Towards Privacy-Preserving Keyword Search via MapReduce
Report on Project Completion

En-Yang Lin(林恩洋)
9822020
Department of Applied Mathematics
NCTU

June 20, 2012

1 Introduction

The idea of privacy-preserving keyword search is pretty straightforward. Suppose we have an *Owner* who possesses some data that he wants to be able to search. However, the amount of data is big, so he decides to outsource them to the *Cloud* (who supposedly has more storage and computational power). It may happen though, that data to be outsourced is sensitive and simply can not be send in plain. Here Owner faces supposed dilemma: *expose sensitive data to cloud or search himself.*
Therefore, natural question arises: *can we find such a way to encrypt our data so that search on them is still possible and no sensitive information leaked during the search?.* The apparent answer is no, since the intuitive idea of search assumes that we have access to the plain text. However, cryptography may be quite counter-intuitive, the famous example being public-key encryption. It may be a coincidence, but the idea of counter-intuitive encrypted keyword search dates to the inventors of RSA algorithm back in 1978. However, they were unable to come up with the solution.
Since then, there were attempts done to solve problem and now we know quite a few more or less satisfactory mathematical settings and schemes that address this problem. To clarify, we treat Cloud as *semi-trusted, but curious*, that is *it will correctly perform assigned task, but may be interested to inspect data it operates on.*. There are few fundamental things that we want to protect Owner from (they are listed in roughly increased difficulty to achieve):

- Cloud should not be able to recover the plain text from the cipher it searches . This is basic. This is why it is called ”encryption”. The exact meaning of ”not be able to know” depends on treat model chosen. The semantic security under chosen keyword attack (SS-CKA) is popular, which means that even if you decryption of few keywords, you still can not decrypt others.

- Cloud should not be able to do search without Owner's agreement. That is, we introduce *search tickets* that are something that grants you with ability to do a particular search query and can be generated by Owner exclusively.
- Cloud should not learn the *access pattern.* In other words, Cloud should not be able to learn statistical information about documents content by analyzing search requests and search results. This is difficult to achieve. Such feature naturally requires that Cloud can not learn about statistics of search queries, which implies that search tickets should be randomized in some way. That is, for same query, search ticket generated is not unique. The scheme we currently use lacks this feature (randomized search tickets) but we know at least one, that has it.

As you see, the problem raised above is very natural and applications one may think of are numerous. As cloud computing become so ubiquitous now, question of whether we may trust cloud storage or not is particularly important, if one does not want to end up in situation, when he discovers that his every step is traced and he basically lives in the world of Orwell's "1984".

2 Our Contribution

Basically our goal was to survey papers and implement one secure search scheme. However, we also wanted scheme implemented to fit our scenario.
In our scenario, Owner possesses two two types of data: images and tags. The later may be seen as description of images, where each image has corresponding set of keywords that describe it. This is exactly tags, that are being securely searched. In such setting if *User* wants to search and he has search ticket from owner, he may ask Cloud to perform search and he will get the index of all images, that contain particular keyword in their description. After that, User may download some or all of the images from Cloud. Images that stored on Cloud side are in fact encrypted (by symmetric cipher known to Owner) so Cloud has no access to them. Therefore, Owner after downloading should acquire the (symmetric) decryption key from Owner to finally enjoy the image.
Finally, to add popular cloud-computing flavor to our project, it was initial design de-

Figure 1: This is how one may depict communications between the entities in our system (image based on that from [2]).

cision to implement Cloud software on top of open-source distributed computing framework Hadoop. Computational model used in Hadoop is called MapReduce, which was included in name of our project to symbolize this decision.
However, soon after we have surveyed enough papers and started to think about particular algorithm to chose, we have noticed that in fact many schemes possess very similar structure, therefore our system may be largely decoupled *from the internals of particular algorithm to use*, therefore the design decision was done: although we still implement

only one secure search scheme, we encapsulate it in as abstract as possible interface, so later we may integrate other schemes painlessly. The advantage was obvious: now we have modular system that secure search schemes can be included in. Therefore, different approaches may be compared and fruitful experimental data would be available. However, this is still something to be done. So far, our contributions include:

