Even more crucial, saving the key is not required, in advance of communication. Actually, this method appends further security permitting to produce several keys in different sessions. Mishra and Bali [15], have proposed a Genetic Algorithm application in cryptography area. Selecting the Key in the public key cryptography (PKC) is a process of selection in which keys may be classified, according to their fitness. Ultimately, it purely created non-repeating and random final keys, in addition to increasing the security and strength of the keys. An algorithm for Public Key Cryptography (PKC) has proposed by Jhajharia and his team [16] with the use of hybrid concept of two evolutionary algorithms which include respectively Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs). PSO-GA algorithms utilized to generate the fittest amongst fine fit keys in the domain of keys which contains the optimal keys with the maximum strength possible. Abu-Mouti and Elhawary [17] have proposed a literature overview which employs the recent algorithm of the Artificial Bee Colony (ABC), which is a meta-heuristic population-based approach of optimization which is inspired by intelligent foraging honeybee swarms' behavior. The performance characteristics and the features of key in the Artificial Bee Colony algorithm have been described as well.

3. MEERKAT CLAN ALGORITHM (MCA)

The through observation of the behavior of some living beings might illustrate how they plan their natural behaviors to the algorithmic routines. Those approaches are meta-heuristics for global optimization and are mainly gathered via selecting the optimal structure and via a structure of randomization. The former regulates the merging of the algorithm to the optimality (i.e. the utilization) and the far ahead avoids both losing the variety and prevents the algorithm from getting bordered in the local optima (i.e. examination). An efficient stability between investigation and utilization could result in global achievement of optimality [10].

Meerkats are the animals, which socially live in colonies of five to thirty members. Due to the fact that they are sociable creatures, they exchange both parental care and toilet duties. Every one of the mobs has a leading alpha male as well as a leading alpha female. Every one of the mobs has their own land where they occasionally transfer in the case where the food is not found or in the case where they are obliged with a tougher mob. In which case, the mob, which is weaker, will try at increasing in another way or remain until they get tougher and recover the lost burrow [18]. Every one of the mobs also has what is known as a "sentry" that indicates someone guarding over the mob and when spotting risk and notifying others in the case of danger. The sentry either observes from climbing a tree or from the ground or in bushes. The sentry is responsible for watching over both the scheme of the burrow and in the case where the other mob members are seeking food. The sentry gives a sound like loud bark in the case of observing a risk and afterwards, the mob will rapidly bolt to the hiding holes.

From earlier explaining concerning Meerkat animal inspired MCA, the following are the general MCA steps, which may be changed based on problem encoded [16]:

Initializing: create individuals' clan arbitrarily and set other clan size parameters, care size, foraging size, and worst foraging and care rate.

1) Calculate the clan fitness.
2) Select the optimal one as the "sentry"
3) Split the clan to two groups (foraging and care)
4) Produce neighbors for foraging group.
5) Select the worst foraging group individuals and swap with the optimal individuals in care group
6) Drop the worst individuals in care group and arbitrarily produce another individual.
7) Substitute the optimal individual in foraging with sentry in the case where it's best.

4. THE PROPOSED SYSTEM

In this system, more than one level is applied to generate the key that is used for encryption, we use a key length of 128-bit consists of 8 blocks, each block contains 16 bits, It's generated using Meerkat which will use the unique features of the person to extract features that is used to generate key. Due to the advancement of information technology, we have generated another key length of 256-bit consists of 16 blocks and each block is 16 bits. Figure 1 shows the proposed system architecture and algorithm (2) shows the main phases in the proposed key generation algorithm.

