Hindawi
Advances in Human-Computer Interaction
Volume 2022, Article ID 9607550, 10 pages
https://doi.org/10.1155/2022/9607550

Research Article

A Proposed Harmony Search Algorithm for Honeyword Generation

Yasser A. Yasser,[1] **Ahmed T. Sadiq**,[1] **and Wasim AlHamdani**[2]

[1]*Computer Science Department, University of Technology-Iraq, Baghdad, Iraq*
[2]*Information Technology Department, University of the Cumberlands, Williamsburg, KY 40769, USA*

Correspondence should be addressed to Yasser A. Yasser; cs.19.28@grad.uotechnology.edu.iq

Received 16 November 2021; Revised 8 March 2022; Accepted 9 March 2022; Published 25 March 2022

Academic Editor: Francesco Bellotti

The honeyword system is a password cracking detection technique that aims to improve the security of hashed passwords by making password cracking simpler to detect. Many honeywords (false passwords) accompany the sugarword (true password) to form the sweetwords (false and true passwords) for every user. If the attacker signs in using a honeyword, a silent alarm trigger shows that the honeyword system might be compromised. Many honeyword generation techniques are presented; each one has a flaw in the generating process, a lack of support for all honeyword characteristics, and a slew of honeyword problems. The harmony search algorithm (HSA), a metaheuristic intelligence algorithm inspired by music, is used in this article to offer a novel method for generating honeyword. The suggested honeyword generation technique will enhance the generating process, enhance honeyword characteristics, and address prior approaches' shortcomings. This paper will show several previous honeyword generation techniques, explain the suggested one, discuss the experimental findings, and compare the new honeyword generation method with the previous ones.

1. Introduction

Because of its simplicity and memorability, password-based authentication is the most widely recognized authentication method. However, numerous attack methods, such as password cracking, have been used to examine this approach [1, 2]. Password cracking is an uncommon and generally unethical method of retrieving passwords from data maintained or transmitted by a computer system [3].

Honeywords is an easy technique to increase the quantity of "honeywords" (false passwords) connected with each user's account, therefore enhancing the security of hashed passwords and making password cracking simpler to detect [4, 5]. An adversary who obtains entry to the hashed passwords database and reverses the hashing will not determine the real password. If a honeyword is used in the login process, a "silent alert" will be activated [6, 7]. Honeychecker is an additional server that can distinguish between the real and honeywords and is linked to the login server through a secure connection [8, 9]. A metaheuristic is a higher-level process or heuristic used in computer science and mathematical optimization to identify, develop, or choose a heuristic (partial search algorithm) that may offer a suitably good solution to an optimization problem [10]. An optimization issue is a problem in mathematics, computer science, and economics where the goal is to identify the optimum answer out of all the possible ones [11]. The metaheuristic algorithms can be swarm, nature-inspired, physics-based, evaluation-based, or unique solutions [12]. Harmony search algorithm (HSA) is a unique music-inspired algorithm that mimics the improvisation process of musicians aiming to resolve optimization problems by obtaining the optimal solution [13, 14].

The suggested honeyword system proposes to use the harmony search algorithm HSA, a metaheuristic music-inspired intelligence algorithm, to provide a novel technique for generating honeyword. Many adjustments have been made to the HSA to suit the problem nature, honeyword

generation, and treating solutions as honeywords. The suggested system tokenizes the real password and then handles every token type in different techniques. Every token type has its generator (alphabet, digits, special characters generator). Alphabet token generator uses the proposed HSA, while the digits and special characters tokens use simple random generators.

The suggested method chooses the harmony search algorithm to produce honeywords benefiting from the algorithm's characteristics in terms of quick convergence and population diversity, ease of implementation, and fewer parameters adjustment.

The contributions of this paper are many. (1) The proposed system uses the harmony search algorithm offering a novel technique for generating honeyword, (2) the suggested generation technique will enhance the honeyword generating process, support honeyword characteristics, and address prior approaches' shortcomings, (3) the password alphabet token of the proposed generating algorithm provided great results for generating meaningful words from meaningful words; the most interesting is that the proposed algorithm can find meaningful words from rubbish words, (4) the proposed algorithm suggests its evaluation criterion for the generated honeyword (alphabet token), called the approximation factor, and (5) the sugarword cannot be guessed even if the attacker knows one of the sugarword tokens. In sweetwords, every token is redundant six times. So, if the attacker knows one of the sugarword tokens, then the chance of picking the sugarword at random is 1/6(≈17%).

This paper will describe a few honeyword generating techniques, offer a basic explanation for the honeyword technique, illustrate the harmony search algorithm, explain the proposed system with proposed HSA, show the experimental findings, compare with the previous honeyword generation methods, with discussion, and end with the conclusion.

2. Related Works

Over the recent few years, much research has presented honeyword generating methods. In addition, there are several asymptotical researches in this area.

