Honey Encryption: Myth, Challenges and Strategies for Designing Convincing Decoy Message to Deflect Attack on a Network Resource

## Abiodun Esther Omolara1, Aman Jantan2, Oludare Isaac Abiodun3, Howard Eldon Poston4

1,2,3School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia

4Department of Computer Science, University of Dayton, Ohio, USA

aeomolara@gmail.com, aman@usm.my, aioludare@usm.my, howard.poston@gmail.com

|  |  |  |
| --- | --- | --- |
| **Article Info** |  | **ABSTRACT** |
| ***Article history:***Received Aug 13th, 2018Revised Aug 20th, 2018Accepted Aug 26th, 2018 |  | The application of decoy is one of the key defensive strategy being explored in contemporary information security. Honey Encryption (HE) is a decoy-based encryption scheme which provides plausible-looking but misleading information to deceive attackers from stealing a network resource. However, it is astonishing to note that experts and researchers in security are yet to maximize the full potential of Honey encryption in fooling the adversary and keeping him away from the resource. Given the anticipated quantum era which is very close, there is an urgent need to research on other variants of encryption schemes as current encryption systems are susceptible to attack from quantum computers. This challenge prompted us to dig further and research other kinds of encryption scheme that may be resistant to quantum computers. We present a detailed literature review on Honey Encryption Scheme and its evolvement throughout the years. Our goal is to foster research in this domain and to furnish researchers with a detailed framework of the HE scheme. Moreover, this research provides the reader with a clear understanding of the current issues, challenges, emerging trends in HE and area of focus for newcomer researchers. We identified two major open areas which are the difficulty of creating semantically and contextually plausible-looking and convincing decoys that is good enough to fool the attacker into believing he has the original resource. Secondly, typo problem; where a fake plaintext appears valid to a legitimate user when he mistakenly enters a wrong key. Our findings consolidate the need for further research as state-of-the-art research fails to produce convincing decoys that are good enough to keep the attacker from acquiring the message. |
| ***Keyword:***Brute-ForceHoney EncryptionDistribution Transforming Encoder (DTE)Password-Based Encryption (PBE)Decoys  |
| *Copyright © 201x Institute of Advanced Engineering and Science. All rights reserved.* |
| ***Corresponding Author:***Abiodun Esther Omolara, School of Computer Sciences, Universiti Sains Malaysia, Penang.Email: styleest2011@gmail.com |

1. **INTRODUCTION**

The craft of deception is indispensable in the event of confronting an enemy. It enables an environment where an adversary is trapped into taking actions that consume/wastes his resources [1-3]. Employing deception and decoy techniques in network systems help to detect, trace, monitor and deter the activities of an adversary [4-6]. It is staged to make the adversary’s life difficult where a false reality is projected as a reality to him. Indeed, Sir Sun Tzu encapsulated the art of deception in a perspicuous sentence when he said, "***The art of war teaches us******not on the likelihood of the enemy’s not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable*,**" [7].

We trace historical examples of the use of decoys to 1943 when the British found the corpse of a homeless man and went through extraordinary length to fabricate his death and created a prior but fake existing personal life for him to deceive the Germans. His realistic but fake persona included him been a captain in the military, having a father whom he sends/receives letters from and a fiancee's letter and photo. Also, the British planted some fake papers on him indicating a false location for an Allied attack. Of course, the Germans found the dead man’s body and the letters on him. They read the letters and believed everything on it based on the ‘*supposed*’ evidence found on him. Subsequently, they diverted their attention and military warfare to some other region. Unknown to them, they were conned and the Allied troop landed. For a long time, the German military continued to think that there was a diversion even after the Allied troop landed. History has it that this British facade of the human decoy saved over forty (40) thousand Allied lives [8].

In recent times, decoy/honey systems such as honeypots/honey servers, honey tokens, honey accounts, honeywords which portray false resource have been deployed in various network systems to detect, observe and thwart attacks from cybercriminals [9-10]. Honey encryption (HE) proposed by Juels and Ristenpart [11-12] is structured under the decoy-framework. It is an encryption scheme that yields valid-looking but fake message upon decryption with a wrong key thus an adversary gains no information about the original message.

The advent of highspeed and supercharged parallel and distributed systems (such as Graphical Processing Unit, Field Programmable Gate Arrays) paved way for gathering, analyzing and processing large chunks of data often referred to as big data [13]. However, this huge advancement put cryptosystems at a disadvantage as attackers leverage on the high computational power of this systems to carry out brute-force attacks [14]. A predominant network attack that often jeopardizes computers connected to a network is the brute force attack. Even though conventional encryption schemes continue to guarantee security by increasing the size of the key or computational infeasibility of searching for the key, most cryptosystems fail to withstand cryptanalysis attack specifically the brute-force attack [15].

Honey encryption was proposed as a countermeasure to brute-force attack on conventional encryption schemes specifically for min-entropy systems like passwords. It was observed from studies of persistent data breach that users chose simple, weak and predictable passwords which makes them susceptible to brute force attack [16-17]. Honey encryption acts as a supplementary encryption to fortify the conventional Password-Based Encryption (PBE) scheme. It involves a two-stage process, the first phase is the encode/decode algorithm using the Honey encryption and the second phase is the encrypt/decrypt using the conventional encryption schemes.

Honey encryption is an encryption scheme that is designed to supply fake but plausible-looking message to an attacker that tries to decrypt a ciphertext with an incorrect key. HE addresses the flaws of password-based encryption schemes and is currently employed in securing most password-based system in form of honeywords. However, honey encryption has not been employed in most systems such as, its application for encrypting human written documents like e-mails, large text, etc. The challenge is how do we create contextually and semantically correct decoy-message that can actually fool an attacker? Other problems like typo-safety have remained unaddressed. For instance, if a legitimate receiver mistakenly enters a wrong password. Given that decoy system can completely address brute force attack which standard encryption schemes are susceptible to, then there is need to foster research in this area. Moreover, the current advance made on quantum computers propels us to search for quantum-safe cryptosystems. Since all our encryption schemes are exclusively based on Mathematical problems which are established based on the difficulty of solving discrete logarithm and number factorization problem [18], this puts us at an ‘encryptionless’ state as soon as quantum computers solve all the underlying Mathematics use to secure our modern cryptosystems. The HE scheme is not devised under the computational difficulty of breaking them alone but the real trickery that cryptography was meant to be built upon [10-12]. Consequently, honey encryption if properly designed and implemented, will be a good supplementary encryption scheme for the quantum era. Therefore, the target of this paper is to capture and present a synopsis of the current state of Honey encryption scheme to identify the gaps in the scheme to enable its optimization for real-life applications.

Our paper is organized as follows. First, we set the stage by presenting a detailed background of honey encryption scheme. Also, we present an up-to-date review of the literature in HE, we present a real-life scenario of the application of honey encryption. Furthermore, we discuss the criteria of honey encryption, issues and challenges. We conclude with our propositions, suggestions and identification of promising areas for future research.

##### Motivation and Justification of this Research

**Backup Encryption Scheme:** Quantum computing is an increasingly hot area for venture both from government and large enterprises like Google and Microsoft [19] and so on. It is evident that the quantum era is coming even faster than expected. Thus, the actual need for quantum-safe security mechanism in some applications.

Modern cryptographic algorithms are formulated on computational hardness assumptions based on hard mathematical problems. They are accepted secured on the ground that there are no known practical means to break them [20]. Universal quantum computers would allow solving this hard mathematical problem in a simple, fast and efficient way [21]. Complex computations like integer factorization and discrete logarithm problems, which are the heart of public-key cryptography (PKC) tools [22] such as RSA, ECC, D-H, and DSA will be solved quickly by quantum computers. Therefore, most information protection products in enterprise security systems are vulnerable to attacks with the use of a quantum computer. There is a need for a backup encryption scheme should that time come. Furthermore, we can justify this research in another context. All our encryption schemes are solely built on Mathematical problems which are founded on the basis of discrete logarithm problems and integer factorization problem, this puts us at an ‘***encryptionless***’ state as soon as quantum computers solve all the underlying Mathematics use to secure our cryptosystems. The HE scheme which is formulated on decoy and deception will be a good backup encryption scheme for the quantum era.

