Modified honey encryption scheme for encoding natural language message

Abiodun Esther Omolara, Aman Jantan
School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia

Article Info

Article history:

Received May 24, 2018
Revised Nov 27, 2018
Accepted Dec 22, 2018

Keywords:

Adversary
Attacker
Brute-force
Ciphertext
Honey encryption
Plaintext

ABSTRACT

Conventional encryption schemes are susceptible to brute-force attacks. This is because bytes encode utf8 (or ASCII) characters. Consequently, an adversary that intercepts a ciphertext and tries to decrypt the message by brute-forcing with an incorrect key can filter out some of the combinations of the decrypted message by observing that some of the sequences are a combination of characters which are distributed non-uniformly and form no plausible meaning. Honey encryption (HE) scheme was proposed to curtail this vulnerability of conventional encryption by producing ciphertexts yielding valid-looking, uniformly distributed but fake plaintexts upon decryption with incorrect keys. However, the scheme works for only passwords and PINS. Its adaptation to support encoding natural language messages (e-mails, human-generated documents) has remained an open problem. Existing proposals to extend the scheme to support encoding natural language messages reveals fragments of the plaintext in the ciphertext, hence, its susceptibility to chosen ciphertext attacks (CCA). In this paper, we modify the HE schemes to support the encoding of natural language messages using Natural Language Processing techniques. Our main contribution was creating a structure that allowed a message to be encoded entirely in binary. As a result of this strategy, most binary string produces syntactically correct messages which will be generated to deceive an attacker who attempts to decrypt a ciphertext using incorrect keys. We evaluate the security of our proposed scheme.

Corresponding Author:

Aman Jantan,
Security and Forensic Research Group (SFRG) Laboratory,
School of Computer Sciences,
Universiti Sains Malaysia, Malaysia.
Email : styleest2011@gmail.com

1. INTRODUCTION

Cryptography lies at the heart of a multitude of processes that incorporate E-Commerce, Internet Banking, Business and Social Networks, Smart Systems and so forth in our information-aware society [1], [2]. The evolution of the digital world has affected the way in which people manage finances, procure goods and access healthcare. People have become increasingly reliant on leveraging technology to enjoy instant access to information, business collaboration, family and friends [3], [4]. This trend of societal dependence on modern infrastructure has created an avenue for constant security threats and risk. This challenge has been demonstrated by incessant attacks on computer networks, credit card fraud, malware attacks, cell phones hack et cetera [5], [6]. To reinforce our computerized-reliant society for the present and future generation, and leverage on the opportunities that these technologies proffer, there is a pressing need for reliable information infrastructure with strong security requirements. An essential building block to achieve information security is cryptography [7], [8].

Cryptography is the study of the science and art of concealing information to prevent a distrusted party from learning the content of the message [9]. State-of-the-art conventional cryptographic schemes are prone to brute-force attack. A brute force attack is a trial and error approach used by an adversary to decrypt a ciphertext, for instance, an attempt to crack a user's password. The conventional encryption schemes make it easy for an adversary to continue to employ this method and achieve success. In hindsight, all conventional encryption schemes are encoded in utf8 (or ASCII) characters. Consequently, an adversary that intercepts a ciphertext and tries to decrypt the message using a wrong key can filter out some of the combinations of the decrypted message by observing that some of the sequences are a combination of characters from different languages and also symbols which are non-uniformly distributed [10], [11]. It is safe to conclude that an adversary can judge if his attack is successful based on the structure/distribution of the message he recovers during his attacks [12], [13]. A non-uniform distribution, which forms no plausible meaning signifies an incorrect plaintext and key. Moreover, the advent of high-processing tools such as FPGA, GPU also makes the brute-force attack to be highly successful [14], [15], [16]. Therefore, an attacker that intercepts an encrypted message has a high chance of recovering the key and subsequently the message encrypted under such key by using the distinguisher technique – which is to judge the decryption output based on the distribution/structure of the message.