(i) In [15], this research suggests several honeyword generation techniques, including modifying a portion of the password, utilizing a dictionary, adding a tail by the system, honeywords supplied by the system, honeywords supplied by the user, and hybrid approaches. These approaches are divided into two groups based on whether or not they impact the user interface (UI), and each category contains a variety of honeyword generation techniques:

(1) Legacy-UI

(Chaffing-by-tail-tweaking, Chaffing-by–tweaking-digits, Simple model, Modeling syntax, "Tough nuts", Hybrid generation methods).

(2) Modified-UI

(Take-a-tail, Random pick).

(ii) In [16], this approach is known as "Storage-index," and it proposes an alternative approach for the honeyword generation that selects honeywords based on existing user passwords in the system to produce realistic honeywords. Honeywords are still employed in the suggested approach to detecting password cracking. This approach imitates honeywords by utilizing existing passwords rather than generating honeywords and saving them in a password file.

(iii) In [17], PDP stands for Paired Distance Protocol, a novel honeyword generating method with a new user interface. To log in, the user will need three pieces of information: a username, a password, and a password-tail. In addition to the username and password, the user chooses a password-tail of $t > 1$ from a list of (1) alphabetic characters (a-z) and (2) digits upon enrolling (0–9).

(iv) In [18], as new honeyword generation approaches, the "evolving-password model," "user-profile model," and "append-secret model" are suggested.

(1) Evolving-password model: The following two separate computation steps can be utilized to finish the process: (a) Counting the number of times password patterns and tokens are used, and (b) generating honeywords from post frequencies and maintaining frequency lists.

(2) User-profile model: Honeywords are made by merging diverse user-profile data by constructing distinct sets from provided data that contain tokens of various types, such as "alphabet-strings," "digit-strings," and "special-character-strings."

(3) Append-secret model: The system requests the user's username, password, and an additional item, such as e, to produce a random string s that includes numbers, characters, and symbols. The model yields r after running the function $f(p|e|s)$. H $(password|r)$ will be saved in the system's password file.

(v) In [19], Akif et al. suggest a new honeyword generation approach that includes all four ways. As a consequence, the system received four groups of honeywords that are generated from the following:

(1) Existing user information: Creating data with two sections of public personal questions. The first section will concentrate on characters, while the second will concentrate on numbers. Honeywords will be constructed by combining the answers to the first and second sections.

(2) A dictionary attack: The fundamental idea behind generating suitable honeywords after scanning through the dictionary attack is to utilize the actual password with a modification of up to three numbers or characters.

(3) A generic password list: Honeywords selected at random from a collection of the 500 worse passwords make up this honeyword group.

(4) Shuffling the characters: Honeyword is created by combining scrambled characters or digits from the ID user.

3. Honeywords

The honeywords method works by creating honeywords (false passwords) from sugarword (real password), then entering them all as sweetwords into the username and password file, and hashing them all [20, 21]. If the adversary gets plain passwords from hashed passwords, he must guess the real password amongst some of the sweetwords correctly; otherwise, a quiet alert to the system administrator may be fired, signaling that password cracking is feasible [22, 23]. The administrators' actions are dictated by the organization's policies and may include banning, deferring, or notifying the account [24].

Flatness, let z be the adversary's estimated chance of accurately predicting the sugarword. Since an adversary can succeed with a chance of $1/k$ by predicting sugarword randomly, the user's password pi is chosen using the generation Gen(k; pi). The adversary has at least a (1-(1/k)) chance of picking a honeyword if the honeyword is as flat as possible (i.e., 1/k flat) [25, 26]. For example, in the complete flat honeywords, if the sweetwrods $k = 25$, the adversary has a (1/25 = 4%) chance of selecting a sugarword and a (1–4% = 96%) for selecting a honeyword [27].

User login, the honeypot is examined by the login server when a user wants to connect to his account (the administrator makes fake accounts to detect the attack) [28]. If the account is fake, the administrator will get a warning as a possible attack; if the account is legitimate, hash the user's password and compare it to the file of sweetwords before submitting checking to the honeychecker [29].

4. Harmony Search Intelligence Algorithm

The harmony search algorithm (HSA) is a metaheuristic optimization algorithm based on a natural event where a musician looks for the best notes to create perfect harmony, comparable to searching for the best solutions to a problem [30, 31]. The HSA is simple and easy to implement, has a population diversity, converges rapidly to the best solution, and finds a good enough one in an acceptable amount of time. It can find a balance between exploration and exploitation [32, 33]. Random search, harmony memory considering rate (*HMCR*), and pitch adjusting rate (*PAR*) are the three operators that make up the HSA performance process [34].

The improvising of musicians for a pitch commonly needs to follow one of these rules: (1) playing any pitch from memory, (2) playing a neighboring pitch from memory, and (3) producing a completely random pitch from the sound range. This process is mimicked in each variable selection of the HSA: (1) selecting any value from the HS memory, (2) selecting a nearby value from the HS memory, and (3) selecting a completely random value from the potential value range [14, 35]. Algorithm 1 shows the general steps of the algorithm, which may be changed based on the problem encoded.

5. The Proposed Harmony Search Algorithm

The proposed honeyword system suggests using the unique metaheuristic music-inspired harmony search algorithm (HSA) as a novel method for the honeyword generating process. The HSA underwent many changes to appropriate the problem space of honeyword generation and handle its solutions as honeywords. This study chooses the HSA because of its simplicity, easy implementation, diversity, rapid converges, providing the best to a good solution, and supplying a balance between exploration and exploitation.

