Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 129 (2018) 263–269

Procedia Computer Science

www.elsevier.com/locate/procedia

2017 International Conference on Identification, Information and Knowledge in the Internet of Things

Privacy Preservation for Dating Applications

Weicheng Wang[a], Shengling Wang[b,*]

[a]Beijing Normal University, Beijing, China, 201411212045@mail.bnu.edu.cn
[b]Beijing Normal University, Beijing, China, wangshengling@bnu.edu.cn

Abstract

Dating applications can satisfy the social contact needs of different users and become tools for developing a social relationship. However, the privacy leakage has turned into an insurmountable obstacle to the market success of social applications. Existing privacy protection schemes for social applications either introduce untrusted third parties or sacrifice information accuracy. In this paper, we put forward the privacy protection mechanism based on zero knowledge. In detail, the server knows nothing about the users information, but can still provide accurate services to users. We also analyze the potential attack methods and propose the corresponding solutions. Our simulation results verify the effectivity of our scheme.

Keywords: privacy preservation; social applications; zero knowledge; homomorphic encryption algorithm; service accuracy

1. Introduction

With the growing popularity of smart-devices, like mobile phone, laptops, tablet, the dating applications gradually play an essential role in people's daily life. The dating applications mostly satisfy the social contact needs of different users and becoming tools for developing a social relationship. They can provide services for people without the limitation of distance. People can find friends who have the same interests and hobbies in the world and communicate whenever and wherever they want.

As for dating applications, the primary purpose is to help users find friends based on the information provided by the users. According to the user's interests and location, the dating applications can recommend suitable people to the users. However, once the user sends his query to the server, it is possible that his information is hacked on the Internet. Furthermore, even if the encryption algorithm is perfect, the malicious people can still act like a user to acquire information. If their information is leaked to the social applications, the users will stop using the social

* Corresponding author: Shengling Wang
E-mail address: wangshengling@bnu.edu.cn

10.1016/j.procs.2018.03.074

applications. When there is a large number of users who stop using it, it is easy to cause the collective panic and the trust crisis on social applications.

Existing privacy protection schemes for social applications either introduce untrusted third parties or sacrifice information accuracy. In the scheme of introducing a third party, the third party acting as the agent of the users submits the query request to the server, and returns the query result to the user. In this case, the privacy information scattered in various service providers is concentrated in a few third parties. The risk and scale of privacy leakage are increased sharply. Besides, with the help of analysis related to the big data, it is still possible that the privacy information which is obscure can be inferred or restored.

Our project puts forward the privacy protection mechanism based on zero knowledge. This mechanism solves the problem of the existing privacy protection scheme which protects privacy at the expense of the accuracy of information. In our schemes, the server knows nothing about the user's information, but can still provide to users with accurate services.

In summary, the main contribution of our paper are as follows:

1. We propose the privacy protection scheme in social applications including the communications processes between several entities and data structures.
2. We propose the encryption algorithm based on Homomorphic Encryption and grouping attributes. In this way, the users' information is protected well.
3. We analyze the potential attack methods and propose the corresponding solutions.
4. We do the simulations and verify the effectivity of our scheme.

The remainder of the paper is organized as follows. Section 2 is for the related work. A summary of the framework is presented in Section 3. Section 4 introduces the detailed process of the algorithms including encryption, decryption, and matching. Section 5 reports the results of our numerical simulations. Finally, our conclusions are summarized in Section 6.

2. Related work

In the social network system, privacy is a rather important issue[5, 6]. There are several methods achieving privacy-preserving, which are based on k-anonymity model. The basic idea of k-anonymity is to remove some features such that each item is not distinguishable among other k items. For data protection, it protects data at the cost of the original data quality. For location-based service, it realizes the privacy protection through blurring user's location [1, 2, 3].

However, k-anonymity model exits a problem of accuracy lose and attackers can still perform trajectory attacks. It may be not suitable in situations where the requirement of information accuracy is extremely high. In addition, k-anonymity does not protect users when they are in a densely populated area.

Moreover, some existing ways of protecting privacy is often introducing a third party[4]. All of the keys are stored in the third part. The information should be encrypted before sent to the server and decrypted before the server sends back. However, in the solution of introducing a third party, the privacy information scattered in various service providers will be concentrated. We can't guarantee the third party is able to protect those keys and the information.

In general, a number of methods for privacy protection on the social network have been put forward[13, 14, 15] such as: bipartite graph[11], data migration[12], disrupting edge weights based on Gauss random multiplicative[7], information confusion[8], etc. But there are still many questions which need to be solved.

