 ISSN:2302-8149
e-ISSN:2540-9719

Performance Evaluating of Honeyword Generation Methods: Traditional versus AI

[1]Shahad A. Saadullah*, [2]Saja J. Mohammed
[1,2]Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Iraq
*e-mail: Shahad.23csp41@student.uomosul.edu.iq

(***received*****:** 16 January 2025, ***revised*****:** 10 February 2025, ***accepted*****:** 10 February 2025)

Abstract

As the number of people using computers has increased, the risks to digital systems have increased too. That has necessitated the development of new techniques to defend against hackers. Authentication systems based on text passwords are an example of digital systems that face risks through internet use. Therefore, it was necessary to provide security and reliability to users while protecting their passwords. In the authentication database, A technique known as honeywords was used. In the digital world, honeywords are a popular technique used to enhance the security of users' actual passwords and are an additional strong layer of security. The benefit of this technique is its ability to detect unauthorized access attempts to the systems. In this paper, three of the most popular techniques for honeyword generation and other well-known intelligence algorithms are put in comparison using some evaluation metrics to study the performance of each one. The results illustrate a contrast in the performance of different techniques and intelligence algorithms based on the Hamming distance value.

Keywords: honeywords, honeywords generation, honey checker, passwords security

1. Introduction:

The world has undergone a tremendous digital revolution in recent years, which has resulted in the digitalization of most activities and the network-based flow of data and information. Speed and efficiency have significantly increased as a result, but new security threats have also been brought about, most notably the possibility of user data being lost to hackers or other unauthorized parties. Although each user is given a unique password or identity to improve security, hackers can still use a variety of methods to obtain these credentials [1][2][3]. Enhancing and improving password generation over time is crucial since password security is a crucial issue. Information is protected from unauthorized users using password protection measures. Brute-force methods can be used by an attacker who has stolen a file containing hashed passwords to look for a password "p" whose hash value "H(p)" corresponds to the user's password's stored hash value. Because of this, the attacker can pretend to be the user [3].

Information is protected from unauthorized users with the aid of password protection. Since compromised passwords can result in a plethora of potential cyberattacks, the disclosure of password files is a significant security concern that has affected countless people and businesses, including Adobe, eHarmony, LinkedIn, Yahoo, and others. According to these breaches, a lot of big businesses employ shoddy hashing methods, which makes it simpler for hackers to decipher user passwords [4]. Two essential factors need to be considered while creating a safe system to successfully address these security issues. First and foremost, to greatly reduce the likelihood that malevolent actors would be able to deconstruct the hashes, passwords must be saved in the database using a robust and appropriate hashing technique. Second, it is essential to promptly detect password file intrusions to enable suitable corrective measures [4].

Using a special password known as "Honeywords," which is a decoy password (a phony password) applied to each account record in a credential database, is one method for maintaining password confidentiality. The idea behind honeywords is that an attacker who gained access to the credentials database is the only person who can enter the honeywords created for a genuine user's account because the legitimate user is unaware of them. As a result, attempts to log in using

 ISSN:2302-8149
e-ISSN:2540-9719

honeywords ought to be regarded as strong proof of a database breach. The workings of honeywords are as follows [1][2][5].

When an attacker attempts to compromise the database, the situation is altered. Let's say the hacker can reverse the hash that is securing the database's credentials. The attacker will receive a list of passwords as a result, but they won't be able to tell which are legitimate and which are honeywords. The honey checker will sound an alarm and notify the administrator if the attacker enters a password that has been marked as a honeyword. The nature of honeywords makes this technique very successful in identifying database breaches. [1][2][6].

Infrequently used by authentic users, honeywords are complex, easily cracked passwords. Because honeywords are rarely used when properly chosen, using them as a login password is a clear sign that the password hash file has been hacked [1][2][6]. It offers a practical and affordable way to improve security by using honeywords. Furthermore, creating honeywords doesn't take up a lot of storage space. Cybersecurity precautions are further enhanced by turning on an alert system to inform administrators in the case of a breach [7].

Data is an important component of authentication systems. It plays a pivotal role across many disciplines. Its use is prevalent in both large and small businesses, financial institutions, and various government frameworks. Advances in the fields of internet and communications are a major driver for its implementation across multiple sectors. There is an imperative need to protect this data and maintain its confidentiality. Therefore, it is necessary to come up with various encryption methodologies and algorithms designed to achieve the highest level of security that researchers have developed and refined over time. All these methodologies and algorithms work to enhance and protect data from cyber threats, thereby protecting users' information and enhancing their trust[8][9][10][11][12].

2. Literature Review:

In 2018, scholarly investigations focused on the concept of "stealth passwords," a concept coined by respected researcher Professor Ronald L. Rivest in collaboration with Ari Jules of RSA Labs, as a tactical framework aimed at mitigating the inherent security vulnerabilities of database systems. This initiative was specifically designed to address the widespread challenge faced by many systems that rely on inadequate passwords, which in turn leads to a variety of security breaches; therefore, stealth passwords were designed as a mechanism to increase the identification of such breaches and provide system administrators with rapid notifications and alerts regarding potential security incidents within their databases [1].

In 2018, research presented advanced methods for generating honeywords, especially "enhanced password" and "user profile" models, to improve and enhance the security of the system. The effectiveness of these models was also tested against various attack vectors, such as brute force attacks. The results of this research showed that passwords enhance security by updating the login frequency data, making it difficult for attackers to predict honeywords. In addition, using a dataset containing a large number of unique passwords, the proposed methods maintained a high degree of flatness, which enhanced the security of passwords while reducing the memory burden on users. These researches demonstrated the need to balance security and usability in generating honey passwords while taking into account user behavior and password selection patterns [15].

In 2018, the research divided honey password generation methods into two main groups and discussed several honey password generation methods such as "Tugh nuts" and "hybrid methods" as well as emphasizing the importance of generating honeywords by making the real password cracking detectable [3].

In 2021, research focused on generating honey passwords with additional security measures, such as providing the attacker with fake files, misleading him into believing that he had successfully compromised the system. The goal of the research was to enhance the security of passwords while protecting password databases. It also aimed to use fake passwords and reduce storage costs [6].

In 2022, the proposed honey word generation method uses the bee algorithm, a swarm intelligence optimization technique, to generate honeywords. This method was able to get rid of previous limitations, such as conditional flatness, denial-of-service resistance, storage overhead, and user information security issues. Experimental results show that the method achieves perfect flatness

 ISSN:2302-8149
e-ISSN:2540-9719

and strong denial-of-service resistance, which improves the overall performance of the honey word system. Future research may explore additional applications of the bee algorithm in optimization problems.[17]

In the year 2022, an advanced honey word generation methodology predicated on the Harmony Search Algorithm (HSA) was developed, which enhances the techniques employed for generating analogous terms while concurrently addressing prior limitations. The merits of this algorithm include rapid convergence, as well as the capacity to maintain an equilibrium between exploration and exploitation within the context of optimization problems. HSA incorporates more sophisticated generators for alphabetic characters, whereas the generators for numeric and special characters remain random as previously established. Another notable benefit is its proficiency in augmenting the non-discovery and accurate guessing of the correct password. This algorithm has significantly refined the methodology for generating honeywords and has escalated the challenge associated with deducing actual terms [20].