The proposed honeyword system was adopted for the legacy-UI, which is more convenient to users because it is just required for username and password to enter. The last one involves alphabets, digits, and special characters. 36 sweetwords are used in the proposed system, which means $k = 36$; the adversary has a 1/36(≈3%) chance of successfully selecting the sugarword and has a (1–3% = 97%) probability of selecting a honeyword. The recommendation for the proposed system is that the attacker will not be able to pick the sugarword even if knowing one of its tokens because every token in the sweetwords has been repeated for five times. In this case, the attacker has a 1/6(≈17%) probability of selecting the sugarword.

The proposed honeyword system aims to enhance the generating process, enhance honeyword characteristics, and address problems of prior approaches (detailed discussion in Section 6.2).

The proposed HSA handles the password tokens in a different technique. For each tokens type, there is a different generator (alphabet, digits, special characters generator). These generators are working in parallel. For the alphabet tokens, the HSA constructs its pitch adjustment technique and evaluation criteria.

5.1. Proposed HSA Tokens Generators. As mentioned before, the proposed HSA has three tokens generators that are working in parallel. The alphabet generator is the most important and complicated one depending on the HSA technique in the solution of the problems, whereas the digits and special characters generator is simpler depending on the simpler random generating technique. The generators are as follows.

5.1.1. The Proposed HSA Alphabet Tokens Generator. It is the most important part of the honeyword since it is the attacker's favorite choice for guessing the true password. It is the most complicated generator that depends on the HSA technique in the solution of the problems; the tokens of the password will be treated as pitches. The sugarword's alphabet token will be used as the input for the generators. It is regarded as the seed that is used to produce the honeywords alphabet tokens. Make six copies for the top five tokens generated by the alphabet generator; then divide the 30 tokens into five groups (columns). Each group has six similar

tokens. Six copies of the alphabet seed should be added. As a result, the HSA will include 36 alphabet tokens.

5.1.2. The Proposed HSA Digits Tokens Generator. This generator is based on random generation, and the seed will be the sugarword's digit token. The generator will generate five tokens of the same length as the seed. Make six copies of each of the five generated digit tokens; then divide the 30 tokens into five groups (rows), each with six similar tokens. The HSA will have 36 digit tokens after adding six copies of the digit seed.

5.1.3. The Proposed HSA Special Characters Token Generator. This generator is based on random generation, and the seed will be the sugarword's special characters token. The generator will generate six tokens of the same length as the seed. Make six copies of each of the five special characters tokens; then divide the 30 tokens into five groups (rows), each with six similar tokens. Thus, the HSA will have 36 special characters tokens after adding six copies of the seed of the special character.

5.2. Pitch Adjustment Technique and Evaluation Criteria of Proposed HSA. For the alphabet token, this section presents the adjacent pitches (tokens) generating technique which consists of four operations (insert, delete, translocation, and swap) and the evaluation criteria used by the proposed HSA.

5.2.1. Pitch Adjustment Technique (Adjacent Token Generating Technique) of the Proposed HAS. The change in token should be concerning *bw* (distance bandwidth), representing the maximum pitch adjustment change. The adjacent token generating for the alphabet tokens depends on four operations. For each seed token, four tokens will be generated; then the best one will take the place of the seed token. The four operations are as follows:

(1) Insert: Randomly choose certain character places on the token; then insert random characters.

(2) Delete: Randomly choose certain characters placed on the token and delete them.

(3) Translocation: Pick a character's place on the token at random; then swap them around.

(4) Swap: Choose several character positions on the token at random; then swap those characters.

Example 1. For the proposed HSA that used (bw = 0.3 ∗ (Token length)) during the adjacent pitches (token) generating, if the sugarword alphabet token is (sea) then bw = 0.3 ∗ (3) = 0.9, so 1 character will be changed. The adjacent pitches (tokens) are (sear, sa, aes, tea) in sequence.

5.2.2. Evaluation Criteria of the Proposed HAS. The initial population's alphabet tokens will be evaluated on the seed token that is taken from the sugarword, but the evaluation of the generated alphabet adjacent tokens will be metric on the pitches token as its seed. The proposed HSA has its evaluation criterion for the generated tokens; it is called the approximation factor. The value of the approximation factor is in the range (0, 1), which is calculated as the sum of the four criteria values. Each criterion has a different value as mentioned in Section 5.4. The four criteria are as follows:

(1) Character similarity: The character similarity between the seed token's characters and the produced token's characters.

(2) The length similarity: The lengths of the characters in the seed token and the produced token are similar.

(3) The PoS (part of speech) similarity: The seed token and the produced token are equivalent in terms of PoS.

(4) Meaningful word: Is the token a word in the English language?

