Al-Rafidain Journal of Computer Sciences and Mathematics (RJCM)

www.csmj.mosuljournals.com

Honey Encryption Security Techniques: A Review Paper

Ammar Abdul Majed Gharbi [1],*Ahmed Sami Nori[2]

Department of Computer Systems Technologies, Nineveh Technical Institute, Northern Technical University, Mosul, Iraq [1] ,
Department of Computer Science, College of computer science and mathematics, Mosul University, Mosul, Iraq [2]
**Corresponding author. Email: ammarmajed@ntu.edu.iq[1]*

Article information

Article history:
Received : 5/5/2021
Accepted :21/6/2021
Available online :

Abstract

From time to time, we hear in the news about a breach or attack on some well-known companies as just news, but it is a serious problem because it is the privacy of citizens, their money in trade and managing their businesses and projects. In this paper, we offer a review of the honey encryption planner. Honey Encryption is the encryption system that ensures flexibility versus the brute-force attack through the provision of plain reasonable text, but false for each key is invalid utilized by a trespasser to decrypt a message, two key areas are open it is difficult to create a compelling message trap that's perfect enough to deceive the striker even when he believes that he has the message in its original form.

The next problem, the typo issue, where a valid phony plain text seem to a lawful user when he accidentally enters the wrong key. Our goal is to have more satisfaction disguised tricks that are perfect enough to prevent a trespasser from getting the original message, We also need new security methods because the attackers are looking for new ways to attack the systems, so we proposed a new way to protect messages and passwords well and difficult to break and take all the possibilities of attack, including the brute-force, and then the data is hidden in an image with a public secret key.

Keywords:
honey encryption, ElGamal algorithm, steganography, brute force attack, typo problem,honeyword,DTE

Correspondence:
Author : Ammar Abdul Majed Gharbi1
Email: ammarmajed@ntu.edu.iq

1. INTRODUCTION

Day after day, our life has evolved, and improvements and facilities are added to it in all life facilities, this progress there must be a weakness, which is the password that is the window on our personal life and our private information[1], but all our lives have become dependent on a number or a group of letters We choose it and create it so it can be a guardian of our data and information portal, In this paper, we will present a review of the honey encryption planner.

The honey algorithm is a cryptographic system that ensures flexibility versus a brute-force attack [2] by presenting a reasonable but fake plain text for each void key that an intruder uses to decrypt a message [3]. Two main keys areas are the difficulty of creating a persuasive trap message that's perfect enough to trick convincing the assailant they have the original message. The next issue, the typo issue, where a valid trap plain text seems to a lawful user when he accidentally input's the void key [4].

Our goal is to have more satisfaction disguised tricks that are good enough to prevent an attacker from getting the message. We also need new security methods because the attackers are looking for new ways to attack the systems, so we proposed a new way to protect messages and passwords well and difficult to break and take all the possibilities of attack, including the brute-force, By integrating more than one method of data security (cryptography and steganography) to benefit from its advantages and overcome the disadvantages of each method to improve data security, which is through by encrypting the message with the ElGamal algorithm that Taher Al-Jamal invented in 1985 and used in Public key encryption. Based on the Diffie-Hellman key exchange principle, this is by a mathematical problem called the discrete logarithm problem [5] [6] .

And then the message is encrypted with a password using the honey algorithm and 100 is generated A honey word password, when entering the correct password, the message will be decrypted and the plain text will appear, but if he entered an incorrect password that is not present in the Distribution Transform Encoding (DTE) and inverse table, the password will appear to be incorrect, but if the attacker entered an incorrect and existing password In the DTE tables, an incorrect password will appear and an alert will be sent to the administrator about the existence of a breach in the user database.

After which all the information of the ElGamal algorithm (the encrypted message, the public key, and the private key) and the honey algorithm (all variables and tables, DTE, inverse table) will be hidden. In the cover image with a public secret key, that is, if the image is sent from the sender to the recipient without mentioning any of the keys or information about encryption, only the location of the public key is sent to hide the data.

The remainder of the paper is arranged as follows: the first section explains the type of attack that we responded to the brute force attack, the second section identifies the concepts of the honey algorithm and its most important elements and the third section is for a review of the most important challenges to the honey algorithm, which is the production of the honey message and typos, and a comparison of researchers' suggestions in dealing with typos, advantages and disadvantages ,the fourth section conclusions and the fifth section include recommendations.

2. Brute force attack:

It is also known that the brute-force attack is the strongest type of attack on encryption methods [4], which is the experience of all possible keys for decryption and also it is the most popular method for breaking the encrypt [7] and some cryptanalysis try to use smart methods, for example trying part of the keys and smart guessing in choosing The keys to breaking the encrypt , as in dictionary attack and brute force attack, for example using the most common words that users use to choose their passwords [8], for example when breaking the database of RockYou.com, which is a site that provides services and programs to users on the social network, it was found that 32 million simple, clear passwords frequently among users and the most important 10 words are (123456, 12345, 123456789, password, iloveyou, princess, rockyou, 12345678, abc123) [1] so they are easy to break in this case or try keys from their personal data, especially when Use their explicit name or the date of their holidays and events or their phone numbers, and this information may be available from their personal pages on social networking sites or their use of slang words or letters and serial numbers [9].

Therefore, the honey algorithm was introduced in 2014 by (Juels & Ristenpart, 2014b) [10] to be one of the methods that deter and prevent a brute-force attack from breaking the encrypt in a way that fools the attacker from using an invalid password that appears to him to be correct it results in a fake message and it appears to the cryptanalysis the encrypt is correct, so the attacker does not know if the password used to decrypt is correct and when it appeared in the possibilities that he used to break the encrypt is successful and he is also unable to distinguish between correct and fake words[11].

Therefore, we achieved two objectives: non-discrimination and confidentiality [12] that guessing the foul for the attacker is similar and gives almost the same or similar results and looks real, so the attacker is misleading about the true[1] and also there is a second line of defense against the attackers, which is that when the database that contains passwords is broken, more than one word will appear and when the attacker uses any one From the honeywords, the attacker system will be converted into an imaginary system that contains false and fake data to deceive the attacker [9] for example when using the honey algorithm in the Social Security number, information can be shown to a young child or a man who has died [4].

There is an additional advantage of honey encryption that the use of weak passwords by the user does not have any fear, harm, or concern about the possibility of breaking the encrypt by attacking with brute force because honey words are what works to delude the attacker and enter it into a fake system and show false data that appears to him to be valid [4].

3. Honey Encryption Concept

The idea of deceiving attackers is very old and is also used as a security method, and there is an algorithm that is not as far away as Kamouflage (Bojinov et al., 2010) [13]. Which is similar to the honey algorithm designed to deceive the attackers by generating fake messages. The honey algorithm was proposed by (Juels and Ristenpart 2014b)[9] that differs from the (Kamouflage) algorithm in that it contains DTE which is an active component [14] and is the substrate in the honey encryption algorithm. The main motive of encrypting honey is confusing the attacker [15], making it difficult to know when to guess the correct key or the correct password [16]. In traditional encryption technology, decrypting the encrypted text using the wrong key provides the attacker with an invalid message, so the attacker can find the incorrect keys via a Brute-force attack [17].

However, if the data is encrypted with honey encryption, outputting the decryption under the wrong choices of the attacker will give invalid data that appears reasonable, thus misleading the attacker. For example, if an attacker tries to obtain a mobile phone number by making 200 attempts, it will get 200 fake numbers for all 200 attempts. Because each decryption will appear acceptable like the others, the attacker will not be able in any way to distinguish between the data Correct and incorrect [18]. The base honey encryption contains two models for HEnc encoding and HDec decoding. HE = (HEnc, HDec) [19]. As shown in the figure (1).

