International Journal of Scientific Engineering and Science
Volume 6, Issue 1, pp. 35-38, 2022.
ISSN (Online): 2456-7361

Privacy of Friendship on Social Media: Application of Honey Encryption

Solomon SARPONG
Department of Biological, Physical and Mathematical Sciences, University of Environment and Sustainable Development, Somanya, Ghana

Abstract— *In recent times, the need to make friends (online or offline) by individuals seems irresistible. However, some people use the type of friends a person keeps as a measure of the character/behavior of that person is. In recent times, there has been the use of friends-of-friends to locate and identify individuals who hitherto would not have been known. Some persons (say security agencies) use the type of friends an individual has to classify the individual into groups. Can a person have friends on social media without any user of that social media knowing? This has become the dilemma of some persons on social media platforms. This then brings to the fore the need to be able to hide friends on social media platforms. This paper proposes algorithms that when incorporated on a social media platform will keep the friends of a User secret/hidden. The algorithm makes use of honey encryption techniques to help persons keep their friends secret/hidden on social media. The usage of this algorithm ensures security and privacy for the Users.*

Keywords— *Honey encryption, Distribution-transforming encoder, Social media.*

I. Introduction

A human being is a social entity. This characteristic prompts people to make friends in the physical and or on social media. Since the advent of social media platforms (example Facebook and Instagram etc.), the ease with which individuals make friends has increased. Adding to the ease of making friends easily, some of these social media platforms provide dating or matchmaking services (Okcupid, Tinder, Adult Friend Finder) to the users. Research have also proposed matchmaking protocols [1-5] where people can make friends based on the similarities in their attributes. All these help the users to make friends and find partners. Even though, these social media platforms ensure some form of security to the users against attacks, it is breached sometimes with its dire consequences. The form of security provided on most of these platforms are password-based which when broken reveals the protected information.

On these social media platforms, in as much as individuals see the need to make friends or be in relationships, sometimes there is the need to keep these relationships private. How can a person keep a relationship private/hidden on social media? It is in lieu of this that this paper is proposing the use of honey encryption (HE) to keep such relationships private and hidden from other social media users. The paper proposes algorithms that will make a person keep some relationships or friends out of public domain.

The rest of the paper consists of introduction (overview of honey encryption and hash functions) in section I. Related research is in section II. The proposed protocol is in section III. The security of the protocol and conclusion are in sections IV and V respectively.

A. Honey Encryption

Honey encryption (HE) is a cryptographic encryption for which decrypting a ciphertext with any number of invalid keys yields fake but plausible plaintext [6]. HE is resilient against cryptographic attacks such as brute force attacks. In HE, even though the decryption key maybe invalid, it outputs a valid-looking message. This property makes attackers gain no information by guessing the password [7]. The nature of HE makes it adequate for use in situations where the resources available for full scale encryption protection is limited making the information prone to brute-force attack. HE can also provide a safeguard against restricted disclosure as a result of its high min-entropy keys [8]. The application of honey encryption was proposed by [9].

In cryptography, "honey" signifies deception, decoy or false resource. Honey tokens are decoy objects that when used signifies that the system has been compromised. In checking for situations of password database compromise, honeywords are used. When honeyword(s) is used in a log-in attempt, it signifies a compromise of the password protecting the database. A honeypot is a decoy system, once it is attacked; it stores information on the attacker. This stored information is then used to prepare for and prevent real attacks in the future.

B. Set-up Scheme for Honey Encryption

Conventionally, HE has a message space M that contains messages $m \epsilon M$ that is mapped to a seed space S through a distribution-transforming encoder DTE. There is a direct proportionality between the size of the seed range, *m*, and the feasibility of $m \epsilon M$. For the DTE to be functional, the cumulative distribution function (*cdf*) of M and some information on its ordering of message must be known. The seed space must be such that, even a message with the least likelihood of occurrence is assigned a seed.

C. Limitations of HE

In as much as HE provides extra security in protecting information from attackers, its limitations include; (i). the security of HE may be compromised if an attacker has inside information about the target messages; (ii). decryption with any password produces a valid-looking message, hence a correct password with typographical errors maybe confusing to a legitimate user; and (iii). the message distribution should be independent. Any dependence between message distribution and decryption key can enable an attacker to know the correct message. This also applies to the case of encryption.

However, a breakdown of these limitations will make the HE a password-based encryption.

