DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

Generating Honeyword Based on A Proposed Bees Algorithm

Yasser A. Yasser[1], Ahmed T. Sadiq[2], Wasim AlHamdani[3]
[1,2] *Computer Sciences Department, University of Technology, Baghdad, Iraq*
[3]*Information Technology Department, University of the Cumberlands, KY, US*
[1]*cs.19.28@grad.uotechnology.edu.iq*, [2]*Ahmed.T.Sadiq@uotechnology.edu.iq*,
[3]*wasim.alhamdani@ucumberlands.edu*

Abstract— *Honeywords are fake passwords that are typically companions to the real password "sugarword." The honeyword technique is a password cracking detection technique that works effectively to improve the security of hashed passwords by making password cracking simpler to detect. The password database will contain many honeywords for each user in the system. A silent alarm will trigger, indicating that the password database has been compromised if the hacker signs in using a honeyword. The honeychecker is a separate server in charge of recognizing the real password and raising the silent alarm. Many honeyword creation techniques have been presented previously. They all have limitations in the generating process, supporting characteristics, and strengths of honeyword. The bees algorithm, an optimization metaheuristic swarm intelligence algorithm, is used in this article to suggest a novel approach for generating honeywords. The proposed bee algorithm succeeded in addressing the limitations of the previous methods by enhancing the honeyword generating process, supporting the honeyword characteristics, and addressing the honeyword system problems. The most important characteristics of the honeyword (flatness, DoS resistance, and storage) were supported by the proposed method to present unconditionally flatness, strong DoS resistance, and moderate storage.*

Index Terms— *Bees Algorithm, Honeyword, Password, Swarm Algorithm.*

I. INTRODUCTION

The most popular authentication technique is password authentication because of its ease, which is saved as a hashed password [1]. The password mechanism is subject to many aggressive actions to break it like password cracking. The process of regaining a plain password from hashed password in an illegal way is called password cracking [2].

The Honeyword system is a technique that supports the security of the hashed password by detecting its cracking. Many honeywords (false passwords) with only one sugarword (real password) will be connected to every user's account [3], [4]. The hacker that succeeds in stealing the password file and cracking the hashed password will be detected at once in case a honeyword using in a login attempt [5], [6]. The detection of illegal login will be discovered by an additional server called honeychecker. It is responsible for distinguishing the real password and sending a quiet alert to the administrator if a honeyword is used [7], [8].

A metaheuristic is an elevated technique in computer science that finds, generates, or selects an intuitive that may give a sufficiently good solution to an optimization issue [9]. An optimization problem is a problem in mathematics, computer science, or economics in

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

which the aim is to find the best solution from a set of alternatives [10]. The metaheuristic algorithms can be nature-inspired, the latter has many subfamilies such as swarm, physical, evolutionary, and immune algorithms [11]. Nature-inspired algorithms are a series of unique approaches inspired by nature to solve problems in artificial intelligence [12]. The collective behavior of decentralized, self-organized systems, natural or artificial, is referred to as swarm intelligence algorithms [13]. The bees algorithm based on the careful examination of bees while searching for food is one of the metaheuristic swarm intelligence algorithms. It tries to address optimization issues by selecting the best solution [14].

II. RELATED WORKS

During the previous several years, a lot of research has suggested honeyword generating approaches. Asymptotical works are included in this section.

- In [15] (2013), Juels and Rivest, This paper proposes many honeyword generation methods, including tweaking a portion of the password, employing a dictionary, attaching to a password, honeywords supplied by the system, honeywords supplied by the user, and hybrid methods. These methods are: (Tail tweaking, Digits tweaking, Simple model, Modeling syntax, “Tough nuts”, Hybrid generations, Choose a tail, and Random choose).
- In [16] (2015), Ergular, The "Storage-index" technique presents an alternate approach for honeyword creation that picks honeywords based on current user passwords in the system to generate realistic honeywords. Instead of producing honeywords and preserving them in a passwords database, this method emulates honeywords by using existing passwords.
- In [17] (2017), Chakraborty and Mondal, Paired Distance Protocol (PDP) is a new honeyword generation mechanism with a new user interface. A user must enter three elements of information to sign in: a username, a password, and a password tail. The user selects a password tail from a selection of alphabetic letters and numerals in addition to the username and password.
- In [18] (2018), Akshima et al., Three generating methods are presented as superior and more useful honeyword generation approaches. (1) Password evolving: There are two parts to this model: tracking how many times password patterns have been used and creating honeywords from post frequencies. (2) User profile: Honeywords are created by combining various user profile data by generating unique sets from given data that comprise tokens of various sorts. (3) Add secret: Three items are entered, the user's login, password, and a third item to generate a random string of numbers, characters, and symbols.
- In [19] (2019) Akif et al., Propose a honeyword creation strategy that incorporates all four methods. As a result, four portions of honeywords were formed and sent to the system. (1) User data that already exists: Using two parts of public personal questions to generate data. The first part will be about letters, and the second will be about numbers. The answers to the first and second parts will be combined to create honeywords. (2) An assault using a dictionary: The basic concept of using the real password to generate acceptable honeywords following a dictionary attack is to use it. (3) A collection of generic passwords: This honeyword group is made up of honeywords chosen at random from a list of the 500 worst passwords. (4) Combining scrambled letters or numerals: Honeyword is made out of combined user ID characters or digits.

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

III. HONEYWORDS

The honeywords technique exists to generate honeywords (False passwords) companion sugarword (Real password), then putting all of them as sweetwrods into the username and password database and hashing them [20], [21]. If the hacker obtains plain passwords from hashed passwords, then should successfully predict the true password from some of the sweetwords. Otherwise, a silent notification to the system administrator is sent by an auxiliary server called honeychecker, indicating that password cracking is possible [22], [23]. The actions of the admins are defined by the organization's policy and may include banning, suspending, or alerting the account [24].