HEnc (K, M)	HDec $(K, (R, C))$
$S \leftarrow\$ \text{ encode}(M)$	$S' \leftarrow H(R, K)$
$R \leftarrow\$ \{0, 1\}^n$	$S \leftarrow C \oplus S'$
$S' \leftarrow H(R, K)$	$M \leftarrow \text{decode}(S)$
$C \leftarrow S' \oplus S$	return M
return (R, C)	

Figure (1): The Models of The Honey Encryption. [19]
" H is a cryptographic hash function, K is a key, M is a message, S is a seed, R is a random string, C is a ciphertext, and ← $ denotes uniform random assignment".[14]

the figure (2) shows an example of Soft Drink Flavors encrypting [16]

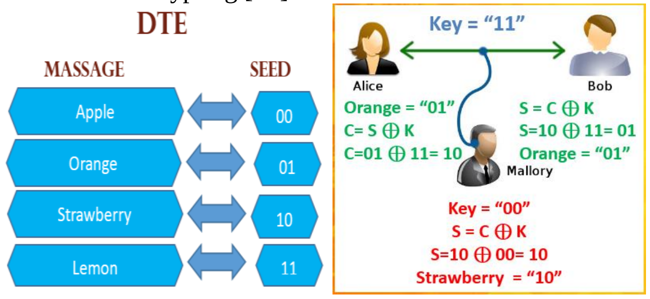


Figure (2) : Example for Encrypting Soft Drink Flavors [16]

3.1. DTE

It is the most important part of the honey encryption but is considered the core and the hart of the encryption [14], so its design must be good and efficient enough and its outputs must be meaningful for the attacker to feel confused [15] which gives the message space the number from the seeds of the n-string-bit.

DTE takes into account the probability distribution of the message space and assigns a corresponding bit stream ratio to the message. The intuition lies in the fact that all potential decryptions, regardless of their authenticity, are associated with some messages, and since the possible decryptions are mapped via the expected probability distribution [20], the attacker does not obtain any information. Creating a DTE suitable for various honey encryption applications requires an understanding of the message space distribution. Example of (HE) If an attacker tries to obtain a credit card number by making 1,000 attempts, he will get 1,000 fake credit card numbers for every 1,000 attempts. Each reasonable decoding will look like the other [1]. The attacker has no way to distinguish a priori and is correct and this is done in two steps

1 - Choose a random set of the seed area

2- Linking this random group of seeds with the original message [21]

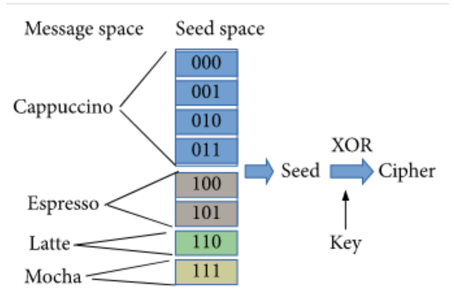


Figure (3) : The seed space for coffee types encryption.[22]

the figure (3) shows an example of seed space for coffee types encryption, Cappuccino, Espresso, Latte, and Mocha are among the coffee message options. These four messages have been arranged in alphabetical order. Assume that 4/8 of Sydney residents choose Cappuccino, 2/8 prefer Espresso, and 1/8 choose Latte and Mocha. A three-digit space is used as the seed space. [22]

DTE also contains two algorithms DTE = (encoding, decoding), encoding takes the message as M∈M input and outputs a set of seed and S values from the seed space, the decoding algorithm takes the S∈S input message and outputs the M∈M message and tracks the correctness of the DTE algorithm. if any M∈M is

Pr [decode (encode (M)) =M] = 1 [23]

3.2. Inverse table

We first built DTE which consists of two algorithms (encoding, decoding) using inverse sampling that uses the cumulative distribution function (CDF) for the encoding algorithm[24], which is a commonly used mechanism for transforming regular random variables into variables from some other distributions[9], and in the decoding algorithm, inverting the cumulative function should be used, distribution function (CDF), we used a reverse table because the cumulative distribution function (CDF) is one-way [25]

3.3. Hashing and salt

As part of the DTE-then encrypt method [26], HE uses traditional hashing [27] and it is, it's important to define the fundamental characteristics. for cryptographic hash functions. The hash function is programmed to make encryption easy, but the decryption is difficult since the hash function plan the set of maladjusted length bit strings into a series of bit strings of fixed length [28]. This makes the one-way features of these functions [29]. SHA-1, SHA-2, SHA-3 and MD5are examples of hash functions. MD5 has currently been described as unsafe [30] [31] and SHA-11 conflicts have been found [31]. conflicts mean that the same outcome will end up with one's values [32].

The process of saving a basic password hash does not guarantee the passwords are saved safely [15]. Two of the benefits of the hashes are also their greatest possible weakness: they are very small to store and easy to produce. The alternative to this is to use the 'salting' process which involves hashing more than just user-accepting passwords. In the salting process, the hashes are randomized by inserting or pre-adjusting a random string called 'Salt' to the password before hashing. As a consequence, the same password hash is converted into a slightly new string every time, in this case, rainbow attack, dictionary attack, and brute force attack are countered [1].

4. Related work

- In [33], A multi-modal biometrics fingerprint and iris are being used. The biometrics is fused using the gradient pyramid technique. The fused template is encrypted using honey encryption. The experimental results show that the proposed algorithm DWTSVDGOA gives an NC value of 1, PSNR as 90.75, and SSIM as 0.99. The performance and the evaluation of the technique are found to be better than existing image watermarking techniques.
- In [34], using honey encryption for the cryptography science in communications networks. by proving hacker a number of fake keys that appears to be original. a more powerful technique is proposed in this research paper when used under honey encryption will give excellent results.
- In [35], This paper proposes source encryption and channel encryption of input data sets to improve the data security in Wireless Sensor Networks (WSN). It is the implementation of honey encryption for the information bits as source encryption and includes Gaussian Frequency Shift Keying (GFSK) for the honey encrypted data to perform Frequency Hopping Spread Spectrum (FHSS) as channel encryption and the output of FHSS is propagated with the help of Frequency Hopping Multiple Access (FHMA) in WSN. So, it is impossible to intrude through channels by the hackers and also there are no possibilities to detect or decode the information by Brute force attack because of honey encryption. It provides dual security to protect the information.
- In [36], The authors of this paper have devised a way to address the problem of data security in Hadoop storage. The authors created Attribute-Based Honey Encryption, which combines attribute-based encryption with honey encryption on Hadoop (ABHE). This method works with files that have been encoded in HDFS and decoded in the Mapper. Furthermore, the authors assessed the new ABHE method by conducting encryption-decryption on a variety of file sizes and comparing it to current methods such as AES and AES with OTP. During the encryption and decryption of data, the ABHE algorithm exhibits a significant boost in performance.
- In [37], The authors of this paper propose a secure privacy protection migration for data that is outsourced to the cloud using a honey-encryption cryptographic algorithm, and we are using migration protocol while migrating data from the existing server storage system to the cloud server storage system, which ensures data integrity and confidentiality.
- In [38], The author of this study focuses on plaintexts, which are non-numeric informational communications. We need to capture the empirical and contextual aspects of the language in order to trick the attacker into thinking the decoy message came from a certain source. That is, there should be no language distinction between genuine and fraudulent communications without disclosing the genuine message's structure.

  To tackle this challenge, he uses natural language processing and extended differential privacy. To model privacy for text documents, primarily focus on machine learning approaches such as keyword extraction, context categorization, bags-of-words, word embeddings, and transformers for text processing. Then, using e-differential privacy, demonstrate the security of this strategy.
- In[39], In this research, we design a security technique based on the notion of honey encryption to protect smart card password authentication from brute force and denial of service threats.
- In[40], An enhanced honey encryption (HE) strategy is described in this research for enhancing the security of instant messaging networks while also confusing the time and resources of hostile individuals. This study improves the HE scheme by utilizing natural language processing techniques to create semantically reasonable but phony chat messages for the adversary to use during his assaults. The unique system is resistant to eavesdropping, according to evaluation results, since an adversary unable to identify decoy messages from plaintext when decrypted with an erroneous key.
- In[41], The authors introduce GenoGuard, a technology for enabling excellent genetic data protection. GenoGuard combines honey encryption (HE), a novel theoretical paradigm for encryption. As a result, GenoGuard solves the open topic of applying HE techniques to extremely non-uniform probability distributions that describe genetic data sequences. GenoGuard provides an information-theoretic security guarantee against message recovery attacks, as well as proving that decryption under any key will give a believable genome sequence. We also look at side information assaults. Finally, we propose a GenoGuard software implementation that is both efficient and parallelized.
- In [42], The honey encryption algorithms are designed and implemented in this research, and they are applied to three categories of private data: Chinese identity numbers, mobile phone numbers, and debit card passwords. We assess the effectiveness of our system and provide a solution to the overhead problem. Lessons learned through the design, implementation, and evaluation of the honey encryption technique are also presented.