Security experts and the cryptographic community have proposed several countermeasures to mitigate the brute-force attack on conventional encryption schemes by increasing the length of the key, computational complexity. Nevertheless, all these methods fail to address the underlying problem, which is using the distinguisher technique to inform the output of the decryption. An attacker can continue to make repeat decryptions on an oracle by trying random keys until he finds a plausible message [17-18].

A breakthrough was made in the year 2014 when Juels and Ristenpart [19-20] proposed the Honey Encryption (HE) scheme in a conference, *Eurocrypt 2014*. Honey encryption is a paradigm that provides and improves security beyond the conventional brute-force bounds. It provides a plausible, valid-looking but fake message when an incorrect key is used for decrypting a ciphertext. It does not yield under a brute-force attack and it was proposed for passwords and PINs and have been implemented in this regard successfully [19-20]. However, HE has not been applied for the security of natural language messages such as emails, human-generated documents and certain formats of message. The applications of the standard HE proposed by [19-20] are precisely for passwords to protect passwords or low min-entropy keys to protect high min-entropy keys. Indeed, the authors of HE left as an open challenge, its adaptation to natural language message when they made this statement,

"*...for human-generated messages (password vaults, e-mail, etc.) (...) is interesting as a natural language processing problem*." [19], [20].

While some research efforts have been made towards adapting the scheme to encode human-generated message, the current proposals are all susceptible to a Chosen-Ciphertext Attack (CCA). A CCA is an attack model for cryptanalysis in which the adversary collects information, at least in part, by selecting a ciphertext and decrypting it by trying random keys. This attack is inevitable when parts or fragments of the plaintext message is revealed in the decrypted message. A practical attacker can easily use the revealed parts of the message to form other parts of the message and recover the plaintext and the key. Moreover, some of the proposed approaches do not scale well; they fail to model human language when used to encode human-generated documents, thus, failing to convince the attacker that he has the target message.

This paper seeks to address the shortfalls of the standard Honey encryption scheme and proposals made by the notable research efforts in this domain. The principal objective is to extend the HE schemes to adapt it for encoding natural language message. Our approach also addresses the CCA attacks which current approach fail to curtail. We employed Natural language processing tools and techniques to build convincing decoy message which will be served to an attacker that tries to decrypt a ciphertext using incorrect keys. This strategy will ultimately allow encryption of other formats specifically human-generated messages, such as emails, written documents, etc. Figure 1 presents how the conventional encryption scheme works during a brute-force attack. In summary, the contributions of this research include the following:

a. Creating a structure that allows a message to be encoded entirely in binary. As a result of the encoding strategy, most binary strings produce a syntactically correct message which is capable of fooling an adversary in search of the plaintext and key.
b. The proposed scheme address CCA attack by generating a message structure that does not reveal any part of the plaintext in the decoy message and failed decryption (incorrect key) produces radically different messages from the plaintext.
c. Length of plaintext is concealed by appending a random binary string to the ciphertext since the decoder runs until the sentence is parsed and then stops. This way the attacker is unable to determine the length of the plaintext.

d. First successful implementation of a provable secured natural-language-based honey encryption scheme and its adaptation to natural language messages.

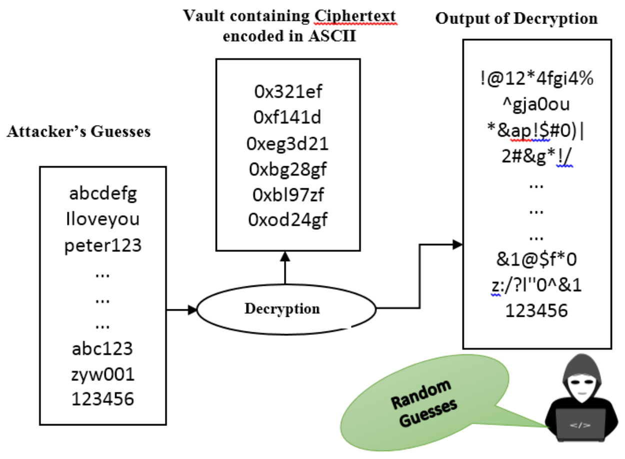

Figure 1. Brute-force attack on current conventional encryption method on a password vault