**An additional layer of Security for Conventional Ciphers:** The Advanced Encryption Standard (AES) is widely considered as one of the best and secured cryptosystem. AES is responsible for a substantial amount of the information security that we benefit from on a daily basis. It is applied by giant institutions such as the NSA, Apple, Microsoft et cetera.

AES is validated by standard security bodies like National Institute of Standards and Technology (NIST) and it is the most important cryptographic algorithm in use today [23]. However, it is susceptible to brute force attack [24-25]. The Honey encryption scheme provide resilience against brute-force attack and can be used alongside AES as an additional layer to reinforce conventional ciphers. The security of large chunks of data stored in the cloud remains a big challenge. The unification of AES and HE will provide optimal security for cloud-based applications.

**The geometric increase of data breaches:** There have been reported cases of data breaches; from large scale to small-scale industries, hackers are stealing passwords everywhere [26]. Large encrypted stashes of sensitive data fall into the hands of cybercriminals daily. In 2013, Adobe and Evernote recorded data breach of about 130 million and 50 million password respectively. In 2014, Yahoo and eBay recorded loss of 273 million and 145 million passwords [27-28]. In 2016, recorded data breaches increased by 40%, Yahoo also announced the most massive data breach in history last year (2016), affecting more than one billion accounts. The year 2017 is not an exception as we have recorded significant data breaches with Equifax been the latest with 143 million accounts compromised [29]. Security breaches are increasing daily in large dimension (Refer to Fig. 1). The cryptographic community is not relaxing as newer securitytechnologies are being developed to protect against this breaches. Attackers are not the less relaxing as they keep evolving with new cryptanalysis techniques and tools. There has been a rat race between the hackers and the cryptographers. As soon as the cryptographers model new counter-attack as defenses against their cryptosystems, the hackers break the cryptosystem and get around those defenses. An extensive study of how HE works and how to deploy it appropriately in enterprise system will mitigate this exponential increase in data breach.



Figure 1: The Figure shows the geometric increase in data breach from the year 2007-2016 in the industry sector. [27]

1. **BACKGROUND OF RESEARCH**

This section outlines some concepts used in the survey for basic understanding and lookup :

**Attacker:** We use words like intruder, adversary, hacker or eavesdropper interchangeably to mean an attacker.

**Plaintext:** It refers to a message in a readable format before it is encrypted. We use any of this term original text/original message, message, true-text to refer to it in this context.

**Ciphertext: C**iphertext refers to the encoded plaintext; a cipher (an algorithm) is used to transform the plaintext to be in an unreadable format. Terms like, decoy, fake-text, false-text, honey messages and also ciphertext is used to refer to it in this context.

**Message Space**: The message space represented by M contains all possible messages M of the plaintext. It is a set of all the fake messages modeled from the original message and tailored to appear like the real message. For each invalid key entered during an exhaustive search, an adversary gets a message from the message space that fools him into believing he has the original message. The message space also contains the original message. The attacker may recover this message during his search, but because he has similar messages generated from the message space, it is difficult for him to determine which is the exact message he wants. Therefore, for every Honey encrypt process, the size of M will be determined by the distribution of the original message to be encoded.

**Seed Space**: The seed space represented by S is the space of all n-bit strings (preferably a binary string) for some predetermine seed of n. The size of S is equivalent to the probability of the message distribution. A uniformly sampled seed should produce a message distributed under the probability of M. The seed space must map to every possible message space M.

**Distribution Transforming Encoder (DTE):** The DTE is defined relative to the message space M and the seed space S. It consists of a pair of algorithm for encoding and decoding.

*DTE = (Encode, Decode)*

**DTE-then-Encrypt:** A DTE-then-Encrypt is a pair of algorithm that encrypts a message by using the DTE algorithm, followed by re-encrypting the output-cipher using any conventional Encryption scheme.

**Exclusive-OR (XOR)**: An exclusive-OR often called an XORis a b[oolean operator](https://www.webopedia.com/TERM/B/Boolean_operator.html) that returns a ‘TRUE’ value only if both its [operands](https://www.webopedia.com/TERM/O/operand.html) have different values.

 For instance 0 ⊕ 0 = 0 and 1 ⊕ 0 = 1.

**Language Model:** A language model assigns [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution) over sequences of words or sentences. It is used to predict the next word/token in a sequence.

**N-gram Language Model**: An n-gram is a close-sequence of n items from a given sequence. A Unigram is an n-gram of size 1, a Bigram is an n-gram of size 2, and a Trigram is an n-gram of size 3. The Unigram for the sentence: “it rained heavily last night”; is “it.” The Bigram is “it rained,” “rained heavily,” “heavily last,” “last night.” The trigram is “it rained heavily,” “rained heavily last,” “heavily last night.”

**Markov Model:** A [stochastic](http://whatis.techtarget.com/definition/stochastic) model used to represent randomly changing systems. It works on the assumption that future states depends on current states and do not depend on past events that occurred before it. They are used to recognize patterns, learn statistics of ordered data and make predictions or estimations.

##### Overview of the Honey Encryption Scheme

We give a brief overview of the conventional encryption to ensure a basic understanding of how Honey encryption evolved.

In a conventional password-based encryption, an adversary performing a brute-force attack to obtain the key used for encrypting a message gets gibberish (non-uniform distribution) or an error symbol as the expected output when he tries a wrong key. This output is a pointer and distinguisher that the key he is trying is incorrect and he continues his search till he gets a plausible looking message which may be the plaintext. During his attack, he quickly discards the message when the distribution is non-uniform. This gives him more time to continue his search, his probability of recovering the message/plaintext is high. Figure 2 shows a detailed explanation of how a conventional encryption scheme specifically Password-based encryption responds to a brute force attack.



Figure 2: An illustration of the setting of the conventional encryption scheme in a brute-force attack.

The sender encrypts his message using a key and a cipher. He sends the ciphertext to the receiver. The receiver decrypts the ciphertext using a decryption algorithm and the same key used by the sender. An attacker that intercepts the ciphertext may try to recover the message by randomly guessing the key. In the conventional setting scenario, the attacker can immediately tell from the suppose plaintext that the key he supplied is incorrect because of the non-uniform distribution of the plaintext.

Honey encryption offers a counter-measure to the brute-force attack which is the predominant attack to conventional cryptosystems. Honey encryption serves a false-text that looks like the original text as a response to any incorrect key the adversary supplies. HE constructs a ciphertext/decoy which looks uniform, valid and legitimate when decrypted with an incorrect key. Thus, the adversary gains no information. Juels and Ristenpart demonstrate how minimum entropy keys such as passwords, credit card numbers can be protected even in the process of an exhaustive key search using the HE scheme [11-12].

**We describe an attack model of the HE scheme briefly;**

Let the message space be M and the keyspace be K. A ciphertext C is generated from a message M which is drawn from the message distribution pm over the message space M and it is encrypted under a key K from the key distribution pk over the key space K. The decryption of C under a wrong key, say K′, produces a fake but valid-looking message M′ from the message distribution pm. Therefore, an adversary with unlimited time cannot distinguish M′ from the original message M.

**Generic Model**: For an encryption C=enc(M, K) of message M. If K and M are drawn from a known distribution. The target of an adversary is to recover the message M. He tries to decode C using different keys. For every key he tries, he gets M1,..., Mn. For a minimum entropy distribution like passwords, M is guaranteed to appear on his list. This is possible because users choose simple passwords that can be easily guessed. Also, attackers are aware of how users choose their passwords (from previously released details of leaked passwords on the internet). Therefore, the security here depends on the probability of the adversary been able to pick the message M from all n possible messages should one of the keys he tried was correct.

In the event that an adversary correctly guesses the key, he is still stuck with a spoof data and cannot ascertain which is the correct message especially when he has no idea of the target message. He wins only if he can determine the message from the list of messages he recovered during his attack. Figure 3 shows a detailed explanation of how honey encryption scheme responds to a brute force attack.



Figure 3: An illustration of the setting of the Honey encryption scheme in a brute-force attack.

However, in the honey encryption setting, he cannot tell if he has recovered the plaintext because the distribution is uniform.

