Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.8 No.3, 2019, pp.19-22
© The Research Publication, www.trp.org.in
DOI: https://doi.org/10.51983/ajcst-2019.8.3.2735

Enhancement of Security for Cloud Based IoT Using XHE Scheme

K. Sailaja[1] and M. Rohitha[2]
[1]Associate Professor, Department of Master of Computer Applications,
Mother Teresa Institute of Computer Applications, Andhra Pradesh, India
[2]PG Student, Department of Electronics and Communication Engineering,
Sri Padmavati Mahila Visva Vidyalayam, Andhra Pradesh, India
E-Mail: sailaja.k20@gmail.com, rohitha9370@gmail.com

(Received 18 July 2019; Revised 10 August 2019; Accepted 2 September 2019; Available online 12 September 2019)

***Abstract* - Security is a million dollar issue for all computer systems. Every week there is news of another major breakin to a commercial or government system. Also it is well known that many governments are actively engaged in cyber-warfare, trying to break into the systems of other governments and other groups. The Internet of Things is increasingly changing into an omnipresent computing service, requiring vast volumes of knowledge storage and process. Unfortunately, due to the unique characteristics of resource constraints, self-organization and short range communication in IoT, it always resorts to the cloud for outsourced storage and computation. Security is one of the major challenges faced by cloud based IoT. The standard file protection technique relies on password-based encryption schemes and they are vulnerable to brute-force attacks. The reason is that, for a wrongly guessed key, the decryption process yields an invalid-looking plaintext message, confirming the invalidity of the key, while for the correct key it outputs a valid-looking plaintext message, confirming the correctness of the guessed key. Honey encryption helps to minimise this vulnerability. Hence, this paper proposed an extended Honey Encryption (XHE) scheme for enhancing the security of the cloud based IoT.**

***Keywords:* IoT, Cloud, Password Based Encryption, Extended Honey Encryption, Brute-Force Attacks**

I. INTRODUCTION

The future Internet will involve large numbers of objects that use standard communications architectures to provide services to end users. It is envisioned that tens of billions of such devices will be interconnected in a few years. This will provide new interactions between the physical world and computing, digital content, analysis, applications, and services. This resulting networking paradigm is being called the Internet of Things (IoT). This will provide unprecedented opportunities for users, manufacturers, and service providers in a wide variety of sectors. Areas that will benefit from IoT data collection, analysis, and automation capabilities include health and fitness, healthcare, home monitoring and automation, energy savings and smart grid, farming, transportation, environmental monitoring, inventory and product management, security, surveillance, education, and many others.

IoT is perhaps the most complex and undeveloped area of network security. Since IoT is carrying more sensitive information across the Internet, there is a need for better security. So by implementing cryptographic algorithms, the user can communicate with IOT devices in safe and secured manner [1]. Traditionally, the data encryption algorithms are considered as “computationally secure”, if the best known method of breaking the algorithms require an unreasonably large amount of computer processing time [2]. However, with the advancement of computing processors, parallelism techniques and distributed algorithms, existing IoT security that relies on the conventional password-based data encryption algorithms (e.g. Advanced Encryption Standard (AES) [3,4], RSA [5,6], etc.) are constantly at risk of being challenged and broken [7].

In computer security, honey commonly denotes a false resource designed to lure or deceive an attacker. Honeypots, for example, are servers designed to attract attackers for observation and study. Honey Encryption creates a cipher text that, when decrypted with an incorrect key or password, yields a valid-looking but bogus message, so that attackers can‘t tell when decryption has been Successful [8]. By implementing our improvised architecture, we can achieve more security compared to simple honey encryption. This paper extends the honey encryption scheme to enhance the security of the IoT, so called as extended Honey Encryption (XHE) scheme.

II. LITERATURE SURVEY

To date, various methods have been proposed to extend the HE scheme to enhance the IoT security. In this proposal, the standard HE scheme is unified with a structural code system. This proposal generates plausible false text relative to the plaintext. However, the anomaly between the plaintext and therefore the false text is way. The attacker may use this vulnerability to recover the difference between the false text and plaintext and acquire the target message. Chatterjee *et al.*, [10] proposed a Natural Language Encoder called the NoCrack. The proposal is specifically for protecting Password Vaults/Manager. The intuition is to come up with pretend however realistic-looking vault to the aggressor within the face of a brute-force attack. The attacker is not able to tell if it is the original or fake vault he has acquired. The system also forces the attacker to go online where his activities can be traced and prohibited. This proposal works reasonably well for password-related settings but cannot be extended to support large human-text.