In the year 2023, the study introduced an innovative methodology that employs the meerkat clan algorithm, which is fundamentally predicated on the tenets of collective intelligence and collaborative problem-solving. This algorithm operates by emulating the behavioral patterns of meerkats to augment the efficacy of honey word generation. It is noteworthy that the investigation incorporated WordNet during the process of generating counterfeit passwords, which significantly contributed to enhancing the diversity of honey word creation as well as its reliability attributes. This research further exhibited unequivocal outcomes regarding the strength, security, and adaptability of the generated words, in addition to their robustness against various forms of attacks [19].

A novel technique for creating similar phrases based on the Separate Swarm Algorithm (SSA) was proposed in 2023. This method addresses the previously identified constraints and improves the production of related phrases by imitating natural swarming behavior. The benefits of this approach, which increases flatness and resistance to denial-of-service attacks, were described in the study. Notably, it tackles security concerns such association troubles and storage expenses. The fact that it strikes a balance between exploration and exploitation is another significant aspect. Regarding the traits of related words, it enhances their traits by using easier techniques for certain symbols. Although this algorithm works well, it lacks diversity, which makes it run through more iterations. Future researchers can use this technique in conjunction with other techniques to enhance and improve generation [18].

In 2023, the challenges of balancing false positives and false negatives were highlighted, especially with human-selected passwords. After analyzing honey word generation strategies, it was noted that the level of protection for passwords generated by administrators was low. The most successful approach was to use the same generator as the user, indicating the need for further study into honey word generation tactics. In this study, a number of honey word generation methods were examined and evaluated, but these systems struggled to maximize the properties of honeywords and find a balance between exploration and exploitation. The use of swarm intelligence techniques, such as the minus algorithm, to generate more secure and effective honeywords was necessary to improve the system [2].

3. Honeyword generation Methods:

Honeywords can be generated with many methods which vary from simple to complex ones. These methods can be divided into two categories:

3.1 Generating Techniques:

The traditional techniques that are used to generate honeywords And Table 1 shows the generation process using traditional techniques.

Table 1. A practical samples of generating honeywords with some traditional techniques

Tech.	pass1	pass2	pass3	Pass4
Original password	c88denum5&3&	abc1426*^sha	st99y!away!s	shahad321!@k

ISSN:2302-8149
e-ISSN:2540-9719

Tweaking tail	c88denum5&3) c88denum5&3! c88denum5&3# c88denum5&3. c88denum5&3- c88denum5&3@ c88denum5&3$ c88denum5&3(c88denum5&3/ c88denum5&3*	abc1426*^shq abc1426*^she abc1426*^shf abc1426*^shg abc1426*^shb abc1426*^shw abc1426*^shr abc1426*^sht abc1426*^shj abc1426*^shk	st99y!away!k st99y!away!f st99y!away!g st99y!away!x st99y!away!d st99y!away!u st99y!away!v st99y!away!j st99y!away!p st99y!away!o	shahad321!@w shahad321!@e shahad321!@y shahad321!@a shahad321!@v shahad321!@u shahad321!@t shahad321!@f shahad321!@q shahad321!@h
Tweaking digits	c98denum5&3& c58denum5&3& c68denum5&3& c78denum5&3& c48denum5&3& c18denum5&3& c28denum5&3& c08denum5&3& c38denum5&3&	abc9426*^sha abc0426*^sha abc8426*^sha abc7426*^sha abc5426*^sha abc2426*^sha abc6426*^sha abc4426*^sha abc3426*^sha	st29y!away!s st19y!away!s st89y!away!s st79y!away!s st59y!away!s st39y!away!s st69y!away!s st09y!away!s st19y!away!s	shahad921!@k shahad521!@k shahad821!@k shahad621!@k shahad121!@k shahad221!@k shahad021!@k shahad421!@k shahad721!@k
Modeling syntax	(bbd3nums&e&	@8c1426*^5h@	5799!y@w@y15	5h@h@d321k
Tech.	Pass1	Pass2	Pass3	Pass4
Original password	wa7ermel0nsh	lionnum4sha%	tajmahal*23+	elf3frazan9@
Tweaking tail	wa7ermel0nsy wa7ermel0nsw wa7ermel0nsk wa7ermel0nsc wa7ermel0nsq wa7ermel0nsa wa7ermel0nsn wa7ermel0nsz wa7ermel0nse wa7ermel0nsef	lionnum4sha/ lionnum4sha) lionnum4sha” lionnum4sha+ lionnum4sha(lionnum4sha$ lionnum4sha’ lionnum4sha! lionnum4sha, lionnum4sha.	tajmahal*23(tajmahal*23’ tajmahal*23. tajmahal*23/ tajmahal*23* tajmahal*23$ tajmahal*23(tajmahal*23- tajmahal*23# tajmahal*23”	elf3frazan9- elf3frazan9% elf3frazan9’ elf3frazan9$ elf3frazan9* elf3frazan9(elf3frazan9+ elf3frazan9! elf3frazan9# elf3frazan9,
Tweaking digits	wa9ermel0nsh wa2ermel0nsh wa4ermel0nsh wa1ermel0nsh wa3ermel0nsh wa0ermel0nsh wa5ermel0nsh wa6ermel0nsh wa8ermel0nsh	lionnum2sha% lionnum5sha% lionnum7sha% lionnum6sha% lionnum1sha% lionnum0sha% lionnum3sha% lionnum8sha% lionnum9sha%	tajmahal*83+ tajmahal*33+ tajmahal*13+ tajmahal*03+ tajmahal*43+ tajmahal*73+ tajmahal*53+ tajmahal*63+ tajmahal*93+	elf8frazan9@ elf7frazan9@ elf5frazan9@ elf9frazan9@ elf6frazan9@ elf31frazan9@ elf0frazan9@ elf4frazan9@ elf2frazan9@
Modeling syntax	w@73rm310n5h	1!0nnum45h@%	7@jm@h@11*23+	31f3fr@z@n9@

These techniques can be classified as:

 ISSN:2302-8149
e-ISSN:2540-9719

3.1.1 **Chaffing by tweaking:** This technique is one of the ways to strengthen the security of systems by applying several strategies to the original passwords. This makes the passwords more complex and harder for attackers to figure out by making the word more confusing [3][4][13].

A. Chaffing-by-tail-tweaking: This technique is used to create "honeywords," which are created by modifying the original password's tail. The final part of a honeyword is slightly different from the original password. Changing the letters, digits, or symbols at the end of the password is the main goal of this technique. The change can be straightforward or intricate, but it must make sure the created honeywords make sense and look like popular passwords. [3][4][13].

B. Chaffing by tweaking digits: This technique involves changing the actual password's numbers by swapping out each one for a different one while leaving the remaining letters and symbols unchanged. The alteration may entail merely shifting the placements of the numerals or altering the digits themselves [3][4][13].

3.1.2 **Chaffing with a Password Model:** This security method relies on a probabilistic model of actual passwords to generate honeywords and is based on modification through a process known as chaffing [3][14].