5.3. The Proposed HAS Algorithm Steps. The system uses the proposed HSA to generate the honeywords as tokens generating process; the sugarword is tokenized into three separate tokens: alphabet, digits, and special characters, then each one is handled in different generators (alphabet, digits, special characters generator), and then the resulting honeywords are collected with the sugarword to provide the sweetwords. The tokens of the password will be treated as pitches. The proposed HSA is showed in Algorithm 2.

Example 2. For the proposed HSA using the parameters listed in Section 5.4. if the sugarword is (killer6+). The generated sweetwords by the proposed HSA will be as follows.

killer6+	filler6+	kinder6+	kicker6+	jailer6+	dealer6+
killer2[	filler2[	kinder2[	kicker2[	jailer2[	dealer2[
killer1-	filler1-	kinder1-	kicker1-	jailer1-	dealer1-
killer8_	filler8_	kinder8_	kicker8_	jailer8_	dealer8_
killer7{	filler7{	kinder7{	kicker7{	jailer7{	dealer7{
killer7+	filler7+	kinder7+	kicker7+	jailer7+	dealer7+

5.4. Parameters. Many parameters that impact the HSA's performance are used in the suggested honeyword generating system. Table 1 lists the parameters utilized in the HSA.

The proposed HSA is experimented with a variety of parameter values before settling on the ones that offer the greatest performance for the suggested system. The parameters tested with many values are as follows:

(i) The population-size *n*: With the HSA experiment having variety of population sizes (20, 40, 60, and 80), the generation of size (80) was chosen.

(ii) The max-generation *MG*: There were no improvements in results after 30 rounds, although using several iterations (10, 20, 30, 40, . . ., 100). So, the alphabet token was given the maximum generation number (30).

(iii) The distance bandwidth (maximum pitch adjustment change) *bw*: Changing in token during

Step 1: Set up the optimization issue and algorithm parameters (N, $HMCR$, PAR, and bw).
Step 2: Create a harmony memory (HM).
N of harmonies are generated (randomly) in the search space and stored in HM at first.
Step 3: Create a new harmony from the HM.
1st stage: A random number in the range (0, 1) is produced ($rand$).
If $rand > HMCR$, the new harmony's decision variable ($x_{\text{new},j}$) is created at random. Harmony memory rate (HMCR) is an acronym for harmony memory rate, which ranges from (0, 1).
If $rand =< HMCR$ is not specified, one of the harmonics stored in HM is chosen at random, for example, k where $1 =< k =< N$. The matching value of harmony k from HM is then used to choose $x_{\text{new},j}$.
2nd stage: Using a pitch adjustment, the improvised note can be moved to a neighboring value within the range of possibilities. Pitch adjusting rate (PAR) is a parameter in HS that is in the range (0, 1). After 1st stage, a random number that is in the range (0, 1) with uniform distribution is produced to execute the pitch adjustment method ($rand$). If $rand =< PAR$ is true, bw should be used to move the improvised note to an adjacent value. Where bw is a random distance bandwidth (a scalar value), bandwidth equals the maximum pitch adjustment change. If $rand > PAR$, the improvised note remains unchanged.
Step 4. Update the HM.
In HM, compare the new harmony to the worst harmony. If the new one has higher fitness than the poorest one in the HM, it will take its position. Otherwise, it will be removed.
Step 5: Continue using Steps 3 and 4 until the termination condition is met.

Algorithm 1: The general steps of the harmony search algorithm [36].

adjacent pitches generation has been attempted in a variety of sizes (1 character, 2 characters, 0.25 ∗ (token length), 0.3 ∗ (token length), 0.5 ∗ (token length)); the changing size (0.3 ∗ (token length)) was chosen.

(iv) Evaluation criteria *Ec*: Many values have been experimented (0.3, 0.2, 0.2, 0.3) & (0.4, 0.1, 0.1, 0.4) & (0.3, 0.2, 0.1, 0.4) & (0.3, 0.1, 0.1, 0.5) & (0.2, 0.2, 0.1, 0.5) & (0.2, 0.1, 0.2, 0.5) & (0.2, 0.2, 0.1, 0.5) for the evaluation criteria (character similarity, length similarity, PoS (part of speech) similarity, and meaningful word), but the values (0.2, 0.1, 0.1, 0.6) were chosen because they led to the production of meaningful words, which disturb the attacker on guessing the password.

6. Results and Discussions

The experimental results, a comparison between the HSA and the prior honeyword generating method, and discussion will be covered in this study section.

6.1. Experimental Results. The HSA is experimented on a variety of password tokens, including the alphabet token, which is the most significant token because guessing the true password is the attacker's primary goal. Table 2 shows the experimental results, using the parameters listed in Section 5.4. The generating procedure for the alphabet token will be based on the HSA approach in the solution of the problem; 80 tokens will be created, but only the best five will be displayed in the results table. A basic random generator will be used for the digit and special characters tokens, with characters changes occurring at random but with the same seed token length. The generated tokes will be six tokens. For the full example, see Example 2.