5. The Challenges of Honey Encryption

5.1- Creating of Honey Messages

first of the problems of honey encryption is to produce false messages, and that the honey message is "plain text that looks reasonable but is fake" [10]. And here, these false messages are called honey terms, as researchers called these types of false messages, although researchers tend to use the term honey message instead. Honey words were added to be used as decoys before encrypting honey, present along with the correct message [2], and thus the attacker was unable to recognize the correct message even though the honey encryption involves a delivery encryption method that maps separate messages instead of storing a bunch of spoofed messages for each recipient, the Honey word concept is still very similar [3].

In general, the difficulty with the development of the honey word is that the honey terms must be identical, which means that they must be distinct from each other but identical in such a way that for example, if the correct message is the SSN, the other honey word may also be some other social security number, the right word will not be picked from the honey message category.

In other words, in the context, connotation, and grammar of the word candy, the words for honey must be identical. The Social Security number must be real, or it may be spotted as a fake by the intruder. It is important to consider the meaning carefully as well. If the SSN belongs to a company employee, then the SSN cannot belong to a nineteenth-century small child or a deceased individual. However, the problem is not with SSNs. For data with low entropy and simple structure, such as an SSN, it is simpler to generate honey word than for data that has a high entropy or lacks any structure (for example, the actual use of natural-world language such as emails). Particularly troublesome are data consisting of human-produced communications, such as emails [12].

and this was seen by many researchers when honey encryption was introduced: "An estimate of message distributions across DTEs is as interesting as a natural language processing problem." The natural language was left unresolved as an issue for future research and remains so today, as a very recent analysis. There is great entropy or a total lack of structure in the messages created by humans. However, there were some solution suggestions, evidently susceptible to the chosen-ciphertext attack by CCA. The intruder can extract bits of information from plain text within the CCA [32]. This contradicts the concept of semantic security, which means that with minimal resources when accessing the corresponding cipher script, the attacker cannot acquire any new information [33], honey encryption can achieve semantic security with low-entropy messages.

Although the issue was also stated in the Honey encryption show, previous to the honey encryption with honey terms, similar problems had occurred. Solution principles were explored, such as chaffing by tweaking, chaffing-with-a-passwords model, hybrid generation approaches, and eventually, the random pick will address certain issues relating to poor passwords. Honey words will be fully flat, but this is not enough appropriate for honey encryption since the message itself varies [2]. For example, if the user selects the "password123" password, the random collection does not simply use it as a password but instead returns the user with a changed version and stores a series of honey words along with the new password. The real password that was initially selected by the user was overlooked. Since the data will change over the life of the data, this will work, It does not, however, deal with genetic data. for example, since the real data must be stored as-is. New categories of data need close thought and preparation since it is necessary to ensure that the DTE process operates with the new data safely. That's what has been found out by many scholars [12]. "In low-entropy settings such as passwords, RSA keys, PINS numbers, and credit cards, Honey encryption is built to work. Expanding it to support other settings usually requires a detailed DTE architecture"[12].

- In [43] The researchers mentioned in the paper, the message space should be built carefully, otherwise, the brute-force vulnerability will not be solved well by honey encryption. While a plain text come by DTE from an improperly estimated key looks like a properly encrypted ciphertext, if the message space has not been built neatly, attackers may use other methods of operation to check whether the estimated key is wrong. In the case of a cell phone number, the intruder may call the cell phone number to verify whether the number is right.

  The deciphered identity number does not refer to a man for the definition of the database in a female hospital. definition of numbers that belong to 0-to-4-year-old babies in an e-commerce sector may be less likely in a report. In a middle school, most pupils should not be younger than 12 years of age and older than 19 years of age. This message should not include any fingerprint that may be utilized by criminals to identify the rightness of the message when the decryption process makes a message from the message space.

- In [44] his thesis, the researcher proposed an algorithm to reduce the cost and storage of honey words, because the current algorithms for generating honey words have a problem with storage and increased cost. So a proposed algorithm was created without generating and storing the individual honey words in a database, then producing two password files in the server database and the honey auditor database, and keeping the real password indexing and hashed passwords in the honey checker folder and the tables are arranged arbitrarily according to the registration location the user.

  The methods of generating honey words in the proposed algorithm at least 20 honey words for one real password, and this number is small because the attacker will get the result with a few attempts and the message space cannot accept more than four messages and also this number is small facilitates the process Breaking the code because the possibilities are very few, and these applications depend on symmetric encryption method and evenly distributed message spaces This is also a problem because there is no randomness in the probability distribution, and only two countermeasures are applied.

- In [9] Researchers have presented a new way to confuse attackers, the application stores the right password with multiple sweet words for each account in the database. so they create several honey words from the user password for every new account allotment and stored them in indexes we called honey indexes. And then if a new person opens a new account, he gets a random index number for actual passwords and sweet words. In the folder, we create one password table for all user accounts. There are two attributes in the table: the first attribute is the account's daily ID and the second is the honey index (real password, honey words). It creates a single password table for all user accounts, and this is a big problem.

  because if the password file is breached, it will detect all the accounts in the server database, and also it did not solve the typo problem and did not solve the typo tolerance problem. And the server is not very secure to provide security to users due to frequent attacks.

- in [2] This paper is considered the origin of the idea of encoding honey and the basis for the honey word and suggests many Gen flat (or approximately flat) generation processes for creating a list of sweet words and selecting an index of the actual password in this list. The processes are separated according to whether password modifications affect the user interface. We differentiate between the two cases:

○ The password-change UI is unchanged by legacy-UI procedures. This is arguably the more important situation. Two legacy-UI techniques are proposed: chaffing by tweaking.

○ The password-change UI is changed with modified-UI procedures to authorize better password/honey word generation.

- Offer attackers an idea of how users write their passwords. Attackers will then optimize their device password collection models and design faster algorithms for cracking passwords.[45] [46]
- Honeywords is not a completely adequate solution to user authentication, Passwords can be replaced with more secure and convenient authentication methods [47], or better authentication methods can be introduced. [48][49].

- In [6] this paper the researchers proposed the honey word mechanism operates as follows: The sweet word list is created utilizing the honey word production algorithm Gene for each user. This procedure takes enter as the number of sweet words and outputs both and password list, where is the right password index (sugar word). The username and sweet word hashes are stored in the main server database while being stored in another honey checker directory. It makes it more difficult to hack the system as a whole by diversifying the hidden information in the system that stores password hashes in one server and in the honey checker, i.e. providing a basic type of distributed protection. Note that the pair is stored for each account in the standard password method, while the is stored in the database for this method.

  We point out that if a rigid protocol is followed when detecting honey words.the system can be vulnerable to distributed denial-of-service (DoS) attacks that damage the whole system.

  When an adversary steals a file of hashed passwords, he or she will also use brute-force search to locate a password whose hash value matches the hash value saved for a given user's password, enabling the adversary to impersonate the user.

  The researchers did not mention the tolerance of typos when the user enters the password

5.2- User Typos

The typographical problems that the user may mistake when entering the password are the most important challenges and important issues in honey encryption, because in this case we do not know or distinguish whether he is the legitimate user or an attacker, in this case, will appear false data to the legitimate user [40], especially If the password is strong, that is, it contains letters, numbers and special characters, the legitimate user may fall into a typo

Where many researchers have mentioned that the user may mistake typos, is another problem. Typos are also a concern for the Kamouflage method as a typo has been identified in the master password in the password store to trigger an alert in the system and probably any behavior depending on the tuning [50][51], for example, if the system is using the password error, it will be blocked The user is from the fake password, and typos in the password may lead to a scenario in which the user enters a password that appears legitimate. The consumer feels perplexed about this scenario and needs to be approached accordingly as typos are so normal in practice [52]. And one study found, when they have to enter passwords 16 to 22 times, 42 percent of users make at least one misspelling. These cases should not be dealt with the same way as the circumstances of abuse, as misinformation provided by DTE will confuse users [53] [4].

The fact that if the key is incorrect, HE creates or returns bogus results will create issues with frequent users when they fortuitously mistype the password. For instance, if honey encryption is used with PBE encrypted password vaults, and the user mistypes their password, the application will create a user's fictitious vault, which is not the intended outcome and is likely to annoy users. Therefore, this situation has two distinct potential consequences. If, yes, the is a customer, then there is no question that we give the option of retyping the password. If he was the intruder, though, we have to show him the bogus data / give an alert. The question is how do we grasp what the situation is? This topic has been discussed in researchers' papers and they have also offered some potential responses to it [4] [10].