Modeling syntax: is a technique for creating honeywords that is thought to be reliable. It involves replacing letters with symbols that are like their shapes, making the method more intricate and difficult to decipher but simple to remember because it is like the original word [3][15].

3.2. Generating artificial intelligence:

The employment of artificial intelligence algorithms is one of the most recent approaches of creating honeywords, which are known for their variety and the difficulty with which attackers can differentiate them [16]. Table 2 shows the generation process using intelligent algorithms below:

Table 2. A practical samples of generating honeywords with some AI algorithms

methods	Pass1	Pass2	Pass3	Pass4
Original password	c88denum5&3&	shahad321!@k	wa7ermel0nsh	tajmahal*23+
Bees algorithm	J64ipbky5-3+ h69tcief5$3) bl2ntyqi8’6’ k25uropy2’8(a15npvzc2,0 a29xwvjv3/4(mo1uekpn1’7+ x71clpoz3/6+ y70ezofe1(8+ e32uzqnv3+4- a54ruiil4”4’ q37uocav3!3) o83glwcp9.3% h83bjvyw3!8& i0betkh4$8# q89sklvt9*7! v03jnlhv0-2#	bxqpuv956*!q vyoybm359),s xrjftf936,.w zophdr298(#k qxdrig111,$e ppdqou854#-j ilwkpu545(+i yvaspq893!#a xhynck388*,0 ajtwuu794*)p nbsovso95%&j oqlvga342()q ucvrta405%%o gaelcg151,.t 15krfb645.”1 pfnlzk797$/d vclqbj771(&e	za8ucqes5czm z98ucqes5czm dx0usbar1rdc hh5ipsor7bnh aq3nnhwq5lck bj92diwt0sbe vk6cowxq5aid bu1hhvvm9kz1 qu5xbqiv4squ ap7ehkzd6vkt fb8yfugk6vkt wf6lzjyt4ckb qm7nqcdu8xgk pv2xetyk80mc q12pspag5yhq ac1adqig2pfy jb2tagkb9iwq	hbidmlg’49+ jhdatucd+99” ysevbrk!92* icjihidw+69+ cjvlnlit’15’ pdysfnox(58* mqywjimt”28/ bldrgtve02 zappdrla/76” epiaqvxp$02’ xblxaqyw#93# kwwkhuhe/45! zaildxyb&05- ojhalcvt$42(noaqowvf+34/ xlysnajt)73# mabytmdth+22-

ISSN:2302-8149
e-ISSN:2540-9719

	q16vrade5,2$ e84ipbbe9)5% m97d2pva5!7%	lzdulw965$)d exqtqi423*”r 2k%a.n109-!b	lm0ulada8mi0 ix3zujhh5vle t/72xe612v7n	zavjfiqs-88- olaxdxra’70(9,mo'm&3#31,
Meerkat algorithm	c28d3num1.2# c53d3num6(9% c91d3num9,3- c62d3num4!2. c51d3num3&61 c46d3num9#1% c31d3num9&3' c85d3num1*1/ c30d3num5-2$ c35d3num4#1$ c76d3num9(3) c82d3num4(4" c66d3num6' 3- c79d3num9"3. c01d3num2"4$ c12d3num8*4& c16d3num3-3" c15d3num3-4# c20d3num9’1$ c39d4num0!4$	$h@h@d640)% $h@h@d588/*3 $h@h@d220"*y $h@h@d728$'e $h@h@d257*)v $h@h@d435+'u $h@h@d222-87 $h@h@d315&,f $h@h@d327”+q $h@h@d456$-h $h@h@d380*"k $h@h@d641&) $h@h@d236/k $h@h@d423*.k $h@h@d789"|k $h@h@d036+& $h@h@d978&k $h@h@d613-#k $h@h@d675/% 3h”h”d281’!b	w@33rm315n$y w@93rm33nSw w@33rm311n$k w@63rm312n$c w@53rm310n$q w@23rm318n$ w@23rm318n$n w@53rm311n$z w@13rm318n$3 w@23rm319n$f w@13rm315n$h w@83rm313n$h w@13rm315n$h w@83rm313n$h w@43rm313n$h w@93rm311n$h w@83rm314n$h w@93rm310n$h w@43rm318n$h w.32rm245n5h	7@jm@h@1*75$ 7@jm@h@1+64, 7@jm@h@1)81* 7@jm@h@1$96" 7@jm@h@1.83% 7@jm@h@1!38" 7@jm@h@1(75/ 7@jm@h@1)67" 70jm@h@1&52. EAssweAd223” 70jm@h@1$03# 72jm@h@1)45- 7@jm@h@1$79/ 72jm@h@1(40. 7@jm@h@1/71& 7@jm@h@1’16$ 7@jm@h@1'64- 7@jm@h@1(37! 7@jm@h@1(83) 0.jm(h(5/48%
Harmony algorithm	HScmweeT2332 Saauree1653' Haasro@1e&3) Ee8mrnydo&t, PacDwoUM#20* PxrureUM833" casmin®le22" PacuEeU12262 Ca2deeud5s6% Cardooed#392 carmwou12.t2 Sesmro@12s3& Pxssenele20) E7sdrirm9st& HeruweyMe24& Pesswn@d#&32 CarDoerm5&7& Pessonr1e33* Hasdonem#9d& Ea8m3oed02t2	Sxsmwee188@2 PaAswn2d13te PHAHwo@d#s$ sasmol4T#3ta Hasswnr2e3tv Shamwo@d20U saaupn3d220T PaauoerT82@F HxAHAo761#0 Hesmwny162t2 Sacurnr11!-2 SxsHaDeTstk sacsre@12282 EHcswerd#2@2 EaAsroyd#32 saauAnde2+k saruwn@12!tK shaswerd#2t2 $h@h@d657/% 5aahpeyd#s”k	Sasmp1@L0s®y warmwor12NSw Ea1mrMel22S2 EacmweyL2n02 warmke@14nsq PeamnE10ns2 HasEwoed2NSN Parswl@L#2s2 wasEwo@d0nsE Pe9sRlEd2n0F WasErne1#n0h Sastpoyde202 wAsmror1#ntH walmol@L.220H Pesony1#sth Px@swMrd22S WaaErMr1030h wesEordeN02 wacsRmeT6N02 Hxruoo@1#Nth	PecmAlad%3t) tajMpeAde20# ExrmAneT#10# PAcurl@d*222 Pasuwna1'28* SasMaor1*302 SAcsAhrL#30(EesuworT20t2 Parmwerd#20(EAssweA223” PeaMwh@d*20+ tejuonad*302 PAsschAd#78% EesuohaTe20™ teamaHr1#702 Hacmwo@de33+ PxjsoHel-50+ tasMwoeTes, PAauAn@L#23+ HxsM@n@d+3t-
Method	Pass5	Pass6	Pass7	Pass8
Original password	abc1426*^sha	lionum4sha%	elf3frazan9@	st99y!away!s