Flatness, is the hacker's expected chance of correctly guessing the sugarword, because a hacker can succeed with a probability of $(1/n)$ by predicting sugarword randomly (n=number of sweetwords). If the honeywords are perfectly flat (i.e., $1/n$ flat), the hacker has at least a $(1-(1/n))$ probability of selecting one [25], [26].

User login, when the user's account is legitimate, then the user's password is hashed and matched to the sweetwords' file before being sent to the honeychecker for verification. If the provided password is the sugarword, the login will be permitted; otherwise, the admin will be notified that a suspected password cracking has occurred [27], [28].

IV. BEES ALGORITHM

The foraging process is the most important colony behavior concept for bees. The honey bee colony is comparable to other social insect colonies in that it possesses similar self-organized characteristics. The bees begin their foraging journey by looking for good flower patches with enough nectar and pollen of sufficient quality. The latter step is similar to exploration, and the bees in this status are referred to as scouts [29], [30]. Scout bees that discovered the better flower patches are considered elite bees have a specific dancing habit for positive feedback, and this waggle dance is vital for the exploitation of good food sources [31], [32]. The waggle dance will attract colony bees to the selected flower patches, here the bees will regard as recruits [33], [34]. Negative feedback is another issue that is balancing the colony's positive feedback impact; this feedback will cause the flower patches to be abandoned and a new random search as exploration to begin [35], [36]. Several NP-Hard problems have been solved using the bees algorithm. The Bees Algorithm is based on honey bees' natural foraging activity to discover the optimal solution [37], [38].

Following up on the bees algorithm's earlier explanation, Algorithm 1. shows the general steps of the algorithm, which may be changed based on the problem encoded.

ALGORITHM 1. THE GENERIC STEPS FOR THE BEES ALGORITHM [39], [40].

Step 1: Setup the parameters, including the number of scout bees (s), the elite bees (e), the number of selected areas out of s points (h), and the number of recruited bees surrounding elite regions (h-e) regions, as well as the stopping criteria.
Step 2: Initialize s bees randomly (scout bees).
Step 3: Evaluate fitness.
Step 4: Assign elite status to the bees with better fitness (e).
Step 5: Neighborhood search,
  a. Neighborhood search sites (h-e).
  b. Determine the neighborhood range.
  c. Recruit bees to specific locations and assess the fitness.
  d. From each place, choose the fittest bee.
Step 6: Assign the remaining bees to search ($s - h - e$) at random.
Step 7: Population of new scout bees ($e + h + (s - h - e)$).

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

Step 8: Continue iterating steps 3 to 5 until the stopping criteria are met.

V. PROPOSED BEES ALGORITHM

The suggested technique chose the bees algorithm as an optimization metaheuristic swarm intelligence algorithm to provide a novel honeyword generating method. The algorithm was chosen because of its robustness, reliability in getting the optimum solution, fast convergence, and ability to maintain exploration and exploitation. The bees algorithm has undergone several modifications to address the alleged problem of honeyword generation.

The proposed honeyword strategy was adopted for the legacy-UI, which is more user-friendly because it just asks for the user's login and password. Consider including alphabets, numerals, and special characters in the password. The suggested method uses 36 sweetwords, which implies that if n=36, the hacker has a (1/49≈2%) c of successfully picking the sugarword and a (1-2%= 98%) chance of selecting a honeyword. The proposed honeyword system will try to address the shortcomings of the preceding generating technique.

According to the kind of password token, the proposed bees algorithm uses different approaches to process it. Each token kind has its own generator (alphabet, digits, special characters generator). The proposed algorithm adopts the bees algorithm strategies in solving problems to handle the alphabet token, even it suggests a neighborhood search technique and evaluation criteria for it. On the other hand, the digits and special characters tokens deal with by simple random generators.

A. Proposed Bees Algorithm Tokens Generators

In the suggested approach, three potential token generators run in parallel. Because the alphabet generator is the most important and challenging, it will use the bees algorithm strategies in solving problems, but the digits and special characters generator will employ a simple random generating strategy.

1. The proposed bees algorithm alphabet tokens generator
   The alphabet token seems to be the most important part of the honeyword since it is the hacker's preferred way of guessing the genuine password. This generator is the hardest to use since it solves issues using the bees algorithm strategy, with password tokens acting as bees. The sugarword's alphabet token will be used as input for the generators. It's supposed to be the starting point for the honeywords alphabet tokens.
   Make seven copies of the top six algorithm tokens for each of the six alphabet tokens, then divide the 42 tokens into six groups (columns). Each group includes seven tokens that are similar. Seven copies of the alphabet root should be added. As a result, the algorithm will have a total of 49 alphabet tokens.
2. The proposed bees algorithm alphabet tokens generator
   The sugarword's digit token is at the root of this generator, which is based on random generation. The generator will produce 48 tokens of the same length as the root. The suggested bees algorithm will have 49 digit tokens after adding the digit root.
3. The proposed bees algorithm special characters tokens generator
   The sugarword's digit token is at the root of this generator, which is based on random generation. The generator will generate 48 tokens with the same length as

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

the root. The suggested bees algorithm will have 49 digit tokens after adding the special character root.