How to differentiate between an attacker's sheer guessing of a password and a misspelling is the key issue with user misspellings? If we can prove that a honey word is not an entry, As a result, we should be certain that the database has not been tampered with. We don't know, though, whether anyone is doing a guessing/brute force. According to the concept of HE, a false result must be the result of a wrong entry. Such as credit card numbers, SSN, addresses, maybe, etc [4] .

The theory is that these misspelled words will not be present as honey words in the database, so these honey words will behave much like the actual records. So these sweet words are also legitimate credit card numbers. If the number of a credit card is misspelled, we may conclude that an assault won't occur. However, a bug will be marked and a warning submitted by Honey checker. This is the reason why honey words are needed.

The password must be checked against the honey credentials to figure out whether the database has been hacked. If your honey word list does not include the misspelled object, it is most likely misspelled and needless for alarm. There must, however, be a way to solve this problem [2] [4] [10] [13] [53].

there are approaches proposed by researchers in this field. and there are methods suggested by researchers in this field, as in the Table (1). and this many types of typos for users [53]

1- Capitalization errors
2- Errors in the first letter
3- Click a key near the intended key
4- Accidentally hit the Caps Lock key
5- Adding letter at the end or beginning of the word

And many user errors [2]:

1- In terms of typographical errors
2- In terms of choosing a weak password
3- Use one password for all their accounts
4- Using personal information to choose a password
5- Use short passwords less than 8 characters' long

It is necessary to identify or take into account the type of device used by the user when entering passwords, for example, mobile, personal computer, or otherwise, because each device has a special keyboard, so errors appear that differ from one device to another device [53]

Table 1: Definition of the typo-based solution offered by various authors

	Authors	*Characterization*	*Advantage and Disadvantage*
1	[53]	- A typo-tolerant tester that fits reasonably well with the current password authentication scheme is introduced in this proposal. - The plan found out that if the typo-tolerant system was introduced, at least a minute would have been spared for 20 percent of consumers.	For the current password-based authentication method, this approach is appropriate because it incorporates a caps lock corrector, a first-case flip corrector, and an additional character at the end corrector to improve usability, but this approach is not suitable for dealing with the HE scheme's typo issue.
2	[54]	In an offline and online environment, the scheme offered two forms of typo-safety to cope with numerous typo problems while also ensuring message recovery in a traditional HE scheme.	The kind A protocol is simpler to enforce since it needs just a server but the main downside is that the size of the key is limited and also there is the difficulty of finding typos in certain environments. kind B is an upgrade over kind A since a user can quickly find typos if he recollect his pin, but a key point here is that to check his address, the user needs to recall the pin.
3	[55]	- A customizable typo resistant password verification is presented in this plan. - This thesis suggests a straightforward blacklisting method in which it is forbidden for a limited number of dangerous typos to be admissible in the typo cache.	This thesis is an expansion of existing typo forgiving password programs but is not meant to function on typos on a decoy device. It can be changed for HE, however.
4	[4]	- rewrite the password - honey checker - honey word –service	- Help the user to retype a second password We give the attacker a bigger chance - How do we know who the attacker is from the legitimate user
5	[27]	- "online verification of plaintexts" - Honey checker	- The legitimate user enables not to make a mistake because he will check directly upon entry and have a clear idea of the structure of the entered data or dealing with it such as (SSN, E-mail, credit card number) - Where the user misspells the way that they create a proper credit card, but not the one they have, online investigation will not mark it as a misspelling and let it advance - A honey checker must be used, it will mark an error and send an alarm. This is the reason why honey words are needed. To find out if the database has been compromised - Since the verification service works with structured data, it is not a practical solution to user typos when the data consists of natural language passwords, as it is usually limited by length and some ASCII characters. This makes the online investigation service useful for PBE technologies in general and even with data with a clear architecture does not offer a

			complete solution.
6	[10]	- error-detecting codes - checksums, - online verification of plaintexts	- - It helps the user to discover and reduce errors - But it helps to reduce the size of the key area, and therefore it causes a security deterioration. Therefore, careful construction and application must be done - Fake passwords/honey tokens are suggested to be shared frankness between password vault apps and providers of services. Applying debugging codes to regular texts in HE can generate honey tags without explicit participation. - Since this technique reduces message space, it degrades protection marginally and should be used with caution. However, it does provide an intriguing way to link HE security to online security tests.
7	[13]	- Honey tokens without explicit sharing. - False passwords/honey tokens are directly exchanged between password vault apps and service providers.	The bogus passwords/honey tokens were proposed to be directly exchanged between password vault applications and service providers. Honey tags can be created without explicit involvement by the application of debugging codes to standard texts in HE. By reducing message space, this method degrades protection slightly, and it should be implemented with caution. However, it provides an intriguing comparison of HE security to online security controls.
8	[2]	- tail-tweaking methods - error-detection code	- When tail-tweaking, it can be beneficial if the password tail is distinct from the honey word tails, so that a typing error would not convert the password to a honey word. - The honey word tails can also be very distinct from one another so that the password does not stand out like the sweet word that is "the most distinctive" from the others. Typos can be detected using an error-detection code (as for ISBN book codes). - This property enables the identification of a single-digit replacement or a transposition of two neighboring digits in the tail.
9	[56]	- The first suggestion is to add any detail to the plaintext that is exclusive and verifiable by the patient but useless to the adversary. We propose adding a string of bits similar to the seed. - -Before encryption, the device should have a pool of N validation images from which the user can select. The validation images do not need to be used in the ciphertext. - is based upon the concealment of a biometric template among decoys.	- This alternative works best if the PIN is a uniformly random string; otherwise, since the PIN is no longer consistent, it will compromise protection. - By merely signaling that a shown image is familiar, the user can validate proper decryption. - With the age of the patient, and if he has a disease on his finger, or if he works in difficult work or difficult manual work, it may damage the fingerprint, and thus it may prevent him from entering and verifying the password
10	[57]	- Each person will create a text using a limited number of characters that are unrelated to some of the health attributes of his or her health record.	- When a user enters a password, a validation text appears, and the user can check to see whether it is the text he or she entered. Otherwise, the recipient must retype his or her email.
11	[44]	the use of honeywords made up of other people's passwords	The user is unable to forget the passwords of other users.
12	[50]	- Type scheme is designed for a conventional client-server model. - - Type scheme is designed for an extended system model with an additional database manager.	- In terms of false-positive score, it has a disadvantage. Even if a user does not make any typos in the password, he or she will see that there are certain typos in the password. - Users are burdened by having to memorize side detail in addition to passwords. - The type scheme was created with a traditional client-server model in mind. Even among the schemes, it is the easiest and most efficient. - The type scheme is intended for a framework model that includes an external database manager. It solves the first form of accuracy problem. By using additional side details, it provides high precision in detecting typos (e.g., PIN).

6. Conclusions

After we made a comprehensive survey on the honey algorithm and mentioned and reviewed the work of researchers in various fields, it was found that the honey algorithm has proven its effectiveness against the attack by brute force and when using the hash functions with it in building DTE and using salting it becomes stronger and works to repel other attacks such as rainbow attack and dictionary attack And other attacks, taking into account the challenges faced, the most important of which is the creation of honey words, typos and the quality of DTE construction, and the honey algorithm can be used or combined with other algorithms to be stronger and more effective and to overcome some of the gaps that the honey algorithm suffers from such as the chosen-ciphertext attack (CCA). Finding several solutions to overcome typographical errors among users and highlighting the most important advantages and disadvantages of each method in detail, and the most prominent features in the honey algorithm that:

1. Verify valid password authentication
2. Achieve data integrity in the valid password
3. Used for PIN, RSA, PW
4. Needs to :

1) The best way to produce honey words
2) How to build an optimization of DTE?
3) What is the formula for calculating a message's probability?

4) How can you be confident that the perpetrator won't be able to tell the difference between a false and a true message?

7.Recommendations

A new encryption method has been system proposed to improve security by integrating cryptography mechanisms and data steganography mechanisms in an effective way to repel some of the most powerful types of attacks such as brute force attack, dictionary attack, and rainbow attack by designing and implementing a new approach.