3. Our Framework

The whole purpose of our framework is to response a user with the accurate social information according to his query while without leaking any query information to the dating application system. Query information is often sensitive because it is related to user's privacy. It always contains a user's personal information such as habits, hobbies, and preferences. Hence, our framework aims to realizes zero-knowledge privacy protection. Here, *zero-knowledge* implies the server knows nothing. Therefore, our aim is challenging, because on the one hand, the server needs information from the user to provide services, and on the other hand, such information needs to be kept private to the server.

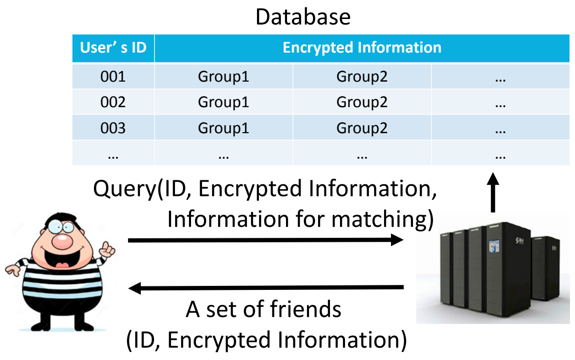

Fig. 1: The proposed privacy preservation framework for social applications.

To realize the seemly conflict aims, we propose a privacy preservation framework for dating application systems. The preservation is described in the following steps. The framework is shown in Figure 1.

1. The dating application server broadcasts all users three pieces of information: a) the attributes they can submit; b) how to group the attributes; c) the part of keys to decrypt users' information, each of which corresponds to one attribute group's value and hence different attribute groups' values have different keys. The reason why part of a key is related to users' attribute value is that this will make users who have the same attribute group values can decrypt the corresponding attribute values of the recommended friends.
2. As required by the server, the user's information like name, age, hobbies, etc.. are divided into several groups. For each group, the information is encrypted using Homomorphic Encryption Algorithm. Then user's ID attached to all of the encrypted information will be sent to the server. The key for encryption consists of two parts. The first part corresponds to the user's attributes' values, which is obtained from the broadcast message sent from the server and the other is selected by the user freely. Hence, the key of each attribute group's value is different. Besides, our framework only provides two matching patterns: Matching with Strict and Matching with Loose.
3. Once receives the information from a user, the server saves his information and finds friends who have the same interests with the user using the Matching Algorithm based on Homomorphic Encryption Algorithm in the database. It is worth noting that because all of the information in the server is encrypted, the user's privacy will be protected.
4. The server returns the recommendations'(recommended friends) ID and information encrypted to the user. The user can select suitable people and get a connection with them using their ID. Moreover, the user can decrypt part of friends' information.[1] This can be achieved because when a group of information of two users is the same, the corresponding keys of two users will be the same. These are related to our matching pattern, especially Loose pattern. How to deal with it will be introduced in next section in detail.

4. Our Algorithm

In order to realize zero-knowledge privacy protection, our framework adopts algorithms based on Homomorphic Encryption[9, 10, 11]. Homomorphic Encryption allows complex mathematical operations on cipher-texts, generating an encrypted result which, matches the result of the operations as if they had been performed on the plaint-text and then be decrypted[10]. In the following, we detail our algorithms based on Homomorphic Encryption.

[1] Through decryption, the user can find out which attributes are the same with those of the recommended friends

4.1. Encryption

To preserve the privacy of the user, the query needs to be encrypted through the Homomorphic Encryption algorithm. In order to guarantee the safety of the user's information, as mentioned in the above section, the key used to encrypt consists of two parts: one corresponds to the attribute groups, and the other is selected by the user randomly. Hence, the key of each attribute group is different. In the algorithm, M represents all the attributes groups($M = M_1, M_2....M_k$) and M_i represents the i attributes group. I means the personal information ($I = I_1, I_2....I_k$) and I_i means the i personal information version. P_i, Q_i, R_i are the keys of i attributes group and K is the number of attributes groups.