ISSN:2302-8149
e-ISSN:2540-9719

Bees algorithm	hs19922*&xg rqy9170*!szr xdc2968+%fwq bxx1650'$vcu htv3845,$¢vor egs3626$%tql bty3723'"jhu e1z8198)(hid cgw1858!)kdx wwb@649(+dsv jok8682()utd tnd8@65)!zcw gwy4453$$ftt zxx7019(,arl ixh1026"&qkw jga8471&.btq zyh9913/!1bh itk2406+'ohg kph6900, !xsn 4!t2437!+6j-	nsvlbzw52wa- vfjkvgq9wok” qojpzcj8fsk! zhydhyx8kzm- dbgrdax0hos, ddntjoj1lml! ugrhuyx1eza” vdthlws4cqg(vtyuvdv6wxm+ sdfhtkm5oxj# lionnum2sha% uzeiqta2ueb. ytatwma7fih& bunvnsm9wpz/ bfvdxkc1qwb/ awtcojp3pei- iyvmqme3rqw- fjldjux6ukq’ cmudfas4thz/ 197wkhx61v!.	sot7pjqnjg8/ gkx2xxuwho4% bww0srbhfo4& hck5gurmgb2) oqv9sgocte5% rau0rdxjva3” jii3xamqlc6/ mln7rpaaca1, cis7sguedc2. teg1kjqvrg4(jsh2youbfk2# mri1jivrew3/ czq4fneyfd6$ x1p5nelcir0/ uww7zyizwd1” rcq9orlfwn9- bfz3ivaslb6/ kqk1mmsdmi7% npe7nuxwp5, 02u9at&k.j9&	Ub43n”cxre+v Oc53u.xfk+e rs67p+jrso%l hy67c’iqpo/i gv27y”ctpq(g fx49k(xohg&d Pg57o’heb)l ew67u&ppff’x ac02s!gqda,i fs860*vgrr.w hz94x.uhlp!l rr14j-ffux!j ij70c%iqfb#g la13f*njlu(m oi56x+nkcx*g qr36z!ztqu-n sj96f&axqu$w so02r$itga%g pl56e”ovst+d 1611c.’l-z+0
Meerkat algorithm	8c7964$.$hh@ 8c1964""$h3@ 8c9848'"$hm@ 8c9742%/¢hr@ 8c2705/*$hn@ 8c7976'!$¢ho@ 8c6042! ,¢hc@ 8c0122+($hf@ 8c3189,&$hx@ 8c7843)&$h1@ 8c2171%$¢h@ 8c9092*#$¢h@ 8c0221*'$h@ 8c6932-'$¢h@ 8c5487/'$¢h@ 8c6623%$$h@ 8c8038,)$h@ 8c3990""$h@ 8c7174()$h@ .1c5585”’1h%	1!@nnum4$h@- 1!@nnum8$hét 1!@nnum2$h@ 1!@nnum3$h@ 1!@nnum3$hés 1!@nnum4$h@! 1!@nnum7$h@\ 1!@nnum3$h@s 1!@nnum3$h@- 1!@nnum@$h@ 1!@nnum9$h@ 1!@nnum5$h@ 1!@nnum2$h@ 1!@nnum6$h@ 1!@nnum8$h@ 1!@nnum3$h@ 1!@nnum8$h@ 1!@nnum3$h@ 1!@nnum5$h@ 7/61nnum85h.)	31f3fr@z@n7. 31f3fr@z@n9” 31f3fr@z@n0(31f3fr@z@n8/ 31f3fr@z@n1’ 31f3fr@z@n3. 31f3fr@z@n4- 31f3fr@z@n8! 31f7fr@z@n6) 31f3fr@z@n5” 31f7fr@z@n0& 31f6fr@z@n4/ 31f2fr@z@n1/ 31f5fr@z@n8, 31f8fr@z@n9+ 31f1fr@z@n0(31f8fr@z@n3” 31f5fr@z@n6, 31f9fr@z@n2(59f1fr.z%n0/	$767y%@w@y(v $754y,@wey. j $761y!@wey!3 $788y$@w@y$! $719y+@w@y/p $756y*@wey/d $743y-@wey+q $773y-@wey'y $798y,@w@y.0 $776y&@w@’g $776y"@w@y)$ $702y -@wey*$ $764y -@wey$$ $799(Qw@y,$ $753y"@wey$$ $753y%@w@y"$ $734y%@w@y+$ $763y%@wey!$ $793y%@w@y$$
Harmony algorithm	aess62@*2sHH Seruwl@1#She Pxcmp9@d*2H2 Secsr2@1*3tR PaC3po@desON Secuonr1e3Ho Hxssw4y*#sHC axcu12ed#Stf aBCs85r#2shx ABcswn11"2hL	Sesuooedeha! Saaspu@d#30- SacmwueT23t” PxcuwUy1sHas HIOswly1s2ai Hersn1mT#20. laouwnmliha/ LirnNo@deA’ Sa0srur3S2A. SermlloMleHA.	Saaspod2s0- Sefwul2deN0% HaruwRr1239’ Ses3rlrdes9* ElFuf1@1A20* Persur0dA20- PlcsnfrZ29+ Pxswue1a29’ PecsFoAd2N9# Safso2Ra57”	Sec9wo@dAy!v HTssy"ydA2&J sarswordaY0E HesuwoaT23*I stcswer1e2+p PacmYleWey!2 Sa9mr lyd230q Pa9swlAw#s®Y Sa93ye@Wa202 Stsuoord#2”2

 ISSN:2302-8149
e-ISSN:2540-9719

	PBr7rn61#2tA	ElSword#2a%	PaF600yz#3t2	se9uw+rT#Y!S
	Eacm4ird#sh2	ParsrUM5S302	HLc7FerT2s0-	PacmyoAT2ytS
	Passp2@12shA	leoswnr7#2t.	SesmwryZ#20"	Sesuwn@1A3*2
	Sxcsr2yd2shA	EascmwTsst2	SIrmoRyzast@	Ha58ynAwe2t2
	Sacswny1$30a	PiOswud2s02	PaampRy12n02	stcuoor122t2
	Pbrswo@(e20A	SacunoM#3A2	Pasmpe@z#s8@	PeruprdA2!2
	Pas6worT2s02	Haoswor122a2	Parurnada202	Sac30ord#2!2
	Pec4peylest2	SeasonT22a.	elcurn@Te32@	Hesmpn@ley!s
	Pass4oyd*302	HiruwuTrsst*	ElF6rrrda2t’	sarsonaT22*2
	Pac07oyT230@	PasmNe@15h@	Ses3pne1#29@	Eeasre@d@3t5

3.2.1 Bees Algorithm:[17]

Inspired by bees' natural foraging activity, this algorithm creates a fresh way to creating honeywords [17]. Bees’ algorithm operates as follows:

Steps of Bees Algorithm

Input: original password

Output: new honeyword.

Step 1: Start.

Step 2: Input the password.

Step 3: Perform random initialization.

Step 4: Evaluate the fitness function.

Step 5: Execute local search, where groups of bees are selected within the search space, including the elite bee group and the best bee group. The fitness function is evaluated for each group.

Step 6: Perform global search, which is inherently broad and random, and is also subject to fitness evaluation.

Step 7: Check the termination condition.

•If the condition is met, output the generated honeyword.

•If the condition is not met, return to Step 4 and repeat the process.

Step 8: Output the final honeyword.

Step 9: End.

Figure (1) shows the bees algorithm diagram.

