IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-9, April 2019 E-ISSN: 2347-2693

Efficiency Analysis of Honey Encryption Algorithm

Bhoomika R[1*], Gowda Deepa Narsimha[2], Abhishek V[3], Bharathi H[4]

[1, 2]Post Graduate Scholar, Dayananda Sagar of Arts Science and Commerce, Bangalore University, Bangalore, India
[3, 4]Post Graduate Scholar, Dayananda Sagar of Arts Science and Commerce, Bangalore University, Bangalore, India

[*]*Corresponding Author: bhoomika.rajshekar@gmail.com, Tel.: 9743813465/7021199726*

DOI: https://doi.org/10.26438/ijcse/v7si9.1214 | Available online at: www.ijcseonline.org

***Abstract*:** In the field of cyber security, we have many algorithms that support cryptography. These algorithms are more prone to cyber-attacks and can be cracked easily by hash cracking tools. One of the methods to prevents these attacks is Honey Encryption technique that has gained popularity in recent years. This method provides another layer of protection when the intruder tries to hack an account with bogus data. In this paper, we are efficiently increasing the probability of Honey Encryption by generating more hash keys(password)using python programming approach. Honey encryption helps in minimization of vulnerability. We have designed an algorithm which eventually increases the buffer size for the randomly generated passwords.

***Keywords*:** Cryptography, Security, Password.

I. INTRODUCTION

Protecting systems, networks, and programs from digital attacks are challenging as there are more devices than people, and eventually attackers have also become more innovative. These cyber-attacks are usually aimed at accessing, changing, or damaging private data and personal information. The common selection of passwords which are opted by the users, which are easy to remember, is the main reason behind the development of honey encryption algorithm.[1] Hash algorithms are one-way cryptographic function which turn any amount of data into a fixed length. These algorithms have a property wherein if the input changes by a small amount the resulting hash changes completely from the original.

There are many techniques to crack plain hashes and obtain the actual passwords and hence just by hashing we cannot provide complete security for the passwords. The Rivest-Shamir-Adleman algorithm, better known as RSA, is now the most widely used asymmetric cryptosystem on the web today. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. [2][3] A Brute Force Attack is the simplest method to gain access to a site or server. It tries various combinations of Usernames and passwords again and again until it get in. This repetitive action is like an army attacking a fort.[4]

The main aspect of cybersecurity was to protect entities like: PC's, smart devices, networks, and cloud. Emerging technologies like next generation firewalls, DNS filtering, antivirus software, primitive methods of data encryption. The problem of these prompted us to analyse the efficiency of honey encryption algorithm.

Honey encryption is the type of data encryption that "produces a cipher text which when decrypted with an incorrect key as guessed by the attacker, presents a plausible-looking yet incorrect plain text password or encryption key". Instead of improvising primitive data encryption techniques, we can implement python programming language is a general purpose, high level programming language which helps us to increase the efficiency of honey encryption algorithm.[5][6][7]

II. RELATED WORK

Honey Encryption scheme was introduced by Juelz and Ristenpart on 2014 to increase protection layer onto the password-based RSA encryption algorithm and credit card applications. Subsequently, extended by Tyagi and Huang to protect the text messaging and genomic data application respectively. More recently, Joseph enhanced the security of this scheme to resist the message recovery attacks. Honey encryption is also related to Format-Preserving Encryption (FPE) and Format-Transforming Encryption (FTE).
In FPE, the plaintext message space is the same as the ciphertext message space, whereas they differ in FTE.

A recent study reports that 1.08% of people chose the same password. Due to the intelligence and upgradation of innovative ideas, the time taken to crack other encryption

algorithms are decreasing. Therefore, there is a need for new encryption scheme that will improve the efficiency of honey encryption.

III. METHODOLOGY

The concept of honey encryption is used to detect attackers when trying to decrypt the encrypted data. In honey encryption, there are many passwords, and only one is right and the rest are all bogus data. These bogus data are called honey words or decoys. The entire list of honey words is called sweet words. This creates an ambiguity in the intruder to distinguish between the real and fake data.

In the improvised version of honey encryption algorithm, we take the last six digits of the debit card and the pin number as an input from the user. It creates two dictionary(lists) to store these inputs. So, the program generates ten random 6-digit pin numbers of the card. This is done to create an ambiguity in the intruder's mind as to what the actual password is. The first dictionary is mapped to the 6-digit of the card number, and the second dictionary stores the pin number. Then it generates additional password prefixing original password and essential validations are performed. When the hacker tries to hack the user account, he gets a randomly generated list of fake passwords. [8] Once the hacker guesses the right password from the randomly generated list then he gets the actual pin. If the hacker guesses the password wrong from the list then he gets a wrong pin, and the user gets an intruder alert for the security purposes. And if he gives any other input apart from the list then a randomly generated pin is returned to the hacker which he assumes to be the actual pin number. Every time the hacker tries to hack an account a fake 4-digit pin is returned if he guesses the wrong account number which creates ambiguity.

