# A Cloud Authentication Protocol using One-Time Pad

Lexus Jun Hong Sim†, Shu Qin Ren‡, Sye Loong Keoh† and Khin Mi Mi Aung‡
†School of Computing Science, University of Glasgow, G12 8QQ United Kingdom
‡Data Center Technologies Division, Data Storage Institute, A*STAR, Singapore 138932
Email: {2167846s, SyeLoong.Keoh}@glasgow.ac.uk, RenShuqin@gmail.com, Mi_Mi_Aung@dsi.a-star.edu.sg

***Abstract*—There is a significant increase in the amount of data breaches in corporate servers in the cloud environments. This includes username and password compromise in the cloud and account hijacking, thus leading to severe vulnerabilities of the cloud service provisioning. Traditional authentication schemes rely on the users to use their credentials to gain access to cloud service. However once the credential is compromised, the attacker will gain access to the cloud service easily.**

**This paper proposes a novel scheme that does not require the user to present his credentials, and yet is able to prove ownership of access to the cloud service using a variant of zero-knowledge proof. A challenge-response protocol is devised to authenticate the user, requiring the user to compute a one-time pad (OTP) to authenticate himself to the server without revealing password to the server. A prototype has been implemented to facilitate the authentication of the user when accessing Dropbox, and the experiment results showed that the overhead incurred is insignificant.**

## I. Introduction

Many business organizations now have a huge dependency on cloud services for their daily operations. Increasingly, terabytes of data are being stored in the cloud for ease of access, thus providing 24/7 availability at any place and anytime. This also makes file sharing a lot simpler, with a URL link, users can now share a file or document in the cloud with their colleagues and friends without requiring any validation of credentials.

However, the convenience brought forth by cloud services comes with a price, i.e., security. Convenience or security has always been a battle of striking a balance, tools that make our life simpler are often less secure, while technologies that safeguard the system are generally inconvenient. Although many cloud authentication schemes have been proposed [1], since 2004, there has been an increasing amount of data breaches [5], compromising the cloud service providers, user credentials, and user's confidential data in the cloud. With users losing personal information, passwords and credit card details, there is an increasing need to not only ensure security but to also reduce and minimize the amount of *information leakage* [7].

The current practise for accessing the cloud service is via *username* and *password* authentication, which is prone to many security attacks and vulnerabilities. Many users tend to use the same password for multiple cloud services, if the password is compromised, all user data hosted on these services will be compromised as well [9]. Additionally, users typically choose a password that is easy to remember, as randomly generated passwords are just not memorizable. Although these random passwords have higher entropy, they are not usable from the user's perspective.

This paper proposes a new approach to secure cloud authentication using One-Time Pad (OTP). A trusted proxy which serves as a mediator between the cloud service provider and the client is responsbile for authenticating the client using a challenge-response protocol. The client is not required to use its PIN or password to authenticate itself, but using a two secrets pre-shared with the client to compute a response to the challenge posted by the proxy. In this way, neither the user's password nor its hash is transmitted via the network to the server for authentication. This also minimizes the *information leakage* on the communication channel. Furthermore, the trusted proxy is not required to store users' (salted or hashed) passwords. Essentially, each challenge is unique, random and not repeated, hence requiring a different response from the client to be authenticated for each access request.

This paper is organized as follows: Section II presents related work, and Section III describes the system architecture of the proposed proxy-based authentication scheme for cloud services. In Section IV, we present the implementation details, while Section V describes the evaluation results. Finally, we conclude the paper with future work in Section VI.

## II. Related Work

Multi-factor Authentication (MFA) [4] is commonly used in the current cloud services to authenticate users. In addition to the use of username and password, additional authentication factors such as biometrics, voice and face recognition are used to further ascertain the authenticity of the user. This is mainly because these biometric traits are easy to provide and hard to replicate. However, using biometrics and password at the same time, demands more computational resources on the server side, and registration of biometrics must be done. This also leads to privacy issue as the clients may not be comfortable with.