Password-based encryption (PBE) is a technique for creating strong cryptographic keys based on passwords provided by the user(s). The robustness of password-based encryption depends on among other things, the length of the key. PBE may be secured but with enough resources, their security can be compromised [10]. Hence, PBEs for data protection are vulnerable to brute-force attacks [7]. This is because when an invalid password is used for decryption, it results in an invalid-looking message confirming that the password is not correct. This enables the attacker to keep trying until a valid-looking plaintext is achieved.

In order to make the PBE less prone to brute-force attack, hashing is added to enhance the security. Even though hashing provides some form of security, the message digest can be compromised by using; (i). dictionary and Brute force attacks; (ii). lookup tables and (iii). rainbow tables.

D. Cryptographic Hash Functions

A cryptographic hash function is a mathematical function that can convert an input of any arbitrary length to an output of a fixed length. The output of a hash function is called message digest or simply hash value. A cryptographic hash function is a one-way function that is easy to generate a hash value from this one-way function but difficult to generate input from the output. A cryptographic hash function must satisfy these criteria; (i). input message of any length outputs a fixed length of hash value – that is, cryptographic hash function maps arbitrary binary strings into binary strings of fixed length: $H: \{0,1\}^* \rightarrow \{0,1\}^n$ such that $x \longmapsto H(x)$; (ii). efficient and fast to compute; (iii). Computationally infeasible to invert – given $y \in \{0,1\}^n$, it is difficult to find $x \in \{0,1\}^*$ such that $H(x) = y$; (iv). should be strongly collision free – that is, it is difficult to find distinct $x, x' \in \{0,1\}^*$ such that $H(x) = H(x')$ and given $x \in \{0,1\}^*$, it is difficult to find $x' \in \{0,1\}^*$, $x' \neq x$ such that: $H(x) = H(x')$; and (v). must possess avalanche effect.

The message digest of a hash function can be computed using some of these techniques: (i). division-remainder; (ii). mid-square; (iii). digit folding; and (iv). radix transformation. In this paper, the digit folding method will be adopted. In the digit folding method, the input password (numeric) is grouped in twos. These groups are added up to represent the hash of the input password. If the input password is a word or phrase, numbers are assigned to the alphabets. These numbers are then grouped in twos. The grouped numbers are added up to represent the hash value of the input password.

II. Related Research

[11] Designed and implemented a honey encryption mechanism and applied it to some data. [12] applied honey encryption concept to MANETs with the aim of preventing adhoc networks from launching brute-force attack. In their paper, [7] proposed the use of HE to protect Credit card numbers and text messages. [13] and [6] used a form of distribution-transforming encoder (DTE) for encryption and decryption. Application of HE to images and videos was proposed by [14].

[15] applied honey encryption in cloud computing. [16] modified the HE scheme to support the encoding of human-generated message by leveraging natural language processing techniques. [17] implemented honey encryption for securing public cloud data storage. [18] proposed fake data generating algorithm (FDGA), a modified honey encryption for providing security to information. [19] proposed integrated attribute-based encryption with the honey encryption on Hadoop.

[20] observed that, there is no available evidence indicating that users choose significantly stronger master passwords. The addition of security questions to the honey encryption to enhance the security was proposal by [21]. There was the usage of honey encryption by [22] in the processing of natural language. [23] proposed the usage of honey encryption for the protection of files. [24] applied honey encryption to protect patient's information from unauthorized users.

The use of honey encryption and structural coding scheme were proposed by [25] to preserve the security and privacy of information against cryptanalysis such as side-channel analysis and brute-force attacks. Using honey encryption, [26] proposed three types of protocols that enable legitimate users detect when a password contains typographical errors. Their protocols are resilient against online and offline brute-force attacks.

III. Proposed Protocol

The proposed protocol in this paper is aimed at helping keep friends on social media platforms hidden from public view. That is, friends a person has but does not want others on the social media to know about can securely be hidden using this proposed protocol.

A. Setup for Hiding Friends

To let this feature work on the social medium platform (e.g. Facebook), two bottoms or icons (*Hide* and *Display*) will be incorporated on the platform. The individual who wants to hide some of the friends from being public needs to activate the *Hide* bottom or icon by clicking it.