Example 1: If the sugarword is (**825^#%test**), use the parameters specified in the (parameters section) for the suggested bees algorithm. The following are the sweetwords generated by the suggested bees algorithm:

825^#%test	434(%@zest	268?”&west	418)_<vest	961(^!text	340+?~tent	227)<{teat
371){:test	715@&”zest	925_>^west	338.<-vest	606_’~text	703(&!tent	025+\!$teat
134+~,test	536>?)zest	959$%-west	745?[\vest	523%$|text	147{;?tent	839=-|teat
016^~{test	735)!^zest	382+?^west	563($\vest	247?>,text	368_^|tent	751)<&teat
193*”]test	530}!’zest	741!_.west	841_:^vest	103?:}text	058!^(tent	185?:’teat
546)&?test	927).<zest	823%^.west	621~”;vest	736~])text	993(!#tent	338.,’teat
840)>!test	475#=-zest	281-/|west	058-{!vest	501]+=text	357)$,tent	631_?.teat

B. Proposed Bees Algorithm Neighborhood Search Technique

For the bees, the proposed method presents a neighborhood search technique that is particular to the alphabet token, the bees will use four moves in their search (Add, remove, locations swap, and change). The honeyword alphabet token will search for the sugarword alphabet token in the same way as bees search for food. The token search will be applied as a modification in the alphabet tokens' characters. The change quantity in the characters of the alphabet token will indicate the neighborhood range of sites. (0.3*alphabet token size) is the suggested neighborhood range.

1. Add: At random, choose certain characters' places on the token, then insert randomly selected characters in those positions.
2. Remove: At random, choose characters from the token and erase them.
3. Locations swap: Pick at random the locations of the characters on the token, then swap them with one another.
4. Change: Choose characters' places on the token at random, then replace them with other characters chosen at random.

Example 2: If the sugarword alphabet token is (bat), then nr =0.3*(3)= 0.9 in the suggested bees algorithm, which employed neighborhood range =(0.3*(alphabet token length)) during the neighborhood search, 1 character will be modified. The generated tokens are in the order (baty, at, tab, bot).

C. Proposed Evaluation Criteria

Only the created alphabet tokens will be reviewed in the proposed evaluation procedure, which will be based on the sugarword's root alphabet token. The approximation element is a suggested assessment criterion for the created alphabet tokens offered by the proposed bees algorithm. The total of the four criteria values, which is in the range, is used to calculate the approximation element (0,1). Each criterion has its value, as stated in the (Parameters section).

1. Character matching: The degree of matching between the root token's characters and the created token's characters.
2. Length matching: The length of the characters in the root token and the produced token are identical.
3. PoS (part of speech) matching: The root token and the produced token are matched in terms of PoS.
4. Is the created token a meaningful term in English?

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

Example 3: If the sugarword alphabet token is (ninja) and the parameters specified in the (parameters section) are used. The approximation element of the honeyword tokens generated will be (ninon/0.919, nina/0.9, punjab/0.883, nanny/0.88, sink/0.86, link/0.86).

D. Proposed Bees Algorithm Steps

The proposed system uses a suggested bees algorithm to generate the honeywords as a token generating process, where the sugarword is tokenized into three distinct tokens: alphabet, digits, and special characters, and each one is treated in a different generator (alphabet, digits, special characters generator), and then the honeywords are collected with the sugarword to introduce the sweetwords. The password tokens will be treated similarly to bees, but with significant modifications, as previously mentioned. The proposed algorithm's general steps are divided into six portions, as shown in Algorithm 2.

ALGORITHM 2. THE GENERAL STEPS OF THE PROPOSED BEES ALGORITHM.

Step one: Tokenization. According to its kind, the sugarword is divided into three tokens: alphabet, digits, and special characters tokens. Each token type will be handled differently, with a separate generator.

Step two: Alphabet generator. Step one's alphabet tokens will be delivered to the following generating sub-steps:

- a: Set the alphabet generator's parameters. Bees' population size *bs*, maximum generation *mg*, optimal nectar *on*= alphabet token received from step1, number of bee's movement *bm*, neighborhood range *nr*, evaluation criteria *e*, elite bees' size *es*.
- b: Generate the bee's initial population (alphabet token) with *bs* randomly.
- c: Calculate the population's fitness (approximation element) using assessment criteria *e* and the optimal nectar *on*.
- d: Assign the bees *es* with the best fitness as elite bees.
- e: Neighborhood search
  1. Every bee makes the *bm* movements.
  2. Considering neighborhood range *nr*.
  3. Compute the fitness of bees considering the evaluation criteria *e* and optimal nectar *on*.
  4. Every bee adopts its best move.
- f: Abandon the population's worst bees and replace them with new ones that generate at random.
- g: Repeat sub-steps c to f until maximum generation *mg*.
- h: Return the best fitness bees with considering to *es*, as the alphabet honeyword tokens.

Step three: Digits generator. The digits token from step one is passed on to the digit's generator, which performs the following sub-steps:

- a: Set the *dt* value for the number of generated digits tokens.
- b: Generate tokens with *dt* considering by selecting special characters at random that have the same length as the root token.
- c: Return the *dt* tokens as the digits honeyword tokens.

Step four: Special characters generator. After receiving the special characters token from step one, the token is delivered to the special characters generator, which includes the following sub-steps:

- a: Set the *st* value of the number of generated special characters tokens.
- b: Generate tokens with st considering by selecting special characters at random that have the same length as the root token.
- c: Return the *st* tokens as the special characters honeyword tokens.

Step five: Collect honeywords. Collect the six alphabet tokens, with the 48 digits, and special characters tokens.

Step six: Return sweetwords. Combine the sugarword with the honeywords. Sweetwrods' positions are permuted at random, giving the total of 49 sweetwords.

E. Proposed Bees Algorithm Pseudocode

A follow-up to the suggested bees algorithm, Algorithm 2. and the explanation of the three token generators. This section shows the suggested bees algorithm pseudocode in Algorithm 3.

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

ALGORITHM 3. THE PSEUDOCODE OF THE PROPOSED BEES ALGORITHM.