Table 2 illustrates the suggested HSA's generated tokens for various token kinds in order to claim the capability to handle any password token type. Token 1 (hunter) demonstrates that the proposed HSA may yield a large number of useful tokens; 16 generated tokens crossed the 0.6 threshold. Token 2 (shadow) displays created tokens in various Pop-size/Max-gen settings; there are usually decent results, but Pop-size = 80/Max-gen = 30 produces the best results. Token 3 (apple) demonstrates how the proposed SSA creates distinct tokens for each try, even when the tokens and Pop-size/Max-gen are the same. Token 4 (fOOtball) demonstrates the proposed SSA's ability to handle the password's capital letters. Tokens 5–10 are alphabet tokens that represent several significant words. Tokens 11–13 display the produced alphabet tokens for trash words. Tokens 14–16 are digit tokens. Tokens 17–19 depict tokens with distinctive characteristics.

6.2. Comparison. A comparison between the proposed honeyword systems included the proposed HSA with the prior honeyword generating methods shown in this section.

The proposed HSA honeyword generation technique is better than earlier honeyword generation methods in terms of honeyword generating because it enhances the generating process benefiting from its characteristics in problem-solving (diversity, rapid convergence, providing the best to a good solution, and supplying a balance between exploration and exploitation).

The proposed HSA enhances the most important honeyword properties (flatness, DoS resistance, and storage), which are not always present in the best possible way in prior honeyword generation techniques. Flatness: The proposed HSA guarantees perfect flatness unconditionally with a 1/36(≈3%) chance for the attacker to correctly pick the sugarword and has a (1–3% = 97%) chance of selecting a honeyword. The recommendation for the proposed HSA is that the attacker has a 1/6(≈17%) probability of selecting the sugarword even with knowing one of its tokens. DoS Resistance: The DoS attack performs by guessing and entering a

```
Parameter
   n pitches size (population-size), HS harmony size (number of pitches that made harmony), HM harmony memory, HMS harmony
memory size (equal to the max-generation), HMCR harmony memory considering rate, PAR pitch adjusting rate, bw distance
bandwidth (the maximum pitch adjustment change), ap number of generated adjacent pitches, Mg max-generation, Ec evaluation
criteria, d number of the generated digits tokens, dl number of digits that changed in the generated token, s number of the generated
special characters tokens, sl number of special characters that changed in the generated token.
Begin
  Tokenization/ * parse the sugarword to the alphabet of, numbers, and special characters token * /
  If the token is an alphabet
    Generate the initial pitches population with n randomly
    Compute the fitness of the population with considering to Ec
    for i = 1 to Mg
      for j = 1 to n
        Generate rand1 in range (0, 1) and HMCR random in range (0, 1)
          if rand < HMCR
            let a pitch selected form the population randomly
            Generate rand2 in range (0, 1) and PAR random in range (0, 1)
            if rand < PAR
            Generate adjacent pitches with ap respect to bw and choose the best adjacent as the pitch
            end if
          else generate pitch randomly
          end if
      end for
        Compute the fitness of the new population with considering to Ec
        Drop the worst pitches of the population generate ones randomly
        Select the best pitches with HS as the harmony then save in HM
    end for
    Return the best harmony in HM as the alphabet honeyword tokens
  end if
  If the token is a digit
    for i = 1 to d
      for j = 1 to dl
        Changes the digits of the token by other digits randomly
      end for
    end for
    Return the d tokens as the digits honeyword tokens
  end if
  If the token is a special character
    for i = 1 to s
      for j = 1 to sl
        Changes the special characters of the token by other special characters randomly
      end for
    end for
    Return the s tokens as the special characters honeyword tokens
  end if
  Collect honeyword tokens
  Provide sweetwords by adding sugarword to honeywords then permutate and hashed the sweetwords
End
```

Algorithm 2: The proposed harmony search algorithm.

honeyword to deny the services of the system. The suggested HSA generates honeywords that the adversary cannot guess. Storage: While the proposed HSA stores usernames and sweetwords, several earlier generating techniques save additional data and information.

Prior honeyword generating systems face several problems. The proposed honeyword system addresses the seven most pressing concerns of honeyword systems. The following are the seven problems:

(i) Conditional flatness problem: It is the satisfaction of some requirements to attain perfect flatness that is regarded as a weakness Unlike unconditional flatness, which indicates not having to meet any conditions, that is considered a strength. On the other hand, most earlier honeyword generating methods give perfect flatness under certain conditions, but the suggested honeyword system guarantees perfect flatness unconditionally.

Table 1: The HSA parameters values.

No	Parameter	Values
1	Pitches size (population-size) n	80
2	Max-generation MG	30
3	Harmony size (number of pitches that made harmony) HS	5
4	Harmony memory size (equal to max-generation) HMS	30
5	Harmony memory considering rate $HMCR$	Random in range (0, 1)
6	Pitch adjusting rate PAR	Random in range (0, 1)
7	Number of generated adjacent pitches ap	4
8	Distance bandwidth (maximum pitch adjustment change) bw	0.3×(Token length)
9	**Evaluation criteria Ec** Character similarity Length similarity PoS (part of speech) similarity Meaningful word	$\begin{pmatrix} 0.2 \\ 0.1 \\ 0.1 \\ 0.6 \end{pmatrix}$
10	Number of the generated digits tokens d	5
11	Number of digits that changed in generated token dl	Token length
12	Number of the generated special characters tokens s	5
13	Number of special characters that changed in generated token sl	Token length

Table 2: Experimental results of the proposed HSA.