When the *Hide* friends bottom or icon is clicked, a pop-up menu will come up with two input spaces provided. The *User* will need to fill in the names of the *Real* friends s/he wants to hide from public view in the designated space. Another space is provided to be filled with names of people who are not friends, *Fake* friends. Care should be taken such that, there is no name(s) in the intersection of the names in the two sections. When done, the *User* clicks on the upload bottom. The *User* is prompted to input the password. The list of friends (*Real* and *Fake*) is encrypted and uploaded. These encrypted names are stored separately in the seed space in the DTE.

When a password is input, the hash of the password is computed using the digit folding technique in the DTE. To encrypt and store the names in the *Seed Space*, the following algorithms are executed;

```
$Encrypt(password,\ names_i)$
Salt $\leftarrow \{0,\ 1\}^{128}$
$K \leftarrow H^c(password||salt)$
```

$C_i \leftarrow K \oplus name_i$

The ciphertext of the names in the list of friends (real and fake) are stored separately in the seed space. Figure 1 shows the schematic representation of the algorithm for *hiding* friends.

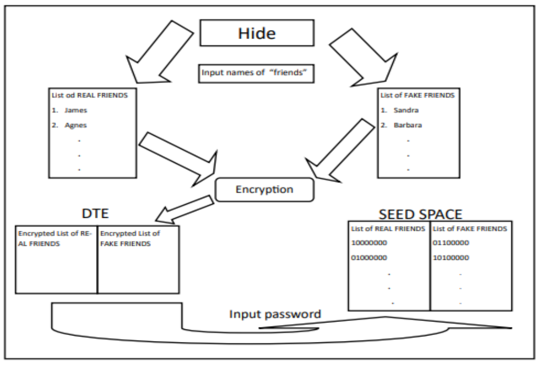

Fig. 1. Schematic representation of the algorithm for *hiding* friends

B. *Setup for Unhiding Friends*

In order to see hidden friends, the *User* needs to click the *Unhide* friends icon and input the password. The input password undergoes fold and add hashing. If the password is correct, the seed space maps to the list of *Real* friends in the DTE. The *User* is further prompted to input a second the password. If the password is correct the DTE displays the list of *Real* friends. The algorithm to *unhide* friends is depicted in Figure 2.

On the other hand, if the password is not correct after the fold and add hashing the list of *Fake* friends in the seed space maps to the list of *Fake* friends in the DTE. When the second password is input, the DTE displays the list of *Fake* friends. In the process of *Unhiding* friends, if the first password is incorrect, the process of displaying *Fake* friends starts and cannot be reversed. Hence, a correct second password will still lead to the display of the list of *Fake* friends. Furthermore, even if the first password is correct but an incorrect second password will also lead to the display of the list *Fake* friends.

To decrypt to *Unhide* the names in the friends (*Real* or *Fake*), the following happens;

$Decrypt(password^*, salt, C_i)$

$K \leftarrow H^c(password^*||salt)$

$names_i \leftarrow K \oplus C$

The list of *Real* friends is displayed if $password^* = password$ (used in the encryption). The list of *Fake* friends is displayed if $password^* \neq password$.

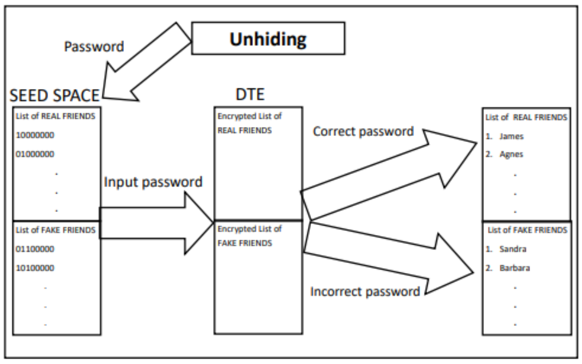

Fig. 2. Schematic representation of the algorithm for *unhiding* friends

C. *Interaction within the Group of Hidden Friends*

The individual who formed the group serves as the trusted third party. In this set up, the trusted third party will be called the *Admin*. The *Admin* generates an RSA keypair (e_{Admin}, d_{Admin}); an identity, ID_{Admin}, and publicizes (e_{Admin}, ID_{Admin}) in the group of hidden friends. Each member also generates an RSA keypair (e_{m_i}, d_{m_i}) and shares with the *Admin* e_{m_i}. The *Admin* sends to each *User* $Encrypt_{e_{m_i}}(ID_{Admin}||C_i)$.