Huang *et al.,* [11] expanded the standard Honey encryption scheme to support encoding of genomic data. This proposal suggested techniques for securing genetic materials and also protecting the genomes from mauling by an attacker with an unbounded time. This proposal uses statistical tools to increase the message space, allowing more instances of online guessing of the original vault. However, this technique does not scale well as it cannot be extended to support the human generated message [12]. Pointed out how message recovery setting in the standard HE scheme is lacking. They suggested ways of strengthening the scheme to conceal partial information of the target message while still providing security and protecting the acquired message from mauling. In this proposal, an adversary eavesdropping on their conversation at another end of the channel is supplied with valid-looking but fake chat message when he tries his incorrect keys. Yoon *et al.*,[14], proposed the visual HE which employs an adaptive DTE in a Bayesian framework. This proposal introduced a novel method of using the Bayesian framework to secure images and videos to produce fake but normal-looking videos to an adversary during transmission of images/videos. Tyagi *et al.,* [15] implemented the standard HE scheme on short messages and PINS. [18] Proposed techniques of solving typo problems in the honey encryption scheme. From our studies, all methods proposed by [9-16] worked relatively well for securing passwords.

III. HONEY ENCRYPTION SCHEME

The two main factors in this construct are the implementation of the message space where all the probable values of passwords are placed. The second factor is the Distribution transforming Encoder that encodes or decodes the message space using the specified functions. The probable values are mapped to a seed; using a specified value of n. The seeds are distributed according to the probability of the occurrence of the password. Like for more common passwords, the seeds are given a higher probability as compared to the unlikely/uncommon passwords.

A. Message Space: As defined in the paper by JR, the class Message Space Probability Functions contains a set of functions that might be used to apply Encryption. They are defined as follows:

1. *Cumulative_Distr (Message)*: gives cumulative probability defining the point where the message lies in ordered messages.
2. *Probability_Distr(Message)*: gives probability of the message that is input.
3. *Next_Message(Message)*: gives the next message in the message space
4. *Get_Inverse_Cumul_Distr_Samples()*: returns list of pre-calculated sampling of cumulative distribution values of messages.

B. Distribution Transforming Encoder: The DTE is constructed keeping the message distribution in consideration. The Encoding yields a “seed” value distributed uniformly. The seeds are mostly taken to be binary strings. The Encoder needs to have a decoder as well which when provided with the seed returns the text message. Encoding is a two-step process called DTE-then-encrypt.

1. The DTE is applied to Message to obtain seed.
2. The obtained seed is encrypted using cipherkey that will give HE Cipher-text.

a. *Loop Hole:* The security of this Encryption relies on the probability that is defined by the Encrypting Party. If by any means this probability is not calculated properly, the method fails. So in cases when the format or distribution of plain text is unknown or there is a large space of plain-texts, HE can’t be applied. Now keeping these factors in mind, the predictability is judged. So before applying this method, the plain-text needs to be monitored and then it needs to be mapped in a large space where all the outputs look plausible and match the likelihood of legitimacy.

b. *Target Area:* The security provided by this encryption is best applicable for places having low-entropy. The developers, JR, proposed this scheme in the context of passwords. They may include generic alphabets, credit cards or plain text messages.

IV. EVALUATION

The platform for evaluating our honey encryption system is the Toshiba Portege-M800 laptop. The processor is Intel Core 2 Duo 2.0 Hz. The memory has a 3 GB RAM. The operating system is Ubuntu Kylin 16.04. The goal of experiments is to study the time taken to encrypt and decrypt a message. In order to make it easy to increase the size of the message space for multiple times, we choose the password message space for evaluation and increase the size from 10^6 to 10^8 .