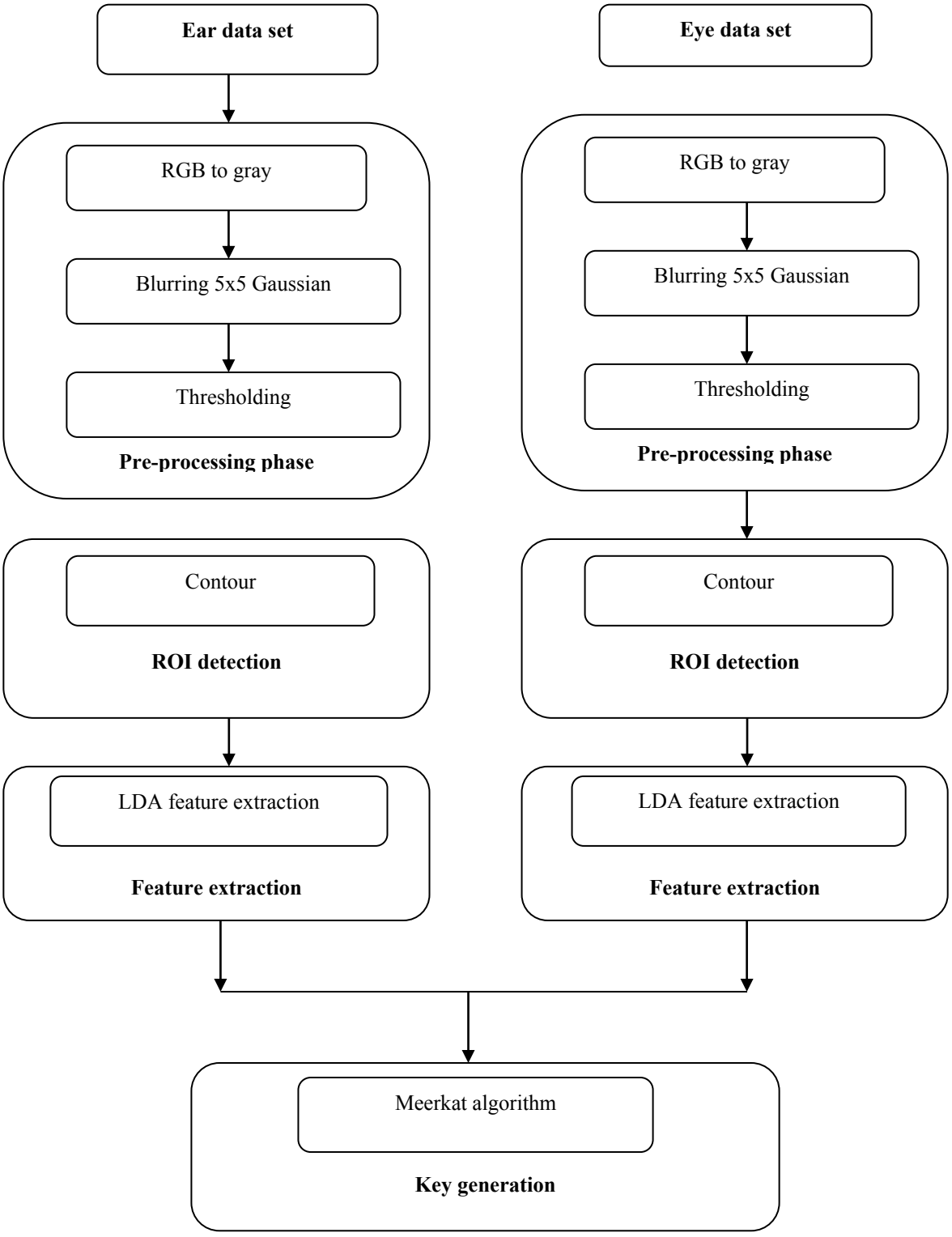

Figure 1: Proposed System Architecture

Algorithm 1: proposed key generation algorithm

Input: ear and eye image
Output: unique features

Start
Step1: read the input image
Step2: apply pre-processing phase which is consist of three internal steps:
RGB to gray conversion Using this equation

$$L=0.299R+0.587G+0.114B$$

Blurring using Gaussian filter
Thresholding Using this equation

$$G(x,y) = \begin{cases} 1 & if\ f(x,y) > T \\ 0 & if\ f(x,y) < T \end{cases}$$

Where value of T chosen carefully through experiment, 115 for eye and 55 for ear images
Step3: detect the region of interest using contour algorithm.
Step4: extract the features using Linear Discriminated Analysis.
Step5: generate the key using Meerkat swarm algorithm.
End

This system consists of five main phases:

I. Pre-processing phase:

This phase consists of three internal steps which used to prepare the data set for further processing each step in this phase applied for a specific task:

1) RGB to gray conversion: in this step the entered colored image will be converted to gray scale to reduce the amount of information processed in the system and remove noise.
2) Blurring: this is done via applying the 2D Gaussian filter to enhance the image and reduce the noise within that image.
3) Thresholding: which will convert the image to binary which will be entered to the region of interest extraction using contour which need binary inputs since it senses the black spots.

II. Region of interest (ROI) detection phase:

The detect part need to be bounded since it is the only part that will need to extracted using feature extraction algorithm, the detection of region of interests done using contour algorithm which will bound the ear or eye and discard other information on the image.

III. Feature extraction phase:

This phase aim to extract features from bounded region of interest founded by contour this is done via algorithm LDA.