In this protocol, the members of the group of hidden friends can see and read the social media posts of other members who are not in that group however, they cannot comment or put-up posts. The members of the group of hidden friends do not communicate directly among themselves. All communications go through the *Admin*. When the *Admin* wants to communicate with a *User A*, the *Admin* sends $Encrypt_{e_{m_A}}(ID_{Admin}||Message||C_A)$ to the *User*. When *User A* receives and decrypts it, the *User* will be able to read the message. The *User A* can also communicate with the *Admin*. For a *User* to communicate with the *Admin*, the *User* sends $Encrypt_{e_{Admin}}(C_A||Message||ID_{Admin})$ to the *Admin*. On receiving and decrypting the message, the *Admin* will know who sent the message and the content of the message received. The *Admin* can link two or more members of the hidden group but all communication among them will be through the *Admin*.

IV. SECURITY

This paper has proposed techniques that can help a person with friends on social media hidden so that, the friends will not be known by other social media users. That is, the identities of the friends (*True* and *Fake*) are encrypted before storing in the seed space. Even if the seed space is compromised, the identities of the friends are not in plaintext making it difficult to know who they are. In order to further enhance the security in the protocol, there is the use of two passwords in the HE.

In the process of *Unhiding* friends, if the first password is incorrect, the process of displaying *Fake* friends starts and cannot be reversed. Hence, a correct second password will still lead to the display of the list of *Fake* friends. Furthermore, even if the first password is correct but an incorrect second password will also lead to the display of the list *Fake* friends. In order to display the list of *Real* friends, the two passwords must be correct. This measure ensures the prevention of the disclosure of the identities of the list of *Real* friends by an attacker.

Furthermore, the use of honey encryption adds another layer of security. The use of honey encryption will prevent an adversary from succeeding when brute-force attack is undertaken. Even if an adversary undertakes brute-force attack, the identities obtained will be a list of fake friends. The honey encryption ensures that even if a wrong password is used, a list of fake identities will be obtained but it will not be the identities of the true friends. Also, when the honey encryption breaks down, it results in password-based encryption. Hence, an attacker will need a lot of resources to be able to obtain the actual identities of the true friends when this proposed protocol is used.

In order to prevent semi-honest attacks, the real friends do not know the identities each other. A semi-honest *User* can leak the identities of the other persons in the group. This is further enforced by preventing the *Users* from communicating with each other. Hence, all communications are done through the *Admin*.

V. Conclusion

The need for keeping friends private has become important in this era of social profiling where an innocent person can become a subject of interest by security agencies by virtue of association with "bad persons". Hence, the proposed protocol in this research paper when incorporated in an existing social media platform can enable a *User* make friends without any other *User* knowing. The proposed algorithm uses a combination of honey encryption and hash function. This has resulted in high security and privacy in the proposed algorithm. Hence, the incorporation of this algorithm on a social media platform will enable the *User* achieve privacy and security of the information.

References

[1] S. Sarpong, "Privacy-Preserving Zero Knowledge Scheme for Attribute-based Matchmaking," 2021.

[2] S. Sarpong and C. Xu, "Privacy-preserving attribute matchmaking in proximity-based mobile social networks," *Int. J. Secur. its Appl.*, vol. 9, no. 5, pp. 217–230, 2015, doi: 10.14257/ijsia.2015.9.5.22.

[3] Y. Wang, H. Li, T.-T. Zhang, and J. Hou, "A Privacy Preserving Matchmaking Scheme for Multiple Mobile Social Networks," in *Algorithms and Architectures for Parallel Processing*, 2013, pp. 233–240.

[4] S. Sarpong, C. Xu, and X. Zhang, "PPAM: Privacy-preserving attributes matchmaking protocol for mobile social networks secure against malicious users," *Int. J. Netw. Secur.*, vol. 18, no. 4, pp. 625–632, 2016.

[5] Y. Wang, J. Hou, Y. W. Tan, and X. Nie, "A recommendation-based matchmaking scheme for multiple mobile social networks against private data leakage," in *Procedia Computer Science*, Jan. 2013, vol. 17, pp. 781–788, doi: 10.1016/j.procs.2013.05.100.

[6] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," *Proc. ACM Conf. Comput. Commun. Secur.*, no. November, pp. 145–159, 2013, doi: 10.1145/2508859.2516671.

[7] N. Tyagi, J. Wang, K. Wen, and D. Zuo, "Honey Encryption Applications," *Netw. Secur.*, no. May, 2015.

[8] A. Juels and T. Ristenpart, "Honey encryption: Security beyond the brute-force bound," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8441 LNCS, pp. 293–310, 2014, doi: 10.1007/978-3-642-55220-5_17.