	Seed token	Pop-size/Max-gen	Honeyword tokens/approximation factor				
1	**Hunter**	80/30	Punter/0.966 Putter/0.933 Mincer/0.9 Hinted/0.833	Hunger/0.966 Punier/0.933 Jitter/0.9	Bunter/0.966 Luster/0.933 Hungry/0.9	Sunder/0.933 Lunger/0.933 Handler/0.842	Sinter/0.933 Hitter/0.933 Puntes/0.833
2	**Shadow**	20/30	Hallow/0.866	Shaaban/0.842	Salw/0.833	Slaw/0.833	Skaw/0.833
		40/30	Shaver/0.9	Slalom/0.9	Khaddar/0.871	Hallow/0.866	Shaaban/0.842
		60/30	Cracow/0.9	Slalom/0.9	Shad/0.9	Shallow/0.871	Shannon/0.842
		80/30	Slalom/0.9	Shaver/0.9	Shad/0.9	Shampoo/0.871	Shallow/0.871
3	**Apple**	80/30	Apace/0.919	Anile/0.919	Angle/0.919	Applier/0.887	Apron/0.88
		80/30	Ample/0.96	Apace/0.919	Anile/0.9199	Apogee/0.883	Spoke/0.88
		80/30	Anile/0.919	Applier/0.887	Apogee/0.883	Spoke/0.88	Apron/0.88
4	**fOOtball**	80/30	sOftball/0.924	gOOfball/0.9	fOOthill/0.9	fOretell/0.875	fOOtman/0.862
5	**Superman**	80/30	Superfine/0.922	Supermen/0.875	Cesarean/0.875	Sumer/0.8624	Sumerian/0.85
6	**Pokemon**	80/30	Pokeweed/0.887	Poker/0.885	Pokey/0.885	Power/0.857	Cowpoke/0.828
7	**Babbygirl**	80/30	Babytalk/0.9	Babyhood/0.9	Babysitter/0.88	Backfield/0.877	Basilisk/0.875
8	**Pepper**	80/30	Popper/0.966	Peeper/0.966	Hepper/0.966	Temper/0.933	Pipped/0.933
9	**Chocolate**	80/30	Iconolatry/0.89	Chowchowl/0.855	Chordate/0.855	Cholesterol/0.836	Cockateel/0.822
10	**Jacket**	80/30	Packet/0.966	Market/0.933	Cackle/0.933	Cachet/0.933	Sallet/0.9
11	**Kebvtco**	80/30	Greatcoat/0.844	Ketch/0.828	Kabob/0.828	Sketchy/0.7999	Jukebox/0.7999
12	**Yothd**	80/30	Moth/0.9	Hoth/0.9	Goth/0.9	Voter/0.88	Roped/0.88
13	**Nusi**	80/30	Nuss/0.95	Nisi/0.95	Suss/0.9	Rust/0.9	Puss/0.9
14	**9368**	N/A	1649	0576	2382	8843	1535
15	**314**	N/A	237	943	971	112	001
16	**52**	N/A	29	44	41	87	63
17	‘ ∗ !	N/A	^_&’	.,@ =	;).,	! ∗ ~(	(]:$
18	/[&	N/A	(`,	} ∗ ”	~| ∗	_,>	`/
19	)<	N/A	#]	;&	∗ -	?|	>&

(ii) Weak DoS resistance problem: The attacker can predict the honeywords, whereas strong DoS resistance implies the adversary cannot guess the honeywords. Many of the earlier honeyword generating techniques have a weak DoS resistance, but the suggested honeyword system has a strong DoS resistance.

(iii) Storage overhead problem: It is the need for more storage space. Many earlier honeyword generating techniques require additional storage costs, but the suggested honeyword system does not.

(iv) Correlation problem: The presence of a correlation connecting username and password is a problem. As a result, the real password may simply be determined from honeywords. The suggested honeyword system solves the problem by keeping the correlated component the same in the honeywords.

Table 3: A comparison in most pressing problems of honeyword systems.

No.	Methods	Cond. flatness problem	Weak DoS resist. problem	Storage overhead problem	Corre. problem	Cons. and frequent numbers problem	Special date problem	User info. security problem
1	Proposed HSA	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	Yes	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

(v) Consecutive and frequented numbers problem: Users prefer rememberable numerical patterns. Thus, many chose to use consecutive or frequented numbers in their passwords, such as '123,' 1234, 111, or 2222,' which results in the sugarword being recognized. The suggested honeyword system provides a list of the most frequented and consecutive numbers to solve this problem. If the sugarword contains consecutive or frequented numbers, the algorithm will select numbers from the list at random for the honeywords.

(vi) Special date problem: Several people like to put a date in their passwords related to their birth dates, anniversary, the greatest year in school, or any other comparable dates that will reveal the sugarword. As a result, the suggested honeyword system will generate a list of the last 50 years. The system will select years at random from the list to put in the honeywords if the year's number is shown in sugarword.