**Distribution Transforming Encoder**: A DTE is a pair of algorithm DTE= (encode, decode) that takes M as an input and returns a value in S as output. Decode takes as input a value S and returns an output message M. Honey encryption involves a DTE-and-then-encrypt process. This means a sender applies the DTE to the original message he intends encoding and then uses any conventional encryption scheme as the second layer of encryption. The scheme works with a cryptographic primitive referred to as the distribution transforming encoder (DTE). The DTE models all possible message relative to the original message and maps them to a seed space such that any key supplied when decrypting a message produces a relative, but fake message from the original message and this makes it difficult for the adversary to determine if he has recovered the original message or not. The DTE represents the model of the message. A good DTE is designed to model the message distribution well such that if a seed is selected uniformly at random and applied to it, the message is recovered. Figure 4 gives a description of the DTE.

****

Figure 4: Framework of DTE for Honey Encoding

The algorithm of the encode and decode process of HE is as follows:

Honey Encryption Algorithm [12]:

HEnc (K, M)

S ¬$ encode(M)

 R ¬$ {0, 1}n

S’ ¬ H (R, K)

C ¬ S’⊕ S

return (R, C)

Figure 5 presents a description of DTE decoding.



Figure 5: Framework of DTE for Honey Decoding

Honey Decryption Algorithm [12]:

HDec (K, (R, C))

 S’¬ H (R, K)

S ¬ C ⊕ S’

M ¬ decode(S)

 return M

H represents the cryptographic hash function, K represents a key, M represents the message, S represents the seed, R represents a random string, C represents the ciphertext, and ¬$ represents uniform random assignment.

2.2 Application and Example of Honey Encryption to a Real-life Setting

Figure 6 shows an HE scheme setting where ‘Álice’ encoded a message to her friend ‘Bob’ telling him where she is spending her holiday and also a description of what an adversary ‘Eve’ recovers when she tries to decrypt the message using invalid keys.

In this example, Alice Honey encrypts the country where she is spending her holiday presently and send to her friend Bob following this steps (Refer to figure 6):

1. She constructs a message space using the DTE. The message space M consists of different countries, M = {CHILE, CANADA, MALAYSIA, SPAIN, ENGLAND, DENMARK, MONGOLIA, FINLAND}.
2. She finds the seed corresponding to the key/password (The key is shared previously between Alice and Bob). Suppose the key/password they agreed on was ‘456,’ then the seed value is ‘011’ (seed can also be ‘100’ or ‘101’ as they map to the key)
3. She wants to tell Bob that she is in ‘MALAYSIA.’ Using the DTE, she finds the seed corresponding to ‘MALAYSIA’ to be the seed value ‘010’.
4. After retrieving the seed value for both the message and the password, a conventional encryption scheme is used to encrypt the message. In this case, the One-time Pad is used to XOR the seed and the message to get the ciphertext.

$$011⊕010=001$$

Therefore, the ciphertext that Bob gets from Alice is 001.

Bob, on the other hand, decrypts the ciphertext following the reverse process to obtain the location of Alice:

1. He uses the DTE to find the seed value for the password ‘456’ to retrieve the seed value ‘011’.
2. He XORs the seed value ‘011’ with the ciphertext ‘001’ to recover the original seed as ‘010’.

$$011⊕001=010$$

1. He maps ‘010’ to the message distribution to recover the specific country Alice is spending her holiday; ‘MALAYSIA.’

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |
| --- |
| Chile |
| Canada |
| Malaysia |
| Spain |
| England |
| Denmark |
| Mongolia |
| Finland |

  123 456 789

|  |
| --- |
| 000  |
|  001 |
| 010 |
| 011 |
| 100 |
| 101 |
| 110 |
| 111 |

|  |  |
| --- | --- |
|  Key |  Seed Space Message Space |

  |

*Figure 6: A Sample DTE showing the encryption of the secret location of Alice*

The adversary, Eve may be eavesdropping through an open channel where she tries to use her tools to brute-force/guess the key of the message transmitted. Assuming she was able to get the ciphertext, she tries using different keys to get the plaintext. Any key used will produce a country in the message space. For instance,

1. if she tries to decrypt the ciphertext 001 using another password ‘123’. She will find the seed value for the password ‘123’ to retrieve ‘000’.
2. XORing the seed value ‘000’ with the ciphertext ‘001’ produces a seed value of ‘001’.
3. Mapping ‘001’ to the distribution, produces ‘CANADA.’

If she tries the ciphertext 001 with key ‘789’ which maps to the seed value ‘110’ and XORs the two seeds, she gets ‘111’ which maps to ‘FINLAND.’ For every incorrect key she uses, she gets a valid looking message, and even when her keys are correct, she cannot tell exactly which of the countries Alice is spending her holiday. If the message space is tailored well and made as large as possible, the attacker’s job will be made more difficult.

2.3 Criteria for a Good Encoder

In order to create a plausible/convincing decoy that does not expose the original message to the adversary, it is fundamental to keep in mind two criteria:

1. Indistinguishability: Decoy messages must be difficult to distinguish from real messages. Successful deception of an adversary lies on the honey (decoy) message. Automated tools and humans should not be able to tell decoy message from true message. Decoys must be drawn from a probability distribution over possible messages similar to the distribution of the plaintext. Modeling human language requires honey messages that appear as they are used in real world. A good DTE must produce convincing decoy messages. The probability of telling decoy from true message must be minimal to successfully deceive the attacker.
2. Confidentiality: A good DTE must model the human language and at the same time hide the structural information of the original text. Human language such as e-mails, password vaults requires a considerable context-and-content relevant information. An encoder that does not reveal the structural skeleton of the true text provides better security.
3. **TOOLS FOR BUILDING ENCODERS**

Several statistical tools employed for constructing the encoders to model natural language are discussed in this section.

**3.1 Probabilistic Language Models**

A language model is the probability distribution over sequences of words. It is used to assign a probability to a sentence. Given a series of words K1, K2, K3.......Kt. It assigns a likelihood

$P\left(K\right)=P(K1,K2,K3,…,Kt)$ (1)

For instance: P(it rained heavily last night) = P(it, rained, heavily, last, night)

 The language model is used to predict the next word or next sentence. Using K1, K2, K3, K4, to determine the probability that the next word will be:

$P\left(K5\K1,K2,K3,K4\right)related to P(K1,K2,K3,K4,K5)$ (2)

An n-gram is a close-sequence of n items from a given sequence. A Unigram is an n-gram of size 1, a Bigram is an n-gram of size 2, and a Trigram is an n-gram of size 3. In an n-gram model, the probability P(K1, K2, K3,...,) of observing the sentence K1, K2, K3,..., Kt is represented as:

$$P(K1,…Kt)=\prod\_{i=1}^{t}P\left(K1,…,Ki-1\right)≈\prod\_{i=1}^{t}P\left(Ki-\left(n-1\right),…,Ki-1\right)$$

 (3)

It is assumed that the probability of observing the ith word Ki in the context history of the preceding i − 1 words is approximated by the probability of observing it in the shortened context history of the preceding n − 1 words.

The conditional probability calculated from n-gram model frequency counts:

$P\left(Ki-\left(n-1\right),…,Ki-1\right)=\frac{count\left(Ki-\left(n-1\right),…,Ki-1,Ki\right)}{count\left(Ki-\left(n-1\right),…,Ki-1\right)}$ (4)

**3.2 Markov Models**

A [stochastic](http://whatis.techtarget.com/definition/stochastic) model used to represent randomly changing systems where future states depend on current states and do not depend on past events that occurred before it. Predictable events that occur over time are fashioned using Markov models. Markov model is the Markov chain which models the state of a system with a stochastic variable that changes over time. Figure 7 presents an example of a simple Markov chain with two states.

 *Figure 7: A Two-State Markov Chain*

With two states (1 and 2) in the state space, there are four possible transitions. State '1' could transition to '2' or stay at '1'. State '2' could transition to '1' or remain at '2'. The probability of transitioning from any state to any other state is 0.5.

**4. PROPOSED METHODS TO IMPROVE THE HONEY ENCRYPTION SCHEME**

This section describes proposals from several researchers in the past. The proposal section is divided into three groups. The first part is focused on the encoder of the DTE. The second part describes other proposals for message recovery, genetic application, real-world applications and the third part describes proposals on typo problems in HE.