A. Time to Encrypt a Message

For encryption in a large message space, DTE should read the message space file line by line, calculate the PDF and CDF, determine the seed range, and randomly select a seed from the range. Finally, the chosen seed is XORed with the key to obtain the ciphertext. We extend the message space size from 10^6 to 10^8 and conduct an evaluation. The time to encrypt a message is measured and displayed in Fig.1. The x-axis presents the message location in the message space. For example, 0.25 stands for the message that is located at 25% of the message space. The y-axis represents the time taken to encrypt a message. It can be observed from the figure that the encryption time increases as the location of the message moves deeper. This is because the encryption algorithm reads the message space line by line until it finds the message to get the probabilities. The larger the message space, the more time it needs for encryption because the most time-consuming work in this encryption is reading and processing the message space. For the message space 10^6 or 10^8 that contains or messages, the time is reasonable, but for a message space of 10^8 messages, the maximum time to encrypt a message can be as high as 70 s, which is too high.

B. Time to Decrypt a Message

During the decryption process, DTE first XORs the key with the ciphertext and obtains the seed. Then it determines the location of the seed in the seed space. Using the location information, it looks up the inverse table and gets the corresponding plaintext message. We measure the time to decrypt a message in three message spaces, ranging from to, and display these statistics in Fig.2. The x-axis stands for the location of a plaintext message in the inverse table, and the y-axis represents the time to decrypt the message. As shown from the fig.2, the decryption time increases as the plaintext message location in the inverse table goes deeper. This is because the decryption algorithm reads line by line the inverse table file until it finds the plaintext message. The larger the inverse table, the slower the decryption process because the most time-consuming part in this decryption is to process the inverse table file. When the inverse table size is10^6, the time to decrypt a message is acceptable, but when the inverse table size is 10^8, the time can reach 160 s. Comparing Fig.1 and Fig. 2 it can be seen that the time to decrypt and encrypt a message is different because the message space file only contains a message in one line, but the inverse table file contains a message and its cumulative probability for each line. Thus processing the latter takes more time.

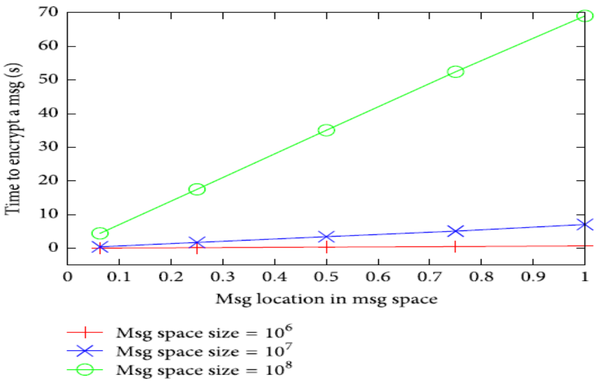

Fig. 1 encryption

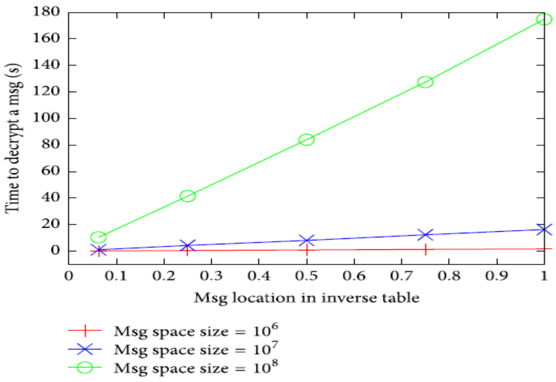

Fig. 2 Decryption

V. ENHANCEMENT

For a large message space, the decryption algorithm needs to read the inverse table file line by line and find the correct plaintext message using the calculated cumulative probability. For a small message space, we can read the whole inverse table into the memory and use the binary search method to find the corresponding plaintext message in the decryption process.

For a large message space, the encryption algorithm needs to read the message space file and determine the message's PDF and CDF. But if the message space is incrementally sorted like the password message space, the value of the message, has a relationship with its location, in the message space; that is, Also, the cumulative probability is related to the message location in the message space; that is, A=V+1.CDF = A/N, where is the number of messages in the message space. Therefore, instead of searching the message space file for CDF, we can calculate the CDF. It should be noted that not all message spaces have such features. Taking the identification number, for example, the CDF of a message is not related to the value of the message itself. We improve the encryption and decryption algorithm and evaluate their performance.