An attacker that intercepts a plaintext and submits random passwords get non-uniformly distributed characters as long as he supplies an incorrect key. This is a clear distinguisher that he has not achieved the plaintext. He continues to try random keys until he gets a valid-looking message. As soon as he gets plausible, uniformly distributed characters, then he is sure he has achieved the message and the key. For example, in Figure 1, when the attacker gets '123456', he knows that he has achieved the key. He is certain he has the correct key because '123456' is distributed evenly and this is made worst as users tend to use simple, easy-to-guess passwords. Also, the attackers are aware of how users choose weak passwords (based on the past release of compromised passwords).

2. HONEY ENCRYPTION

The intuition of Honey encryption was borne by Juels and Ristenpart [19-20] after observing the high rate of password been compromised. For instance, using conventional encryption scheme, if plaintext, P is a 16-digit Master card number encoded via ASCII and the conventional password-based encryption scheme is used as cipher, the probability that any $Pi \neq P$ is a valid ASCII encoding of a 16-digit string which is negligible, at t $(10/256)^{16} < 2^{-74}$. Hence, an adversary can reject incorrect messages and recover P with a high probability [19-20].

Honey encryption provides security beyond the brute-force barrier by producing ciphertexts which, upon decryption with incorrect keys, yields plausible-looking plaintexts. The strategy here is to confound the life of the adversary by making message recovery even after trying every candidate keys to be impossible. The central component of HE is called a distribution transforming encoder (DTE). This component is constructed with an estimate of the message distribution pm. Encoding a message sampled from pm should produce a "seed" value which is uniformly distributed. The DTE is a pair of encoding and decoding algorithm.

The encoding algorithm is randomized, it takes as input a message $M \in M$ and outputs a value in a set S. Also, the decoding algorithm is deterministic and takes as input a value $S \in S$ and outputs a message $M \in M$. When the seed is supplied, a DTE must decode correctly to the plaintext but for every incorrect key, it should decode to a plausible message sampled from the message distribution. A DTE scheme is correct if for any $M \in M$, Pr[decode(encode(M)) = M] = 1 [19-20].

The HE scheme consists of two steps usually referred to as DTE-then Encrypt. In the first step, the DTE is applied to the Message, M to obtain a Seed S. In the second step, the seed S is encrypted under a conventional encryption scheme using the key K, yielding an HE ciphertext C. The steps are as follows:

a. The encoding process, where the Distribution Transforming Encoder (DTE) is applied to the plaintext followed by a conventional encryption scheme.
b. The decoding process, where the Distribution Transforming Decoder (DTD) is applied to the ciphertext followed by a conventional encryption scheme.

The HE algorithm showing the encode and decode process is presented in Figure 2.

Honey Encryption Algorithm:	**Honey Decryption Algorithm:**
HEnc (K, M)	HDec (K, (R, C))
S ¬$ encode(M)	S'¬ H (R, K)
R ¬$ {0, 1}n	S ¬ C ⊕ S'
S' ¬ H (R, K)	M ¬ decode(S)
C ¬ S'⊕ S	return M
return (R, C)	

Figure 2. Honey encoding and decoding algorithms

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.