IV. Key generation phase:

For the ear image 7 keys is generated and for eye image 10 keys generated which will result 17 keys for each person (due to our dataset). To generate these keys, we use Meerkat Clan Algorithm (MCA).
In the beginning MCA generate clan of keys from features extracted from eyes and ears, the generation process done by using logistic map function.
The logistic map can be expressed as:

$$Xn+1 = r\ xn\ (1- xn) \ldots\ldots (1)$$

whereas xn stands for a number which could range from 0 to 1 representing the existing population ratio to the maximal population possible, and the interest values for parameter r (in some cases written as μ as well) are the ones in the interval [3.7,4].

After generate clan algorithm compute fitness for clan using fitness function, the value of fitness of every one of the individuals is computed. The value of the fitness is computed based on the symbol that is repeated maximally. The function of fitness may be represented as:

$$F = n + (€ / m) \ldots\ldots (2)$$

Where
F stands for the Fitness Function.
n stands for the Total number of symbols used in key formation.
€ stands for the Ideal Percentage of each symbol.
m stands for the Percentage of maximum appeared symbol.
The best one of clan has chosen as sentry (best solution) according fitness value, the remaining clan also split to two groups: foraging group that all the operations have performed on it, and care group. For each key in foraging group find the three neighbors for it and choose the best one according to the fitness value to replace with the original. After completion, all keys in foraging group algorithm choose the worst group (worst fitness value) and replace it with the best group of care group.
In care group, the algorithm drops the worst group in care and generate other keys using logistic map. The last step of algorithm is compared sentry (fitness value) with the best key in foraging and choose the best one of them to ne new sentry.
These steps are repeated several times to reach the best key, in the proposed system several experiments were conducted and it was found that when the number of cycles is ten, we will find the optimal solution. If the number of sessions increases further, the solution will not improve.
The pseudocode algorithm shown in the Algorithm (2).

Algorithm 2: Meerkat Clan pseudocode

Input
n clan size (number of keys)
c care size n-m-1
m foraging size where m < 80% of n
Cr worst care rate
Fr worst foraging rate
k neighbor solution
Output
Best Generated Key (Sentry)
Process
Produce random clan of Keys from extracted feature clan(n)
$x_n+1 = r\, x_n\, (1- x_n)$
Calculate the fitness for each Key in clan using
$F= n + (€ / m)$
Sentry = optimal clan Key
Split the clan to 2 sets (which are foraging and care)
While not end of generations
For i=1 to m
Call **neighbor generated** (k, Sentry, foraging(i), best key)
foraging(i)= best key from k neighbor
End For
Swap the worst for Fr Keys in foraging group with the optimal ones' Keys in care group;
Drop worst Cr Keys from care group and produce ones' Key from extracted features;
Choose optimal Key of foraging call it best key
If best key <= Sentry then
Swap the Sentry with best key
End If
End While
End

5. RESULT AND DISCUSSION

The proposed system was applied on 46 persons, each person has 7 snapshots for ear and 10 snapshots for eye and each snapshot present one key. The keys generated by MCKGA for ear (128-bit) can be shown in Table I and II and those generated for eye (128-bit) demonstrated in Table II and IV. The same for keys with (256-bit) length shown in Tables V- VIII. Table IX and Table X present the result of test keys that generated from eye and ear respectively with key size 128 for one-person (10, 7 keys), the columns show Frequency test results, Frequency Test within a Block results, The Runs Test results, The Binary Matrix Rank Test results, The Discrete Fourier Transform (DFT) Test results, Non-overlapping Template Matching Test results, and Overlapping Template Matching Test results. The results for each test present in two value test present test result, and erfc present error ratio function results, as shown in Table IX, all values of erfc column in each test is greater than 0.01, which mean the generated keys are random. Table XI and Table XII present the result (1 person) of test keys that generated from eye and ear respectively with key size 256.

TABLE I: Keys generated by MCKGA for ear with key size 128 bit (person 1)

Snapshot	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7	Block 8
1	0.195963	0.580398	0.897098	0.340048	0.826664	0.52783	0.918056	0.277118
2	0.915933	0.283638	0.748469	0.693493	0.782994	0.625902	0.862519	0.436806
3	0.010212	0.037232	0.132041	0.422168	0.898594	0.335663	0.821426	0.540336
4	0.414347	0.893884	0.349414	0.837378	0.501624	0.920899	0.268331	0.723206
5	0.409566	0.890783	0.358376	0.847025	0.477303	0.919011	0.274172	0.73305
6	0.190405	0.567837	0.903957	0.319808	0.801304	0.586494	0.893351	0.35096
7	0.608689	0.877392	0.396267	0.88127	0.385429	0.872555	0.409629	0.890825

TABLE II: Keys generated by MCKGA for ear with key size 128 bit (person 2)