Fuzzy vault, digital signature, and zero-knowledge combination (FDZ) [8] first establishes a secure session between the client and the server in order to derive a session key using Diffie-Hellman Key Exchange protocol in combination with RSA to prevent man-in-the-middle attack. Subsequently, a fuzzy password scheme is employed in which the client is asked to choose five out of seven images in order to authenticate itself to the cloud server. If the user selects five images correctly, he will be granted to access to server resources; in

other cases, the server will reject the authentication request. [6] proposed an alternative to replace graphical passwords with QR codes.

SeDiCi 2.0 [3], [2] uses Zero-Knowledge Proof (ZKP) to safeguard the user's privacy, thus enabling the authentication to take place without requiring the user to reveal its user name and password. Therefore, it provides an improved solution for phishing attempts. The technology also supports REST-based API that enables taking advantage of the service by mobile phones, web applications and other client applications. Instead of requiring the user to carry a hardware token, a browser plug-in must be installed in order to run SeDiCi.

## III. Overview

This section describes the architectural overview of the proposed authentication protocol.

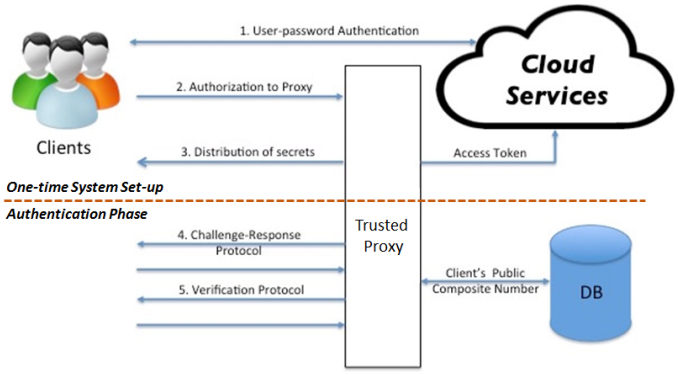


Fig. 1. System Architecture of the cloud-based authentication protocol

As illustrated in Figure 1, we introduce a trusted proxy for cloud services such as Dropbox, to handle authentication using our novel challenge-response protocol that is based on prime-number factorization. The system is first set-up by requiring the client to authenticate itself to the cloud service using the normal user-password authentication protocol, and then requests for an access token. It then authorizes the trusted proxy to mediate the authentication between himself and the cloud service, by passing the access token to the trusted proxy. In this one-time set-up phase, two secrets, which are randomly generated large prime numbers are distributed to the client, together with a client secret, to be stored in a secure storage. A public composite number, which is the product of the two prime numbers is stored in the trusted proxy to identify the client. Knowing this composite number does not reveal any information about the secret prime numbers.

Once this proxy-based cloud authentication has been set-up, all subsequent authentication will not require the user to transmit its *username-password*, but a security challenge will be issued to the client based on the composite number, and a residue computed based on the client's secret. Only the correct response which must be generated using the client's secrets will be accepted. The challenge is generated using a random number that is 3072 bits long, and hence ensuring that each access request is unique and will not be re-used. The database is used to store the mapping of the public composite number to the corresponding user account. Using this protocol, the clients do not need to use their *username-password* to perform authentication anymore.

The following sections describe the security requirements to be achieved, the procedure of generating the security parameters and the details of the challenge-response protocol for cloud authentication.

### A. Security Requirements

This section outlines the security requirements for cloud authentication.

- *Transmission of password* – the client must not be required to transmit its password to the server for authentication.
- *Storing of password* – the trusted proxy which mediates the authentication should not store the user's password, but only public information that even if it is compromised, the client's confidential credential is not revealed.
- *Secrets with high entropy*[1] – Access to the cloud services should be based on secrets distributed to the client, which have high entropy. A master password maybe required to grant access to the client to retrieve these secrets. Alternatively, the secrets may be stored using a hardware token, e.g., a secure USB, that should be in possession of the client.

### B. Generation of Security Parameters

The security of system is based on large prime factorization.