(vii) User information security problem: Many of the preceding honeyword generating approaches rely on personal knowledge-based questions, which need users to supply personal information and detail for the methods to create honeywords. If the system is hacked and personal information is exposed, it might be utilized on another system, posing a risk to the user. As a result, employing this approach is a security concern when viewed as a weakness, but not utilizing it is a strength. Therefore, the suggested honeyword system does not need the user to provide any personal information.

Table 3 shows a comparison between the proposed HSA and prior honeyword generation methods in the most critical honeyword system problems.

7. Discussion

The results of the experiments showed that the proposed method effectively produces passwords with all of its tokens (alphabet, digits, and special characters), particularly the alphabet token, with its difficulties in relating to meaningful phrases. The alphabet token generation technique had excellent results in terms of creating meaningful words from meaningful words; perhaps most notably, the system was able to create meaningful words from rubbish words. As a consequence of the analysis of the results, the proposed HSA determines that the Pop-size should be more than the Max-gen; as a result, the proposed system picks Pop-size = 80/Max-gen = 30 based on experience. The results reveal that

Pop-sizes of 20, 40, 60, and 80 produce good results, while Pop-size = 80 produces a better approximation factor. The produced honeywords have a lot of desirable qualities, according to the results. (1) Independent tokens generation: it generates each password token type separately. (2) Different solutions generation: Even if the Pop-size/Max-gen is set to (80/30), every generating operation generates distinct honeywords. (3) Manipulation of several password patterns: it can carry out a variety of token order password patterns. (4) Sweetwords with a high level of security: They have strong security against attacker guessing. (5) Capital letters handling: it can carry out the capital letters of alphabet tokens.

The comparisons between the proposed HSA and prior generating techniques demonstrate that the current approach is superior in three dimensions: Honeyword producing process, honeyword characteristics, and resolving previous method problems. The essential property, flatness, shows a significant improvement for the suggested system; the proposed system has a better flatness 1/36(≈3%). Furthermore, even if he knows one of the sugarword tokens, the adversary has a 1/6(≈17%) probability of picking the sugarword.

8. Conclusion

The suggested system uses the harmony search algorithm, a unique metaheuristic music-inspired algorithm to offer a novel approach for the honeyword generation process, which is modified numerous times to match the problem. Furthermore, it effectively employs an intelligence algorithm (HSA) for security reasons, namely, password cracking detection system (honeyword system). The proposed HSA enhances the generating process, satisfies honeyword characteristics, addresses prior approaches' shortcomings, generates meaningful words from rubbish words, and suggests its evaluation criterion called the approximation factor.

The alphabet token is the most significant and complicated part of the sugarword. Therefore, the suggested system is used for the proposed HSA technique to generate the alphabet token in the solutions of the problems. On the other hand, the digit and special characters tokens depend on a simple random generating technique.

A kind of limitation can infect the proposed method if the initial population is not well-diversified, which may lead to more iterations of implementation.

Employing the knowledge gained from this research of using metaheuristic algorithms, this paper makes suggestions for honeyword generating techniques and seeks to identify another intelligence methodology that may give ideal solutions (honeywords). Researchers can use HSA in their study and attempt to figure out how to use it for solving multiobjective optimization issues. Further study in this subject may aim towards identifying and resolving other problems that honeywords systems face, with few places where HSA might be improved and hybridized with another algorithm.

Data Availability

The research used a new factor to measure the approximation between the generated word (honeyword) and the original word (password). So, the results are dedicated to the research and cannot be supported by other data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. A. Mohammed, A. K. Abdul-Hassan, and B. S. Mahdi, "Authentication system based on hand writing recognition," in *Proceedings of the 2019 2nd Scientific Conference of Computer Sciences (SCCS)*, pp. 138–142, Baghdad, Iraq, March 2019.

[2] J. Bozeman, "Cyber security essentials," *Control Engineering*, Auerbach Publications, vol. 62, no. 1, pp. 90-91, 2015.

[3] A. Kadhim and H. Imad Mhaibes, "A new initial authentication scheme for kerberos 5 based on biometric data and virtual password," in *Proceedings of the ICOASE 2018-International Conference on Advanced Science and Engineering*, pp. 280–285, Baghdad, Iraq, October 2018.

[4] Z. A. Genç, S. Kardaş, and M. S. Kiraz, "Examination of a new defense mechanism: honeywords," in *Lecture Notes in Computer Science*, G. P. Hancke and E. Damiani, Eds., vol. 10741, pp. 130–139, Springer International Publishing, Cham, Switzerland, 2018.

[5] A. B. Kusuma and Y. R. Pramadi, "Implementation of honeywords as a codeigniter library for a solution to password-cracking detection," *IOP Conference Series: Materials Science and Engineering*, vol. 508, no. 1, Article ID 012134, May 2019.

[6] T. Win and K. S. M. Moe, "Protecting private data using improved honey encryption and honeywords generation algorithm," *Advances in Science, Technology and Engineering Systems Journal*, vol. 3, no. 5, pp. 311–320, 2018.

