Enhanced Honey Encryption Algorithm on e-mail with Increased Message Space

Apoorv Singh¹, Chandraket Singh², Samridhi Mishra³
¹,²,³Final Year Student, Department of Computer Science and Engineering, Galgotias College of Engineering and Technology, Greater Noida, India

Abstract: In the present scenario privacy is provided but they do not provide enough protection against attacks like dictionary attack and brute force attack. Honey encryption (HE) proposed by Juels and Ristenpart in 2014 as a solution against these attacks. As honey encryption provides fake messages when a wrong password is entered, it gives an extra layer of security to the user as all these honey messages which consist of sweet words in distribution transforming encoder (DTE). We are using an enhanced honey encryption algorithm which increases message space for DTE (with discrete distributed function) which we have used on email to encrypt the given email. We propose an approach for email encryption using Stanford dependency parser, wordnet from Princeton enhance, honey encryption method and SHA-II 256-bit encryption method for the secure transmission of email. We ensure that no one except the sender and receiver gets access to the original message.

Keywords: Honey words, Sweet words, Ciphertext, Distribution transforming encoder, (DTE), Honey encryption (HE), Natural language processing (NLP), Plane text, Dictionary attack, Brute force attack, SHA-II.

1. Introduction

In today's technologically advanced society email has now become one of the official ways to communicate in the industries. Not only that email has almost become one of the most prominent official ways of communication, but it is also now considered an authentic official document. Although there are many ways for securing your email and text from getting attacked by a hacker or snooped of there still some ways to breach those authentications.

Offline and online attacks can easily break weak passwords through brute force or dictionary author example, user passwords are stored in a database in the form of hash H(P). When the user enters the password for login a fresh hash key is generated which is matched with the hash key stored in the database if the key matches the user gets login. Although due to the password being longer it could become more difficult to guess. If the entropy of the password is high then also it could become hard to guess the password by brute force method. But as the technology is progressing with higher processing power both these problems of long passwords and high entropy of password could be solved by an attacker to crack the password of any security with sufficient time and unbounded attempt.

To solve these problems Honey encryption (HE) proposed by Juels and Ristenpartion could be used to serve the attacker with valid looking but fake information. It could become a very useful tool to encrypt passwords even if the user uses weak passwords. As it provides decoy to the attacker then even if the password format becomes obsolete then also it could provide a significant amount of security for the encrypted passwords as the decoys would be enough for an attacker to get confused to find and confirm which information is real and which is fake.

In this paper, we have surveyed some of the many applications of HE, we have discussed all the papers which we deemed necessary to explain the advantages, disadvantages and improved application of honey encryption on latest technical development such as group data sharing in the cloud environment, natural language processing not only that we have also included some results for comparison between some common encryption and honey encryption.

In this paper, we have discussed the paper, Honey Encryption (Ari Juels, Thomas Ristenpart) discussing the basics of honey encryption, Cracking-Resistant Password Vaults using Natural Language Encoders which discusses various password-based encryption (PBE) comparing various methods with honey encryption.

2. Terminology and background concepts

A. Terminologies

This section outlines some terminologies used in the paper for basic understanding:

Attacker: Words like adversary, intruder eavesdropper will be used interchangeably to mean attacker.

Ciphertext: Ciphertext refers to the encoded message which is to be made secure, a cypher (an algorithm) is used to transform the message/information to be in an unreadable format. Terms like, decoy, fake-text, false-text, honey message, honey word is used to mean the ciphertext in this paper.

Message Space: The message space represented by m or M contains all possible message honeyword of the information to be encrypted. Hence, it is a set of all fake messages modelled from the original message.

Distribution Transforming Encoders (DTE): It is the encoder used by honey encryption and discussed in detail section B.
B. Properties for a good encoder

The main thing to keep in mind while making an encoder are:

a) Indistinguishability: The fake message should be difficult to differentiate from the original message by using any means i.e. by using any software or by seeing it by any person. The fake message should be difficult to differentiate between honey message and the original message. Distributed transform encoders should produce a fake message which is difficult to distinguish from the original message, so the attacker can be dought.

b) Security: Distributed Transform Encoder should be able to produce the same message like a human being and it should also hide the real message detail from the unknown user or attacker.

c) Message space: To be a good encoder for honey encryption it should also be able to encode messages such that the memory occupied by message space should be minimal.

3. Honey encryption

It addressed the concept of weak password and their protection from an intruder using hash tools for generating more complex password in a vault and their protection using honey encryption [1]. It introduced the concept of Distribution Transforming Encoders(DTE) for message space in HE schema.

A. Distribution transforming encoder

This is used to encode messages in message space M, the procedure to encode over a message space is as follows:

Let probability distribution be Pd on the message space M, indicating that a user selects m Î M for encryption with probability Pd(m).

A DTE encoder encodes m as an n-bit seed Se{0, 1}n. This encoding needn’t be unique: many seeds might correspond to M, in which case encoder selects one such seed uniformly at random. (however, Every seed corresponds to a unique message.)

We require that encode be efficiently invertible. In other words, given Se, we can decode through the inverse DTE decode(Se) = m, which returns Se’s unique corresponding message.