- Two large prime numbers ($P_{ii}$, $P_{ij}$) – are randomly generated for each client. The generated prime numbers must be at least 1024-bit. These two primes form two parts out of three of the client's secret.
- Public composite numeer ($N_i$) – is a product of the primes, $P_{ii}$ x $P_{ij}$. $N_i$ is public, and it is stored in the database hosted by the trusted proxy.
- Secret ($s$) – is a randomly generated number that is co-prime with the composite number $N_i$. $s$ is generated such that it is $1 < s < N_i$. Essentially, $s$ forms the third part of the client's secret.
- Residue ($I_i$) – is generated as $I_i = s^2 \ mod \ N_i$. Similarly, this residue is public, and it is known to the trusted proxy and is used for verification of the client's secret, $s$.

The primes and the secret, ($P_{ii}$, $P_{ij}$, $s$) are distributed to the client, and these secrets must be stored in a secure storage of the client, while the public information ($N_i$, $I_i$) are distributed to the trusted proxy, and they are mapped to the client's cloud service account for identification.

[1]The randomness collected by an operating system or application for use in cryptography or other uses that require random data. This randomness is often collected from hardware sources, either pre-existing ones such as mouse movements or specially provided randomness generators.

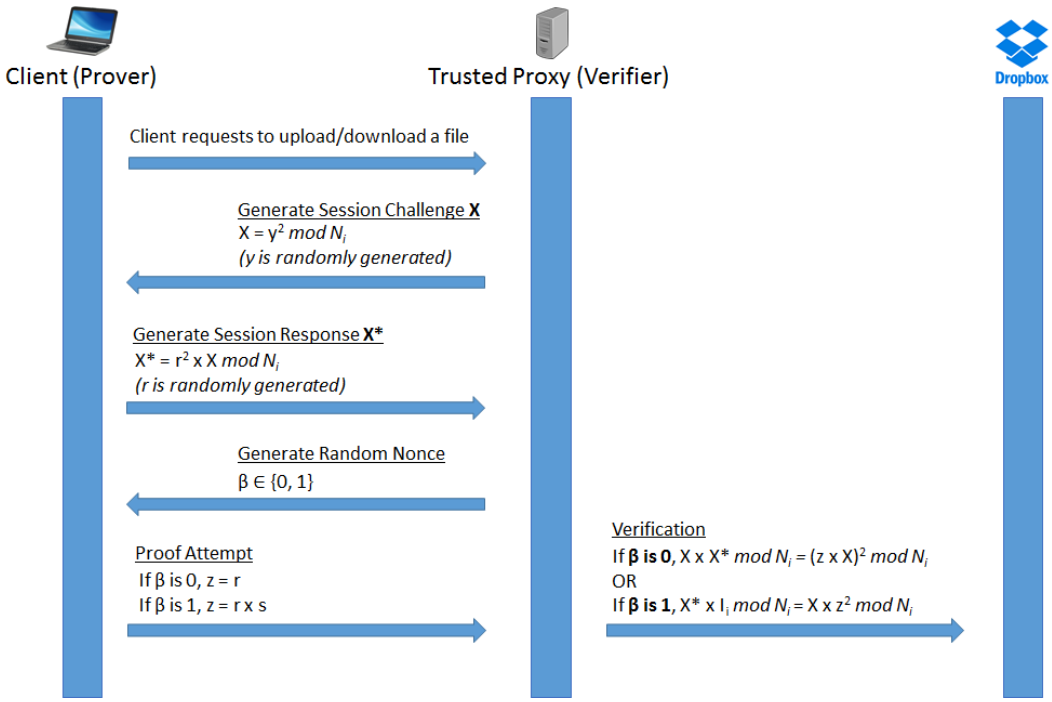


Fig. 2. Challenge-Response Protocol for cloud authentication

### C. Challenge-Response Authentication Protocol

Figure 2 shows the sequence diagram of the challenge-response protocol for authenticating the clients. Assuming that the client has been configured with the secrets, and the public-information are mapped to the client's account. When the client requests to read/write/modify/download its files on the cloud service, the trusted proxy generates a session challenge X with the following:

$$X = y^2 \mod N_i \tag{1}$$

where $y$ is a random number per access request, and it is typically 3072-bit.