Parameter
Bees' population size *bs*, maximum generation *mg*, optimal nectar *on*= alphabet token received from step1, number of bee's movement *bm*, neighborhood range *nr*, evaluation criteria *e*, elite bees' size *es*, number of generated digits tokens *dt*, number of changing digits in the generated token *dl*, number of generated special characters tokens *st*, number of changing special characters in the generated token *sl*

```
Begin
    Tokenization
    If the token is an alphabet
    Generate the initial bees initial population with bs randomly
    Calculate the population's fitness considering e and on
      for i=1 to mg
      Let the best bees with es as elite bees
        for j=1 to bs
          Makes the bm movements
          Considering neighborhood range nr
          Evaluate fitness of bees considering e
          if the movements are better than the previous site then adopt the better move
          else cancel it
          end if
        end for
          Calculate the population's fitness considering e and on
          Abandon the population's worst bees and replace them with new ones that generate at random
      end for
    Return the best fitness bees with considering to es, as the alphabet honeyword tokens
    end if
    If the token is digits
      for i=1 to dt
        for j=1 to dl
          Changes the digits of the token by other digits randomly
        end for
      end for
      Return the dt tokens as the digits honeyword tokens
    end if
    If the token is special characters
      for i=1 to st
        for j=1 to sl
          Changes the special characters of the token by other special characters randomly
        end for
      end for
    Return the st tokens as the special characters honeyword tokens
    end if
    Collect honeyword tokens
    Return sweetwords by adding the sugarword to the honeywords then permutate and hashed the sweetwords
End
```

F. Parameters

Many parameters are used in the suggested honeyword generation approach to impact the performance of the proposed bees algorithm. Table I lists the parameters utilized in the suggested bees algorithm.

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

TABLE I. THE PARAMETERS VALUES OF THE PROPOSED BEES ALGORITHM

No	Parameter	Value
1	Bees population size *bs*	100
2	Maximum generation *mg*	40
3	Number of bees movement *bm*	4
3	Neighborhood range *nr*	0.3*(token length)
4	Elite bees' size *es*	6
5	Evaluation criteria *e* Character matching Length matching PoS (part of speech) matching Meaningful term	$\begin{pmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.6 \end{pmatrix}$
6	Number of generated digits tokens *dt*	48
7	Number of changing digits in the generated token *dl*,	Token length
8	Number of generated special characters tokens *st*	48
9	Number of changing special characters in the generated token *sl*	Token length

The suggested bees algorithm evaluated a variety of parameter values before settling on the ones that deliver the optimum results for the proposed system. The parameters that have been examined with a variety of values are as follows:

- Bees population size *bs*: The suggested bees algorithm tested a variety of population sizes (40, 60, 80, and 100), the generation size (100) was chosen.
- Maximum generation *mg*: There were no improvements in results after 40 rounds, even using numerous iterations (10,20,30,40...,100). As a result, the alphabet token was given the maximum possible round number (40).
- Neighborhood range *nr*: Many different sizes of the change in token during bee movements have been tested. Tested sizes were (1 character, 2 characters, 0.25*(token length), 0.3*(token length), and 0.5*(token length)), the changing size (0.3*(token length)) was chosen.
- Evaluation criteria *e*: Many values have been tested for the evaluation criteria (Character matching, length matching, PoS (part of speech) matching, and Meaningful term), (0.3, 0.2, 0.2, 0.3) & (0.4, 0.1, 0.1, 0.4) & (0.3, 0.1, 0.1, 0.5) & (0.2, 0.1, 0.5) & (0.2, 0.1, 0.5) & (0.2, 0.1, 0.5) & (0.2, 0.1, (0.2, 0.2, 0.1, 0.5). The chosen evaluation critera values were (0.2, 0.1, 0.1, 0.6).

VI. RESULTS AND DISCUSSIONS

This research part will contain experimental results, discussion, and comparison of the proposed and the previous honeyword generating approaches.

A. Experimental Results

Because the hacker's primary goal is to guess the true password, the suggested bees algorithm examined a variety of password tokens, including the alphabet token, which is the most crucial token. Table II shows the experimental results utilizing the parameters listed in Table I. To generate the alphabet token, the bees algorithm strategy will be utilized to solve the problem; 100 tokens will be created, but only the top six will be displayed in Table II. Simple random generators will be used to generate the digits and special characters tokens,

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

with character changes occurring at random with the same root token length, resulting in six tokens. For a full example, check example1.