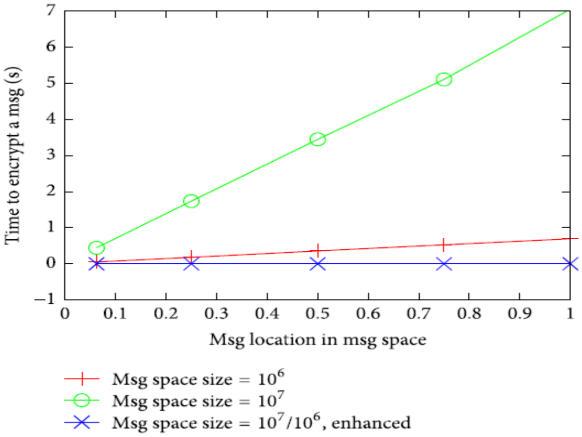

Fig. 3 Encryption enhanced

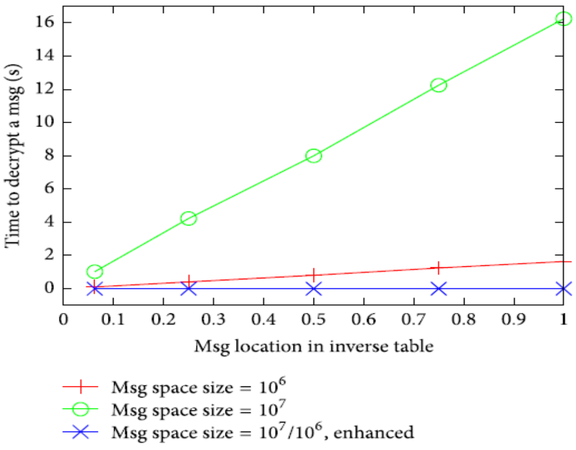

Fig. 4 Decryption enhanced

Fig. 3 shows the encryption time of the enhanced encryption algorithm. For 10^6 and 10^8 and message spaces, no matter where the message is located, the encryption time is only around 136 microseconds. The lines for both 10^6 and 10^8 and message spaces overlap. This means the encryption time is independent of the message space size.

Fig.4 shows the time taken for the enhanced decryption algorithm. No matter where the message is located in the message space, the decryption time is around 45 microseconds. The lines for both 10^6 and 10^8 and message spaces overlap, which means the decryption time is independent of the message space size. This may be because the difference of the binary search algorithm in the 10^6 and 10^8 and message spaces is not significant. We tried the 10^8 message spaces to verify this case, but the system fails due to memory error because the available memory is too small to hold the inverse table file containing 10^8 messages.

VI. CONCLUSION

Security is one of the major challenges faced by IoT industry. In future most of the standard encryption algorithms that exist today can be broken within seconds. Hence by implementing this type of encryption technique, both security and integrity is preserved. While the existing file protection relies on password-based encryption, which is vulnerable to password guessing attack such as brute-force, dictionary or rainbow table attack. The recent development of honey encryption offers many password based security schemes resilience to brute force offline attacks by yielding plausible plaintexts under decryption by invalid keys. This paper proposed an extended Honey Encryption (XHE) scheme for securing the IoT. The proposed XHE scheme provides an additional protection layer to existingencrypted file. When the attacker attempts to access the encrypted data with his guessing password, instead of rejecting their data access as conventional file encryption scheme, the extended HE algorithm generates an indistinguishable bogus file that are closely related to the original file. It is noticeable that the message space of the proposed scheme is pre-fixed and the complexity and the size of inverse sampling tables is growing exponentially with the increase of the file names and its extension sizes.

In future, several aspects of this work can be furtherexplored, such as working with a flexible message space and further extends into folders protection. Besides that, whether the proposed XHE scheme can be further adapted to work with the recent advancement of cryptography algorithm such as Homomorphic Encryption for supporting the computation on encrypted data, as well as Attribute-Based Encryption (ABE) to fine grained control access on encrypted data are another interesting topic to be explored.

REFERENCES

[1] Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and SiobhánClarke, "IEEE, Middleware for Internet of Things", *IEEE Internet of Things Journal*, Vol. 3, No. 1, pp. 70 - 95, 2016.