Upon receiving the challenge, the client accesses its secrets, and compute the following:

$$X^* = r^2 \times X \mod N_i \tag{2}$$

where $r$ is also a randomly generated number with the same length as $y$. $N_i$ can be derived by multiplying the two secret prime numbers in possession of the client. X* is then sent back to the trusted proxy for verification. This serves as a commitment of authentication for the access request.

The trusted proxy then generate a random $\beta \in \{0, 1\}$, in order to verify the client. Depending on the value of $\beta$, the client generates a different response to authenticate itself. If $\beta$ = 0, then the client returns the previously generated random number, $z = r$. Conversely, if $\beta$ = 1, the client computes:

$$z = r \times s \tag{3}$$

At the trusted proxy, the verification is performed depending on the value of $\beta$. When $\beta$ = 0, it does not require the client to use the third part of its secret, $s$, and by returning a correct $z = r$ is sufficient. This is because $r$ which was committed as part of $X^*$ must have been *generated* by the correct client that possesses the authentic prime numbers ($P_{ii}$, $P_{ij}$). The trusted proxy verifies $z$ such that it satisfies the following equation:

$$X \times X^* \mod N_i = (z \times X)^2 \mod N_i \tag{4}$$

Essentially, this is equivalent to $y^4 \times r^2 \mod N_i$.

On the other hand, if $\beta$ = 1, this requires the client to use the third part of its secret, $s$ to compute the response $z$. The received $z$ must satisfy the folloiwng:

$$X^* \times I_i \mod N_i = X \times z^2 \mod N_i \tag{5}$$

which results in $y^2 \times r^2 \times s^2 \mod N_i$.

## IV. Prototype Implementation

We have implemented a prototype of the secure challenge-response protocol, integrated with Dropbox in order to demonstrate the feasibility of this new authentication for cloud services. Dropbox was chosen as it provides a core API for most of the functionalities needed to perform file upload and download from the trusted proxy.

This prototype was implemented using Java, while the trusted proxy was deployed as an RMI server. The functions for generating the security parameters were implemented using

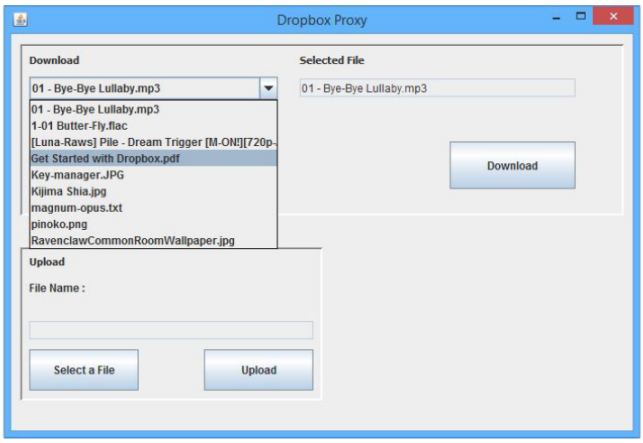


Fig. 3. GUI of a simple file upload/download client

Java Cryptographic library to generate secure random numbers, prime numbers, and big integer multiplications, modular arithmetics and exponentiation.

The client secrets, i.e., two prime numbers and a random secret, $s$ were stored in a USB, protected with a password. Whenever the client requests to access his files in Dropbox, the client will be prompted to insert the USB containing the secrets in order to complete the authentication process.

Using resource upload and download as our use case, we have implemented a client GUI to facilitate these file operations using the proposed challenge-response authentication protocol. Figure 3 shows the GUI of the client application.