7.1 Design the Proposed System

The proposed system as shown in figure (4), is divided into two main parts: the first part is the encryption and hiding of the message (data) and the second part is to decrypt and extracting the message (data)

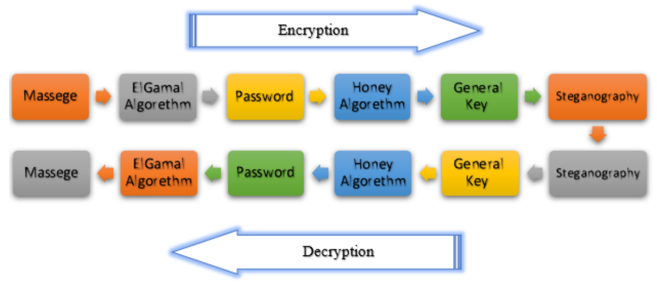


Figure (4): the Design of the Proposed System

7.2 The Flowchart for The Proposed System

the following flowchart in figure (5), shows the steps, operation of the system, and the stages Implementation

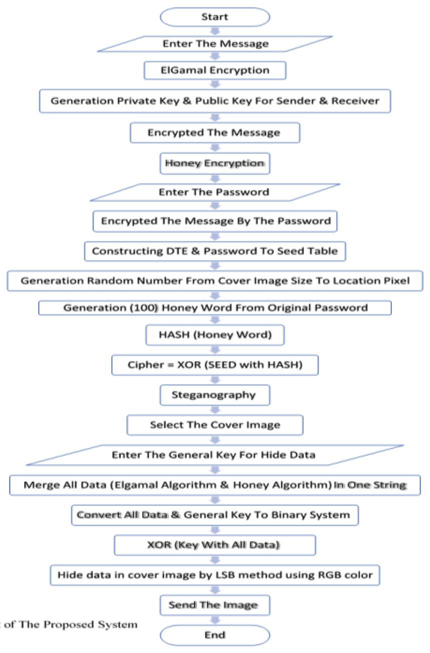

Start
Enter The Message
ElGamal Encryption
Generation Private Key & Public Key For Sender & Receiver
Encrypted The Message
Honey Encryption
Enter The Password
Encrypted The Message By The Password
Constructing DTE & Password To Seed Table
Generation Random Number From Cover Image Size To Location Pixel
Generation (100) Honey Word From Original Password
HASH (Honey Word)
Cipher = XOR (SEED with HASH)
Steganography
Select The Cover Image
Enter The General Key For Hide Data
Merge All Data (Elgamal Algorithm & Honey Algorithm) In One String
Convert All Data & General Key To Binary System
XOR (Key With All Data)
Hide data in cover image by LSB method using RGB color
Send The Image
End

Figure (5) : The Flowchart of The Proposed System

7.3 Algorithm of the Proposed system

```
1.  Input the massage (msg)
2.  Let q = random (1024, 2048) , Let g = random (2, q)
3.  Generation the Private key for sender k = gen_key(q)
4.  Generation Public key receiver h = power (g, key, q)
5.  Generation masking key s = power (h, k, q)
6.  Encryption the massage c=msg*s
7.  Input the password (userpass)
8.  ps=The sum of the lengths of all the parameters
9.  Generation ri random (imgsiz([0]*imgsiz[1],ps)
10. While lop <100 for 100 honey word
        a. For i in range (len(userpass))
        b. Honeyword[lop]=Userpass[i].random (a-z) or (A-Z) or (0-9) or Special characters
        c. Hash(honeyword[lop])
        d. Next i
11. Loop while
12. Cipher= trueseed XOR honeytohash[userpass]
13. Input the stego-key (master key)
14. Input the name of the resulting image file with the Extension
15. Keyrep=Repeat the stego-key to length message
16. secret_message= merge(all data(p,key,q,honeytohash, cipher…etc)
17. bin_sec_msg = messageToBinary(secret_message)
18. kbin=messageToBinary(keyrep)
19. bin_xor = bin_sec_msg XOR kbin
20. for pixel in image
21. (r,g,b)= message to binary (pixel)
22. Pixel[7]= bin_xor[i]
23. Next pixel
```

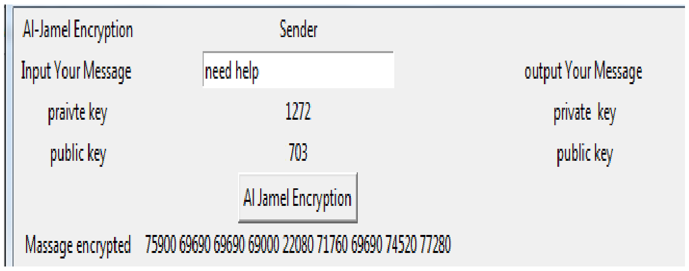
7.4 Resources for Proposed Method

To implement the proposed methods in this paper, Python 3.7 and using compiler Spyder (anaconda3) was used as a tool for programming language. A Windows10-based intel (R) Core i5 gen 1 processor (2.53 GHz) and system type 64-bit operating system. A laptop with 8 Gigabyte RAM has been used to perform all the experiments reported in this paper.

7.5 System Implementation

To evaluate the system, examine its performance, study its feasibility, and compare it with previous research, the system must first be implemented and data entered for all stages of the system.

A. ElGamal Encryption Parameter as shown in the figure (6):


Al-Jamel Encryption — Sender

Input Your Message: need help — output Your Message

praivte key: 1272 — private key

public key: 703 — public key

Al Jamel Encryption

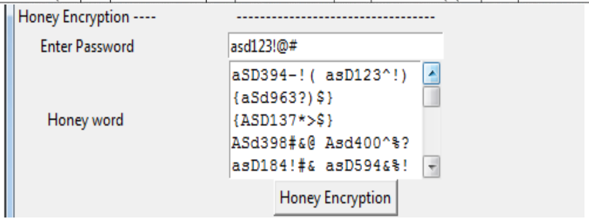
Massage encrypted: 75900 69690 69690 69000 22080 71760 69690 74520 77280

Figure (6) : first stage, the ElGamal Encryption

B. Honey encryption as shown in Table (2) and Figure (7):

Table (2): The Honey Word

1	AsD800=)^	21	asd684\?_	41	ASD266->?	61	aSd162_(%	81	ASD832(@?
2	ASd126?)$	22	aSd381%=?	42	asD266<)@	62	Asd183<$?	82	AsD424<?-
3	aSD569\<?	23	Asd192>!$	43	aSD125!=$	63	asD452>)$	83	Asd862_-<
4	asD912)>&	24	ASD248>%/	44	AsD310@*^	64	ASD404%<&	84	asd921=)=
5	asD703\-*	25	aSD867(#@	45	asd774*#(	65	aSd635&>@	85	AsD610@!<
6	Asd221!?-	26	AsD143!>@	46	AsD481#*&	66	aSd635&>@	86	asD987/(/
7	asD520\<=	27	AsD739!(=	47	aSd622(!/	67	aSd937	87	Asd297=_&
8	aSD706<#$	28	{asd626@$<	48	aSd153$$<	68	asd143-#&	88	aSD821/?<
9	AsD210)!_	29	ASD066/_(	49	asD177))\	69	ASD793-?$	89	aSd379$#^
10	Asd775>$!	30	asd123!@#	50	Asd638?#@	70	aSd499*\$	90	asD714)$<
11	Asd419>?_	31	aSd210)?-	51	Asd358&\$	71	asD181(@#	91	aSd974=>>
12	AsD325*^%	32	Asd548*\?	52	aSD737=!-	72	asD421?-^	92	aSD818@<>
13	AsD259*/	33	Asd249?!&	53	asD529^=/	73	Asd614(@\	93	asD900!=@
14	aSd600/=_	34	aSd921>@_	54	ASd543>$^	74	Asd528?=\	94	aSd484!!@
15	aSD615\<>	35	asd857(_	55	asD840%?(	75	aSD873\##	95	ASd454_<>
16	ASd813=?\\	36	aSD757^\#	56	ASD605@^>	76	ASD941?_/	96	AsD540\\@
17	asd054(-$	37	aSD268_?/	57	Asd100^\?	77	AsD393^#&	97	aSD584&#<
18	AsD957?_	38	Asd604@$#	58	Asd078?(>	78	ASD574@_)	98	AsD215-^-
19	Asd705%%^	39	asD631%)\	59	AsD424<?-	79	ASD047?))	99	Asd341%<?
20	Asd965\(-	40	asD565?*#	60	Asd952?#$	80	aSD122(?(	100	aSd789&-%