3. PROPOSED APPROACH

Our approach leverages the power of deception to persuade and confuse the attacker that he has the original message. The experiment of this research are implemented with Python 3.63 and Natural Language Processing (NLP) libraries. The NLP tools capture the relationship between words and how they combine to form meaningful sentences. Readers interested in detail treatment of NLP should see [21]-[22]. The decoy message will have a completely different message structure from the original message (plaintext) but will have a semantic and contextual meaning. Each message to be encoded is treated as a string of sentence and processed as a grammatical feature in the English Language. Each word is classified based on the part of speech by its syntactic functions. In the English Language, the main parts of speech are noun, pronoun, verb, adverb, adjective, preposition, interjection, conjunction and determiner. Figure 3 shows how nouns are processed and encrypted, and Figure 4 shows a flowchart of the complete system of our proposed scheme. Figure 5 shows how nouns are processed and decrypted and Figure 6 shows a flowchart of the complete system of our proposed scheme. In summary, we create a structure that allowed a sentence to be encoded completely in binary. As a result of the encoding strategy, most binary strings produce a syntactically correct sentence.

```
Algorithm 1: Noun Encoding
1: Get synsets for the noun
        if personal noun
                Drop synsets with non-personal meaning for the noun (i.e., using mum to refer to a flower)
        Else
                Drop synsets with personal meaning for the noun (i.e., using a dog to refer to a person)
        if synset list is non-empty
                Randomly select synset from list
        Else
        if the noun is a pronoun
                Set offset to the length of the list of nouns in wordnet plus one and encode in binary
                Encode pronoun
        Else
                Set offset to the length of the list of nouns in wordnet plus two and encode in binary
                Encode unknown word
        return binary string
2: Get the path from the root of wordnet noun tree to selected synset
        if personal noun
                Drop nodes in the path that are parents to "person noun" synset
        Else
                Drop nodes in the path until "person noun" synset is not in the path
                Randomly select a node in the remaining path to be subtree root
                Encode offset of the selected node in the list of wordnet's nouns in binary
        while not at the desired node (noun passed in)
                Append 1 to binary encoding to indicate that search is not complete
                Find the index of next node in the path in the list of current root's children
                Encode index in binary and append to a string
                Append 0 to binary encoding to indicate that search is complete
                Find index in the list of synset's lemmas of lemma containing the desired noun
                Encode index in binary and append to a string
        3: return binary string
```

Figure 3. Pseudocode for implementing ***Encrypt*** Algorithm of the Proposed (Modified) Honey Encryption

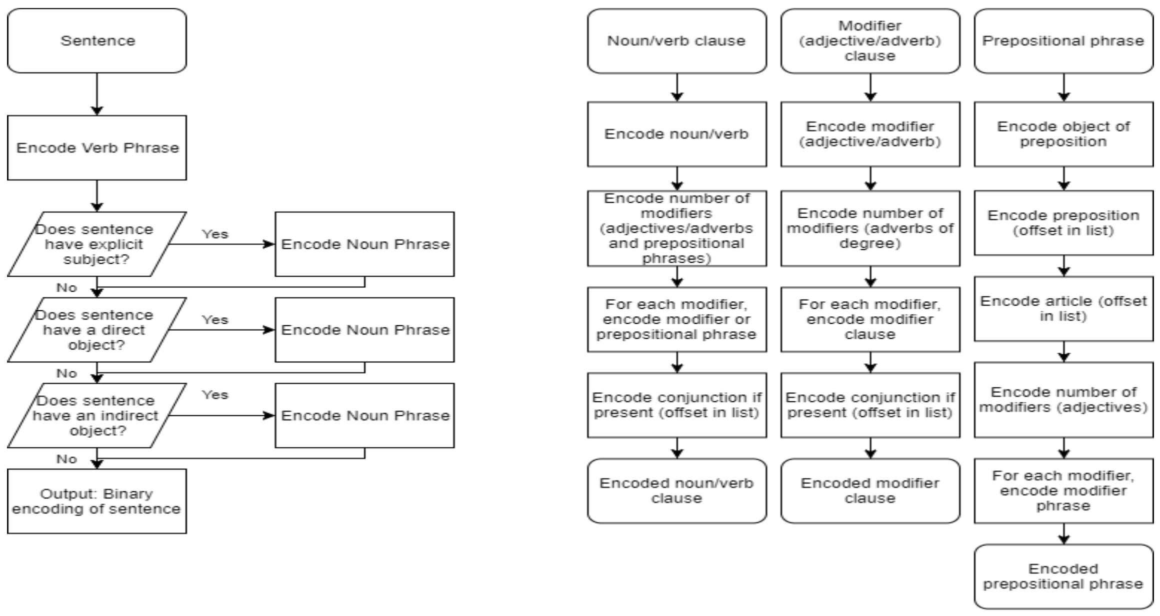
Sentence
Encode Verb Phrase
Does sentence have explicit subject?
Yes
Encode Noun Phrase
No
Does sentence have a direct object?
Yes
Encode Noun Phrase
No
Does sentence have an indirect object?
Yes
Encode Noun Phrase
No
Output: Binary encoding of sentence
Noun/verb clause
Encode noun/verb
Encode number of modifiers (adjectives/adverbs and prepositional phrases)
For each modifier, encode modifier or prepositional phrase
Encode conjunction if present (offset in list)
Encoded noun/verb clause
Modifier (adjective/adverb) clause
Encode modifier (adjective/adverb)
Encode number of modifiers (adverbs of degree)
For each modifier, encode modifier clause
Encode conjunction if present (offset in list)
Encoded modifier clause
Prepositional phrase
Encode object of preposition
Encode preposition (offset in list)
Encode article (offset in list)
Encode number of modifiers (adjectives)
For each modifier, encode modifier phrase
Encoded prepositional phrase