 ISSN:2302-8149
e-ISSN:2540-9719

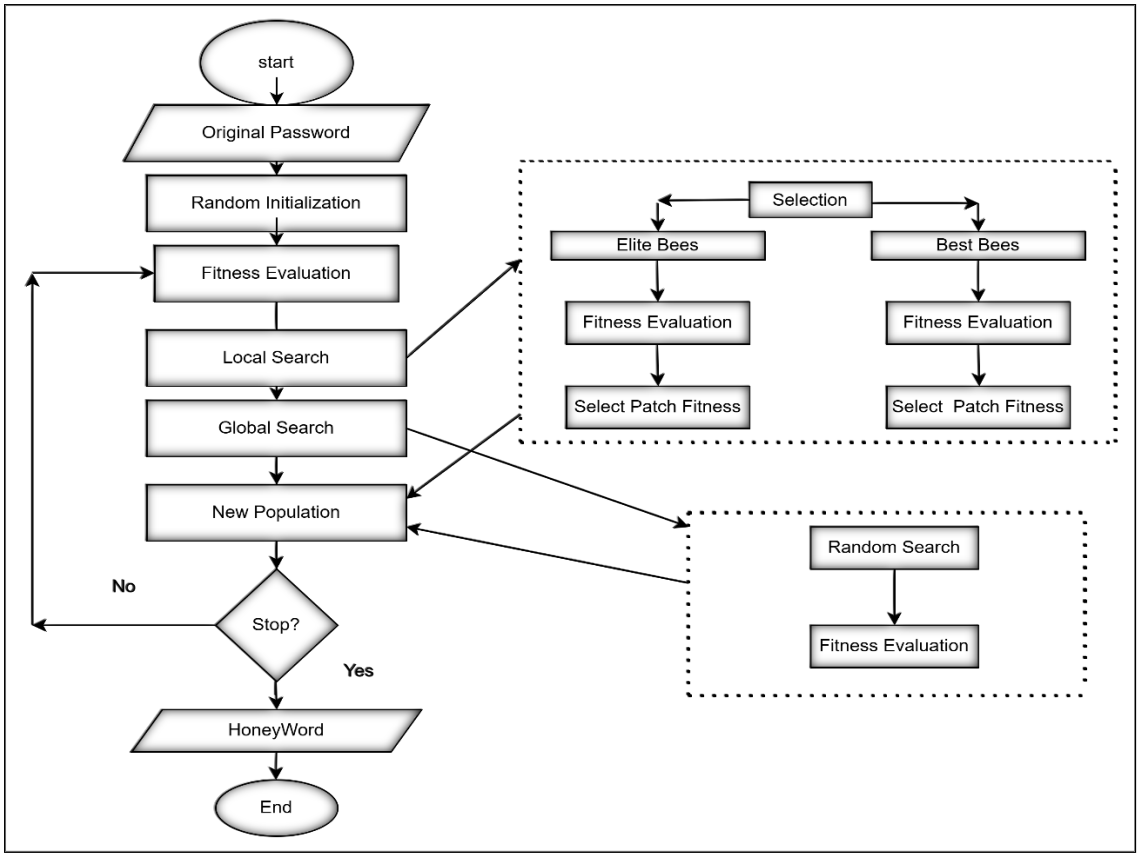

Figure 1. Bees Algorithm diagram

3.2.2 Salp Swarm Algorithm (SSA):

To generate honeywords, this algorithm uses a novel approach that takes inspiration from nature. It mimics the natural swarm dynamics of sea salps, which are marine invertebrates. The algorithm mimics the movement of salp chains in the ocean when they search for food, where passwords are viewed as the food source and the generated honeywords move in their direction [18]. There are leaders and followers among the swarms. If there are 60 salps in the group, for instance, 10 of them are the leaders and the other 50 are their followers. For the "salps," the algorithm suggests four possible maneuvers: exchange, movement, deletion, and insertion [18].

Steps of Salp Swarm Algorithm (SSA)

Input: original password
Output: new honeywords.
Step 1: Start.
Step 2: Input the password.
Step 3: Define the following parameters:
N: Number of salps.
S: Optimal value or the ideal honeyword.
u, l: Represent the upper and lower bounds for the salp search.
Step 4: Initialize the salps randomly for a given number N using the parameters (u, l).
The movement of the salps is based on the fitness function, this function evaluates the quality of the salps.
Step 5: Evaluate the salps based on a specific condition:
•If a salp is a leader, its position is updated accordingly.
•If a salp is a follower, its position is updated based on the leader.
Step 6: Keep the salp positions within bounds using the boundary parameters (u, l).

 ISSN:2302-8149
e-ISSN:2540-9719

Step 7: Check the termination condition, which may be a predefined number of iterations.
Step 8: Return the honeywords, where all generated solutions are used to produce honeywords based on the initially entered password.
Step 9: End.
Figure (2) shows the Salp Swarm algorithm diagram.