TABLE II. EXPERIMENTAL RESULTS OF THE PROPOSED BEES ALGORITHM

	Root Token	Pop-Size/ Max-Gen.	Honeyword Tokens/Approximation Element					
1	**chicken**	100/40	thicken /0.971	whicker /0.942	quicken /0.942	chicle /0.928	flicker /0.914	chiffon /0.914
			blacken /0.914	chicness /0.9125	chickpea /0.9125	chickenpox /0.909	chichi /0.9	chicer /0.9
			chickweed /0.888					
2	**chatbox**	80/30	saltbox /0.914	postbox /0.914	chatter /0.914	cashbox /0.914	chatroom /0.912	chateaux /0.887
		80/30	saltbox/ 0.914	gearbox/ 0.914	chattel/ 0.914	chariot/ 0.914	cashbox /0.914	chatroom/ 0.912
		80/30	whatnot /0.914	chatter /0.914	cashbox /0.914	chatroom /0.912	flatboat /0.887	chateaux /0.887
3	**mustang**	40/40	mutton /0.9	sultan/ 0.9	tussahs /0.885	musas /0.857	mutt/ 0.842	mutant /0.842
		60/40	mustard /0.942	mistake /0.914	Luoyang /0.914	musty /0.885	juntass /0.885	estrange/ 0.862
		80/40	mustard /0.942	sustain /0.9142	mustache /0.912	mutton /0.9	siamang /0.885	musty /0.885
		100/40	sustain /0.914	busying /0.914	bustard /0.914	bushing/ 0.914	luoyang /0.914	mustache /0.912
4	**basebaLL**	100/40	fastbaLL /0.95	pushbaLL /0.924	beanbaLL /0.924	baseborn /0.924	sourbaLL /0.9	bastilLe /0.9
5	**secret**	100/40	secrete /0.957	socket /0.933	septet /0.933	sachet /0.933	secrecy /0.928	secretor /0.924
6	**master**	100/40	waster /0.966	taster /0.966	paster /0.966	mister /0.966	matter /0.966	masker /0.966
7	**starwars**	100/40	starworts /0.922	stalwarts /0.922	stadiums /0.9	slipways 0.9/	/stargazers 0.88	starts/ 0.875
8	**letmein**	100/40	letdown /0.914	letterer 0.887/	mullein /0.885	fitment /0.885	lutetium /0.862	leukemia /0.862
9	**login**	100/40	yogin /0.96	logic /0.96	logia /0.96	logan /0.96	logan /0.96	lysin /0.919
10	**space**	100/40	spice /0.96	spare /0.96	spade /0.96	spite /0.919	spire /0.919	specs /0.919
11	**hbgod**	100/40	hyson /0.88	hoeed /0.88	blood /0.88	wagon /0.88	hogg /0.86	hobo /0.86
12	**odnsf**	100/40	odess /0.919	oddss /0.919	odour /0.88	odder /0.88	oozy /0.82	owns /0.8
13	**lqpamev**	100/40	lipase /0.9	empale /0.871	leprose /0.857	pyramid /0.857	lepanto /0.857	tamale /0.842
14	**7362**	N/A	8611	4826	3649	8068	7121	… 48tokens
15	**814**	N/A	460	240	729	311	359	… 48tokens
16	**64**	N/A	72	79	39	18	17	… 48tokens
17	_^@?	N/A	^{%#	.|;;	}+^(	-$)(_	|;\$	… 48tokens
18	(/~	N/A	_{/	!$+	<?|	\{\	~~|	… 48tokens

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

	Root Token	Pop-Size/ Max-Gen.	Honeyword Tokens/Approximation Element					
19	&>	N/A	.,	@}	,%	*/	<{	… 48tokens

The suggested bees algorithm-generated tokens of various types in Table II. assert the capability to handle any password token type. Token 1 (chicken) demonstrates that the suggested technique may generate a large number of good tokens; 13 tokens passed the 0.6 value with an approximation element over the 0.6 value. Even though the same tokens and Pop-size/Max-gen are used, Token 2 (chatbox) demonstrates that the suggested method generates distinct tokens for each attempt. Token 3 (mustang) displays generated tokens in various Pop-size/Max-gen parameters; there are always good results, but Pop-size=100/ Max-gen=40 generates the best results. Token 4 (basebaLL) demonstrates the suggested algorithm's ability to handle the password's capital letters.

Tokens 5–10 are alphabet tokens that represent several meaningful words. Tokens 11-13 display the generated alphabet tokens for rubbish words. Tokens 14 - 16 are digit tokens. Tokens 17-18 depict tokens with special characters.

B. Comparison

In this section, the proposed honeyword system was compared to the previous honeyword generation methods.

- The proposed bees algorithm honeyword generation approach exceeds earlier strategies in terms of honeyword generation because it enhances the generating process by utilizing its problem-solving characteristics (Robustness, reliability in getting the optimum solution, fast convergence, and ability to maintain exploration and exploitation).
- The proposed bees algorithm enhances the most important three honeyword characteristics (Flatness, DoS resistance, and storage), which aren't always present in the best possible way in earlier honeyword generation approaches. three honeyword characteristics are:
    1. Flatness: The proposed method guarantees perfect flatness unconditionally, with the hacker having a (1/49≈2%) chance of catching the sugarword and a (1-2%=98%) chance of selecting the honeyword. While the hacker's chance of predicting the sugarword in Juels' [15] original honeyword system was (1/20=5%), the suggested method had a lower chance (1/492≈%).
    2. DoS Resistance: A DoS attack denies the system's services by predicting and typing a honeyword. For hackers, the proposed method generates honeywords that are hard to predict.
    3. Storage: whereas the proposed technique saves usernames and sweetwords, some prior generating methods saved more data and information.
- As a comparison between previous honeyword generation methods and the proposed one, the seven most serious problems that honeyword systems face will be illustrated. Then a table will encompass the generating techniques to illustrate which approach experiences the problem and which does not. Getting into a problem represents a weakness but avoiding each problem represents a strength. The proposed honeyword system addresses the seven most common problems of the previous honeyword systems. The following are the seven problems:
    1. The conditional flatness problem: Some conditions must be satisfied to attain perfect flatness, while unconditional flatness states that no condition that is a

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

strength must be met. Most earlier honeyword generating systems only guarantee perfect flatness in select conditions, but the suggested honeyword system guarantees perfect flatness in all cases. The proposed method provides unconditional flatness, as it is not set conditions to the real password to provide perfect flat honeywords. For illustrating follow the Table II Experimental Results of The Proposed Bees Algorithm.