Figure 4. Proposed encoding algorithm

```
Algorithm 2: Noun Decoding
1: Extract offset from binary string
        if pronoun
                return decoding of the pronoun
        if unknown word
                return decoding of the unknown word
2: Get synset of a noun at given offset in the list of wordnet's nouns and set as root
        while not at the desired node (the first character in the binary string is non-zero)
                Extract offset from binary string
                Set child at offset in the list of root's children to new root
3: Extract lemma number from a binary string
4: Get specified lemma
5: return the name of the lemma (desired noun)
```

Figure 5. Pseudocode for implementing ***decrypt*** algorithm of the proposed (modified) honey encryption

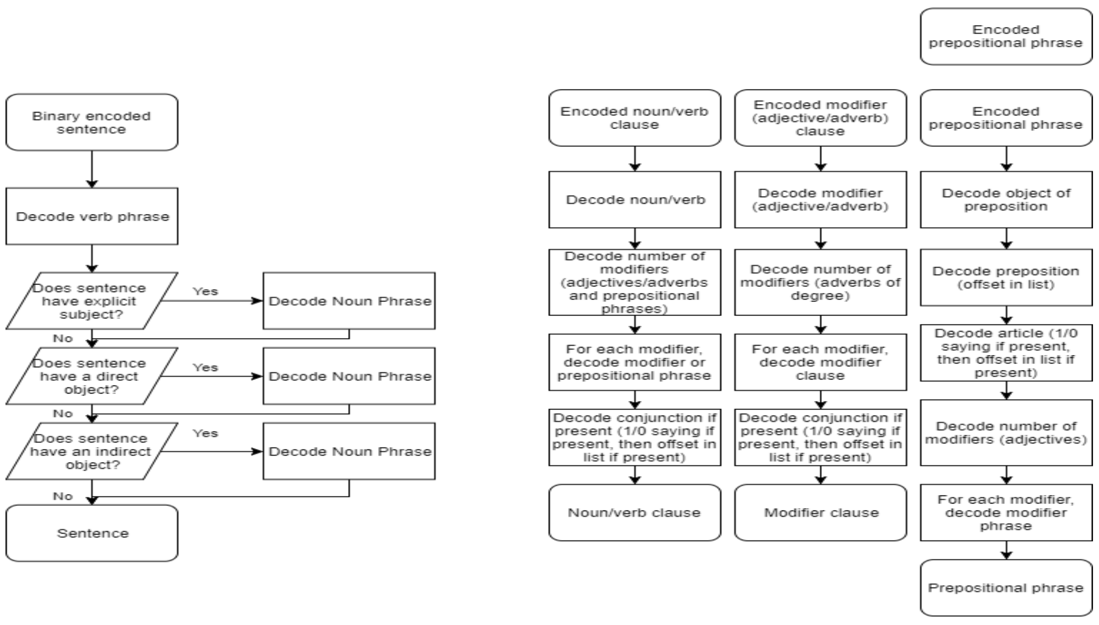
Binary encoded sentence
Decode verb phrase
Does sentence have explicit subject?
Yes
Decode Noun Phrase
No
Does sentence have a direct object?
Yes
Decode Noun Phrase
No
Does sentence have an indirect object?
Yes
Decode Noun Phrase
No
Sentence
Encoded prepositional phrase
Encoded noun/verb clause
Decode noun/verb
Decode number of modifiers (adjectives/adverbs and prepositional phrases)
For each modifier, decode modifier or prepositional phrase
Decode conjunction if present (1/0 saying if present, then offset in list if present)
Noun/verb clause
Encoded modifier (adjective/adverb) clause
Decode modifier (adjective/adverb)
Decode number of modifiers (adverbs of degree)
For each modifier, decode modifier clause
Decode conjunction if present (1/0 saying if present, then offset in list if present)
Modifier clause
Encoded prepositional phrase
Decode object of preposition
Decode preposition (offset in list)
Decode article (1/0 saying if present, then offset in list if present)
Decode number of modifiers (adjectives)
For each modifier, decode modifier phrase
Prepositional phrase