Snapshot	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7	Block 8
1	0.457158	0.914147	0.289099	0.757063	0.677489	0.804865	0.578542	0.898185
2	0.322543	0.804907	0.578448	0.898239	0.336704	0.822682	0.537354	0.915769
3	0.290074	0.758575	0.674617	0.808591	0.570122	0.902796	0.32326	0.805842
4	0.574671	0.900369	0.330438	0.814999	0.555401	0.909602	0.30289	0.777791
5	0.709734	0.758872	0.674051	0.809317	0.568469	0.903639	0.320753	0.802555
6	0.618838	0.868886	0.41965	0.897127	0.339964	0.826565	0.528068	0.918007
7	0.33611	0.821967	0.539053	0.915291	0.285606	0.751591	0.687742	0.791072

TABLE III: Keys generated by MCKGA for eye with key size 128 bit (person 1)

Snapshot	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7	Block 8
1	0.456847	0.914049	0.289399	0.757529	0.676605	0.806019	0.575945	0.899663
2	0.679209	0.802606	0.583598	0.895165	0.345689	0.833194	0.511958	0.920382
3	0.560989	0.907207	0.310098	0.788066	0.615232	0.871996	0.411164	0.891838
4	0.67802	0.804169	0.580102	0.897273	0.339536	0.826059	0.529284	0.91775
5	0.57767	0.898686	0.335392	0.821097	0.541114	0.914682	0.287466	0.754517
6	0.379646	0.867551	0.423274	0.899224	0.333813	0.819174	0.54565	0.913232
7	0.569337	0.903199	0.322062	0.804278	0.579858	0.897417	0.339115	0.825562
8	0.025618	0.09195	0.307564	0.784498	0.622758	0.865398	0.429085	0.902384
9	0.898169	0.336912	0.822932	0.536759	0.915931	0.283645	0.748479	0.693474
10	0.709416	0.759362	0.673115	0.810514	0.565737	0.90499	0.316729	0.797182

TABLE IV:Keys generated by MCKGA for eye with key size 128 bit (person 2)

Snapshot	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7	Block 8
1	0.413318	0.893231	0.351307	0.839465	0.496419	0.920861	0.268447	0.723405
2	0.455055	0.913467	0.291172	0.760268	0.671381	0.812714	0.560685	0.907343
3	0.57832	0.898313	0.336487	0.822421	0.537975	0.915596	0.28467	0.750109
4	0.804588	0.579163	0.897824	0.337921	0.824141	0.533878	0.916681	0.281345
5	0.453337	0.912888	0.292937	0.762972	0.666169	0.819195	0.5456	0.913249
6	0.636225	0.852551	0.463062	0.915883	0.283793	0.748715	0.693041	0.783638
7	0.569814	0.902954	0.322788	0.805228	0.577726	0.898655	0.335485	0.821211
8	0.422709	0.898903	0.334756	0.820325	0.542939	0.914117	0.289192	0.757208
9	0.597488	0.8859	0.372347	0.860883	0.441166	0.908158	0.307242	0.78404
10	0.805487	0.577143	0.898987	0.334508	0.820022	0.543651	0.91389	0.289885

TABLE V: Keys generated by MCKGA for ear with key size 256 bit (person 1)

Snapshot	Block 1	Block2	Block3	Block4	Block5	Block6	Block 7	Block8
1	0.398822	0.883199	0.379997	0.867862	0.42243	0.898744	0.335223	0.820892
2	0.092294	0.3086	0.785962	0.619681	0.868146	0.421661	0.898302	0.33652
3	0.885197	0.374344	0.862746	0.436199	0.905914	0.31397	0.793429	0.603746
4	0.030631	0.109376	0.358833	0.847501	0.476084	0.918802	0.274818	0.734123
5	0.409566	0.890783	0.358376	0.847025	0.477303	0.919011	0.274172	0.73305
6	0.398959	0.883301	0.379709	0.867607	0.423121	0.899137	0.334067	0.819484
7	0.538223	0.915527	0.284883	0.750448	0.689856	0.78813	0.615097	0.872111
Snapshot	**Block9**	**Block10**	**Block11**	**Block12**	**Block13**	**Block14**	**Block15**	**Block16**
1	0.541598	0.914534	0.287917	0.755222	0.680963	0.800278	0.588766	0.891884
2	0.822461	0.53788	0.915623	0.284589	0.74998	0.690718	0.786923	0.617655
3	0.881261	0.385455	0.872578	0.409569	0.890784	0.358372	0.84702	0.477314
4	0.718995	0.744245	0.701159	0.771851	0.648677	0.839483	0.496375	0.92086
5	0.720842	0.741254	0.706508	0.763818	0.664528	0.821194	0.540884	0.914752
6	0.544919	0.913476	0.291145	0.760227	0.671459	0.812616	0.560911	0.907242
7	0.410849	0.891632	0.35593	0.84445	0.48386	0.919949	0.271273	0.728196