[2] W. Diffie and M.E. Hellman, "New Directions in Crytography", *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644 - 654,*IEEE Press, New Jersey, 1976.*

[3] G. Irazoqui, M.S. Inci, T. Eisenbarth and B. Sunar, "Wait a Minute! A Fast, Cross-VM Attacks on AES", *LNCS, Springer, Switzerland,* Vol. 8688, pp. 299-319, *2014.*

[4] Y. Wei, J. Lu and Y. Hu, "Meet-in-the-Middle Attack on 8 Rounds of the AES Block Cipher under 192 Key Bits. LNCS", *Springer, Heidelberg,* Vol. 6672, pp. 222-232, *2011.*

[5] A. Nitaj, M.R.K. Ariffin, D.I. Nassar, H.M. Bahig, "New Attacks on the RSA Cryptosystem. LNCS, Progress in Cryptology – AFRICACRYPT", LNCS, *Springer, Swtizerland*, Vol. 8469, pp. 178-198, 2014.

[6] Y. Lu, L. Peng, S. Sarkar, "Cryptanalysis of an RSA variant with Moduli N = prq", *In: 9thInternational Workshop on Coding and Cryptography 2015 WCC2015, Apr 2015, Paris, France. 2016.*

[7] S.F. Tan and A. Samsudin, "*Enhanced Security for Public Cloud Storage with Honey Encryption", Advanced Science Letters. Accepted Manuscript*.

[8] A. Juels and T. Ristenpart, "Honey Encryption: Security beyond the Brute-Force Bound,‖ Advances in Crypto logy—Euro crypt 2014", *LNCS 8441*, *Springer*, pp. 293–310, 2014

[9] H. Jo and J. Won, "A new countermeasure against brute-force attacks that use high-performance computers for big data analysis",Hindawi Publishing Corporation, *International Journal of Distributed Sensor Networks*, pp. 7, 2015. [Online] Available at: http://dx.doi.org/ 10.1155/2015/406915.

[10] R. Chatterjee, J. Bonneau., A. Juels and T. Ristenpart, "Cracking Resistant Password Vaults using Natural Language Encoders," *Proceedings – IEEE Symposium on Security and Privacy*, No. 7163043, pp. 481-498, July 2015.

[11] Z. Huang, E. Ayday, J. Fellay, J. Hubaux and A. Juels, "Genoguard: Protecting genomic data against brute-force attacks," *IEEE Symposium on Security and Privacy*, pp. 447-462, 2015. DOI 10.1109/SP.2015.34.

[12] J. Jaeger, T. Ristenpart and Q. Tang, "Honey encryption beyond message recovery security," *International Association for Cryptologic Research, Fischlin and J.-S.Coron (Eds.): EUROCRYPT 2016, Part I, LNCS 9665*, pp. 758–788, 2016. DOI: 10.1007/978-3-662-49890-3 29.

[13] J. Kim and J. Won, "Honey chatting: A novel instant messaging system robust to eavesdropping over communication," *IEEE In Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2184-2188, 2016.

[14] J.W. Yoon, H.S. Kim, H.J. Jo, H.L. Lee, and K.S. Lee, "Visual honey encryption: Application to steganography," *in Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, New York, NY, USA, 2015, IH & MM Sec '15,* pp. 65–74, ACM.

[15] N. Tyagi, J. Wang, K. Wen and D.Zuo, "Honey Encryption Applications. 6.857 Computer and Network Security", *Massachusetts Institute of Technology*. [Online] Available at: http://www.mit.edu/~ ntyagi/papers/honey-encryption-cc.pdf, 2015

[16] M. Golla, B. Beuscher and M. Durmuth, "On the security of cracking resistant password vaults," *Proceedings of the ACM Conference on Computer and Communications Security*, Vol. 24, No. 28, pp. 1230-1241, Oct. 2016.

[17] R. Chatterjee, A. Athalye, D. Akhawe, A. Juels, and T. Ristenpart, "Password typos and how to correct them securely," *In Security and Privacy (SP), 2016 IEEE Symposium*, pp. 799–818, 2016.

[18] H. Choi, H. Nam and J. Hur, "Password Typos Resilience in Honey Encryption," *IEEE Symposium. The 31st International Conference on Information Networking (ICOIN 2017)*, pp. 593-597, 2017.