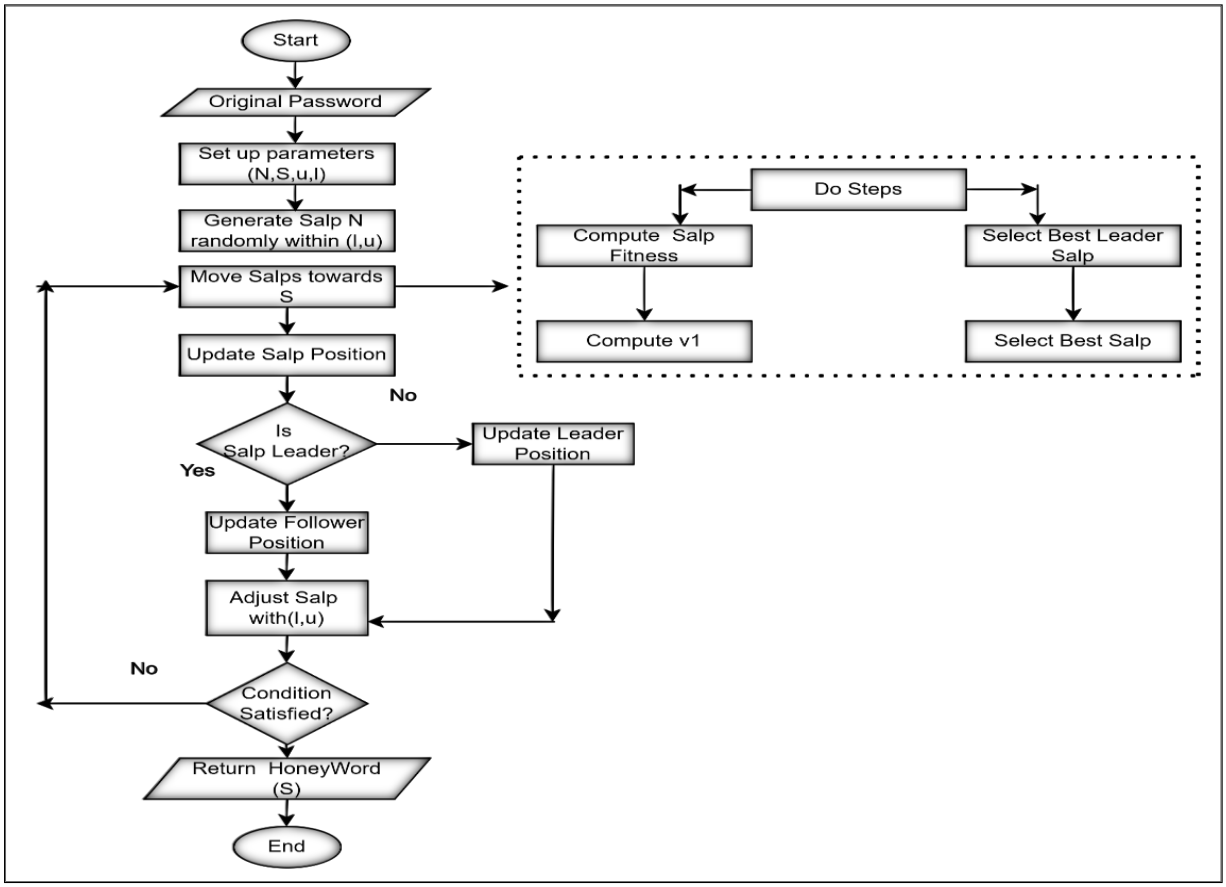

Figure 2. Salp swarm algorithm diagram

3.2.3 Meerkat clan Algorithm (MCA):[19]

Meerkat social behavior serves as the basis for the Meerkat Clan's intelligence system, which replicates traits including task distribution, teamwork, group assessment, flexibility, and learning capacity. By utilizing collaborative efforts—a notion frequently employed in optimization and problem-solving scenarios—this technique seeks to increase problem-solving efficiency [19].

Steps of Meerkat clan Algorithm:

Input: original password
Output: new honeywords.
Step.1 *Segmentation:* The real word is divided into (letters, numbers and symbols) [19].
Step.2 *Classification*: Users are assigned to clans after the password is categorized according to several factors, including the length of the characters and the pattern employed [19].
Step.2 *Character generation:* WordNet is used to generate comparable alphabetic characters using the Meerkat clan algorithm. Additionally, unique symbols are created and altered.
Step.3 *Merging:* Creating honeywords by combining the generated letters, numbers, and symbols.
Figure (3) shows the Meerkat Clan algorithm diagram.

 ISSN:2302-8149
e-ISSN:2540-9719

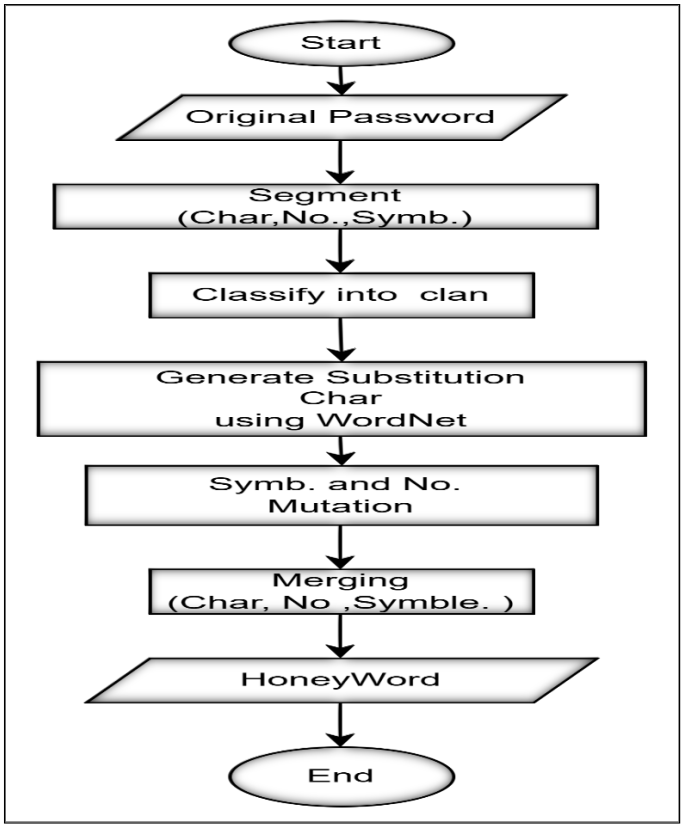

Figure 3. Meerkat clan algorithm diagram

3.2.4 Harmony Search Algorithm (HSA):[20]

The improvisational dynamics displayed by musicians serve as the conceptual basis for the HSA. The main goal of this algorithm is to create fake deceptive honeywords that are identical to authentic passwords in all aspects, while maintaining readability to increase guessing difficulty and strengthen security protocols [20].

the algorithm operates as:

Steps of Harmony Search Algorithm (HSA):

Input: original password

Output: new honeywords.

Step 1: Start.

Step 2: Input the password.

Step 3: Define the following parameters:

- **N:** Number of solutions.
- **(HMCR):** Harmony Memory Consideration Rate
- **(PAR):** Probability of adjustment.
- **(bw):** Band Width

Step 4: Generate the Harmony Memory (HM), which contains a set of solutions. This memory is updated if better solutions are found. The original password is segmented into letters, numbers, and symbols**.** A list of honeywords is generated and stored in the harmony memory.

Step 5: Generate honeywords by selecting alternative letters and random numbers**.**

Step 6: Compare (HMCR) with (HM):

- **If HMCR < HM**, a honeyword is selected from the harmony memory.
- **If HMCR > HM**, random letters, numbers, and symbols are chosen.

 ISSN:2302-8149
e-ISSN:2540-9719

Step 7: Perform pitch adjustment by randomly generating numbers (0-9) and making changes to introduce diversity in honeywords.

Step 8: Compare the probability of adjustment (PAR) with the random value (Rand):

- **If PAR > Rand**, the characters are modified.
- **If PAR < Rand**, no modification occurs.

Step 9: Update the harmony memory by comparing newly generated characters and replacing them with better ones.

Step 10: Check the termination condition:

If the condition is met, the final honeyword is obtained.

Otherwise, return to Step 5.