**4.1 A New Countermeasure against Brute-Force Attacks that Use High-Performance Computers for Big Data Analysis [30]**

Jo et al. [30] proposed a statistical code system where the standard HE scheme was unified with a structural code system. The argument was that HE does not reflect human language as it uses a fixed distribution DTE. HE has a semantic security feature but has no support for Natural Language Processing (NLP) which is needed before it can be extended to allow encoding human-generated text. The proposition was to look for a system that supports NLP and integrate it with the standard DTE of Honey encryption. Therefore in this proposal, the structural code scheme was unified with the HE scheme to form a statistical code scheme.

 The structural code scheme is built using natural language processing tools; a probabilistic language model and Markov process. The process is as follows:

1. The sender takes his plaintext (usually a text-data) and trains it to get the frequency of each character from the plaintext to create a probabilistic language model. This step is used to reveal the features of the text data.
2. Then generate a corpus.
3. The final step is to encode or decode the plaintext using the corpus. The sender selects a character from the plaintext and then searches for an equivalent bit string in the corpus. From this design, a character having a high probability will be the first letter, and the conditional probability will create meaningful words as decoys. Therefore, a false plaintext is generated making it difficult for the adversary to distinguish the original plaintext from the false-text if he eventually tries the correct key.

4.2 Cracking-Resistant Password Vaults using Natural Language Encoders [31]

In a more advanced research in 2015, Chatterjee et al., [31] proposed a Natural Language Encoder (NLE) which they called NoCrack. The proposal was explicitly for protecting password vaults (usually referred to as Password Managers). A password vault is used for [generating](https://en.wikipedia.org/wiki/Password_generator), storing and retrieving complex [passwords](https://en.wikipedia.org/wiki/Password) stored in an [encrypted](https://en.wikipedia.org/wiki/Encryption) database or calculating them on demand. Password managers were introduced for two reasons; the need to relieve users from having to memorize many complex passwords of their accounts online and also to let the software generate complex passwords for all their online websites. The user only needs to memorize one secure master password to access all other passwords/accounts.

This proposal describes a scenario whereby an adversary was able to lay his hands on an encrypted vault due to either a vulnerability, lost or stolen vault. The adversary does not know the passwords encrypted inside the vaults but is aware of the distribution of human-generated passwords based on leaked passwords that are publicly displayed online. He also has some knowledge of the encryption scheme used to secure the vault. The target is to get the original master password which if he successfully gets (using brute-force or hacking tools) gives him access to all other passwords secured in the vault. The proposal maintained that the best countermeasure in such scenario is to generate a false (plausible-looking) password vault for each master password as the adversary tries his guesses. This false-vault forces the adversary to go online.

In this proposal, a tool called the NoCrack was built from existing natural language models using tools like N-gram Markov models and probabilistic context-free grammars of language models. The objective of their research was to generate a realistic but fake vault on the fly during brute-force attacks.

The general process to build a simple DTE in this proposal,

The NLE takes a users password as input

Encodes it to a random bit string

Decoding any random bit of the string produces a realistic but fake vault

This proposal requires a considerable amount of time decrypting the vault and re-encoding the password. The larger the vault size, the more time is required to recover a vault.

4.3 On the Security of Cracking-Resistant Password Vaults [32]

**Golla et al. [32] in 2016 highlighted the major drawback of the NLE proposed by Chatterjee et al.**

They pointed out that the model fails to give security as statistical tools can be used to reveal the differences between real and false vaults. According to the findings by Golla et al., the framework of Chatterjee [31] is structured on a static NLE and as such the distribution of the false vault is fixed making it possible to distinguish the false vault from the real vault with high probability. The adaptive NLEs was proposed as an improvement.

 An optimization of the NLE is the adaptive NLEs which is constructed using Markov Models. The n-gram Model was modified not to give a very low probability to the passwords that appear in the vault. This is required so that the distribution of the decoy vault depend on the passwords stored in the vault. Modifying the NLEs hides the structural differences between passwords and the correlations. This makes it difficult to filter decoy from real passwords.

 The process to build the adaptive NLE is as follows:

1. Each password that appears in the vault is assigned one n-gram at random and multiplied by 5 to increase the probability.
2. A factor of 5 increases the remaining n-gram with a probability of 20%.
3. All probabilities are re-normalized.

**4.4 Honey Encryption for Language Robbing Shannon to Pay Turing? [33]**

In their cutting-edge paper of 2016, Beunardeau et al.[33], demonstrated why the previous schemes proposed by [30-32] could only work in the context of password security but fail to accommodate reasonable sized, human-written documents like emails. To our knowledge, this research is the first to explore this open problem as previous models were built to work only for passwords. The approach used by previous authors fail to model human language to produce convincing decoys as used in real life settings.

The corpus quotation DTE is proposed where users are restricted to quote only from a known public document. This approach produces realistic sentences and does not reveal the structural information as the words used to encode the message is realized inside the public document. Therefore, the document takes the structure of the human language from the specified document.

The approach is to use the grammar of a language to simplify its structure and using a re-writing rule to shuffle the message. This approach uses an existing book with text and intervals to encode a combination of words. This proposal describes a list of grammatical roles where words are tagged using the clause level, phrase level, and word level labels. Table 1 highlights all the descriptions, pros and cons of proposals suggested by various authors.

Table 1: Description of models given by different authors.

|  |  |  |  |
| --- | --- | --- | --- |
| **Authors** | **Description** | **Justification** | **Flaws** |
| **Jo et al. [31]** | * The authors proposed the statistical code scheme which is a unification of the structural code scheme and honey encryption scheme. In this proposal, the HE provides the semantic feature of language while the structural code scheme provides the syntactic features of natural language.
* The performance was evaluated to find how many times it was required to generate meaningful false text from the original text in the corpus. The probability was 0.38 which means at least four repetitions are required to generate convincing decoy message.
 | * This approach provides plausible false text relative to the original text as decoys.
 | * It does not support other data-format but generates decoys only for short length message.
* The ambiguity between the original plaintext and the false text is much, and the adversary may use this to figure out the difference between the false text and original text when he has a little knowledge of the target message.
* The proposal works for only short messages and cannot be used to encode long-length human texts.
 |
| **Chatterjee et al. [31]** | * This proposal describes how to build a Natural Language Encoder (NLE) called NoCrack using existing password models.
* The proposal also refuted the claim of a previous proposal called the Kamouflage and proved that it leaks structural information; thereby degrading security.
* The performance was evaluated by measuring the time required to recover a particular vault and the time to add a password to a vault.
* The analysis showed that, the smaller the vault, the faster the recovery. Large vaults require a long time to generate a decoy vault.
 | * This proposal performs best when the vault is small.
* A single password can be recovered quickly.
* This approach creates realistic decoy vaults on the fly during brute-force attacks.
* This proposal serves as a basis for constructing better decoy vaults using constructive language models.
 | * This approach does not support large vault as it is very slow.
* The system reflects human language but was borne in the context of password security and does not support human-written documents like emails or long messages.
* Generating vaults on the fly impacts time and also require large space, thereby increasing computational cost.
* This method suffers from a plethora of pitfalls as proven by [9].
 |
| **Golla et al. [32]** | * This proposal used Kullback–Leibler (KL) divergence to prove that the approach by [8] degrades security.
* This proposal pointed out how to distinguish decoy vault from the original vault using the probability distribution with a high accuracy.
* The proposal also pointed out that all fixed NLE is susceptible to the KL divergence attack and proposed the adaptive NLE.
* The adaptive NLE was constructed using Markov model. It was evaluated using KL divergence attack, and the analysis showed that the real vault ranks among 40.12% of the most likely vault.
 | * This proposal improves previous methods of securing the vaults for password security.
* This proposal defines an easy way of building the adaptive NLE.
* The adaptive NLE increases the message space, allowing more instances of online guessing of the original vault.
* It serves as a basis to build better NLEs for better decoys.
 | * One of the major drawbacks of adopting the fixed or static NLE is intersection attack.
* There is no adequate closure of providing maximal security.
 |
| **Beunardeau et al. [33]** | * This proposal contends that the proposed method by [7-9] works reasonably well to secure short passwords but fail to model natural language as used in real-world scenarios such as e-mails and written documents.
* This proposal explains that context-relevant information is required to model human language. Also, convincing decoy messages that fool human and automated tools are required to model human language.
* The corpus Quotation DTE is proposed.. Grammar model of language is used to build the DTE and users are required to only quote from a known public document.
 | * This proposal suggests how to extend the scheme to allow encoding human-written texts.
 | * It uses a quotation DTE.
* Quoting from a public document restrict users to the vocabulary of the document domain.
* Fixed codebook is not able to provide all the required combinations of words. For instance, it is unlikely that a user will be able to encode a text from a computer science domain using a code-book from a flower-domain.
 |