Figure 6. Proposed decoding algorithm

Note that:
Nouns in Wordnet are organized using a tree structure with general concepts at the root, moving to specific nodes at the leaves. We utilize this structure for encoding of all nouns within the encoding algorithm.
Verb lemmas in Wordnet are stored with lists of verb frames. These are the possible structures of sentences using that verb. For example, "Somebody appears" is a valid sentence structure but "Somebody appears something" is not.
Pronoun encoding/decoding: Have a list of pronouns, encoded as an offset in the list.
Unknown encoding/decoding: Each letter encoded as a position in the alphabet (0-25)
Article encoding/decoding: 0/1 for not exists/exists, then 1/0 if the/a
Conjunction encoding/decoding: Have a list of conjunctions, encoded as an offset in the list.

A general use case of the proposed algorithm is to combine it with any conventional cipher. For instance, the sender will encrypt this binary string with a cipher (AES, RC4, Twofish, etc.). A wrong key will produce an incorrectly encoded string which should decode to a semantically correct message. Only when the correct key is applied should the plaintext decode correctly. Any incorrect key (flipping a bit) will decode correctly to a valid message with completely different length and structure from the plaintext. Our approach is completely different from existing approach because no part of the plaintext will be revealed during decryption with incorrect key. This approach makes it impossible for the adversary to carry out a chosen ciphertext attack.

We consider a case where an email is to be encoded using our approach. For instance, each email usually starts with some greeting and will usually end with a goodbye message. An adversary expecting an email who gets instead a text not starting from this type of greetings will quickly discard the message. In this case, we define encryption protocol differently. We assume that email-message should always start from the greeting and first few bits will encode one of the possible greeting’s variations. This way it is not possible for any encryption key to get the results without greetings and goodbye. Therefore, the higher probability of the decoy message is done by restricting potential user format of the message.

An easy implementation will be to define three parts of the message which are allowed. For instance, 1) Greetings with one of the following options: 00 - "Hello", 01 - "Hi", 10 - "Greetings", 11 - "Good time of day"; 2) middle sentence with the following options: 00 - "how are you? "; 01-"Fine"; 10-"having a hard day"; 11 - "let's meet"; 3) Good byes: 00 - "Bye"; 10 - "Looking forward"; 01 - "have a nice day"; 11-"Best wishes” This protocol will limit amount communication very much, but every combination or key can lead to a meaningful message. There is no way for the attacker to guess if his key was incorrect.

4. RESULTS AND ANALYSIS

To evaluate our proposed system, we test with a controlled amount of noise added to a ciphertext (i.e., 1% wrong, 5% wrong, etc.) and compare the sanity of resulting messages.