Algorithm 1: Encryption

```
  Input:
  $M, M_i, I, I_i, P_i, Q_i, R_i, K$;
  Output:
  $C([M])$: all the encrypted attributes ( $M = M_1, M_2....M_k$ );
  $C([I])$: all the encrypted personal information ( $I = I_1, I_2....I_k$ );
1 for $i = 1; i \leq K; i++$ do
2     $N_i \leftarrow P_i \times Q_i$;
3     if $R_i \neq 0$ then
4         $C([M_i]) \leftarrow (M_i + P_i \times R_i) mod N_i$;
5         $C([I_i]) = (I_i + P_i \times R_i) \mod N_i$;
6     end
7 end
8 return $C([M]), C([I])$;
```

4.2. Matching Algorithm

When receives the user's ID and the information encrypted, the server runs the Matching Algorithm to find friends for the user in the database. The Matching Algorithm mainly includes two parts: Location Matching and Attributes Matching.

4.2.1. Location Matching

The purpose of location matching is to find people whose district overlaps that of the user. We consider the user's district as a square. If two districts overlap, the overlapping can appear in the upper left, upper right, lower left and lower right in the square. To simplify the matching algorithm, we see the user's location as the center of a rectangular coordinate system and divide the district into four parts based on the quadrant of the rectangular coordinate system. Each vertex in the subdistrict is evenly distributed. The basic idea of the proposed location matching algorithm is: we compare two users' district. If each subdistrict has the same vertexes with the corresponding one, then we consider these two users' district overlap.[2]

4.2.2. Attributes Matching

When the user makes a query, the server will receive query information which is in the form of cipher-text from the user. We use $C([M])$ to represent it. Because the server has stored the information of other users before, the matching algorithm will compare each record in the database with the query information one by one attributes groups to find the suitable recommended fiends. The operation result is computed according to the following algorithm.

[2] Because homomorphic encryption algorithm can only judge whether the information is equal, we adopt the local discretization method for the location matching. The more vertexes are in the district, the more precise the matching algorithm is. Vice versa.

Algorithm 2: Attributes Matching

```
K: the number of attributes groups.
Input:
C([Ma_i]): the i attributes group of user_a in the form of cipher-text( i = 1, 2, 3...k );
C([Mb_j]): the i attributes group of user_b in the form of cipher-text( i = 1, 2, 3...k );
Q_i, N_i: the i attributes group's paraments set by the user_a;
Output: res: the matching result ( res = res_1, res_2....res_k )
1 if $Q_i \neq 0$ AND $N_i \neq 0$ then
2     for $i = 1; i \leq K; i++$ do
3         $res_i = ((C([Ma_i]) - C([Mb_j])) \times Q_i) \mod N_i$;
4     end
5 end
6 return res;
```

Considering matching, the server could request all of the attributes be matched, but most times we do not need this. Objectively, it is tough to find two people whose attributes are the same with each other. Besides, subjectively, the purpose of dating is not to find a mirror image of the user, but to find suitable friends. Therefore, when it comes to matching algorithm, we mainly consider part of the attributes which are the same with each other. The partial matching is divided into two parts: Strict Matching and Loose Matching.

(1) Strict Matching: When the user is sending his query, he must have specific demand looking for friends. The user needs to appointed some attributes groups. The Matching with Strict Attributes algorithm should make sure that these attributes groups are matched. In this way, although the server saves a large number of attributes of users, only part of attributes will be used in the matching algorithm. It not only meets the needs of the users but also saves the operation cost to a great extent, which makes the matching algorithm more efficient.

(2) Loose Matching: As for the Matching with Loose Attributes algorithm, it only needs to make sure that part of the attributes groups which user appoints are matched. If the user appoints N attributes and he gives a number $M(N > M)$, the server only needs to check N attributes groups and make sure whether there are M attributes groups are matched. It not only doesn't care whether all of the attributes groups are matched but also doesn't pay attention to which attributes groups appointed are matched. In another word, the Matching with Strict Attributes is one of a particular case in Matching with Loose Attributes.

4.3. Decryption

Speaking to this, we should divide the situation in different part. On the one hand, it is not necessary to present the information to the users. In order to make sure that the user is not malicious, we can hand the right of showing the information to the recommended people. The user can only get an "ID" of the recommended friends and an Internet connection which is used for chatting. Therefore, if the user sincerely wants to make friends, the server will tell him this is the suitable person, but the next things should depend on himself. In this circumstance, the decryption is useless.

On the other hand, sometimes it is necessary to give the user some information of the recommendations. This will help them know each other and make friends more quickly. In this situation, the decryption plays an important role. The user will receive the cipher-text packet, decrypt it by decryption algorithm and get the plain-text. In some situation, the matching algorithm does not need all of the attributes groups are matched; some attributes groups will be different. Therefore, the decryption is only used for those attributes groups which are matched. The query user will send a request to the recommended user asking for information. If the recommended friends agree the query user's request, then the query user can use the decryption algorithm to obtain plain-text of the information.