Honey Encryption ---- ----------------------------------

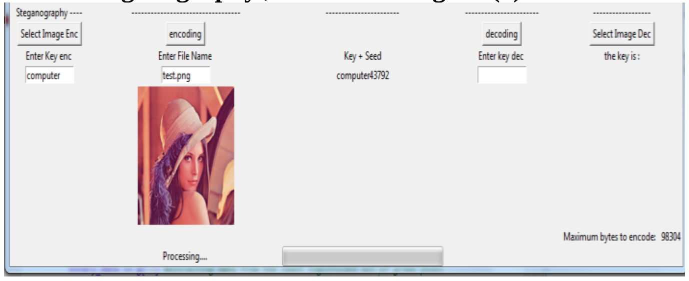
Enter Password: asd123!@#

Honey word:
```
aSD394-!( asD123^!)
{aSd963?)$}
{ASD137*>$}
ASd398#&@ Asd400^%?
asD184!#& asD594&%!
```

Honey Encryption

Figure (7): second stage, the Honey encryption

C. Steganography , as shown in figure (8)


Steganography ----

Select Image Enc — encoding — decoding — Select Image Dec

Enter Key enc: computer

Enter File Name: test.png

Key + Seed: computer43792

Enter key dec:

the key is :

Maximum bytes to encode: 98304

Processing...

Figure (8): third stage, the Steganography

7.6 System Evaluation

In order to know the efficiency of the proposed system, the outputs and mechanisms used in the success of the desired goal in this paper must be examined, so both the steganography and cryptography were taken separately, and evaluation measures such as (PNSR, MSE, HISTOGRAM, SSIM) were discussed and compared with previous work regarding Hiding data and NIST metrics for encryption

7.6.1 Evaluation Cryptography

In this section, the evaluation of cryptography according to the NIST standards, the examination of 10 standards in the proposed system, and the achievement of the objectives of the paper will be discussed

- **NIST Test**

The standard Statistical test NIST applied on the results in text, private key, the public key for sender and receiver. Two types of Frequency tests, that count the number of ones in the key-stream, run test that depends on the run length of ones and zeros, The Discrete Fourier Transform (DFT) Test, The Non-overlapping Template Matching (partition block) Test, The Approximate Entropy Test, and The Cumulative Sums (forward, reverse) Test. All four keys and message of encrypted tests passed the tests (Success) and P-Value was presented in table (3)

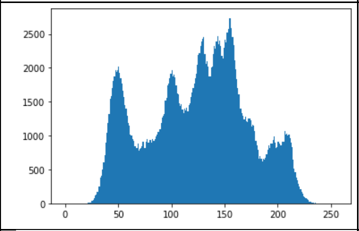
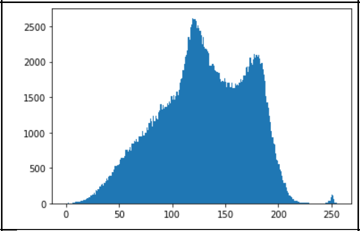
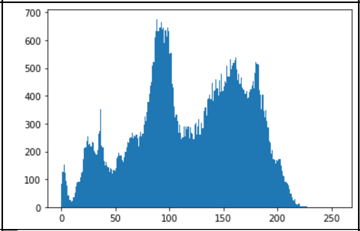
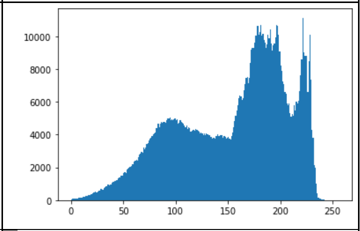
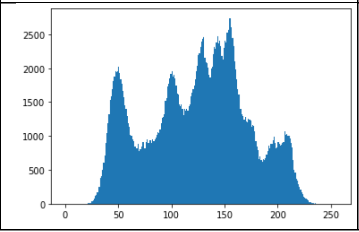
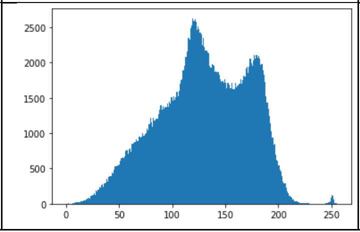
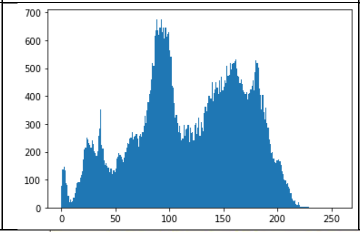
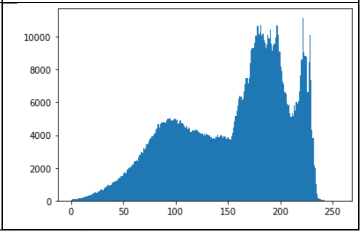
Table (3): NIST test for ElGamal encrypted

	NIST	P-Value				
		Private Key Sender	Public Key Sender	Private Key Recipient	Public Key Recipient	The Message
1.	Frequency Test	0.47950	0.68309	0.28884	0.47950	0.74397
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
2.	Frequency Test (partition block)	0.47950	0.68309	0.28884	0.47950	0.85968
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
3.	Runs Test	0.65346	0.43072	0.83669	0.92845	0.63013
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
4.	Serial test	0.49896	0.49896	0.49896	0.49896	0.49896
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
5.	DFT Test	0.74560	0.45369	0.74560	0.74560	0.03935
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
6.	Non-overlapping Template Matching Test	0.99999	0.99999	0.99999	0.99999	0.99999
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
7.	Approximate Entropy Test	1.0	1.0	1.0	1.0	1.0
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
8.	Cumulative Sums Test (forward)	0.57549	0.80701	0.31457	0.57549	0.79990
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
9	Cumulative Sums Test (reverse)	0.89202	0.80701	0.57549	0.89202	0.99254
		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
10	Longest Run of Ones in a Block	-	-	-	-	0.36099
						SUCCESS

7.6.2 Histogram analysis:

By applying a histogram analysis, the accuracy of images could be visually noted. We embedded all data (6240-byte for data such as ElGamal algorithm, Honey algorithm, Master Key) in each of the images in this test, then applied the histogram statistical analysis to get the histograms for all images before encryption and their corresponding image after encryption.as shown in table (4).

Table (4): Histogram for all images (after, before)

	Lena	Baboon	Peppers	Animal
Before				
After				

7.6.3 PSNR and MSE

Table (5) PSNR, MSE compare with proposed system

Text (kB)	Method	Pepper		Lena		Animal	
		PSNR(dB)	MSE	PSNR(dB)	MSE	PSNR(dB)	MSE
1	proposed system	65.1516	0.02001	77.1771	0.00502	89.1127	0.00128
	[58]	60.883	0.07543	66.960	0.01667	73.063	0.00405
2	proposed system	62.0701	0.04068	74.0754	0.01025	86.0960	0.00258
	[58]	57.985	0.14986	64.009	0.03317	70.109	0.00820
4	proposed system	58.9713	0.08305	71.0164	0.02074	83.0353	0.00522
	[58]	54.996	0.29764	61.078	0.06402	67.118	0.01569
8	proposed system	55.9520	0.16644	67.9774	0.04176	80.0205	0.01045
	[58]	51.930	0.60484	58.040	0.12784	64.105	0.03171
16	proposed system	52.928	0.32268	64.9658	0.08355	77.0214	0.02085
	[58]	48.989	1.14495	55.048	0.25534	61.061	0.06473

7.6.4 Correlation analysis

Table (6): correlation coefficients for image (animal 1024*1024)

Byte	Correlation Coefficients (Original Image)			Correlation Coefficients (Our)			Different		
	Horz	Vert	Dig	Horz	Vert	Dig	Horz	Vert	Dig
1000	0.9967	0.9968	0.9943	0.9974	0.9970	0.9936	0.0007	0.0002	0.0007
1024	0.9973	0.9972	0.9934	0.9975	0.9973	0.9948	0.0002	0.0001	0.0014
2012	0.9972	0.9977	0.9927	0.9974	0.9973	0.9945	0.0002	0.0004	0.0018
2024	0.9976	0.9969	0.9943	0.9980	0.9969	0.9941	0.0004	0.0000	0.0002
2048	0.9969	0.9972	0.9935	0.9968	0.9969	0.9939	0.0001	0.0003	0.0004
4096	0.9978	0.9973	0.9942	0.9973	0.9972	0.9936	0.0005	0.0001	0.0006
6240	0.9969	0.9973	0.9937	0.9969	0.9969	0.9944	0.0000	0.0004	0.0007
8192	0.9974	0.9972	0. 9931	0. 9975	0. 9972	0.9941	0.0001	0.0000	0.0010
16384	0.9974	0. 9970	0. 9934	0. 9973	0. 9976	0. 9928	0.0001	0.0006	0.0006
32768	0.9973	0.9969	0.9937	0.9974	0.9969	0.9928	0.0001	0.0000	0.0009