2. Weak DoS resistance problem: The hacker can guess the honeywords, but strong DoS resistance implies the intruder won't be able to guess them. The honeyword proposed method has a strong DoS resistance compared to several of the previous honeyword production systems. The proposed method generates honeywords that are different from the sugarword at least in two tokens (Alphabet, digits, special characters tokens). Therefore, it is so hard for the hacker to guess the honeyword. For illustration follow "Example 1."
3. The problem of storage overhead: More storage space is necessary. The suggested honeyword generating method, unlike many earlier honeyword generation methods, does not necessitate additional storage costs. The proposed method only saves the sweetwords without any additional information or details, such as user information, lists, or indexes.
4. Correlation problem: The presence of a link between the username and the password is a concern. Honeywords can therefore be deduced from the real password. The suggested honeyword method solves the problem by keeping the associated component the same across all honeywords. For example, if the (Username: earth853 & Password: &^56earth), then the generated alphabet tokens are: (earth, earth, earth, earth, earth, earth,).
5. The problem with consecutive and frequent numbers: Is that users prefer numerical patterns that are easier to remember. As a consequence, many users choose numbers in their passwords that are consecutive or frequent, such as '123, 1234, 111, or 2222,' which makes the sugarword easily identifiable. To overcome this problem, the honeyword method proposes a list of the most often repeating and consecutive numbers. If the sugarword has this problem, the suggested algorithm will pick numbers for the honeywords at random from the list. For example, if the (Username: spider & Password: 111$&spider), then the generated digit tokens are: (123, 333,222,789,22, 456, 234,345,555,567,222,66,999,0000,2345,... 48tokens).
6. The problem of a special date: Several users want to put a date in their passwords that is meaningful to them, such as their birthday, anniversary, finest year in school, or any other date that will reveal the sugarword. As a result, the suggested honeyword system will compile a list of the preceding 50 years. If the year numbers are utilized in sugarword, the system will pick years at random from the list to use in honeywords. For example, if the (Username: spider & Password: 1982$&spider), then the generated digit tokens are: (2001,1966,1990,1980,2007, 1977, 1986,,1999,1969,... 48tokens).
7. User's information, security problem: Many of the preceding honeyword-generation systems rely on personal information queries, which need the users to submit personal information and detail for the system to operate. If the system is hacked, personal information may be disclosed and utilized on another system, placing the user at risk. So, utilizing this method as a security risk is considered a weakness, but not using it is a strength. The proposed method does not ask the user to submit any personal information.

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

For the most critical honeyword system problems, Table III compares the proposed bees algorithm with the previous honeyword generating methods.

TABLE III. A COMPARISON IN THE MOST CRITICAL PROBLEMS OF HONEYWORD SYSTEMS

No	Methods	Cond. Flatness problem	Weak DoS resist. problem	Storage overhead problem	Correlation problem	Consecutive and frequent numbers problem	Special date problem	User Info. security problem
1	Proposed bees algorithm	No	No	No	No	No	No	No
2	Chaffing-by-tail-tweaking [15]	Yes	Yes	No	Yes	Yes	Yes	No
3	Chaffing-by -tweaking-digits [15]	Yes	Yes	No	Yes	Yes	Yes	No
4	Simple model [15]	Yes	No	No	Yes	No	No	No
5	Modeling syntax [15]	Yes	No	No	Yes	Yes	Yes	No
6	Chaffing with "tough nuts" [15]	N/A	No	Yes	No	N/A	N/A	No
7	Take-a-tail [15]	No	No	No	No	No	No	No
8	Random pick [15]	Yes	No	No	Yes	No	No	No
9	Hybrid generation methods [15]	Yes	No	No	Yes	Yes	Yes	No
10	Storage-index [16]	Yes	Yes	Yes	Yes	No	No	No
11	PDP [17]	Yes	No	Yes	No	Yes	No	No
12	Evolving password model [18]	Yes	No	No	Yes	Yes	Yes	No
13	User-profile model [18]	Yes	Yes	Yes	Yes	Yes	No	Yes
14	Append-secret model [18]	Yes	No	No	No	Yes	No	No
15	User information method [19]	Yes	No	Yes	Yes	Yes	No	Yes
16	Dictionary attack method [19]	Yes	Yes	No	Yes	No	No	No
17	Generic password list method [19]	Yes	No	No	Yes	No	No	No
18	Shuffling characters method [19]	Yes	Yes	No	Yes	Yes	Yes	No

C. Discussion

The results of the experiments demonstrated that the proposed approach generates passwords using all of its tokens (alphabet, digits, and special characters), particularly the alphabet token, which is difficult to relate to meaningful words. When it came to constructing meaningful words out of meaningful words, the alphabet token generator looked excellent; words most importantly, the system was able to produce meaningful words out of rubbish words.

As a consequence of the analysis of the result, the proposed bees algorithm determines that the population size should be bigger than the maximum generation; the proposed system picks population size=100/maximum generation=40 based on experience. Per the results, population sizes of [40, 60, 80, 100] generate good results, whereas population size=100 produces a superior approximation element. According to the testing results in Table II, the honeywords created have several positive aspects: each password token type

Received 24/January/2022; Accepted 28/March/2022

ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

is generated independently, even though the population size=100/maximum generation is constant for every generating operation, the generated tokens vary from one generating operation to another, can perform a variety of token order password patterns, has great protection against hacker predicting, and can deal with capital letters of alphabet tokens.

The suggested bees algorithm outperforms earlier producing methods in three areas: honeyword generation, honeyword characteristics, and resolving previous techniques difficulties, according to the comparisons. The most essential attribute of the system, flatness, has improved significantly; the suggested system has a superior flatness (1/49≈2%).

VII. CONCLUSIONS

Bees algorithm is a metaheuristic swarm intelligence optimization algorithm that is used to create a novel technique for honeyword generation. It's gone through a lot of adjustments to meet the problem space, and a suggested bees algorithm generates honeywords as solutions in this study. As a consequence, the suggested system effectively employs the bees algorithm for security objectives, namely the detection of password cracking (Honeyword system). The suggested system generates honeywords by utilizing the features of bee algorithm solution creation (Robustness, reliability in getting the optimum solution, fast convergence, and ability to maintain exploration and exploitation).

The suggested bees algorithm enhances the generation procedure, optimizes honeyword characteristics, and solves earlier systems' flaws. The alphabet token is the most significant and challenging token in the sugarword. As a consequence, the suggested system is used to generate the alphabet token in the suggested algorithm approach's solution to the problem. The digit and special characters tokens, on the other hand, are generated using a more easy random approach.