IV. RESULTS AND DISCUSSIONS

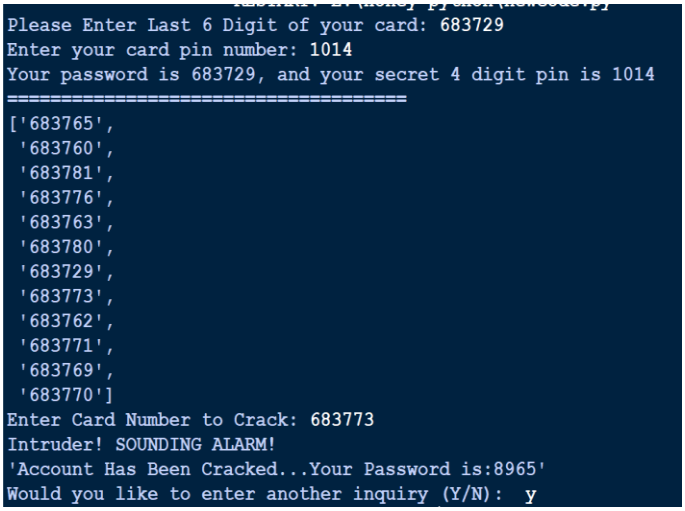
Here the program prompts the user to enter their card details, and the pin number.

And when intruder tries to hack he's given a list of password to guess among. If he guesses the password wrong it prompts the account is hacked and the user receives an alert but not in reality because the intruder believes that to be the right password.

Once when the intruder guesses the password then he actually gets the pin number of the card. Therefore the account is hacked.

Algorithm for improvised honey encryption method:

Step 1: Input last 6-digit card number & pin code from user
Step 2: create 2 lists to store these
Step 3: generate random integer
Step4: map passwordToSeed = {userpass: trueSeed} and SeedToPassword = {trueSeed: Message}
Step4: then it generates additional password prefixing original password
Step 5: Validation
Cipher = int (passwordsToSeeds[userPass]) XOR with trueSeed
Step 6: check password------ // hacker prompt
Query = input ("Enter a password to crack: ")
Step 7: if hacker guesses the right password from the randomly generated list
Then he gets the actual pin
Else
If hacker guesses the password wrong from the list
Then
He gets a wrong pin and user gets an intruder alert
Else if he gives any other input apart from the list then a random pin is returned to the hacker.
Here the program prompts the user to enter their card details and the pin number.
And when intruder tries to hack he's given a list of password to guess among. If he guesses the password wrong it prompts the account is hacked and the user receives an alert but not in reality because the intruder believes that to be the right password.



Once when the intruder guesses the password then he actually gets the pin number of the card. Therefore the account is hacked.


```
Please Enter Last 6 Digit of your card: 693481
Enter your card pin number: 2045
Your password is 693481, and your secret 4 digit pin is 2045
====================================
['693463',
 '693468',
 '693465',
 '693466',
 '693474',
 '693473',
 '693479',
 '693472',
 '693483',
 '693476',
 '693484',
 '693481']
Enter Card Number to Crack: 693481
'Account Has Been Cracked...Your Password is:2045'
Would you like to enter another inquiry (Y/N):  n

Thank you for testing Honey Encryption
>>>
```

V. CONCLUSION

All the existing encryption techniques are unique and rely on password-based encryption. These are vulnerable to attacks as these passwords can be hacked by hash cracking tools. Honey encryption stands out of many encryption methods and is optimal as it provides an extra layer of protection to the user data. Also, this program increases its efficiency by extending the buffer size for randomly generated data. It can be also applied to secure IOT devices and other computer peripherals as well.

REFERENCES

[1] Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In Advances in Cryptology–CRYPTO 2012, pages 312–329. Springer Berlin Heidelberg, 2012.

[2] M. Preetha et al, International Journal of Computer Science and Mobile Computing Vol.2 Issue. 6, June- 2013, pg. 126-130

[3] M. Bakker and R. V. D. Jagt, "GPU-based password cracking. Technical report," Univ. of Amsterdam, 2010

[4] Joseph Jaeger, Thomas Ristenpart, Qiang Tang. Honey Encryption Beyond Message Recovery Security. Presented in EUROCRYPT2016 pages 1 and 2, 2016.

[5] TIOBE Software Index (2011). "TIOBE Programming Community Index Python". 1

[6]"Programming Language Trends - O'Reilly Radar". Radar.oreilly.com. 2 August 2006.

[7] "The RedMonk Programming Language Rankings: January 2011 – ecosystems". Redmonk.com.

[8] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh. Kamouflage: loss-resistant password management. In ESORICS, pages 286–302, 2010.

AUTHORS PROFILE

Bhoomika.R, PG scholar, studying MCA at Dayanada Sagar institutions, interested in certification, data science and researches.

Gowda Deepa Narasimha, PG scholar, studying MCA at Dayanada Sagar institutions, interested in data science and data analysis.

Abhishek.V, PG scholar, studying MCA at Dayanada Sagar institutions, interested in coding, android studio, data science and data analysis, ethical hacking, linux, raspberry, pi network security.

Bharathi.h, pg scholar, studying mca at dayanada sagar institutions, interested in webpages designing, webapp development and coding.