Word Count: The number of words encoded

Noise Added: Percentage of random bits flipped (incorrect key)

Change in the Word Count: When the adversary tries to decrypt with the key, how many sane words were generated.

$P_{important}$: How many important words from the underlying plaintext appear in the decrypted message when the adversary tries incorrect keys. We define $P_{important}$ as very important keywords in the plaintext (nouns, pronoun, verb, adverb, adjective) which must not appear in the event of decryption with wrong keys.

$P_{regular}$: How many regular words from the underlying plaintext appear in the decrypted message when the adversary tries incorrect keys. We define $P_{regular}$ as common words in the plaintext (determiner, conjunction, interjection, preposition). This is essential because regular words help form a sentence and they appear in every message and cannot be used to recover the plaintext. For example, a, the, of, on etc are essential to form sentence. Time-taken: The time it takes to decrypt using our proposed method. Performance of the proposed modified honey encryption scheme as shown in Table 1.

Table 1. Performance of the Proposed Modified Honey Encryption Scheme

Word Count	Noise Added (%) (brute-forcing with incorrect keys)	Change in the word and word count during decryption	Check if CCA is possible (Any revealed word from the plaintext in the decrypted ciphertext?) $P_{important}$	$P_{regular}$	Time-taken to decrypt (ms)
4	4	53	0	0	1.90
7	2	59	0	4	2.23
10	3	54	0	5	3.12
15	2	62	0	4	3.23
20	3	71	0	7	4.40
25	4	88	0	6	4.89
28	5	97	0	9	7.10
38	1	105	0	6	9.30
43	8	127	0	8	10.11
48	7	156	0	9	13.21
82	5	304	0	12	31.11

5. DISCUSSION

The proposed modified honey encryption scheme employs a novel approach for adapting the honey encryption scheme to support the successful encoding of human-generated message which has been an open problem since the scheme was proposed [19,20]. This novel invention brings new benefits as well as some limitations.

In summary, we were able to create a structure where messages are encoded in binary such that message recovery becomes impossible for an adversary that intercepts a ciphertext and tries to decrypt it using random keys. The standard HE scheme was designed and provides provable security only in the context of Passwords, Pins and RSA Keys. Research efforts over the years to extend it for encoding larger domains (such as human-generated message) has failed to produce convincing decoy message, do not scale well. Worst is they reveal structural information of the underlying plaintext, thus, the vulnerability to chosen ciphertext attack (CCA) which makes the probability of message recovery high. Our proposed approach is the first successful implementation of adapting the honey encryption scheme for supporting encoding of human-generated message. It produces a plausible message that is good enough to deceive the adversary. More importantly, the critical message (keywords) that the sender might want to protect is completely concealed and cannot appear during decryption with incorrect keys, thereby, countering the CCA. However, we must state that our approach impacts time when the message gets larger. For this limitation, we are thinking of other ways to curtail this in our next research efforts.

6. CONCLUSION

In the conventional encryption scheme, messages are encoded in ASCII (or UTF-8, etc.) which makes them prone to a brute-force attack, as an adversary who tries to decrypt a ciphertext can easily filter out correct messages from incorrect ones based on the non-uniform distribution of the characters. Honey encryption (HE) offers a countermeasure to the conventional encryption scheme by offering a barrier against brute-force attacks. HE is a decoy-based encryption scheme which yields plausible looking plaintexts under decryption with incorrect keys such that decryption attempts alone are insufficient to recover the correct plaintext. As a result, an adversary is made to believe he has the plaintext when what he has is a decoy-message.

In this paper, we modified the HE scheme to support the encoding of human-generated message by leveraging natural language processing techniques. Our approach models the connection between words with the specific goal of capturing the patterns of language to form syntactically and semantically meaningful message as a framework to build realistic but fake messages as decoys which is served to an adversary trying to decrypt the ciphertext using an invalid-key. For instance, an attacker that intercepts an encrypted email message and tries to decrypt the message gets a plausible email message, and he is fooled into believing he has the original email message. The message structure and message length contained in the underlying message is kept entirely secret and failed decryption produces radically different message from the original message as decoys message, and such features can be useful for successfully concealing and protecting emails and human-written documents that are top secret.