5. Simulation

We build a project which is based on the C/S system to realize our framework. The client resides in the user's machine, used to encrypt and decrypt information. The server is used to find friends for users. The experiments are

Algorithm 3: Decryption

```
K: the number of attributes groups.
M_i: the i attributes group of recommended friends in the form of plain-text( $i = 1,2,3...k$ )
Input:
$C([M_i])$: the i attributes group of recommended friends in the form of cipher-text( $i = 1,2,3...k$ );
$P_i$: the i attributes group's key of user;
res: the matching result ( $res = res_1, res_2....res_k$ );
Output: M: all the attributes groups in the form of plain-text( $M = M_1, M_2....M_k$ );
1 if $P_i \neq 0$ then
2     for $i = 1; i \leq K; i++$ do
3         if $res_i = 0$ then
4             $M_i = C([M_i]) \mod P_i$;
5         end
6     end
7 end
8 return M;
```

run on two Windows machines with 2.4GHz CPU, 8GB memory, 1TB 7200 RPM hard-disk. One is used for the server with TOMCAT. The other runs the client.

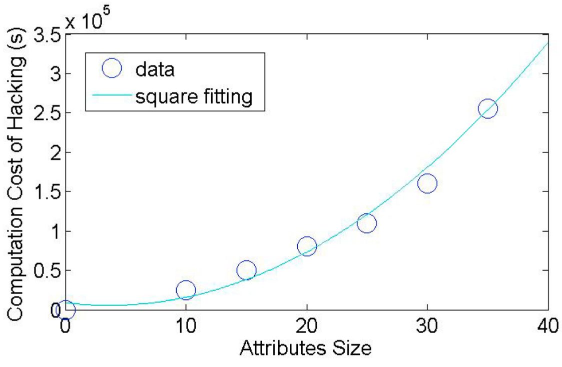

Fig. 2: Hacking cost of decryption with different attributes size

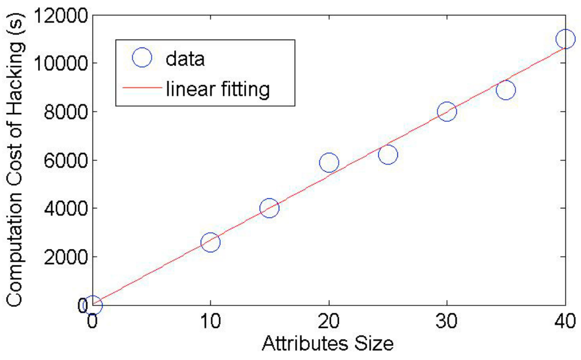

Fig. 3: Hacking cost of decryption with different answers size

In our simulation, we focus on the possibility to decrypt the information by the server or the hacker.The possibility of decrypting the information by the server is important since we need to make sure that with the current technology, there is low chance to leak the information of the user. Therefore, we try to decrypt the cipher-text and record the cost from a hacker's angle(assuming that the server does an inside job to act as a hacker). We put M attributes in an attributes groups where M=10, 15, 20, 25, 30, 35 and 40, and each attribute has 5 answers. For example, as for 30 attributes in one group, we use 300 big prime numbers divided into 150 groups to encrypt attribute groups. For each different combination of the answers to one attribute group, we use different groups of prime numbers to decrypt. If the server wants to decrypt the information, it will need to try 75 times on average (searching in disordered arrangement[13]). Figure 2 shows the time the server can decrypt the information with different attributes sizes. It is clear that the time of decryption is almost squarely proportional to the numbers of attributes. As the number of attributes added, the difficulty of the server decrypting the information will be increased.

Figure 2 shows the cost of hacking when the number of attributes answers is invariant and the number of attributes changes. In this part, we test the hacking cost when the number of attributes answers changes and the number of attributes is invalid. We take 10 attributes in an attributes group, and each attribute has 10, 15, 20, 25, 30, 35 and 40 answers respectively. For example, as for 30 answers, we use 300 big prime numbers divided into 150 groups to encrypt attribute groups. For each different combination of the answers to one attribute group, we use different groups of prime numbers to encrypt. If the server wants to decrypt the information, it needs to try 75 times on average (searching in disordered arrangement[13]). Figure 3 shows the time the server can decrypt the information. It is obvious that the time of decryption by the server is almost linearly proportional to the numbers of answers of the attributes. As the number of answers of attributes added, the difficulty of the server decrypting the information will be increased. We can conclude that the chances of decrypting by the server will be low if the numbers of attributes and the numbers of answers of attributes are enough.