## V. Preliminary Evaluation

In terms of performance, we measured the time taken to *upload* and *download* files of variable size to determine the performance overhead incurred by the authentication performed at the trusted proxy. We compared the total data transfer time from the challenge-response protocol, data upload/download with our proxy solution against the existing Dropbox solution. The experiment results are listed in Table I and Table II. Obviously, our solution is slower than the current Dropbox solution for upload as shown in Table I, we observed that by adding a challenge-response protocol for each file upload, it incurred an overhead of 0.02 sec to 6 sec. Although the performance is highly reliant on the network speed, we can conclude that the overhead for having an additional step to authenticate the client using our scheme is insignificant.

| File Size | Without Proxy | With Proxy |
|---|---|---|
| 100Kb JPG File | 2.156 secs | 2.178 secs |
| 8.35Mb Audio File | 7.706 secs | 7.953 secs |
| 33 Mb Flac Clip | 16.844 secs | 19.57 secs |
| 127 Mb Video File | 53.8 secs | 59.223 secs |

TABLE I. Results of Upload Speed

Similar results were recorded for resource download. As shown in Table II, the performance between our system and pure Dropbox solution is comparable.

An increase in uploading or download time per file will be significant if the client were to be an enterprise user. Processing more files drastically increases the processing time thus might not be feasible. But our solution is scalable, we can add more proxy nodes to perform the authentication in order to safeguard the security of the cloud services.

| File Size | Without Proxy | With Proxy |
|---|---|---|
| 100Kb JPG File | 1.608 secs | 1.621 secs |
| 8.35Mb Audio File | 5.965 secs | 5.982 secs |
| 33 Mb Flac Clip | 16.434 secs | 20.917 secs |
| 127 Mb Video File | 54.023 secs | 60.965 secs |

TABLE II. Results of Download Speed

## VI. Conclusions and Future Work

This paper has proposed a novel challenge-response protocol to authenticate clients that utilize cloud storage in a secure and economical means. With the security concerns, we have proposed an authentication scheme similar to OTP that is based on prime factoring hard problem; we have also devised and implemented a practical challenge-response protocol to allow for each access request from the client to be authenticated without requiring the client to transmit his password.

Through preliminary experiments, we have also shown that the proposed solution incurs reasonable amount of overhead and its performance can be scaled up by deploying multiple proxy nodes, and this is particularly useful for meeting the requirements of the enterprise tenants, where it can better support performance oriented applications for large scale enterprises.

In the proposed system, the three part client secrets must be stored in a secure manner. In the future, we plan to investigate the protection of these secrets, for example using Bitlocker on USB drive or file vault. Another interesting future work is to look into secrets recoverability to ensure better usability of this protocol.

## References

[1] M. Alizadeh, S. Abolfazli, M. Zamani, S. Baharun, and K. Sakurai. Authentication in mobile cloud computing: A survey. *Journal of Network and Computer Applications*, 61:59 – 80, 2016.

[2] S. Grzonkowski. Sedici: an authentication service taking advantage of zero-knowledge proofs. In *Proceedings of the Financial cryptography and data security*, LCNS. Springer, 2010.

[3] S. Grzonkowski, P. Corcoran, and T. Coughlin. Security analysis of authentication protocols for next-generation mobile and ce cloud services. In *Proceedings of the 2011 IEEE international conference on consumer electronics. Berlin, Germany*. IEEE, 2011.

[4] Y.-S. Jeong, J. S. Park, and J. H. Park. An efficient authentication system of smart device using multi factors in mobile cloud service architecture. *International Journal of Communication Systems*, 28(4):659–674, 2015.

[5] D. McCandless. World's biggest data breaches. https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/, mar 2016.

[6] D.-S. Oh, B.-H. Kim, and J.-K. Lee. *A Study on Authentication System Using QR Code for Mobile Cloud Computing Environment*, pages 500–507. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[7] J. S. Park, K. J. Yi, and J. H. Park. *SSP-MCloud: A Study on Security Service Protocol for Smartphone Centric Mobile Cloud Computing*, pages 165–172. Springer Netherlands, Dordrecht, 2011.

[8] D. Schwab and L. Yang. Entity authentication in a mobile-cloud environment. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW '13, pages 42:1–42:4, New York, NY, USA, 2013. ACM.

[9] Z. H. Zhang, J. Xue-Feng, J. J. Li, and W. Jiang. An identity-based authentication scheme in cloud computing. In *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, pages 984–986, Aug 2012.