TABLE VI:Keys generated by MCKGA for ear with key size 256 bit (person 2)

Snapshot	Block 1	Block2	Block3	Block4	Block5	Block6	Block 7	Block8
1	0.289098572	0.757062728	0.677489467	0.804864888	0.578542056	0.898184819	0.336864138	0.822874092
2	0.366039347	0.854804122	0.457190074	0.914157876	0.289066785	0.757013334	0.677583002	0.804742549
3	0.674616748	0.808590893	0.570122229	0.902795726	0.323259566	0.805842267	0.576343324	0.899439287
4	0.330437928	0.814999358	0.555401481	0.909602357	0.302890196	0.777791045	0.636650417	0.852122879
5	0.362334102	0.851096773	0.466830891	0.916855928	0.280807064	0.743927335	0.701730365	0.771002603
6	0.392282308	0.878167041	0.394110909	0.879605874	0.390094061	0.876413287	0.398985642	0.883321185
7	0.324941597	0.808022008	0.571414382	0.902122038	0.325257068	0.808428505	0.570491316	0.902604551
Snapshot	**Block9**	**Block10**	**Block11**	**Block12**	**Block13**	**Block14**	**Block15**	**Block16**
1	0.536896308	0.915893955	0.283758537	0.748660511	0.693141953	0.783495008	0.289098572	0.757062728

2	0.578816778	0.898025576	0.337331189	0.823435449	0.535561816	0.916250143	0.366039347	0.854804122
3	0.333178318	0.818395016	0.547478756	0.912604857	0.29379649	0.764280901	0.674616748	0.808590893
4	0.464172862	0.916180371	0.282880654	0.747259 1	0.695702015	0.779828081	0.330437928	0.814999358
5	0.650373628	0.837613442	0.501037516	0.920904 6	0.268313238	0.723175709	0.362334 12	0.851096773
6	0.379653307	0.867557344	0.423255493	0.899213052	0.333843887	0.819211383	0.392282308	0.878167041
7	0.323826746	0.806579604	0.574679938	0.900364652	0.330452019	0.815016 6	0.324941597	0.808022008

TABLE VII: Keys generated by MCKGA for eye with key size 256 bit (person 1)

Snapshot	Block 1	Block2	Block3	Block4	Block5	Block6	Block 7	Block8
1	0.562606	0.906471	0.312305	0.791136	0.608684	0.877397	0.396254	0.881261
2	0.135102	0.430431	0.90308	0.322415	0.80474	0.578823	0.898022	0.337342
3	0.284432	0.749732	0.691174	0.786281	0.61901	0.868736	0.420059	0.897368
4	0.67802	0.804169	0.580102	0.897273	0.339536	0.826059	0.529284	0.91775
5	0.376845	0.865038	0.430053	0.902886	0.322991	0.805493	0.57713	0.898994
6	0.379646	0.867551	0.423274	0.899224	0.333813	0.819174	0.54565	0.913232
7	0.096972	0.32257	0.804943	0.578366	0.898286	0.336566	0.822517	0.537748
8	0.025618	0.09195	0.307564	0.784498	0.622758	0.865398	0.429085	0.902384
9	0.63545	0.853326	0.461047	0.915319	0.285518	0.751451	0.688	0.790714
10	0.364297	0.853073	0.461704	0.915506	0.284946	0.750548	0.689672	0.788388
Snapshot	**Block9**	**Block10**	**Block11**	**Block12**	**Block13**	**Block14**	**Block15**	**Block16**
1	0.385457	0.872579	0.409565	0.890782	0.358378	0.847027	0.477297	0.91901
2	0.823448	0.535532	0.916258	0.282643	0.746878	0.696395	0.778827	0.634527
3	0.339257	0.82573	0.530075	0.917577	0.278592	0.740332	0.708144	0.761319
4	0.27806	0.739462	0.709681	0.758953	0.673896	0.809517	0.568015	0.903868
5	0.334487	0.819997	0.543711	0.913871	0.289943	0.758372	0.675003	0.808094
6	0.291888	0.761368	0.669267	0.815367	0.554547	0.909948	0.301846	0.77627
7	0.91566	0.284476	0.749801	0.691048	0.786459	0.618634	0.869065	0.419165
8	0.324481	0.807428	0.572762	0.901406	0.327375	0.811139	0.564306	0.905676
9	0.609588	0.87667	0.398274	0.88279	0.381153	0.868879	0.41967	0.897139
10	0.614549	0.872574	0.409578	0.890791	0.358353	0.847	0.477365	0.919021

