Author’s response to reviews

Title: A Privacy-preserving Distributed Filtering Framework for NLP Artifacts

Authors:
Md Nazmus Sadat (nazmus.cse@gmail.com)
Md Momin Al Aziz (azizmma@cs.umanitoba.ca)
Noman Mohammed (noman@cs.umanitoba.ca)
Serguei Pakhomov (pakh0002@umn.edu)
Hongfang Liu (Liu.Hongfang@mayo.edu)
Xiaoqian Jiang (Xiaoqian.Jiang@uth.tmc.edu)

Version: 1 Date: 03 May 2019

Author’s response to reviews:

Response to Reviewers’ Comments

Dear Editor,

We would like to sincerely thank you and the reviewers for their insightful comments. The manuscript has been revised to address the comments. Here, we provide detailed feedback to the comments from each reviewer.

Reviewer 1:

Comments:

I think this paper in general is OK, but some issues are needed to be solved:

1. Theoretical analysis of your detailed system protocol is not enough, you need to revise the part of security analysis with more detail

2. Why the hash function can be used for removing the sensitive terms? I am not convincing on this method. Please detailed give your analysis of this part.

3. Some references closely related with this work can be referred such as:


Response to Reviewer 1:

Thank you for your feedback.

1. Now, we have elaborated security analysis. We also addressed the hash function collision attack and presented the preventive measures in the Discussion Section, as follows:

Another desirable property of a hash function is collision resistance. A hash function is said to be collision resistant if it is computationally infeasible to find two different inputs \( m_1 \) and \( m_2 \) with \( H(m_1) = H(m_2) \). It seems if the hash function has an output length of \( b \) bits, we have to check about \( 2^b \) messages. However, it turns out that an attacker needs only about \( 2^{b/2} \) messages. This is a quite surprising result, which is due to the birthday attack. This attack is based on the birthday paradox, which is a powerful tool that is often used in cryptanalysis.

Collision search for a hash function \( H() \) is exactly the same problem as finding birthday collisions among party attendees: how many people are required at a birthday party such that there is a significant chance that at least two attendees have the same date of birth?. The question is how many messages \( (m_1, m_2, \ldots, m_k) \) does an attacker need to hash until he has a chance of finding \( H(m_i) = H(m_j) \) for some \( m_i \) and \( m_j \) that he chooses. The most significant consequence of the birthday attack is that the number of messages needed to hash to find a collision is approximately equal to the square root of the number of possible output values, i.e., about \( 2^{b/2} \). Hence, for a security level of \( u \) bit, the hash function needs to have an output length of \( 2^u \) bit. In order to prevent collision attacks based on the birthday paradox, the output length of a hash function must be at least 128 [36]. As mentioned previously, we are using SHA-256 in this work, which has output length 256.

In 2004, collision-finding attacks against MD5 and SHA-0 were demonstrated by Xiaoyun Wang [43]. One year later, it was claimed that the attack could be extended to SHA-1 and a collision search would take 263 steps, which is considerably less than the 280, achieved by the birthday attack (the output width in this case is 160 bit). In this work, we are using SHA-2 (precisely, SHA-256) against which no attacks are known to date.

2. Hashing is one of the (not the only one) cryptographic primitives that we used in our proposed system protocol to remove privacy-sensitive terms.
At the system initialization phase, data owners receive salt, public and private keys from the CSP. Also, the central server receives only the public key. Then, each data owner sends the hashes of bigrams to the central server. After receiving the hashes from each data owner, the central server computes the intersection of the hashes. Then, the central server sends the elements of this intersection to data owners. Upon receiving the intersection of the hashes from the central server, data owners encrypt the local frequency of the intersected bigrams by using the ciphertext packing technique. To do so, they follow the order received from the central server. After encrypting the counts, data owners send the packed ciphertexts to the central server. Once the central server receives the ciphertexts, the central server performs homomorphic addition operation on these packed ciphertexts. At this point, the central server does the thresholding for the sum of homomorphically encrypted counts so that after decrypting the encrypted response, the data owners can understand whether the sum of counts of the corresponding record is less than the threshold.

The primary objective of this work is to identify globally infrequent bigrams, that can be used to scrub privacy-sensitive contents from clinical notes before sharing. This usage will be not be the same for different application scenarios. In some cases, only the low frequency bigrams will be removed from the sentences, whereas in some applications, the entire sentence containing those bigrams will be eliminated.

3. The closely related works have been referred in the revised manuscript.

Reviewer 2:

Comments:

Require thorough proofreading. And, the conclusion section require tidying up.

Response to Reviewer 2:

Thank you for your feedback. We now have addressed these issues.