1. We have implemented modular system such that secure search schemes may be integrated inside it. System consist of three pieces of software communicating via network:
    - **Owner** software, that performs initial encryption of data and later distributes search tickets and image decryption keys.
    - **Cloud** software, that is built on top of Hadoop framework and used to do the search and image retrieval.
    - **User** software, that is written for Android OS. This mobile phone application (unlike previous two) has simple GUI that allows user to do two basic tasks, such as search and image retrieval (and decryption). The emphasize was done to hide as much complexity as possible, exposing only simple interface to user.
2. We have implemented one of secure search schemes (the one in [3]) and integrated it in our system. The implementation is complete in the sense that all of described above functionality (search and image retrieval) works correctly, and in reasonable time. We should admit that although we use Hadoop framework, excessive testing on *real* cluster with *big* load is still something to be done. However, as Hadoop framework takes care of distributed computations for us, these tests are something not difficult to do - we do not have to modify the system to run on real cluster.

Note, that we do not own the algorithm we have implemented, it is **not** our invention. Implementation was done by two people: me, and my teammate Oleksii Leontiev (9822058, Department of Applied Mathematics, NCTU). During the project I was responsible for User mobile client coding, whereas my teammate implement the Cloud and Owner.

3 Extensions

We should admit that this project was different. We have learned a lot. First, this is by no means the largest (in size of code) system we wrote ever. Second, this was the first system, each of us wrote not by himself - teamwork had its difficulties. Third, this project was quite eclectic, as we have done some system programming, cryptography (using OpenSSL) and network programming. Designing this system was also a lot of fun. Therefore, naturally we do not want to finish the project and we propose further work that may give this project new dimension and far greater value. Everything below will be included in our proposal for the next semester scholarship, that we hope will be given to us to continue this fascinating and (we believe) not entirely useless project.
As emphasized above, our system is modular and different secure search schemes may

be included with more or less effort. Therefore, aside of purely technical upgrades to Cloud, Owner and User software components (i.e. optimize network communications and make the whole system more stable) we also propose to include the algorithms from the following three papers in our system:

1. N. Cao, C. Wang, M. Li, K. Ren, W. Loe *"Privacy-Preserving Multi-keyword Ranked Search over Encrypted Data"*
2. E.-J. Goh, *"Secure indexes"*, Cryptology ePrint Archive, Report 2003/216, 2003, `http://eprint.iacr.org/`
3. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, *"Public key encryption with keyword search"*, in *Proc. of EUROCRYP.04, volume 3027 of LNCS*, Springer, 2004.

We believe, all these may be implemented and integrated in our system in one semester and after that we will be able to make comparisons of four algorithms at hand. This may give interesting experimental information, as practical comparison is important and we are not aware of any system that would embrace this methods together to allow such comparison.
We should emphasize that term "extension" we use here is not completely correct one. Although system we have now works and may be considered as software, the very way it was build suggests that this "extension" (include more algorithms) should be done. Without it our system is just software, with it it may become source of valuable experimental data, and after all even better software (we still want to improve some technical aspects).

4 Conclusions

As mentioned above, this was interesting project. Although not always perfectly, we have done and learned something, and we are willing to continue. We hope this won't be our last report on behalf of this project.

References

[1] Eu-Jin Goh *"How to Search on Encrypted data"*, slides available at `http://crypto.stanford.edu/~eujin/papers/secureindex/2003nov-encsearch.pdf`

[2] N. Cao, C. Wang, M. Li, K. Ren, W. Loe *"Privacy-Preserving Multi-keyword Ranked Search over Encrypted Data"*

[3] D. Song, D. Wagner, and A. Perrig, *"Practical techniques for searches on encrypted data"*, in *Proc. of IEEE Symposium on Security and Privacy'00*, 2000.