6. Conclusion

In order to better protect the user's personal information on the social applications, we put forward the privacy protection mechanism based on zero knowledge. The client encrypts the information using homomorphic encryption and sends the cipher-text to the server. Then the server helps the user find people who have the same attributes based on homomorphic encryption. In the whole process, once the user's information leaves the client, it is in the form of cipher-text. Meanwhile, the server can still finish his work smoothly. The defenses of grouping attributes and using the location information particularly largely reduces the risk of being leaked by the hacker. Besides, the Loose matching also provides much protection. This mechanism can realize that although a social system server doesn't know any users' information, it can help user match friends who are in accordance with the same attributes. This scheme not only has reference and practicality for practical application, but also has excellent scalability. It provides a strong basis for the future perfection and enhancement.

Acknowledgement

This work has been supported by the National Natural Science Foundation of China (No. 61772080, 61472044).

References

[1] Zaobo He, Zhipeng Cai, and Jiguo Yu. "Latent-data Privacy Preserving with Customized Data Utility for Social Network Data." IEEE Transactions on Vehicular Technology. 67(1): 665-673 (2018).

[2] K. Vu, R. Zheng, and J. Gao. "Efficient algorithms for k-anonymous location privacy in participatory sensing." IEEE Infocom , 2012 , 131 (5) :2399-2407.

[3] L. Sweeney et al.. "k-anonymity: A model for protecting privacy." International Journal of Uncertainty Fuzziness and Knowledge-Based Systems, vol. 10(5), pp. 557-570, 2002.

[4] Jiang Wen-guang, Sun Yu-qing. "Personalized privacy protection for third-party service platform." Journal of Lanzhou University Natural Sciences, 2012, Vol.48(4), pp.85-90.

[5] Zhipeng Cai, Zaobo He, Xin Guan and Yingshu Li. "Collective Data-Sanitization for Preventing Sensitive Information Inference Attacks in Social Networks." IEEE Transactions on Dependable and Secure Computing.

[6] Xu Zheng, Zhipeng Cai, Jianzhong Li and Hong Gao. Location-Privacy-Aware Review Publication Mechanism for Local Business Service Systems. The 36th Annual IEEE International Conference on Computer Communications (INFOCOM 2017).

[7] Tamersoy A, Loukides G et al.. "Anonymization of longitudinal electronic medical records." IEEE Trans Inf Technol Biomed, 2012, 16(3), pp.413-23.

[8] LiJ,LiJ, ChenXetal. "Identity-based encryption without sourced revocation in cloud computing." IEEE Trans Comput, 2015, 64 (2) :425-437.

[9] Naoki Ogura, Go Yamamoto, Tetsutaro Kobayashi. "An improvement of key generation algorithm for Gentry's homomorphic encryption scheme from ideal lattices." Journal of Math-for-Industry, 2011, Vol.3, pp.99-106.

[10] Meiyun Li, Jian Li, Chao Huang. "A Credible Cloud Storage platform based on Homomorphic Encryption." Beijing University of Posts and Telecommunication, 2012.

[11] LiJ, LiX, YangB, SunX. "Segmentation-based image copy-move forgery detection scheme." IEEE Transactions on information forensics and security, 10(3): pp. 507-518.

[12] Stergiou. C,PsannisKE. "Efficient and secure big data delivery in cloud computing." Multimedia tools and applications, Springer, 2017 pp :1-20.

[13] Xu Zheng, Zhipeng Cai, Jiguo Yu, Chaokun Wang, and Yingshu Li. "Follow But No Track: Privacy Preserved Profile Publishing in Cyber-Physical Social Systems." IEEE Internet of Things. 4(6): 1868-1878 (2017).

[14] Xu Zheng, Zhipeng Cai*, Jianzhong Li and Hong Gao. "Location-Privacy-Aware Review Publication Mechanism for Local Business Service Systems ." The 36th Annual IEEE International Conference on Computer Communications (INFOCOM 2017).

[15] Lichen Zhang, Zhipeng Cai, and Xiaoming Wang. "FakeMask: A Novel Privacy Preserving Approach for Smartphones." IEEE Transactions on Network and Service Management.13(2): 335-348 (2016).