[7] N. Chakraborty and S. Mondal, "Towards improving storage cost and security features of honeyword based approaches," *Procedia Computer Science*, vol. 93, pp. 799–807, 2016.

[8] S. Palaniappan, V. Parthipan, S. Stewart kirubakaran, and R. Johnson, "Secure user authentication using honeywords," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 31, pp. 896–903, 2020.

[9] I. Erguler, "Some remarks on honeyword based password-cracking detection," *IACR Cryptol. ePrint Arch.*vol. 2014, p. 323, 2014, https://eprint.iacr.org/2014/323.pdf.

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

[11] B. T. Tezel and A. Mert, "A cooperative system for metaheuristic algorithms," *Expert Systems with Applications*, vol. 165, Article ID 113976, 2021.

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

[13] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[14] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36-38, pp. 3902–3933, 2005.

[15] A. Juels and R. L. Rivest, "Honeywords," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security-CCS'13*, pp. 145–160, Berlin, Germany, October 2015.

[16] I. Erguler, "Achieving flatness: selecting the honeywords from existing user passwords," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 284–295, 2016.

[17] N. Chakraborty and S. Mondal, "On designing a modified-UI based honeyword generation approach for overcoming the existing limitations," *Computers & Security*, vol. 66, pp. 155–168, 2017.

[18] A. Akshima, D. Chang, A. Goel, S. Mishra, and S. K. Sanadhya, "Generation of secure and reliable honeywords, preventing false detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 5, pp. 757–769, 2019.

[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," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 3, pp. 28–37, 2019.

[20] J. Brindtha, K. R. Hithaeishini, R. Komala, G. Abirami, and U. Arul, "Identification and detecting of attacker in a purchase portal using honeywords," in *Proceedings of the 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pp. 389–393, Chennai, India, March 2017.

[21] 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," *Communications in Computer and Information Science*, vol. 977, pp. 125–151, 2019.

[22] 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*, pp. 83–95, Icissp, Funchal, Portugal, January 2018.

[23] L. Catuogno, A. Castiglione, and F. Palmieri, "A honeypot system with honeyword-driven fake interactive sessions," in *Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 187–194, Amsterdam, Netherlands, July 2015.

[24] T. Nathezhtha and V. Vaidehi, "Honeyword with salt-chlorine generator to enhance security of cloud user credentials," *Communications in Computer and Information Science*, vol. 746, pp. 159–169, 2017.

[25] K. S. M. Moe and T. Win, "Improved hashing and honey-based stronger password prevention against brute force attack," in *Proceedings of the 2017 International Symposium on Electronics and Smart Devices (ISESD)*, pp. 1–5, Yogyakarta, Indonesia, October 2017.

[26] P. B. Shamini, E. Dhivya, S. Jayasree, and M. P. Lakshmi, "Detection and avoidance of attacker using honey words in purchase portal," in *Proceedings of the 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pp. 260–263, Chennai, India, March 2017.

[27] P. D. Shinde and S. H. Patil, "Secured password using honeyword encryption," *The IIOAB Journal*, vol. 9, no. 2, pp. 78–82, 2018, https://www.iioab.org/IIOABJ_9.2_78-82.pdf.

[28] A. Karthik and M. D. Kamalesh, "Rat trap: inviting, detection & identification of attacker using honey words in purchase portal," in *Proceedings of the 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM)*, pp. 130–132, Chennai, India, March 2017.

[29] A. Juels, "A bodyguard of lies," in *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies-SACMAT '14*, pp. 1–4, Ontaria, Canada, February 2014.

[30] Q. Zhu and X. Tang, "An ameliorated harmony search algorithm with hybrid convergence mechanism," *IEEE Access*, vol. 9, pp. 9262–9276, 2021.

[31] L. Fu, H. Zhu, C. Zhang, H. Ouyang, and S. Li, "Hybrid harmony search differential evolution algorithm," *IEEE Access*, vol. 9, pp. 21532–21555, 2021.

[32] D.-W. Kang, L.-P. Mo, and K.-Q. Zhou, "An improved harmony search algorithm with segmented search," *IOP Conference Series: Materials Science and Engineering*, vol. 864, no. 1, Article ID 012063, 2020.

[33] A. K. Al-Shamiri, A. Sadollah, and J. H. Kim, "Harmony search algorithms for optimizing extreme learning machines," *Advances in Intelligent Systems and Computing*, vol. 1275, pp. 11–20, 2021.

[34] I. F. Faeq, M. G. Duaimi, and A. T. Sadiq Al-Obaidi, "An efficient artificial fish swarm algorithm with harmony search for scheduling in flexible job-shop problem," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 8, pp. 2287–2297, 2018.

[35] M. Dubey, V. Kumar, M. Kaur, and T.-P. Dao, "A systematic review on harmony search algorithm: theory, literature, and applications," *Mathematical Problems in Engineering*, vol. 2021, Article ID 5594267, 22 pages, 2021.

[36] A. Askarzadeh and E. Rashedi, "Harmony search algorithm: basic concepts and engineering applications," *Intelligent Systems*, IGI Global, Hershey, PA, USA, 2017.