Table (7): correlation coefficients for image (peppers 256*256)

Byte	Correlation Coefficients (Original Image)			Correlation Coefficients (Our)			Different		
	Horz	Vert	Dig	Horz	Vert	Dig	Horz	Vert	Dig
1000	0.9734	0.9615	0.9287	0.9716	0.9696	0.9363	0.0018	0.0081	0.0076
1024	0.9728	0.9660	0.9389	0.9693	0.9568	0.9430	0.0035	0.0092	0.0041
2012	0.9702	0.9685	0.9425	0.9715	0.9678	0.9338	0.0013	0.0007	0.0087
2024	0.9743	0.9633	0.9452	0.9712	0.9627	0.9365	0.0031	0.0006	0.0087
2048	0.9739	0.9719	0.9481	0.9727	0.9623	0.9396	0.0012	0.0096	0.0085
4096	0.9692	0.9742	0.9403	0.9731	0.9648	0.9393	0.0039	0.0094	0.001
6240	0.9679	0.9644	0.9410	0.9753	0.9651	0.9414	0.0074	0.0007	0.0004
8192	0.9721	0.9681	0.9491	0.9718	0.9680	0.9310	0.0003	0.0001	0.0181
16384	0.9742	0.9656	0.9482	0.9715	0.9616	0.9436	0.0027	0.004	0.0046

1. Do not use the keys for the hash function in the encryption because that will complicate the issue to the user by using additional keys that we do not need
2. We encoded the Seed with the hash function after adding "salt" to it to be strong to repel brute force attack, rainbow attack, and dictionary attack.

7.6.5 SSIM Test

Table (8): SSIM test for all images

Byte	Animal (1024×1024)	Baboon (499×499)	Lena (512×512)	Peppers (256 ×254)
849	-	0.9999998	0.9999962	0.9999992
1,000	0.9999996	-	0.9999927	0.9999888
1,024	0.9999999	-	-	0.9999878
1,648	-	0.9999982	0.9999615	0.9999865
2,012	0.9999943	-	0.9999448	0.9998867
2,024	0.9999941	-	0.9999441	0.9998859
2,048	0.9999938	-	0.9999426	0.9998827
2,547	-	0.9999960	0.9999171	0.9999821
3,396	-	0.9999936	0.9998743	0.9999686
4,096	0.9999582	-	0.9998385	0.9996061
4,287	-	0.9999912	0.9998286	0.9999428
6,240	0.9999167	0.9999851	0.9997270	0.9993498
8,192	0.9998791	-	0.9996315	0.9991722
16,384	0.9997262	-	0.9992997	0.9991722
32,768	0.9994481	-	0.9987354	-

8. References

[1]. Noorunnisa, N. S., & Afreen, D. K. R. (2016). Review on Honey Encryption Technique. International Journal of Science and Research (IJSR) ISSN (Online), 2319-7064.

[2]. Juels, A., & Rivest, R. L. (2013, November). Honeywords: Making password-cracking detectable. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 145-160).

[3]. Juels, A. (2014, June). A bodyguard of lies: the use of honey objects in information security. In Proceedings of the 19th ACM symposium on Access control models and technologies (pp. 1-4).

[4]. Lindholm, R. (2019). Honey Encryption: implementation challenges and solutions.

[5]. ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 31(4), 469-472.

[6]. Meier, A. V. (2005, June). The elgamal cryptosystem. In Joint Advanced Students Seminar.

[7]. Oppliger, R. (2011). Contemporary cryptography. Artech House

[8]. Taneski, V., Heriˇcko, M., & Brumen, B. (2014). Password security—No change in 35 years?In 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 1360–1365). IEEE. Retrieved from https://ieeexplore.ieee.org/document/6859779

[9]. Abdalla, N. A. A. (2019). Preservation of Data Confidentiality Using Honey Encryption (Doctoral dissertation, Sudan University of Science and Technology).

[10]. Juels, A., & Ristenpart, T. (2014, May). Honey encryption: Security beyond the brute-force bound. In Annual international conference on the theory and applications of cryptographic techniques (pp. 293-310). Springer, Berlin, Heidelberg.

[11]. SAHU, S. (2020) Providing Information Security Using Honey Encryption.

[12]. Omolara, A. E., Jantan, A., & Abiodun, O. I. (2019). A comprehensive review of honey encryption scheme. Indonesian Journal of Electrical Engineering and Computer Science, 13(2), 649-656.

[13]. Bojinov, H., Bursztein, E., Boyen, X., & Boneh, D. (2010, September). Kamouflage: Loss-resistant password management. In European symposium on research in computer security (pp. 286-302). Springer, Berlin, Heidelberg.

[14]. Yajun, G. U. O., & Dongqi, P. U. (2019). Privacy Data Protection Based on the Honey Encryption. Netinfo Security, 19(12), 38.

[15]. Latha.K, Sheela.T (September 2019) . Reducing Cloud Data Breaches and Improving Data Security using Honey Encryption Algorithm. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue-11S

[16]. Rani, D. N. U., Ahmad, S. N., Reddy, C., & Lakshmi, P. J. (2018, February). Honey Maze Encryption (Home). In 2018 IADS International Conference on Computing, Communications & Data Engineering (CCODE).

[17]. Noorunnisa, N. S., & Afreen, D. K. R. (2019). Honey Encryption based Password Manager. Journal of Emerging Technologies and Innovative Research (JETIR) ISSN: 2349-5162.

[18]. Srinivasu, N., Sahil, M., Francis, J., & Pravallika, S. (2017). Security enhanced using honey encryption for private data sharing in cloud. International Journal of Engineering & Technology, 7(1.1), 675-678.

[19]. Fun, T. S., Samsudin, A., & Zaaba, Z. F. (2017). Enhanced security for public cloud storage with honey encryption. Advanced Science Letters, 23(5), 4232-4235.

[20]. Huang, Z., Ayday, E., Fellay, J., Hubaux, J. P., & Juels, A. (2015, May). GenoGuard: Protecting genomic data against brute-force attacks. In 2015 IEEE Symposium on Security and Privacy (pp. 447-462). IEEE.

[21]. Arun, S., & Shanker, N. R. (2006). Data Security In Cloud Storage Using Advanced Encryption Standard And Honey Cryptography.

[22]. Srilatha Komakula, V.Shobha Rani (2020).Honey Encryption With Quantum Key Distribution. International Journal For Innovative Engineering and Management Research- IJIEMR (Vol 09 Issue01, Jan 2020 ISSN 2456 – 5083)

[23]. Mok, E., Samsudin, A., & Tan, S. F. (2017). Implementing the honey encryption for securing public cloud data storage. In First EAI International Conference on Computer Science and Engineering (Vol. 10).

[24]. Tan, S. F., & Samsudin, A. (2018). Enhanced security of internet banking authentication with extended honey encryption (XHE) scheme. In Innovative Computing, Optimization and Its Applications (pp. 201-216). Springer, Cham.

[25]. Tyagi, N., Wang, J., Wen, K., & Zuo, D. (2015). Honey encryption applications. Network Security, 2015, 1-16.

[26]. Rajalakshmi, M., & Parthasarathy, C. (2006). An Implementation Of Fhma For Honey Encrypted Datasets In Wireless Sensor Networks.

[27]. A. Juels and T. Ristenpart, "Honey Encryption: Encryption beyond the Brute-Force Barrier," in IEEE Security & Privacy, vol. 12, no. 4, pp. 59-62, July-Aug. 2014, doi: 10.1109/MSP.2014.67.

[28]. Jaeger, J., Ristenpart, T., & Tang, Q. (2016, May). Honey encryption beyond message recovery security. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 758-788). Springer, Berlin, Heidelberg.