REFERENCES

[1] Gamido HV, Sison AM, Medina RP., "Modified AES for Text and Image Encryption," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 11(3), pp. 27, May 2018.

[2] Chandrakala N, Rao BT., "Migration of Virtual Machine to improve the Security of Cloud Computing," *International Journal of Electrical and Computer Engineering (IJECE)*. vol. 8(1), pp. 210-9, Feb 2018.

[3] Sultanpure KA, Gupta A, Reddy LS., "An Efficient Cloud Scheduling Algorithm for the Conservation of Energy through Broadcasting," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8(1), pp.179-88, Feb 2018.

[4] Singh JP, Kumar S. "Authentication and encryption in cloud computing," In *Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015 International Conference on 2015 May 6 , IEEE*, pp. 216-219.

[5] Jaber AN, Zolkipli MF, "Use of cryptography in cloud computing. In Control System," *Computing and Engineering (ICCSCE), 2013 IEEE International Conference on 2013 Nov 29. IEEE*, pp. 179-184.

[6] Chen D, Zhao H., "Data security and privacy protection issues in cloud computing," In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on 2012 Mar 23. IEEE*, vol. 1, pp. 647-651.

[7] Omolara OE, Oludare AI, Abdulahi SE., "Developing a modified Hybrid Caesar cipher and Vigenere cipher for secure Data Communication," *Computer Engineering and Intelligent Systems*, vol. 5, pp. 34-46, 2014.

[8] Stallings W, Brown L, Bauer MD, Bhattacharjee AK, "Computer security: principles and practice," *Pearson Education* 2012.

[9] Lorek P, Zagórski F, Kulis M., "Strong stationary times and its use in cryptography," *arXiv preprint arXiv:1709.02631*, Sep 2017.

[10] 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*. vol. 11(6), pp. 406915, Jun 2015.

[11] Beunardeau M, Ferradi H, Géraud R, Naccache D., "Honey Encryption for Language," In *International Conference on Cryptology in Malaysia. Springer, Cham*, pp. 127-144, Dec 2016.

[12] 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*. IEEE, pp. 2184-2188.

[13] 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 2018*, vol. 1, 2018.

[14] Cousins DB, Rohloff K, Sumorok D., "Designing an FPGA-accelerated homomorphic encryption co-processor," *IEEE Transactions on Emerging Topics in Computing*. vol. 5(2), pp. 193-206, Apr 2017.

[15] Kestur S, Davis JD, Williams O, "Blas comparison on FPGA, CPU and GPU. VLSI (ISVLSI)," *2010 IEEE computer society annual symposium on 2010 Jul 5. IEEE*, pp. 288-293.

[16] Sutikno T, Idris NR, Jidin A., "High-Speed Computation using FPGA for Excellent Performance of Direct Torque Control of Induction Machines," *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, vol. 14(1) pp. 1-3, Mar 2016.

[17] Marks M, Jantura J, Niewiadomska-Szynkiewicz E, Strzelczyk P, Góźdź K., "Heterogeneous GPU&CPU cluster for high-performance computing in cryptography," *Computer Science*, vol. 13, pp. 63-79, 2012

[18] Couture N, Kent KB, "The effectiveness of brute force attacks on RC4," *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on 2004 May 19. IEEE*, pp. 333-336.

[19] 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, Springer, Berlin, Heidelberg*, pp. 293-310.

[20] Juels A, Ristenpart T., "Honey Encryption: Encryption beyond the brute-force barrier," *IEEE Security & Privacy*, vol. 12(4), pp. 59-62, Jul 2014.

[21] Bird S, Loper E., "NLTK: the natural language toolkit," In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions 2004 Jul 21, Association for Computational Linguistics*, pp. 31.

[22] Chowdhury GG., "Natural language processing," *Annual review of information science and technology*, vol. 37(1), pp. 51-89, Jan 2003.