Based on the information acquired from this study of using metaheuristic algorithms, this work presents a honeyword-generating technique and seeks to discover another intelligence approach that may supply optimal solutions. Researchers can experiment with the bees algorithm to see if they can find out how to use it to address multi-objective optimization issues. More study into this area might lead to the discovery and solutions of additional problems that honeywords systems face. In a few places, the bees algorithm might be improved and hybridized with another approach.

REFERENCES

[1] A. A. Mohammed, A. K. Abdul-Hassan, and B. S. Mahdi, “Authentication System Based on Hand Writing Recognition,” in *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, Mar. 2019, pp. 138–142, doi: 10.1109/SCCS.2019.8852594.

[2] T. M. Abed and H. B. Abdul-Wahab, “Anti-Phishing System Using Intelligent Techniques,” in *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, Mar. 2019, pp. 44–50, doi: 10.1109/SCCS.2019.8852601.

[3] Z. A. Genç, S. Kardaş, and M. S. Kiraz, “Examination of a New Defense Mechanism: Honeywords,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10741, G. P. Hancke and E. Damiani, Eds. Cham: Springer International Publishing, 2018, pp. 130–139.

[4] A. B. Kusuma and Y. R. Pramadi, “Implementation of honeywords as a codeigniter library for a solution to password-cracking detection,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 508, no. 1, p. 012134, May 2019, doi: 10.1088/1757-899X/508/1/012134.

[5] T. Win and K. S. M. Moe, “Protecting private data using improved honey encryption and honeywords generation algorithm,” *Adv. Sci. Technol. Eng. Syst.*, vol. 3, no. 5, pp. 311–320, 2018, doi: 10.25046/aj030537.

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 ISSN (Online) 2617-3352

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

[6] S. Palaniappan, V. Parthipan, S. Stewart kirubakaran, and R. Johnson, "Secure User Authentication Using Honeywords," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 31, 2020, pp. 896–903.

[7] Z. A. Genç, G. Lenzini, P. Y. A. Ryan, and I. V. Sandoval, "A Security Analysis, and a Fix, of a Code-Corrupted Honeywords System," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, vol. 2018-Janua, no. Icissp, pp. 83–95, doi: 10.5220/0006609100830095.

[8] T. Nathezhtha and V. Vaidehi, "Honeyword with Salt-Chlorine Generator to Enhance Security of Cloud User Credentials," *Commun. Comput. Inf. Sci.*, vol. 746, pp. 159–169, 2017, doi: 10.1007/978-981-10-6898-0_13.

[9] B. T. Tezel and A. Mert, "A cooperative system for metaheuristic algorithms," *Expert Syst. Appl.*, vol. 165, no. May 2020, p. 113976, 2021, doi: 10.1016/j.eswa.2020.113976.

[10] H. Malik, A. Iqbal, P. Joshi, S. Agrawal, and I. B. Farhad, *Metaheuristic and Evolutionary Computation: Algorithms and Applications*, vol. 916. Singapore: Springer Singapore, 2021.

[11] S. S. Sabry and N. M. Thaker, "Particle Swarm Optimization Based LQ-Servo Controller for Congestion Avoidance," *Iraqi J. Comput. Commun. Control Syst. Eng.*, pp. 63–70, Feb. 2019, doi: 10.33103/uot.ijccce.19.1.8.

[12] S. M. Homayouni and D. B. M. M. Fontes, "Metaheuristic Algorithms," in *Metaheuristics for Maritime Operations*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2018, pp. 21–38.

[13] H. A. H. Akkar, W. A. H., and I. H. Al-Dosari, "PSO Trained Hybrid Intelligent Classifier Using Wavelet and Statistical Features for Pipeline Leak Classification," *Iraqi J. Comput. Commun. Control Syst. Eng.*, pp. 1–9, Feb. 2019, doi: 10.33103/uot.ijccce.19.1.1.

[14] R. A. Mohammed, M. G. Duaimi, and A. T. Sadiq, "Modified Bees Swarm Optimization Algorithm for Association Rules Mining.pdf." Iraqi Journal of Science, 2017, [Online]. Available: https://www.iasj.net/iasj/download/e1498ea57bb18ffd.

[15] A. Juels and R. L. Rivest, "Honeywords: Making Password-Cracking Detectable," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, 2013, no. October 2015, pp. 145–160, doi: 10.1145/2508859.2516671.

[16] I. Erguler, "Achieving Flatness: Selecting the Honeywords from Existing User Passwords," *IEEE Trans. Dependable Secur. Comput.*, vol. 13, no. 2, pp. 284–295, Mar. 2015, doi: 10.1109/TDSC.2015.2406707.

[17] N. Chakraborty and S. Mondal, "On designing a modified-UI based honeyword generation approach for overcoming the existing limitations," *Comput. Secur.*, vol. 66, pp. 155–168, 2017, doi: 10.1016/j.cose.2017.01.011.

[18] A. Akshima, D. Chang, A. Goel, S. Mishra, and S. K. Sanadhya, "Generation of Secure and Reliable Honeywords, Preventing False Detection," *IEEE Trans. Dependable Secur. Comput.*, vol. 5971, no. c, pp. 1–13, 2018, doi: 10.1109/TDSC.2018.2824323.

[19] O. Z. Akif, A. F. Sabeeh, G. J. Rodgers, and H. S. Al-Raweshidy, "Achieving flatness: Honeywords generation method for passwords based on user behaviours," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 3, pp. 28–37, 2019, doi: 10.14569/IJACSA.2019.0100305.

[20] Y. Guo, Z. Zhang, and Y. Guo, "Superword: A honeyword system for achieving higher security goals," *Comput. Secur.*, vol. 103, p. 101689, Apr. 2021, doi: 10.1016/j.cose.2019.101689.