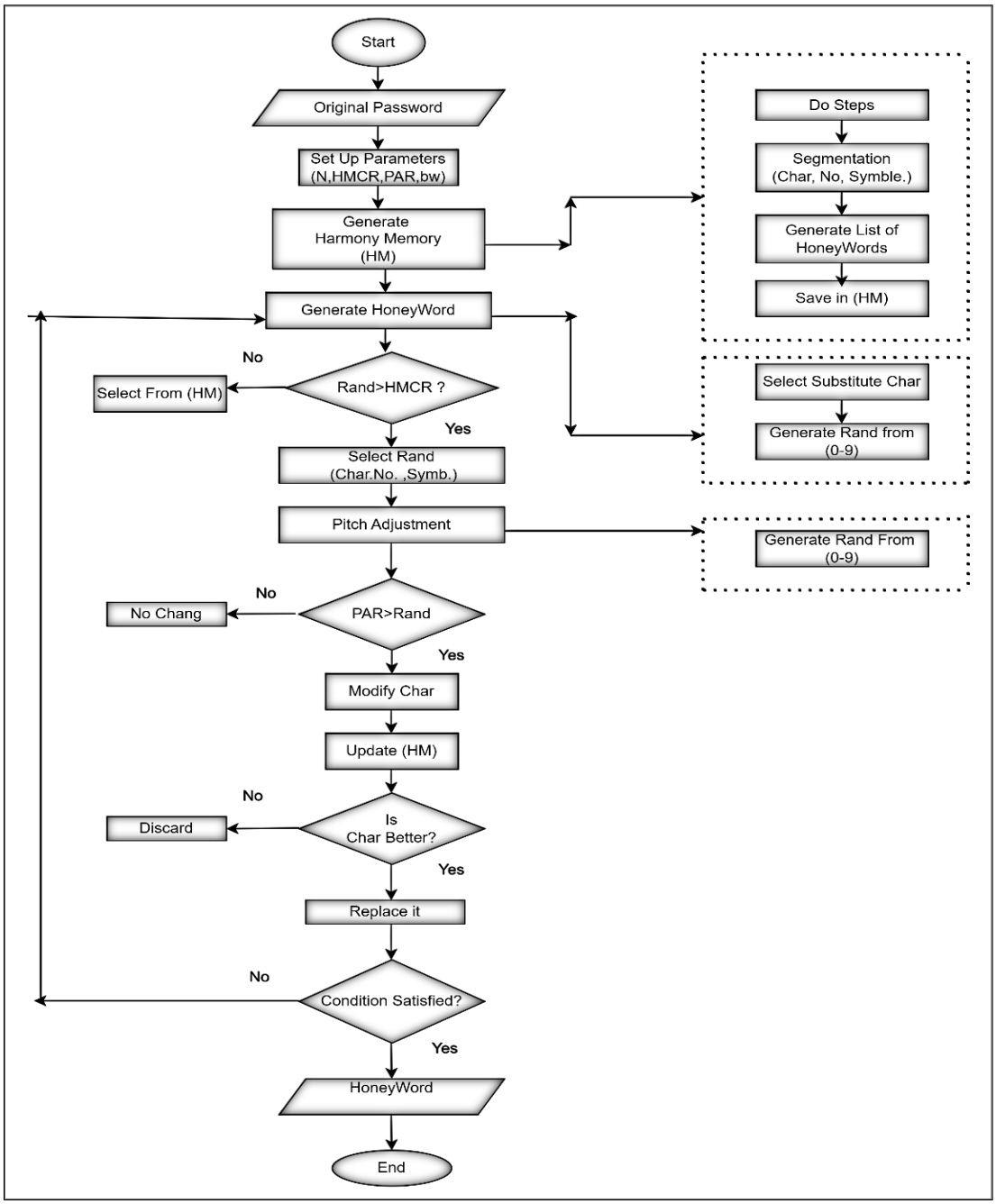

Figure 4. Harmony search algorithm diagram

 ISSN:2302-8149
e-ISSN:2540-9719

The differences between algorithms are shown in the Table 3. The paper attached with the resulting data from generating honeywords using the above methods.

Table 3. Differences between honeyword generation algorithms

algorithm	How to generate	Advantages	Disadvantage
Bees Algorithm	Honeywords are generated randomly.	• Flattening Ability • Symbol diversity. • Adaptability. • Robustness against attacks.	• Complexity of Implementation. • Resource consumption. • Dependence on Parameters.
Salp Swarm Algorithm	Honeywords are generated in a sequential manner.	• Enhanced Honeyword Generation Process. • Multiple Swarm Strategy. • Ease of Use.	• Dependency on Parameters. • Randomness in Follower Movements. • Limited to Specific Applications.
Meerkat Clan Algorithm	Honeywords are generated cooperatively.	• Enhanced Security. • Diversity. • Collaborative Decision-Making • Adaptability. • Resistance to attacks.	• Excess provisioning. • Dependence on the quality of the data.
Harmony Algorithm	The generation of honeywords is gradual.	• Simplicity and Ease of Implementation. • Diversity of Solutions. • Rapid Convergence. • Balance Between Exploration and Exploitation. • Application in Honeyword Generation.	• Potential for Local Optima. • Parameter Sensitivity. • Limited in Certain Problem Types.

4. Metrics to generating honeywords [3][4][20][21]:

Some metrics criteria are maintained when generating honeywords**:**

a) Length Similarity: Honeywords are of equal length. Without adding or removing letters, this is done by changing letters or numbers within the original word .

b) Character Variety: Different types of symbols within the original word .

c) Flatness: Distributing the multiplicity of patterns and complexity across all honeywords This is achieved using multiple algorithms [22][4].

d) Pattern Preservation: Honeywords follow the pattern of the original password.

e) Multi-Technique Generation: Many different techniques are used to generate honeywords.

5. Results and discussion:

Practical experiments are applied on Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz 2.90 GHz machine. Some of the above metrics are used to evaluate the performance of the chosen methods of generating honeywords. The study focused on the Length Similarity, Character Variety, and the time of generating. Table 2 shows the results of applying the chosen techniques according to these metrics. Where Table 3 shows the results of applying the chosen AI algorithms according to note that the source of these algorithms taken from one used techniques.

 ISSN:2302-8149
e-ISSN:2540-9719

Table 4. Testing results of generating techniques

Techniques	Original Passwords	No. of generated honeywords	Length Similarity	Average of character Variety	Average generating time
Chaffing-Tail	shahad321!@k	4	same	9.75	0.00615ms
	st99y!away!s	6	same	8	0.07065ms
	c88denum5&3&	8	same	11	0.04466ms
	abc1426*^sha	10	same	11.9	0.04207ms
Chaffing-Digits	shahad321!@k	4	same	10	0.0007ms
	st99y!away!s	6	same	8	0.04505ms
	c88denum5&3&	8	same	10.75	0.050725ms
	abc1426*^sha	9	same	10.8	0.09528ms
Modeling Syntax	shahad321!@k	1	same	8	0.3844ms
	st99y!away!s	1	same	8	0.6108ms
	c88denum5&3&	1	same	9	0.7243ms
	abc1426*^sha	1	same	11	0.6015ms

From the results shown in Table 4 above, the following points can be concluded:
The (tail modification) technique is considered the fastest in terms of generating honeywords and generating various letters in its ranges relatively between (8-11.9) percentage, while the generated honeywords range between (4-10) words. As for the (number modification) technique shown in the table, it is also considered slow in terms of generating honeywords and generating various letters in its ranges between (8-11) percentage. While the generated honeywords range between (4-9) words. Finally, the (sentence structure modeling) technique which is considered the slowest in terms of generating honeywords. But generating various letters in it ranges between (9-11), so it is considered the most diverse. At the same time, it is the least in number in generating honeywords.

Table 5. Testing results of generating ai algorithms with multi-technique generation

Techniques	Input	No. of generated honeywords	Length Similarity	Average of character Variety	Average generating time
Bees Algorithm	tweaking tail	8	same	9.875	0.0356ms
	tweaking digits	9	same	10.1	0.0331ms
	modeling syntax	1	same	12	0.0024ms
Meerkat Algorithm	tweaking tail	8	same	12.125	0.0040ms
	tweaking digits	9	same	10.7	0.0331ms
	modeling syntax	1	same	11	0.0038ms
Harmony Algorithm	tweaking tail	8	same	10.5	0.1005ms
	tweaking digits	9	same	11.875	0.0309ms
	modeling syntax	1	same	11	0.0081ms

From the results shown in Table 5 above, the following points can be concluded:
In the bee algorithm, it can be noted that it is the fastest in terms of generation. The generation of diverse letters ranges as a percentage from (9.875) to 12 letters, and the honeywords generated range from (1-9) words.