TABLE VIII: Keys generated by MCKGA for eye with key size 256 bit (person 2)

Snapshot	Block 1	Block2	Block3	Block4	Block5	Block6	Block 7	Block8
1	0.174513	0.530658	0.917446	0.278993	0.740985	0.706985	0.763091	0.665939
2	0.542955	0.914112	0.289208	0.757232	0.677168	0.805284	0.577599	0.898727
3	0.50465	0.920829	0.268548	0.723576	0.736778	0.714391	0.751596	0.687732
4	0.580071	0.897292	0.339482	0.825996	0.529437	0.917717	0.278162	0.739629
5	0.570494	0.902603	0.323831	0.806586	0.574666	0.900372	0.33043	0.814989
6	0.428836	0.902253	0.324868	0.807927	0.571631	0.902008	0.325595	0.808863
7	0.569814	0.902954	0.322788	0.805228	0.577726	0.898655	0.335485	0.821211
8	0.422709	0.898903	0.334756	0.820325	0.542939	0.914117	0.289192	0.757208
9	0.443375	0.909098	0.304412	0.779993	0.632126	0.856602	0.452479	0.91259
10	0.805487	0.577143	0.898987	0.334508	0.820022	0.543651	0.91389	0.289885
Snapshot	**Block9**	**Block10**	**Block11**	**Block12**	**Block13**	**Block14**	**Block15**	**Block16**

1	0.819477	0.544936	0.913471	0.291162	0.760253	0.671409	0.812679	0.560766
2	0.335272	0.820952	0.541456	0.914578	0.287784	0.755014	0.681354	0.799757
3	0.791086	0.608791	0.877311	0.396494	0.881444	0.384942	0.872143	0.410761
4	0.709387	0.759408	0.673028	0.810625	0.565483	0.905113	0.316363	0.796687
5	0.555426	0.909592	0.30292	0.777834	0.636562	0.852212	0.463942	0.916119
6	0.569503	0.903114	0.322314	0.804608	0.57912	0.897849	0.337848	0.824054
7	0.540845	0.914763	0.287219	0.754129	0.683014	0.797529	0.594821	0.887789
8	0.677215	0.805223	0.577736	0.898649	0.335502	0.821231	0.540796	0.914778
9	0.293842	0.76435	0.663493	0.822445	0.537919	0.915612	0.284622	0.750033
10	0.758282	0.675174	0.807872	0.571754	0.901943	0.325787	0.80911	0.568942

TABLE IX:Test Results of Keys Generated from eye with key size 128 bit (1 person)

Frequency Test		**Freq-uency Block**		**Run Test**		**Rank Test**		**DFT**		**Non overlying**		**Overlapping Test**	
Tests	***Erfc***	***tests***	***Erfc***	***tests***	***Erfc***	***Tests***	***erfc***	***Tests***	***Erfc***	***tests***	***erfc***	***tests***	***Erfc***
0.625	0.441942	21.75	10.875	118	27.68953	1.679739	2.316064	-43.232	30.56965	0.164678	0.082339	82.713	41.3565
0.25	0.176777	28	14	124	11.22258	2.037079	2.769148	-47.6931	33.72411	0.128083	0.064042	86.25817	43.12908
0.875	0.618718	25.75	12.875	131	9.539423	5.396498	14.8537	-45.2598	32.00349	0.17154	0.08577	87.43411	43.71705
0.5	0.353553	28	14	126	5.298122	5.396498	14.8537	-45.6653	32.29026	0.141806	0.070903	89.06829	44.53415
0.375	0.265165	30.25	15.125	141	36.94812	1.679739	2.316064	-42.4209	29.99611	0.144094	0.072047	90.96091	45.48045
1.625	1.149049	45.75	22.875	130	9.294392	1.679739	2.316064	-46.4764	32.8638	0.093775	0.046888	85.9992	42.9996
0.375	0.265165	30.25	15.125	134	17.16001	1.679739	2.316064	-46.4764	32.8638	0.15553	0.077765	89.23934	44.61967
0.5	0.353553	32	16	139	31.43552	2.037079	2.769148	-46.4764	32.8638	0.102924	0.051462	92.62908	46.31454
1	0.707107	39.5	19.75	136	23.94772	5.396498	14.8537	-43.6376	30.85642	0.091488	0.045744	86.37091	43.18546
0.625	0.441942	21.75	10.875	118	27.68953	1.679739	2.316064	-43.232	30.56965	0.164678	0.082339	82.713	41.3565