[29]. Menezes, A.J., van Oorschot, P.C., & Vanstone, S.A. (1997). Handbook of Applied Cryptography (1st ed.). CRC Press. https://doi.org/10.1201/9780429466335

[30]. Dougherty, C. R. (2009). Vulnerability Note VU# 836068 MD5 vulnerable to collision attacks. Retrieved August, 26, 2009.

[31]. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A. K., Molnar, D., Osvik, D. A., & de Weger, B. (2008). MD5 considered harmful today, creating a rogue CA certificate. In 25th Annual Chaos Communication Congress (No. CONF).

[32]. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2017, August). The first collision for full SHA-1. In Annual International Cryptology Conference (pp. 570-596). Springer, Cham.

[33]. Devi, R., & Sujatha, P. (2020). A Hybrid Watermarking System for Securing Multi-modal Biometric Using Honey Encryption and Grasshopper Optimization Technique. In Intelligent Computing and Innovation on Data Science (pp. 725-734). Springer, Singapore.

[34]. Piyush, "Advanced Honey Encryption: An Escape-less Trap for Intruders," 2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018, pp. 1-4, doi: 10.1109/CCAA.2018.8777468.

[35]. Rajalakshmi, M., & Parthasarathy, C. (2006). AN IMPLEMENTATION OF FHMA FOR HONEY ENCRYPTED DATASETS IN WIRELESS SENSOR NETWORKS.

[36]. Kapil, G., Agrawal, A., Attaallah, A., Algarni, A., Kumar, R., & Khan, R. A. (2020). Attribute based honey encryption algorithm for securing big data: Hadoop distributed file system perspective. PeerJ Computer Science, 6, e259.

[37]. Ravindranadh, K., Kiran, M. S., Kumar, B. D. S. P., & Priyanka, D. (2018). Data migration in cloud computing using honey encryption. International Journal of Engineering & Technology, 7(2.8), 230-234.

[38]. Panchal, K. (2020). Differential Privacy and Natural Language Processing to Generate Contextually Similar Decoy Messages in Honey Encryption Scheme. arXiv preprint arXiv:2010.15985.

[39]. KURNAZ, S., & Mohammed, A. H. (2020, June). Secure Pin Authentication in Java Smart Card Using Honey Encryption. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) (pp. 1-4). IEEE.

[40]. Abiodun, E. O., Jantan, A., Abiodun, O. I., & Arshad, H. (2020). Reinforcing the Security of Instant Messaging Systems Using an Enhanced Honey Encryption Scheme: The Case of WhatsApp. Wireless Personal Communications, 112(4), 2533-2556.

[41]. Huang, Z., Ayday, E., Fellay, J., Hubaux, J. P., & Juels, A. (2015, May). GenoGuard: Protecting genomic data against brute-force attacks. In 2015 IEEE Symposium on Security and Privacy (pp. 447-462). IEEE.

[42]. Yin, W., Indulska, J., & Zhou, H. (2017). Protecting private data by honey encryption. Security and Communication Networks, 2017.

[43]. Yin, W., Indulska, J., & Zhou, H. (2017). Protecting private data by honey encryption. Security and Communication Networks, 2017.

[44]. Win, T., & Moe, K. S. M. Protecting Private Data using Improved Honey Encryption and Honeywords Generation Algorithm.

[45]. Sawant, S., Saptal, P., Lokhande, K., Gadhave, K., & Kaur, R. (2018). Honeywords: Making Password Cracking Detectable. International Journal of Engineering Research and Advanced Technology-IJERAT (ISSN: 2454-6135), 4(4), 01-06

[46]. Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L., ... & Lopez, J. (2012, May). Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In 2012 IEEE symposium on security and privacy (pp. 523-537). IEEE.

[47]. A. Czeskis, M. Dietz, T. Kohno, D. Wallach, and D. Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In ACM CCS, pages 404–414, 2012.

[48]. O. Kharif. Innovator: Ramesh Kesanupalli's biometric passwords stored on devices. Bloomberg Businessweek, 28 March 2013.

[49]. T. Wadhwa. Why your next phone will include fingerprint, facial, and voice recognition. Forbes, 29 March 2013.

[50]. Choi, H., Jeong, J., Woo, S. S., Kang, K., & Hur, J. (2019). Password typographical error resilience in honey encryption. Computers & Security, 87, 101411.

[51]. Chatterjee, R., Bonneau, J., Juels, A., & Ristenpart, T. (2015, May). Cracking-resistant password vaults using natural language encoders. In 2015 IEEE Symposium on Security and Privacy (pp. 481-498). IEEE.

[52]. Burgess, J. (2017). Honey Encryption Review. Queen's University Belfast.

[53]. Chatterjee R, Athayle A, Akhawe D, Juels A, Ristenpart T. pASSWORD tYPOS and how to correct them securely. In Security and Privacy (SP), 2016 IEEE Symposium on 2016 May 22 (pp. 799-818). IEEE.

[54]. Choi H, Nam H, Hur J. Password typos resilience in honey encryption. In Information Networking (ICOIN), 2017 International Conference on 2017 Jan 11 (pp. 593-598). IEEE.

[55]. Chatterjee R, Woodage J, Pnueli Y, Chowdhury A, Ristenpart T. The TypTop System: Personalized Typo-tolerant Password Checking. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security 2017: 329-346

[56]. Huang, Z., Ayday, E., Fellay, J., Hubaux, J. P., & Juels, A. (2015, May). GenoGuard: Protecting genomic data against brute-force attacks. In 2015 IEEE Symposium on Security and Privacy (pp. 447-462). IEEE.

[57]. Choi H, Nam H, Hur J. Password typos resilience in honey encryption. In Information Networking (ICOIN), 2017 International Conference on 2017 Jan 11 (pp. 593-598). IEEE.

[58]. Al-Qwider, W. H., & Salameh, J. N. B. (2017). Novel technique for securing data communication systems by using cryptography and steganography. Jordanian Journal

of Computers and Information Technology (JJCIT), 3(2), 110-130.

تقنيات أمان تشفير العسل: ورقة مراجعة

احمد سامي نوري

قسم علوم الحاسوب، كلية علوم الحاسوب والرياضيات، جامعة الموصل، الموصل، العراق

ahmed.s.nori@uomosul.edu.iq

عمار عبد المجيد غربي

قسم تقنيات أنظمة الحاسوب، معهد التقني نينوى، الجامعة التقنية الشمالية، الموصل، العراق

ammarmajed@ntu.edu.iq

تاريخ استلام البحث 2021/5/5

تاريخ قبول البحث 2021/6/21

الخلاصة:

بين الحين والآخر نسمع في الأخبار عن اختراق أو هجوم على بعض الشركات المعروفة على أنه مجرد خبر، لكنها مشكلة خطيرة لأنها خصوصية المواطنين وأموالهم في التجارة وإدارة أعمالهم ومشاريعهم. في هذه الورقة، نقدم مراجعة لمخطط تشفير العسل. تشفير العسل هو نظام التشفير الذي يضمن المرونة مقابل هجوم القوة الغاشمة من خلال توفير نص معقول ولكنه خطأ لكل مفتاح غير صالح يستخدمه المتسلل لفك تشفير رسالة، هناك مجالان رئيسيان مفتوحان، من الصعب إنشاء رسائل مزيفة مقنعة ويكون مثاليًا بما يكفي لخداع المهاجم، حتى عندما يعتقد المهاجم أن لديه الرسالة في شكلها الأصلي.

المشكلة الأخرى، مشكلة الخطأ المطبعي، حيث يبدو النص المزيف صالح للمستخدم الشرعي، عندما يقوم بإدخال المفتاح الخطأ عن طريق الخطأ. هدفنا هو الحصول على مزيد من الحيل المقنعة التي ترضي الجميع والتي تكون مثالية بدرجة كافية لمنع المتسلل من الحصول على الرسالة الأصلية، ونحتاج أيضًا إلى طرق أمان جديدة، لأن المهاجمين يبحثون عن طرق جديدة لمهاجمة الأنظمة، لذلك اقترحنا طريقة جديدة وجيدة لحماية الرسائل وكلمات المرور ويصعب كسرها وأخذ كل احتمالات الهجوم بما في ذلك هجوم القوة الغاشمة، ثم يتم إخفاء البيانات في صورة بمفتاح سري عام.

الكلمات المفتاحية: تشفير العسل، خوارزمية الجمل، إخفاء المعلومات، هجوم القوة الغاشمة، مشكلة مطبعية، كلمة العسل، DTE.