4.5 Genoguard: Protecting genomic data against brute-force attacks [34]

 In a major advance in 2015, Huang et al. [34] extended the Honey encryption scheme to support encoding of genomic data. There is an increased use of genomic data in a diverse sphere from healthcare to research, legal cases, forensics and so on. With such great use comes the problem of securing the genomic data. A tool called GenoGuard was proposed to secure genomic data. The tool is designed to yield a plausible but fake genome sequence should an intruder try to decrypt the key/passwords used to secure the genome. The tool provides an information-theoretic security guarantee to protect the message from retrieval during message-recovery attacks. An adversary that knows the physical trait (phenotype) of a patient cannot recover the patients' phenotype as he will continue to get a corresponding realistic-looking but fake phenotype for every key he supplies during his attack. This way, he gains no information.

 The structure of genomic data is complicated, and as such, the approach used was to combine the HE scheme with the features of genomic data to develop secure genomic data storage and retrieval. The contributions of this proposal are as follows:

1. a technique to provide security for genomic data against data breaches given a computationally unbounded attack.
2. Schemes to mitigate mauling of genomic sequences and to prevent an intruder from exploiting partial information of the patients in question is provided.

This proposal also suggested ways to deal with typos in honey encryption.

 Typos in HE scheme is very serious if not handled properly. If a legitimate user for instance- the doctor or patient provides a wrong password, he gets a valid-looking genome sequence, and this can be very dangerous if the doctor/patient did not realize he made the typo. The user may take the sequence to be correct, and medical decisions may be made based on that. This proposal provided two ways of dealing with typos in securing genomic data.

1. A personal information known only to the patient such as a 4-digit Pin can be appended to the plaintext and encrypted together.
2. A recall memory or a recognition memory can be used by the user to identify a typo. In the recognition memory, the system can generate several numbers of confirmation images. The user then chooses one from this images. During retrieval, a user confirms he correctly decrypted his genome by indicating that the displayed image is familiar. Biometrics like fingerprints can also be used and concealed among the decoys, and only the legitimate owner can decode his/her message.

**4.6 Honey encryption beyond message recovery security [35]**

Jaeger et al. [35], in their recent review, pointed out that the standard HE scheme proposed by Juels and Ristenpart [12-14] provided message recovery security in settings where keys are of minimum entropy. However, message recovery property in this setting is weak as an attacker may learn partial information about the plaintext and even attack the ciphertext during a brute-force attack.

This proposal studied the HE scheme from the min-entropy settings and maintained that the HE proposed by [12-14] could not be prevented from a known plaintext attack. This is because as the attacker tries all possible keys, he gets plaintexts from the message space that seems relevant to the plaintext. He may use this similarity to determine the plaintext, and it also gives him an idea of the target message. In a setting where the attacker can exhaust all the keyspace, a known-message attack is inevitable.

The HE scheme can be prevented from unknown message attack. A simple proof of this is to pick an encrypted message from the message space as a challenge message, and if the adversary can determine what the plaintext is from the challenge message, then he wins. This proposal strengthened the HE scheme by building simple models like encode-then-encrypt and encode-then-encipher to achieve target distribution semantic-security and target distribution non-malleability as a countermeasure for message recovery.

 This proposal concludes that there is no way to secure the HE scheme from known-message attacks, but unknown-message can be secured from attack.

**4.7 Honey chatting: A novel instant messaging system robust to eavesdropping over communication [36]**

In [36], the authors proposed a chatting application using an HE-based statistical scheme. This proposal unified a statistical scheme with the HE scheme to generate plausible but fake chat messages for every key supplied by an adversary eavesdropping during communication. This proposal involves a three-way process:

1. The N-gram language model is used to represent the chat messages.
2. The message is encrypted using the conventional password-based encryption.
3. An encoder and decoder share the same code table for their messages.

The performance of the scheme was evaluated using sample movie subtitles. The system generated fake messages from the scenes as dialogue to fool the attacker.

**4.8 Visual Honey Encryption: Application to Steganography [37]**

Yoon et al.[37], proposed a visual HE which employs an adaptive DTE in a Bayesian framework. The idea behind this is to let HE accommodate complex distributions likes images and videos. The process of the Visual Honey encryption (VHE) is as follows:

1. A plain image and a public image is selected. The format and size of the plain image is made to be the same as that of the fake image. The plain image is secret while the public image is made public. A sender and receiver share the public image. The adversary is believed to know the public image. The public image is shared between the sender and the receiver since they have an identical DTE.
2. A codebook is constructed using statistical formula.
3. Encoding and decoding between the transmission parties is done using the codebook.
4. Two keys K1 and K2 are used for encryption and decryption respectively. Encryption is done using any of the conventional encryption schemes. If K1=K2, the encoded message is correctly decrypted and decoded to the hidden plain image via the Bayesian DTE. If K1≠K2, then the encoded cipher will be decrypted to a random sequence different from the actual bit streams and then fake images are generated.
5. The encrypted data is transmitted.
	1. **Honey Encryption Applications [38]**

Tyagi et al. [38] showed the implementation of HE on RSA and basic text messaging. This proposal showed how to implement the standard HE scheme and how they are applied in real-world settings. The proposal also demonstrated how to use honey encryption to support Public-key encryption.

4.10 Typo-Problems in Honey encryption

Password refers to a secret phrase or word(s) used for user [authentication](https://en.wikipedia.org/wiki/Authentication) to prove identity or that allows access to a system or resource. It is designed to protect accounts or resources from unauthorized access. In the HE scheme, a user is supplied with a seemingly real but false text for every key he supplies. A significant drawback with the HE scheme is typo safety when a legitimate user mistakenly enters an incorrect key. Typo error is a serious problem in HE if not adequately provided for. This section provides a review of typos.

4.11 Password Typos and How to Correct Them Securely [39]

Preliminary work on typo safety was carried out in 2016 by Chatterjee et al. [39]. This proposal pointed out that there have been lots of research on usability of passwords, but a neglected area is password typos. It was observed that 3% of users were not able to login to their accounts based on easily correctable typos. Simple typos- for instance- if a user tries to log in to his account using the first character of the password as an uppercase letter instead of a lower case letter, then he will be denied access to the system. A study of users authentication at Dropbox showed that typos were a huge problem.

 Chatterjee et al. [39] argued that building a server that corrects simple typos without degrading security will improve usability and also save time. The approach was to investigate:

1. how to build typo-tolerant password checking system
2. how much would tolerating the typos help users
3. if the typo-tolerant system degrades security?

In a standard authentication, a user is granted access to the server when he submits correct credentials to the server. However, he is denied access to the server if the credentials are incorrect. This incorrect credentials maybe due to a first character flip in the password; for instance an uppercase character instead of a lowercase character. This proposal suggests a relaxed checker as a countermeasure to such problem. It works like the standard authentication scheme but has the added feature to check a failed login based on the setting. The relaxed checkers apply one or more correctors on the password entered to check if the user supplied a lower case character instead of an upper case character in any position of the string. The server is designed to decide what type of typos it wants to correct, and the existing password-based authentication system was used as a framework to build them on. This approach is easier to build as it also improves usability. However, the hash function used on the existing system is slow and computationally expensive and as such too many corrections cannot be made.