TABLE X: Test Results of Keys Generated from ear with key size 128 bit (1 person)

Frequency Test		Frequency Block		Run Test		Rank Test		DFT		Non Overlying		Overlapping Test	
Tests	*Erfc*	*tests*	*Erfc*	*tests*	*Erfc*	*Tests*	*erfc*	*Tests*	*Erfc*	*tests*	*erfc*	*tests*	*Erfc*
0.125	0.088388	32.75	16.375	139	31.1329	1.679739	2.316064	-46.4764	32.8638	0.146381	0.07319	84.50759	42.25379
0.875	0.618718	36.75	18.375	127	1.740449	1.679739	2.316064	-45.2598	32.00349	0.093775	0.046888	84.99745	42.49872
0.125	0.088388	48.25	24.125	137	25.47639	1.679739	2.316064	-42.0153	29.70934	0.118934	0.059467	94.53594	47.26797
0.5	0.353553	32.5	16.25	142	39.91252	1.679739	2.316064	-44.8542	31.71673	0.116647	0.058324	89.40633	44.70317
0.125	0.088388	36.75	18.375	131	8.506859	5.396498	14.8537	-45.6653	32.29026	0.21271	0.106355	89.23713	44.61857
1	0.707107	33.5	16.75	131	9.860825	1.679739	2.316064	-44.0431	31.14319	0.189838	0.094919	86.46985	43.23492
0.125	0.088388	32.75	16.375	139	31.1329	1.679739	2.316064	-46.4764	32.8638	0.146381	0.07319	84.50759	42.25379

TABLE XI:Test Results of Keys Generated from eye with key size 256 bit (1 person)

Frequency Test		Frequency Block		Run Test		Rank Test		DFT		Non Overlying		Overlapping Test	
Tests	*Erfc*	*tests*	*Erfc*	*tests*	*Erfc*	*Tests*	*erfc*	*Tests*	*Erfc*	*tests*	*erfc*	*tests*	*Erfc*
0.625	0.441942	27.75	13.875	136	23.14447	1.679739	2.316064	-46.4764	32.8638	0.153242	0.076621	96.21435	48.10717
1.375	0.972272	33.25	16.625	123	11.38369	2.037079	2.769148	-47.2875	33.43734	0.153242	0.076621	89.9182	44.9591
1.375	0.972272	25.25	12.625	135	22.30677	2.037079	2.769148	-45.6653	32.29026	0.153242	0.076621	91.9033	45.95165
1	0.707107	19.5	9.75	126	4.226068	5.396498	14.8537	-44.4487	31.42996	0.176114	0.088057	85.77936	42.88968
0.375	0.265165	29.25	14.625	124	11.10873	1.679739	2.316064	-45.6653	32.29026	0.166966	0.083483	89.189	44.5945
0.125	0.088388	29.25	14.625	121	19.77569	0.559932	1.323085	-43.6376	30.85642	0.139519	0.06976	89.05807	44.52903
1.25	0.883883	24.5	12.25	118	25.91542	1.679739	2.316064	-46.0709	32.57703	0.146381	0.07319	89.7356	44.8678
0.375	0.265165	36.75	18.375	127	2.628109	2.037079	2.769148	-42.0153	29.70934	0.166966	0.083483	86.46677	43.23339
0.125	0.088388	29.75	14.875	118	28.26045	1.679739	2.316064	-45.6653	32.29026	0.139519	0.06976	92.54524	46.27262
1.5	1.060660	31.5	15.75	134	19.97542	0.559932	1.323085	-44.8542	31.71673	0.166966	0.083483	92.63931	46.31966

TABLE XII: Test Results of Keys Generated from ear with key size 256 bit (1 person)

Frequency Test		Frequency Block		Run Test		Rank Test		DFT		Non Overlying		Overlapping Test	
Tests	*Erfc*	*tests*	*Erfc*	*test*	*Erfc*	*Tests*	*erfc*	*Tests*	*Erfc*	*tests*	*erfc*	*tests*	*Erfc*

				s									
0.375	0.265165	32.75	16.375	126	5.454982	0.559932	1.323085	-44.8542	31.7167 3	0.118934	0.059467	89.67829	44.8391 4
2.5	1.767767	34	17	121	10.69257	1.679739	2.316064	-48.0987	34.0108 8	0.176114	0.088057	88.00944	44.0047 2
2	1.414214	40.5	20.25	131	13.92116	0.559932	1.323085	-44.0431	31.1431 9	0.166966	0.083483	91.95743	45.9787 2
0.25	0.176777	24.5	12.25	123	14.05032	5.396498	14.8537	-43.232	30.5696 5	0.107498	0.053749	90.01068	45.0053 4
1.25	0.883883	28.5	14.25	118	25.91542	1.679739	2.316064	-46.882	33.1505 7	0.130 37	0.065185	89.13904	44.5695 2
0.125	0.088388	26.25	13.125	120	22.60394	5.396498	14.8537	-45.6653	32.2902 6	0.134945	0.067472	90.78845	45.3942 2
0.25	0.176777	35.5	17.75	124	11.22258	0.559932	1.323085	-46.882	33.1505 7	0.141806	0.070903	92.71809	46.3590 5