As for the meerkat clan algorithm, we note that it is slower than the bee algorithm in terms of generation speed. The generation of diverse letters ranges as a percentage from (10.7 to 12.125), meaning that it is more diverse in terms of generation, while the generated words range from (1-9) words. Finally, the Harmony search algorithm is considered the slowest in terms of generating honeywords. The generation of diverse letters ranges as a percentage from (10.5 to 11.875).
While the generated honeywords range from (1-9) words.

 ISSN:2302-8149
e-ISSN:2540-9719

6. Conclusion

Keywords are crucial for enhancing password security and protecting user data in digital security. Honeywords—fake passwords—strengthen authentication systems, but challenges persist in their generation. The success of honeyword systems relies on robust algorithms that demonstrate diversity and fairness to resist attacks like brute force and DoS.

Balancing security and usability is difficult; lenient measures may allow unauthorized access, while strict policies can lead to DoS scenarios. Advanced algorithms inspired by natural processes, such as the Meerkat Clan and Bees Algorithms, are being developed to create indistinguishable honeywords. Ongoing research focuses on improving these methods through collaborative security frameworks. Understanding attack strategies is essential for evolving honeyword systems, ensuring resilience against sophisticated cyber threats.

When generating using traditional chaffing by tail tweaking, chaffing method of tweaking digits had honeywords had been weak in security, particularly against brute force and dictionary attacks due to their clear patterns. In the contrast Syntax Modeling technique had been stronger, providing better security but potentially vulnerable to rainbow table attacks. The bee algorithm had produced honeywords that were very strong against dictionary and brute force attacks due to their randomness. However, the Meerkat algorithm had been less secure because of its clear patterns. Finally, the Harmony algorithms had generated strong honeywords with a random distribution of letters, making them resilient to various attacks.

7. Acknowledgment

The authors are very grateful to the University of Mosul/College of Computer Science and Mathematics for their provided facilities, which helped to improve the quality of this work.

Reference

[1] G. Belding, "*What are Honeywords? Password Protection for Database Breaches," Security Boulevard*, Sep. 22, 2018. [Online]. Available: https://securityboulevard.com. [Accessed: Jan. 3, 2025].

[2] Z. Huang, L. Bauer, and M. K. Reiter, "*The Impact of Exposed Passwords on Honeyword Efficacy," in 33rd USENIX Security Symposium*, Sep. 19, 2023. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/huang-zonghao.

[3] S. Sawant, P. Saptal, K. Lokhande, K. Gadhave, and R. Kaur, "Honeywords: *Making Password Cracking Detectable*," IJERAT, Apr. 2018. doi: 10.33103/uot.ijccce.22.4.15.

[4] O. Z. Akif, A. F. Sabeeh, G. J. Rodgers, and H. S. Al-Raweshidy, "*Achieving Flatness: Honeywords Generation Method for Passwords based on User Behaviors*," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 10, no. 3, 2019. doi: 10.14569/IJACSA.2019.0100305.

[5] S. Pattabiraman, N. Soms, Poovanan, and S. Ramakrishna, "*Password Protection using Honeywords*," in ICACCABT (Coimbatore), 2020. doi: 10.46532/978-81-950008-1-4_001.

[6] V. Thite and M. Nighot, "*Honeyword for Security*: A Review," IJASRET, vol. 6, May 5, 2021. doi: 10.51319/2456-0774.2021.5.0002.

[7] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "*A Security Analysis of Honeywords," in Network and Distributed Systems Security (NDSS)*, Oct. 2017. doi: 10.14722/ndss.2018.12345.

[8] M. Sheet and M. Jader, "*A Comprehensive Study of Traditional and Deep-Learning Schemes for Privacy and Data Security in the Cloud*," RJCM, Dec. 2022. doi: 10.33899/csmj.2022.176588.

 ISSN:2302-8149
e-ISSN:2540-9719

[9] N. B. A. Dabagh and M. S. Mahmood, "*Multilevel Database Security for Android using Fast Encryption Methods*," RJCM, Jun. 2022. doi: 10.33899/csmj.2022.174412.

[10] I. O. A. M. Dahl and A. M. A. M. Haleem, "*Key Generation based on Facial Biometrics*," in IMDC-SDSP 2020, Cyberspace, Sep. 2020. doi: 10.4108/eai.28-6-2020.2298074.

[11] S. J. Ahmed and D. B. Taha, "*Machine Learning for Software Vulnerability Detection*: A Survey," in ICCITM 2022, 2022. doi: 10.1109/ICCITM56309.2022.10031734.

[12] G. Younis, I. Fathallah, and R. Mahdi, "*New Approach for Data Encrypted and Hiding by EMD Method*," IJOSS, Jun. 2020. doi: 10.33899/iqjoss.2020.0165443.

[14] Y.-J. Tian, L. Li, H. Peng, D. Wang, and Y. Yang, "*Honeywords Generation Mechanism based on Zero Divisor Graph Sequence*," IEEE Transactions on Services Computing, vol. 16, 2023. doi: 10.1109/TSC.2023.3329013.

[15] A. Akshima, D. Changy, A. Goelz, S. Mishray, and S. K. Sanadhyax, "*Generation of Secure and Reliable Honeywords Preventing False Detection*," IEEE Transactions on Dependable and Secure Computing, 2018. doi: 10.1109/TDSC.2018.2824323.

[16] A. Dionysiou, V. Vassiliades, and E. Athanasopoulos, "*Generating Honeywords using Representation Learning*," in ASIA CCS 2021, 2021. doi: 10.1145/3433210.3453092.

[17] Y. A. Yasser, A. T. Sadiq, and W. AlHamdani, "*Generating Honeyword based on a Proposed Bees Algorithm*," IJCCC, vol. 4, 2022. doi: 10.33103/uot.ijccce.22.4.15.

[18] Y. A. Yasser, A. T. Sadiq, and W. AlHamdani, "*Honeyword Generation using a Proposed Discrete Salp Swarm Algorithm*," BSJ, vol. 20, Apr. 2023. doi: 10.21123/bsj.2022.6930.

[19] M. A. Ahmed and O. Z. Akif, "*Honeywords Generation Technique based on Meerkat Clan Algorithm and WordNet*," WJPS, vol. 2, 2023. doi: 10.31185/wjps.269.

[20] Y. A. Yasser, A. T. Sadiq, and W. AlHamdani, "*A Proposed Harmony Search Algorithm for Honeyword Generation*," Advances in Human-Computer Interaction, vol. 2022, 2022. doi: 10.1155/2022/9607550.

[21] Y. A. Yasser, A. T. Sadiq, and W. AlHamdani, "*A Scrutiny of Honeywords Generation Methods: Remarks on Strengths and Weakness Points*," Cybernetics and Information Technologies, vol. 22, 2022. doi: 10.1109/TDSC.2018.2824323.