The proposal suggested an improvement on the relaxed checkers. Therefore, some 4,300 employees of Amazon Mechanical Turk (MTurk) was tasked with the transcription of over 100,000 passwords drawn from the RockYou password leak. The study was performed to find a small set of correctors that can correct a large collection of typos. Analysis of the password showed that 11% typos were committed on Capslock, 4.6% is committed based on adding character at the end of the password, 4.5% was on flipping first letter case and others. As this top three-factor comprises of all the top typo problems that existed, it was used as the basis for creating the typo-tolerant scheme.

 The impact of this factors in a real-life scenario using the Dropbox showed that in a 24-hour interval:

1. 3% of Dropbox users were denied login into the server because of one of the three factors
2. 20% of the users would have saved at least a minute of logging into Dropbox if these 3 typos are corrected.

A security-sensitive typo correction scheme is presented as an improvement over the existing authentication scheme. It was concluded that allowing typo correction in the passwords will add several months of login time for a person.

4.12 Password Typos Resilience in Honey Encryption [40]

In a recent attempt, Choi et al. [40] in 2017 proposed two protocols that alert a legitimate user of typo errors in their passwords. This protocol was designed specifically for the HE scheme. The protocols include an A-Type and B-Type scheme. Both schemes were constructed differently with a system and threat model.

In the A-Type scheme, the system model has a user and a server entity. It describes a standard setting where a user sends his message to the server and sets a password. The message is encoded under the password using the HE scheme and the ciphertext stored. When the user wants to recover his data after successful login, the server sends the ciphertext with some side information to check the password integrity.

A key problem with this scheme is that an online brute-force attack can be used to exploit the system. An intruder who successfully logs in to the system may exploit the system and gain knowledge of the ciphertext based on the side/partial information he is supplied with during the authentication.

 The A-Type involves a three-phase process:

1. An enrolment phase; a user sends his message m with the password p. The server encrypts the message under the password using the HE scheme and generates a ciphertext Cm (H Enc m,p).
2. A retrieval phase: the user types in his password p’ and sends it as a request to the server. The server checks if the password for the decryption is the one created by the user in (i) using a Check-Typos function defined by the authors. Given a password p', Check-Typos function determines the type of p.' This supplies two types of information; the original password and typos; denoted by α. If p’=p as entered in (i), it returns the original password, otherwise, it returns a typo. The server then decodes the ciphertext Cm under the password p’ and generates the original message m’= HDec Cm, p’ and encrypts α by exploiting HE scheme under the original password p which yields the ciphertext Cα = HEnc α, p. Then the server sends m′ and Cα to the user.
3. A verification phase: the user types in the password p” to decode Cα. He then generates α HDEc Cα, p’’. Should the user gets the original password from the typo information then he has the original message m’. If he gets another password, then m’ is not the original message and he has to retrieve again following the process.

 In the B-Type scheme, the system has a user, a server and database manager entity. The user uploads their message, password, and partial information to the server. A 4-digit Pin (Personal Identification Number) is used to represent the partial information. The server encodes the message and sends to the database manager.

The threat model in the B-Type scheme can be a user that has access to the database. The database manager may be a threat, and he may try to recover the message by trying every possible key.

 The B-Type was constructed using the same three phases but different approach

1. An enrolment phase: the user sends his message m, a password p and a pin (side information) to the server. The server encodes the pin and the message under the password p using the HE scheme and generates a cipher Cm. H Enc m, p and Cpin HEnc pin, p. The ciphertext is sent and stored on the server.
2. Retrieval phase: a user that successfully logs in gets an encrypted data as a response to the user's request.
3. Verification Phase: The user supplies the password p’ to decrypt Cm and Cpin and generates m’ HDec Cm, p’ and pin’ HDec Cpin, p’ respectively. If the pin’ is the same as the one he supplied while enrolling, then he verifies the message m.' A pin which is different from the original one Pin’ means that there was a typo when entering the password.

4.13 The TypTop System: Personalized Typo-tolerant Password Checking [41]

Recent work on typo tolerant system was carried out in 2017 by Chatterjee et al. [41]. This proposal introduces a personalized password-tolerant system called TypTop. This proposal pointed out that usability of passwords can be improved with a slim security loss by permitting a small number of typographical errors. According to them, it was observed that existing typo-tolerant system only corrects typo errors such as first letter capitalization or if the Capslock key is on which does not offer maximal usability. The aim of the research is to improve the usability of the existing system by storing failed logins and training the system to learn the typo-errors committed by specific users. The proposal gives a basic criterion that, if met, guarantees no security loss in offline and online attacks. Therefore, the system was modeled to work under real-life setting.

Simulations were conducted with data gathered from the study on Amazon MTurk to validate if it secures real-life password applications and if it performs better than current typo-tolerant password checking systems.

A cryptographic security concept was proposed showing a reduction of brute-force attacks against the registered password or the typos entered into the typo cache.

It was concluded that the TypTop system offers better usability compared to the current typo-tolerant scheme as substitution errors, transposition errors and capitalization errors in other parts of the password is handled effectively by the proposed system. Table 2 briefly summarizes the typo schemes.

Table 2. Description of Typo-Safety Protocols Proposed in the Past

|  |  |  |
| --- | --- | --- |
| **Authors** | **Description** | **Positives and Negatives** |
| **Chatterjee et al. [39]** | * 3% of users failed to login based on a Capslock problem (where they either login with a lower case letter instead of an uppercase letter), first case letter flip and also applying an extra character at the end of their password.
* This proposal pointed out that at least a minute would have been saved for 20% of the users if the typo-tolerant scheme is adopted.
* This proposal presents a typo-tolerant checker which works relatively well with the existing password authentication system.
 | This approach is suitable for existing password-based authentication system as it applies caps lock corrector, first case flip corrector and also extra character at the end corrector to improve usability but this approach is not suitable to handle the typo problem in the HE scheme. This research had a quick positive impact as Dropbox introduced a Caplock’s indicator on their login page to reduce the capslock error but as mentioned by Choi et al.,[29] this approach does not allow a broader spec of typos to be detected and does not handle other types of typo errors. |
| **Choi et al. [40]** | * This proposal presents the Type A and Type B protocols.
* The scheme provided two types of typo-safety both in an offline and online setting to handle different typo problems while still retaining message recovery in a typical HE scheme.
 | The Type A protocol is easier to implement as it requires only a server but the major drawback of adopting the type A is that the size of the key is small and also there is the uncertainty of detecting typos in some settings.Type B is an improvement over Type A as a user can easily notice typos if he remembers his pin, however, a key problem here is that user has to remember the pin to verify his message. |
| **Chatterjee et al. [41]** | * This proposal presents a personalized typo tolerant password checking.
* The system is designed to store failed log ins and learns the typo-errors committed by specific users
* This research proposes a simple blacklisting procedure in which a small set of risky typos is prohibited from being admissible into the typo cache.
 | This research is an improvement over existing typo-tolerant password schemes but is not designed to work on typos committed on a decoy system. However, it can be modified for Honey encryption. |

**5. COMPARATIVE ANALYSIS**

The Table 3 shows various approaches and also a summary of each proposal. It also shows if their output produces fake plaintext that meets the indistinguishability and confidentiality criteria required for producing convincing decoys.