6. CONCLUSION

Produce randomness in cryptography is very difficult. In cryptography, the randomness gives better security. Multibiometric-based cryptographic key generation have been proposed in this work, and the proposed system carried out for hundreds of samples. Each population varies greatly from another, thus, key length for which test carried out 128 bit long and longer key sequence will also work but time constraint does not permit to check.

The present research presents the MCA, and explains its ability in generating Keys from multi-biometric and obtain the random keys with a smaller number of iterations. Results show that the keys resulting from eye features are more random than those resulting from the ear and thus become difficult to detect by the hackers and give strength to work for encryption processes. To determine which is better between the eye and the ear, the error rate of the test result was tested and compared with 0.01.

References

[1] A. Ross and P. Flynn, Handbook of biometrics. Springer Science+ Business Media, LLC, 2008

[2] Y. C. Feng, P. C. Yuen, and A. K. Jain, "A hybrid approach for face template protection," in Biometric Technology for Human Identification V, 2008, vol. 6944, p. 694408.

[3] P. Balakumar and R. Venkatesan, "A survey on biometrics based cryptographic key generation schemes," Int. J. Compute. Sci. Inf. Technol. Secur., vol. 2, no. 1, pp. 80–85, 2012.

[4] P. Balakumar and R. Venkatesan, "A survey on biometrics based cryptographic key generation schemes," Int. J. Compute. Sci. Inf. Technol. Secur., vol. 2, no. 1, pp. 80–85, 2012.

[5] Biggio, "Adversarial Pattern Classification," Doctoral dissertation, Sardinia Univ. Cagliari, 2010.

[6] D. D. Salman, R. A. Azeez, and A. J. AH, "BUILD CRYPTOGRAPHIC SYSTEM FROM MULTI-BIOMETRICS USING MEERKAT ALGORITHM", Iraqi Journal for Computers and Informatics, Vol 45, Issue 2, 2019.

[7] 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), 2004, vol. 3, pp. 2203–2206.

[8] Juels and M. Sudan, "A fuzzy vault scheme," Des. Codes Crypto. vol. 38, no. 2, pp. 237–257, 2006.

[9] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in International conference on the theory and applications of cryptographic techniques, 2004, pp. 523–540.

[10] Ahmed. T. S. Al-Obaidi, H. S. Abdullah, and Z. Othman., "Meerkat clan algorithm: A new swarm intelligence algorithm," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 10, no. 1, pp. 354–360, 2018.

[11] G. Panchal and D. Samanta, "Comparable features and same cryptography key generation using biometric fingerprint image," in *2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, 2016, pp. 691–695.

[12] G. Panchal and D. Samanta, "A novel approach to fingerprint biometric-based cryptographic key generation and its applications to storage security," *Comput. Electr. Eng.*, vol. 69, pp. 461–478, 2018.

[13] G. Panchal and D. Samanta, "Directional area based minutiae selection and cryptographic key generation using biometric fingerprint," in *Proceedings of the First International Conference on Computational Intelligence and Informatics*, 2017, pp. 491–499.

[14] S. Barman, D. Samanta, and S. Chattopadhyay, "Fingerprint-based crypto-biometric system for network security," *EURASIP J. Inf. Secur.*, vol. 2015, no. 1, p. 3, 2015.

[15] S. Mishra and S. Bali, "Public key cryptography using genetic algorithm," *Int. J. Recent Technol. Eng.*, vol. 2, no. 2, pp. 150–154, 2013.

[16] S. Jhajharia, S. Mishra, and S. Bali, "Public key cryptography using particle swarm optimization and genetic algorithms," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 6, pp. 832–839, 2013.

[17] F. S. Abu-Mouti and M. E. El-Hawary, "Overview of artificial bee colony (ABC) algorithm and its applications," in *2012 IEEE International Systems Conference SysCon 2012*, 2012, pp. 1–6.

[18] M. Gheni, "An Approch to Identify and classify Iraqi Papers currency". a PhD. Thesis, Department of computer Science, University of Technology, 2019.