[21] V. R. Pagar and R. G. Pise, "Strengthening password security through honeyword and Honeyencryption technique," *Proc. - Int. Conf. Trends Electron. Informatics, ICEI 2017*, vol. 2018-Janua, pp. 827–831, 2018, doi: 10.1109/ICOEI.2017.8300819.

[22] J. Brindtha, K. R. Hithaeishini, R. Komala, G. Abirami, and U. Arul, "Identification and detecting of attacker in a purchase portal using honeywords," *ICONSTEM 2017 - Proc. 3rd IEEE Int. Conf. Sci. Technol. Eng. Manag.*, vol. 2018-Janua, pp. 389–393, 2017, doi: 10.1109/ICONSTEM.2017.8261414.

[23] Z. A. Genç, G. Lenzini, P. Y. A. Ryan, and I. Vazquez Sandoval, "A Critical Security Analysis of the Password-Based Authentication Honeywords System Under Code-Corruption Attack," in *Communications in Computer and Information Science*, vol. 977, 2019, pp. 125–151.

[24] P. D. Shinde and S. H. Patil, "Secured Password Using Honeyword Encryption," *Iioab J.*, vol. 9, no. 2, SI, pp. 78–82, 2018, [Online]. Available: https://www.iioab.org/IIOABJ_9.2_78-82.pdf.

[25] P. B. Shamini, E. Dhivya, S. Jayasree, and M. P. Lakshmi, "Detection and avoidance of attacker using honey words in purchase portal," in *2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, Mar. 2017, vol. 2018-Janua, pp. 260–263, doi: 10.1109/ICONSTEM.2017.8261290.

[26] N. Chakraborty, S. Singh, and S. Mondal, "On Designing a Questionnaire Based Honeyword Generation Approach for Achieving Flatness," in *2018 17th IEEE International Conference On Trust, Security And*

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*

DOI: https://doi.org/10.33103/uot.ijccce.22.4.15

Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Aug. 2018, pp. 444–455, doi: 10.1109/TrustCom/BigDataSE.2018.00071.

[27] K. Akshaya and S. Dhanabal, "Achieving flatness from non-realistic honeywords," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Mar. 2017, pp. 1–3, doi: 10.1109/ICIIECS.2017.8276120.

[28] M. A. Fauzi, B. Yang, and E. Martiri, "Password Guessing-Based Legacy-UI Honeywords Generation Strategies for Achieving Flatness," *Proc. - 2020 IEEE 44th Annu. Comput. Software, Appl. Conf. COMPSAC 2020*, pp. 1610–1615, 2020, doi: 10.1109/COMPSAC48688.2020.00-25.

[29] L. Baronti, M. Castellani, and D. T. Pham, "An Analysis of the Search Mechanisms of the Bees Algorithm An Analysis of the Search Mechanisms of the Bees Algorithm (preprint)," no. October, 2020, doi: 10.1016/j.swevo.2020.100746.

[30] Ahmed T. Sadiq and N. T. Mahmood, "A Hybrid Estimation System for Medical Diagnosis using Modified Full Bayesian Classifier and Artificial Bee Colony," vol. 55, no. 3, pp. 1095–1107, 2014.

[31] M. A. Nemmich, F. Debbat, and M. Slimane, "An enhanced discrete bees algorithms for resource constrained optimization problems," *Intel. Artif.*, vol. 22, no. 64, pp. 123–134, 2019, doi: 10.4114/intartif.vol22iss64pp123-134.

[32] S. Zeybek, D. T. Pham, E. Koç, and A. Seçer, "An improved bees algorithm for training deep recurrent networks for sentiment classification," *Symmetry (Basel).*, vol. 13, no. 8, pp. 1–26, 2021, doi: 10.3390/sym13081347.

[33] A. M. Sagheer, A. T. Sadiq, and M. S. Ibrahim, "Improvement of scatter search using Bees Algorithm," *6th Int. Conf. Signal Process. Commun. Syst. ICSPCS 2012 - Proc.*, no. December, 2012, doi: 10.1109/ICSPCS.2012.6507943.

[34] A. H. Ismail, N. Hartono, S. Zeybek, M. Caterino, and K. Jiang, "Combinatorial Bees Algorithm for Vehicle Routing Problem," *Macromol. Symp.*, vol. 396, no. 1, 2021, doi: 10.1002/masy.202000284.

[35] S. Kamaruddin, M. Naqiuddin Rosdi, and N. Aiman Sukindar, "Optimization of Drilling Path Using the Bees Algorithm," *Manuf. Technol.*, vol. 21, no. 6, 2021, doi: 10.21062/mft.2021.095.

[36] J. Jamhuri, K. Norizah, M. I. Hasmadi, and A. A. Siti, "Timber transportation planning using bees algorithm," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 463, no. 1, 2020, doi: 10.1088/1755-1315/463/1/012171.

[37] A. T. S. Alobaidi and A. G. Hamad, "Exploration-Balanced Bees Algorithms to Solve Optimization and NP-Complete Problems," vol. 2, no. 1, pp. 108–113, 2012.

[38] A. Tariq Sadiq and A. Ghazi Hamad, "BSA: A Hybrid Bees' Simulated Annealing Algorithm To Solve Optimization & NP-Complete Problems," *Eng. & Tech. Journal*, vol. 28, no. 2, pp. 271–281, 2010.

[39] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "Bee Algorithm A Novel Approach to Function Optimisation," *Tech. Note MEC 0501*, no. June 2016, 2005.

[40] A. H. Ismail, N. Hartono, S. Zeybek, and D. T. Pham, "Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 847, no. 1, 2020, doi: 10.1088/1757-899X/847/1/012027.

Received 24/January/2022; Accepted 28/March/2022
 ISSN (Print) 1811-9212 *ISSN (Online) 2617-3352*