Table 3. Proposals of HE Techniques

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Authors** | **Approach** | **Indistinguishability** | **Confidentiality** | **Summary** |
| **Jo et al. [30]** | Statistical code System: Unification of the HE and structural code system. | Yes | Yes | This approach supports only short length message. |
| **Chatterjee et al. [31]** | NoCrack: Natural Language Encoder (NLE) | Yes | Yes | This scheme supports only passwords.  |
| **Golla et al. [32]** | Adaptive NLEs | Yes | Yes | This approach improves [31] and it is proposed in the context of passwords. |
| **Beunardeau et al. [33]** | Corpus Quotation DTE | Yes | Yes | Supports long length message. However, users are restricted to code only from a public document. |
| **Huang et al. [34]** | HE schemes to secure genomic data | Yes | Yes | This scheme was proposed for the security of genomic data during transmission. |
| **Jaeger et al. [35]** | Target distribution semantic-security and target distribution non-malleability as a countermeasure for message recovery | Not Applicable | Not Applicable  | Formal proofs to show how fake plaintext can be hidden from revealing some structure of the original message. |
| **Kim et al. [36]** | Honey Chatting as a countermeasure to eavesdropping. | Yes | Yes | This application is designed to provide fake chat message to an adversary that is eavesdropping on two parties while they are chatting. |
| **Yoon et al.[37]** | Visual Honey encryption that uses a Bayesian DTE framework | Yes | Yes | This application is designed to provide fake images to an adversary that tries to get hold of the original image during transmission. |
| **Tyagi et al. [38]** | Implementation of the HE scheme in RSA keys and short messages | Yes | Yes | This proposal demonstrated the implementation the HE scheme on RSA keys and short messages. |
| **Chatterjee et al. [39]** | Password Typos | Not Applicable | Not Applicable | This proposal describes how typo errors can be resolved.  |
| **Choi et al. [40]** | Proposed the A-Type and B-Type protocols for typo problems in HE | Not Applicable | Not Applicable | This proposal suggested two protocols that can be used to counter typo problems in HE scheme. |
| **Chatterjee et al. [41]** | Password Typos | Not Applicable | Not Applicable | This proposition presents a customized password-tolerant framework called TypTop. This proposal was built from the concept that allowing a small number of typographical errors can improve usability. Experimental analysis of a real-life system using Mechanical Turk shows that 45% of users would benefit from the personalized system. |

6. DISCUSSIONS

From our findings, all methods proposed by [30-32] worked relatively well for passwords but failed to accommodate its extension for the support of human-generated messages as effective decoys. Each approach was an improvement over the previous approach. To our knowledge, the only research that tried to address the open problem was by Bernadeau et al.[33]. Each approach presented by [30-32] cracked the previous proposal, and so we shall discuss the state-of-the-art approach proposed by Bernadeau et al.[33]. This proposal contended and proved why the methods of [30-32] failed to support the human generated message. According to them; the approaches used by [30-32] failed to model even simple messages – let alone entire documents. For more details on their work, see [33]. They proposed the Corpus Quotation Distribution Transforming Encoder. However, our findings with the research are highlighted below;

1. The user is restricted to quote only from a known-public document; As pointed out in Table 3, this poses a severe problem as it is unlikely that a user will be able to encode a text from a computer science domain using a code-book from a grocery-related domain. Also, subjecting users to quote only from a particular source document does not model human language at all [33], and a single vulnerability where the language distribution of the document is known will compromise the whole setting.
2. Their method, when used in some context, produces weird sentences as honey/decoy messages. For instance the sentence; “During his youth, Peter was tutored by a skilled architect until the age of 16”. In their scheme, architect was only truly defined as a noun, so a decoy version might result in “Peter being tutored by a Baboon,” since Baboon is a noun. This is obviously wrong and the attacker will easily discard this decoy. Our suggestion here is to define architect as a “person Noun” and provide a wordlist of a similar word. An idea for autogenerating this label is using a large database of text (Like the download of Wikipedia). Then create a vector of values that describes the word. For instance, the number of times used as a subject when the verb is an action verb, number of times acted upon (like stapler in “Peter used the stapler”). The vector should be launched into a machine learning construct for sorting data based on similarities to generate the wordlists.
3. The original idea of the HE scheme is to act as an additional layer to another known scheme. The basis of their scheme was that having the correct key will reproduce the original message. This fails to give security in some instances. For instance- if AES is used with their corpus method proposed (in this case, intervals represents the position in the codebook), the model impacts the text and fail to generate valid-looking decoys/fake text. For instance, if we encode the interval position with 8-bit numbers we will allow only 256 symbols of the text to be used for encoding; in 16 bit that would be 65636 symbols and in average 10000. If the attacker tries to decode the message interval position with a large number of small text, he will be able to discard the key right away. When testing, some keys will be canceled out. In this case, it will be possible to make an option to generate results produced by all keys combinations of a specific size. To handle this problem, we suggest an interface is built for the user of the application where the interval parameters can be checked and changed and the user can make decisions based on this.

They also made some suggestions regarding improving their work:

1) Automated Plaintext Pre-Processing - This option is required to search for the phrases in the codebook. In a setting where it is impossible to find a specific entry in the phrase, the text application should try to simplify the phrase while preserving the meaning so that the phrase can be encoded as a whole instead of encoding by a combination of multiple words. For this option, we have two suggestions;

a) remove or ignore help words like (a the,...) and punctuation during the search;

b) during the search, additional test words for synonyms and word forms

c) automatic summarization: There are some drawbacks to this approach. First, it will affect search time; with a more significant number of combinations to test, it will take more time. Some procedures like synonym testing will depend on the dictionaries sometimes generated automatically and will not contain a full set of possible synonyms or can contain synonyms which have slightly different meaning. Automatic summarization procedures can also lose some important details.

2) Adding Syntactic Defenses - Adding additional words and clauses to the application is possible, but there is a limitation to the content that can fool the attacker. For example, the sentence "It will rain" changing it to the negative sentence "It will not rain" will affect the content.

However, changing it as "it will likely rain" does not hide the content and so the adversary will still be able to understand the message. Some sentences will not allow any modification without syntactical structure failure. For example, introducing any words in the sentence "Attack at 6 p. m." will be visible. Therefore, modification is not applicable to all possible data.

3) Adding Phrase-Level Defenses - here the same comment is applicable. Not all the modification of the syntactic tree changes the results significantly to modify its content. This modification will only affect the results considerably in case it will later require generating an encryption key from it. In this case, people will usually use rather small messages which will not be much different from the passwords, so they might not have pervasive syntactic structure.

7. ISSUES AND CHALLENGES

As discussed in the introduction section, HE provides security beyond brute-force bound; in settings where minimum entropy keys are used to secure messages. However, there are open areas in HE which need to be researched. In this section, we briefly describe some of the problems of HE.

1. Honey encryption is difficult to apply in a setting where the plaintext is large or the distribution of the message is unknown. A large plaintext requires a substantial content to be used to construct the DTE so that fake-texts that looks like the original texts can be used as the decoy message. The contents of the decoy message also need to have a good contextual meaning relative to the original message. The authors acknowledged this difficulty in their paper when they pointed out that *“...The key challenges of honey encryption scheme are development of appropriate instances of a new type of randomized message encoding scheme called a distribution-transforming encoder (DTE)”* [11-13]. This problem was also strengthened when Juels [9] pointed out the complexity of encoding honey documents, for instance, e-mails require generating fake but semantically and contextually realistic natural language message.
2. Having a context-sensitive decoy produces good instances of the message but if not correctly constructed will reveal the structure of the original message giving the adversary an upper hand/high probability of recovering the message. A known-plaintext attack seems inevitable in a setting where the message is of significant length like email. It is difficult to hide the structure of the message from an adversary who has no idea of the target message.
3. HE is tailored to work in low-entropy settings like passwords, RSA keys, PINS and Credit card numbers. Extending it to support other settings and file type requires an extensive design of the DTE to meet the criteria discussed in the previous section.
4. HE is fashioned to produce fake but valid-looking text for every key supplied by anyone. Therefore, a legitimate user that made a typo error while trying to retrieve an encrypted message will recover a valid-looking but fake message and he has no way of knowing this.

8. CONCLUSIONS AND FUTURE CHALLENGES

In this paper, issues, challenges and detailed literature review of the Honey Encryption (HE) scheme is provided. The aim is to furnish current/aspiring researchers and practitioners with a comprehensive overview of the state-of-art research in the scheme. From the survey of various proposals, we conclude that the current techniques used in producing decoy message do not model human language entirely and so fail to produce decoys that are acceptable and convincing to lure the attacker away from the genuine resource.

We have investigated different HE schemes, noted the methods used to develop the decoys and proved their positive sides and flaws. We also surveyed protocols to protect against typos in the scheme, message recovery and the implementation of HE in RSA keys and Pins. Presently, honey encryption scheme has been implemented for credit card numbers, passwords and RSA pins. There is an urgent need for HE to be adapted for other settings such as decoys for human generated message such as e-mails, convincing decoys to confront eavesdropping attack during online chatting, etc.