[9] H.-J. Jo and J. Won Yoon, "Poster: Statistical coding scheme for the protection of cryptographic systems against brute-force attack," *Proc. 35th IEEE Symp. Secur. Priv.*, pp. 3–4, 2015.

[10] Z. Huang, E. Ayday, J. Fellay, J. P. Hubaux, and A. Juels, "GenoGuard: Protecting genomic data against brute-force attacks," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2015-July, no. June, pp. 447–462, 2015, doi: 10.1109/SP.2015.34.

[11] W. Yin, J. Indulska, and H. Zhou, "Protecting Private Data by Honey Encryption," *Secur. Commun. Networks*, vol. 2017, 2017, doi: 10.1155/2017/6760532.

[12] V. P. P and N. M. A, "Avoiding Brute Force attack in MANET using Honey Encryption," 2013. Accessed: Apr. 13, 2021. [Online]. Available: www.ijsr.net.

[13] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers, "Format-preserving encryption," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5867 LNCS, pp. 295–312, 2009, doi: 10.1007/978-3-642-05445-7_19.

[14] J. W. Yoon, H. Kim, H. J. Jo, H. Lee, and K. Lee, "Visual honey encryption: Application to steganography," in *IH and MMSec 2015 - Proceedings of the 2015 ACM Workshop on Information Hiding and Multimedia Security*, 2015, pp. 65–74, doi: 10.1145/2756601.2756606.

[15] S. D. Ulhe, "Security in Cloud Computing using Encryption .," no. August, pp. 24–27, 2016.

[16] A. E. Omolara and A. Jantan, "Modified honey encryption scheme for encoding natural language message," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 3, pp. 1871–1878, 2019, doi: 10.11591/ijece.v9i3.pp1871-1878.

[17] E. Mok, A. Samsudin, and S. F. Tan, "Implementing the honey encryption for securing public cloud data storage," *COMPSE 2016 - 1st EAI Int. Conf. Comput. Sci. Eng.*, no. January, 2017, doi: 10.4108/eai.27-2-2017.152270.

[18] S. Padhye, R. A. Sahu, V. Saraswat, S. Padhye, R. A. Sahu, and V. Saraswat, "Hash Function," *Introd. to Cryptogr.*, pp. 143–163, 2020, doi: 10.1201/9781315114590-8.

[19] G. Kapil, A. Agrawal, A. Attaallah, A. Algarni, R. Kumar, and R. A. Khan, "Attribute based honey encryption algorithm for securing big data: Hadoop distributed file system perspective," *PeerJ Comput. Sci.*, vol. 2020, no. 2, pp. 1–31, 2020, doi: 10.7717/peerj-cs.259.

[20] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, "Cracking-resistant password vaults using natural language encoders," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2015-July, pp. 481–498, 2015, doi: 10.1109/SP.2015.36.

[21] D. N. U. Rani, S. N. Ahmad, C. H. K. Reddy, and P. J. Lakshmi, "Honey Maze Encryption (Home)," *SSRN Electron. J.*, pp. 1–4, 2018, doi: 10.2139/ssrn.3173517.

[22] M. Paliwal, "Natural Language Processing by Enhanced Honey Encryption Technique," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12S, pp. 159–163, 2019, doi: 10.35940/ijitee.l1048.10812s19.

[23] P. Khandelwal, "The Research on File Encryption Method Based Honey Encryption Technique," vol. 5, no. 10, pp. 905–906, 2017.

[24] A. Esther Omolara, A. Jantan, O. I. Abiodun, H. Arshad, K. V. Dada, and E. Emmanuel, "HoneyDetails: A prototype for ensuring patient's information privacy and thwarting electronic health record threats based on decoys," *Health Informatics J.*, vol. 26, no. 3, pp. 2083–2104, 2020, doi: 10.1177/1460458219894479.

[25] H. J. Jo and J. W. Yoon, "A new countermeasure against brute-force attacks that use high performance computers for big data analysis," *Int. J. Distrib. Sens. Networks*, vol. 2015, 2015, doi: 10.1155/2015/406915.

[26] H. Choi, J. Jeong, S. S. Woo, K. Kang, and J. Hur, "Password typographical error resilience in honey encryption," *Comput. Secur.*, vol. 87, 2019, doi: 10.1016/j.cose.2018.07.020.