As DTE gives strong security, decode accurately generates Pd. In this case, selecting Se uniformly at random from {0, 1}n and decoding to obtain m = decode(Se) returns approximately the original Pd. In other words, the DTE is a good model of message distribution [1] in this DTE we are using Discrete distributive function(DDF) rather than cumulative distributive function for enhancing message space in DTE [7].

B. HE Construction

Since DTE is a smart encoder as shown during this paper, then we are able to construct a HE schema simply by using DTE to distribute “original message” and “honey messages” with their corresponding “passwords” and so encrypt it. Thus let the goal be to encrypt m under K to Yield C.
(a) Honey encryption. (b) Honey decryption. H indicates the cryptographic hash function, K is a key, m is a message, Se is a seed, R is a random string, C is a ciphertext, and ¬$ denotes uniform random assignment.

C. Security

HE aims to secure the message of a user by providing a fake message for the wrong password entered by the attacker such that he/she is not able to distinguish between real and fake message and DTE help it in preventing an attacker from deciding m from C using K with high probability. For Example:

Sam is reviewing a confidential report he wants to send a response to Lucy about the state of the report which is “under consideration”. Therefore, Sam constructs DTE that maps responses (“Section 1”, “Accepted”, “not accepted”, “Section 2”, “under consideration”, “need summarization”) on space of n-bit string where n is such that 2n>=5 therefore 3-bit strings which are {000,001,010,011,100,101,110,111}. Sam encodes his response “under consideration” to obtain seed encode (“under consideration”) = Se = 010. Now he selects a random string R and computes Se’= H(R, K) where K represents a key and here Se’=H(R,2509) = 110. Then sam computes C = Se⊕Se’=C=Se⊕Se’=010⊕110=100Sam sends C to Lucy. Lucy receives C from Sam since she knows the key she generates Se’ using function for Se’ as given above we get Se’=H(R,2509)=110 and thus we compute C=Se⊕Se’=100⊕110=010. she decode (010)= “under consideration” from seed space to recover the response sent by sem. Suppose Tom is an attacker who intercepted the message C sent by Sam to Lucy and tries to decrypt the cypher. Since he doesn’t know Sam’s key hence the types key 2496 he gets Se’=H(R,2496)=001and computes Se=C⊕Se’=100⊕001=101 and get the response and get decode(101)= “Section 1”. Hence he gets different results every time with a valid response for the confidential document. Hence he won’t be able to get the correct response sent by Sam and could only rely on guess but even with guessing he won’t be able to deduct the correct response from all the other valid responses which was sent by sam [1].

Step 3: Ciphertext is encrypted by using a randomly generated key K which is stretched using SHA-II, then the encrypted ciphertext is sent to the receiver.

Step 4: Through the key which it is encrypted is sent to the server.

Step 5: For accessing the message the user sends the ciphertext and password to the server.

Step 6: The ciphertext is decrypted using key K from the server which is again stretched using SHA-II, as a result, a binary-coded ciphertext is generated.

Step 7: The password and ciphertext received by the user is passed to DTE.

Step 8: If the password is correct then the right message is generated and if the password is wrong then the honey message is generated and passed to the user.

4. Proposed method

We are applying the honey encryption on email and for applying the honey encryption the following steps are followed:

Step 1: Sender will encode the text using HE and ciphertext is generated.

Step 2: The Seed space Se generated by HE is sent to the server.

Step 3: Ciphertext is encrypted by using a randomly generated key K which is stretched using SHA-II, then the encrypted ciphertext is sent to the receiver.

5. Implementation

As we are using Enhanced HE for increased message space on natural language messages for generating and encrypting the mail and then encrypting the cypher generated by XOR ing message and password binary seed which is then encrypted by a random key which is stretched using SHAII hashing method to generate the encrypted ciphertext. This encrypted ciphertext is then sent to the receiver.

The receiver uses this encrypted ciphertext and password to retrieve the message from the server. The server decrypt the encrypted ciphertext and stretched key to get the binary ciphertext which is used to decrypt the message from seed space using the XOR of password seed and binary ciphertext which is then sent to the receiver.

For implementation of this we are using python with flask framework for server side scripting and fernet symmetric key encryption for key encryption. For honey encryption we are using stanford dependency parser which is used for tokenization of sentences and word in the natural language message for knowing the parts of speech and then wordnet would be used for generating synt set of given words from grammar which in turn would be used for generating Fake looking real natural message.
6. Result

As such by using the methods of enhanced HE for natural message and the proposed model for encryption the email could be provided with an extra layer of security which has following benefits over current methods:

1) Real looking fake messages for wrong password.
2) By using the encrypted ciphertext the user/intruder would have no idea about the binary ciphertext. Thus increasing the number of cases for intruder even for brute attack/dictionary attack etc.
3) Even in rare cases, if the seed space is provided to the intruder it would not be possible for the intruder to know the real message from the fake message.
4) And the work for finding the real message ciphertext nor the password is known to the hacker.
5) Even if the password is known to the hacker he would not be able to get the correct real message unless the hacker knows even the encrypted ciphertext.

Hence we can say that without both the encrypted ciphertext and password it would not be easy to retrieve the real message from the Honey Encrypted message even if it is known one only guesses whether the retrieved message is real or fake.

References