This study has raised many questions in need of further investigation. Therefore, we propose further research in the following areas:

1. Natural Language Processing in Honey Encryption: How do we capture the empirical properties of language? How do we model the human language itself as an effective tool for designing convincing decoys?
2. How do we honey encrypt (produce decoys) without revealing the structure of the message?
3. How do we generate decoy/honey messages that fool machines and human from realizing real messages from decoy messages?
4. How do we handle mauling and prevent adversaries from learning partial information of the original message from the decoy during an exhaustive key-search?
5. How do we address typo problems in the H.E scheme? This problem requires immediate attention and extensive research even before any implementation of the HE scheme.

**Acknowledgement**

This work is partially supported by the Fundamental Research Grant Scheme (FRGS) for ‘‘Content-Based Analysis Framework for Better Email Forensic and Cyber Investigation’’ [203/PKOMP/6711426], Security and Forensic Research Group (SFRG) Lab, School of Computer Sciences, Universiti Sains Malaysia.

**REFERENCEs:**

1. Omolara AE, Jantan A, Abiodun OI, Poston HE. *A Novel Approach for the Adaptation of Honey Encryption to Support Natural Language Message.* In Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, 2018; 1 : 134-140.
2. Disso JP, Jones K, Bailey S. A plausible solution to scada security honeypot systems. In *Broadband and Wireless Computing, Communication and Applications* (BWCCA), 2013 Eighth International Conference on 2013 Oct 28 (pp. 443-448). IEEE.
3. Bringer ML, Chelmecki CA, Fujinoki H. A survey: Recent advances and future trends in honeypot research. *International Journal of Computer Network and Information Security.* 2012 Sep 1;4(10):63.
4. Dhanji PK, Singh SK. Assault Discovery and Localizing Adversary in Remote Networks. *Indonesian Journal of Electrical Engineering and Computer Science*. 2018 Jan 1;9(1): 81-84
5. Zhang D. Inconsistency: the good, the bad, and the ugly. In *Information Reuse & Integration*, 2009. IRI'09. IEEE International Conference on 2009 Aug 10 (pp. 182-187). IEEE.
6. Pawlick J, Colbert E, Zhu Q. A Game-Theoretic Taxonomy and Survey of Defensive Deception for Cybersecurity and Privacy. *arXiv* preprint arXiv:1712.05441. 2017 Dec 14.
7. S. Tzu. "The art of war." In *Strategic Studies*, pp. 63-91. Routledge, 2008.
8. Montagu, E. The Man Who Never Was, J. B. Lippincott Company, Philadelphia, PA (1954).
9. Juels A. 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* 2014 Jun 25 (pp. 1-4). ACM.
10. Juels A, Rivest RL*. Honeywords: Making password-cracking detectable*. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security 2013 Nov 4 (pp. 145-160). ACM.
11. Juels A, Ristenpart T. Honey Encryption: Encryption beyond the brute-force barrier. *IEEE Security & Privacy*. 2014 Jul;12(4):59-62.
12. Juels A, Ristenpart T. Honey encryption: Security beyond the brute-force bound. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* 2014 May 11 (pp. 293-310). Springer, Berlin, Heidelberg.
13. Kwok SH, Lam EY. Effective uses of fpgas for brute-force attack on rc4 ciphers. *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*. 2008 Aug;16(8):1096-100.
14. Couture N, Kent KB. The effectiveness of brute force attacks on RC4. In *Communication Networks and Services Research*, 2004. Proceedings. Second Annual Conference on 2004 May 19 (pp. 333-336). IEEE.
15. Najafabadi MM, Khoshgoftaar TM, Kemp C, Seliya N, Zuech R. Machine learning for detecting brute force attacks at the network level. In *Bioinformatics and Bioengineering* (BIBE), 2014 IEEE International Conference on 2014 Nov 10 (pp. 379-385). IEEE.
16. Bonneau J. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Security and Privacy* (SP), 2012 IEEE Symposium on 2012 May 20 (pp. 538-552). IEEE.
17. Florencio D, Herley C. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web* 2007 May 8 (pp. 657-666). ACM.
18. Gamido HV, Sison AM, Medina RP. Modified AES for Text and Image Encryption. *Indonesian Journal of Electrical Engineering and Computer Science*. 2018 May 27;11(3).
19. S. Tom, “Microsoft's Nadella Wants To Help Coders Take a Quantum Leap,”2017 <http://business.financialpost.com/technology/googles-quantum-computing-push-is-opening-a-new-front-in-cloud-battle>.
20. Shor PW. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science*, 1994 Proceedings., 35th Annual Symposium on 1994 Nov 20 (pp. 124-134). IEEE.
21. Vandersypen L, van Leeuwenhoek A. 1.4 Quantum computing-the next challenge in circuit and system design. *In Solid-State Circuits Conference* (ISSCC), 2017 IEEE International 2017 Feb 5 (pp. 24-29). IEEE.
22. Ahmad J I, Din R, Ahmad M. Analysis Review on Public Key Cryptography Algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*. 2018 Nov 2;12(2): 447-454.
23. Shao F, Chang Z, Zhang Y. AES encryption algorithm based on the high-performance computing of GPU. In *Communication Software and Networks*, 2010. ICCSN'10. Second International Conference on 2010 Feb 26 (pp. 588-590). IEEE.
24. Alghazzawi DM, Hasan SH, Trigui MS. Advanced encryption standard-cryptanalysis research. In *Computing for Sustainable Global Development* (INDIACom), 2014 International Conference on 2014 Mar 5 (pp. 660-667). IEEE.
25. Deshpande AM, Deshpande MS, Kayatanavar DN. FPGA implementation of AES encryption and decryption. In *Control, Automation, Communication and Energy Conservation*, 2009. INCACEC 2009. 2009 International Conference on 2009 Jun 4 (pp. 1-6). IEEE.
26. K. Anjali, Sailing Safe in Cyberspace: Protect Your Identity and Data. SAGE Publications India, 2013
27. A. Stevenson, “These are the 19 worst data breaches in history,” 2015; http://www.businessinsider.com/target-ebay-and-ubisoft-a-list-of-the-biggest-data-breaches-in-history-2015-7/?IR=T/#19-evernote-number-of-records-compromised-50-million-1
28. G. Cluley, “Evernote Hacked – almost 50 million password reset after security breach,” 2013: https://nakedsecurity.sophos.com/2013/03/02/evernote-hacked-almost-50-million-passwords-reset-after-security-breach/
29. C.Johnson, “Equifax Data Breach,” 2017: http://clark.com/tag/equifax-data-breach/
30. Jo HJ, Yoon JW. A new countermeasure against brute-force attacks that use high performance computers for big data analysis. *International Journal of Distributed Sensor Networks*. 2015 Jun 1;11(6):406915.
31. Chatterjee R, Bonneau J, Juels A, Ristenpart T. Cracking-resistant password vaults using natural language encoders. In *Security and Privacy* (SP), 2015 IEEE Symposium on 2015 May 17 (pp. 481-498). IEEE.
32. Golla M, Beuscher B, Dürmuth M. On the security of cracking-resistant password vaults. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* 2016 Oct 24 (pp. 1230-1241). ACM.
33. Beunardeau M, Ferradi H, Géraud R, Naccache D. Honey Encryption for Language. In *International Conference on Cryptology in Malaysia* 2016 Dec 1 (pp. 127-144). Springer, Cham.
34. Huang Z, Ayday E, Fellay J, Hubaux JP, Juels A. Genoguard: Protecting genomic data against brute-force attacks. 2015.
35. Jaeger J, Ristenpart T, Tang Q. Honey encryption beyond message recovery security. In Annual *International Conference on the Theory and Applications of Cryptographic Techniques* 2016 May 8 (pp. 758-788). Springer, Berlin, Heidelberg.
36. Kim JI, Yoon JW. Honey chatting: A novel instant messaging system robust to eavesdropping over communication. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2016 IEEE International Conference on 2016 Mar 20 (pp. 2184-2188). IEEE.
37. Yoon JW, Kim H, Jo HJ, Lee H, Lee K. *Visual honey encryption: Application to steganography*. In Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security 2015 Jun 17 (pp. 65-74). ACM.
38. Tyagi N, Wang J, Wen K, Zuo D. Honey encryption applications. *Network Security*. 2015.
39. 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.
40. 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.
41. 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 Security2017: 329-346