A PASS Scheme in Cloud Computing - Protecting Data Privacy by Authentication and Secret Sharing

Jyh-haw Yeh
Dept. of Computer Science
Boise State University
Boise, Idaho 83725, USA

Abstract - *Cloud computing is an emerging IT service paradigm. Instead of developing their own IT departments, business sectors purchase on-demand IT service from external providers in a per-use basis. From business cost perspective, companies are shifting capital expenses (CapExp) on hardware equipments to operational expenses (OpExp). Many companies, especially those startups, found this cloud IT service is economically beneficial. However, this IT service paradigm still requires to overcome some security concerns before it can be fully deployed. One of the main security issues is how to protect client's data privacy from cloud employees. This paper proposes a PASS scheme for this purpose using Authentication and Secret Sharing.*

Keywords: Cloud Security, Data Privacy, Authentication, Secret Sharing

1 Introduction

Cloud computing is an emerging research field to facilitate the deployment of a new IT service paradigm. Cloud providers offer variety of IT services to subscribers and charge them in a per-use basis. Many researches [1-5] have identified the potential benefits of this IT paradigm such as cost saving, innovative technology fast development, better resource utilization, and arguably better security protection.

Data storage is a popular IT service provided by the cloud. For example, the Amazon's S3 - Simple Storage Service [6]. In traditional data management models, people store and protect their own data under their own authority, whereas in cloud computing, the responsibility of data management and protection no longer belongs to the data owner. This fundamental change brings some new security challenges that traditional security solutions might not work. One of the most difficult challenges is the protection of data privacy [7-11]. Cloud's data center stores data from different clients. These data may physically reside in the same hardware. Thus, cloud providers must have an effective data isolation mechanism to prevent illegal data accesses from outsiders, other clients, or unauthorized cloud employees. The protection against malicious cloud employees is a difficult problem and may require a fully homomophic encryption algorithm. Unfortunately, cryptologists were not able to find any such encryption algorithm for years. Actually, they are not even sure whether such encryption algorithm exists.

Without homomophic encryption algorithms, it is unlikely for clients to have 100% data privacy against cloud employees. Thus, the scheme presented in this paper is intended to reduce the data privacy risk to a minimum. The proposed scheme is named "PASS" because it protects clients' data Privacy by Authentication and Secret Sharing. The PASS scheme combines several coherent security components such as public key cryptosystem, key agreement, key management, authentication, and access control. This paper will present the design choices of each component and describe its relationship to others. In general, the design of the PASS scheme is based on the following guidelines:

1. Ensure secret isolation (or secret compartment). That is, a reveal of a secret should not result in a reveal of another secret.

2. Security protection should be considered with a higher priority than efficiency.

3. Select a design choice that would benefit as many security components as possible.

For example, in trying to ensure the main security goal "data privacy", the PASS scheme chooses not to store encryption keys anywhere in the cloud

because of the secret isolation guideline. More detailed explanation of this design choice will be described later in Section 3.

The rest of this paper is organized as follows: The five security components are described from Section 2 to Section 6. Section 7 presents the PASS scheme by providing a sequence of walk through steps to show how to integrate these five security components. Section 8 concludes the paper.

2 Public Key Cryptosystem

RSA [12] and elliptic curve [13] are two popular public key cryptosystems. In the proposed PASS scheme, elliptic curve was chosen over RSA. Elliptic curve cryptosystem (ECC) provides the same level of security as RSA, but with much smaller key size [14]. In addition, based on ECC, there is a convenient and efficient key agreement algorithm, which can be used in the key agreement security component (design guideline 3).

2.1 ECC Setup

The cloud provider, based on the recommendation from National Security Agency [15], chooses an appropriate prime curve $E(F_p) : y^2 = x^3 + ax + b$ over a prime p and a base point G with an order n, where $nG = O$. The cloud publishes the chosen ECC domain parameters $< p, a, b, G, n >$.

2.2 Public Key Generation

The cloud's server chooses a random number $d_s \in F_p$ as its private key and then computes its public key $D_s = d_sG$. Similarly, each client i establishes his own private-public key pair $d_i \in F_p$ and $D_i = d_iG$.

2.3 ECC cryptography

There are many ECC encryption and signature algorithms available in the literature. All these algorithms are based on the hardness assumption of the Elliptic Curve Discrete Logarithm Problem (ECDLP) that states: *Given two points D and G on the curve, where $D = dG$ for some $d \in F_p$, it is computational hard to find d.*

ECC encryption and decryption will be used in the process of client authentication. Below are the notations used in this paper for these two operations. Let c be a ciphertext of a message x, encrypted by a public key D_i. That is,

$$c = ENC_{D_i}(x) \quad (1)$$

The ciphertext c can only be decrypted by an entity with a private key d_i, where $D_i = d_iG$. The notation of decrypting the ciphertext c back to the original message x is

$$x = DEC_{d_i}(c) = DEC_{d_i}(ENC_{D_i}(x)) \quad (2)$$

3 Key Agreement

Though there is a public key ECC established, for the purpose of efficiency, clients' data stored in the cloud should still be encrypted by a symmetric encryption algorithm such as AES [16]. Thus, the cloud's server needs to share a secret symmetric key k_i with each client i. The PASS scheme chooses not to store k_i anywhere in the cloud because of the following two secret isolation advantages:

- If the cloud's server is compromised, the privacy of stored data can still be preserved.
- If there are malicious employees inside the cloud, not storing the shared keys in the cloud reduces their chances of stealing sensitive data.

Of course, in response to a client's query, the server needs to derive the shared key so that the client's data can be retrieved for further processing. Only during this time frame, the malicious employees may have the chance to steal the key. To further protect keys during these time frames, it requires a strong access control component in the PASS scheme as a second defense, which will be described later in Section 6.

Now, the remaining question is how the server derives the shared key while receiving a client's query. The PASS scheme uses Shamir's secret sharing algorithm [17] to accomplish this. For each client i, the server has a secret share SS_i and the client has the other secret share CS_i. With both SS_i and CS_i, any entity is able to derive the shared secret key k_i. However, lack of any one of these two shares, k_i cannot be recovered.

For each client i and the server, the key agreement component in the PASS scheme is responsible for generating k_i and its two secret shares SS_i and CS_i. From the established ECC, the two parties (the server and each client i) can easily agree on the two secret shares, even without the need to talk to each other. However, the agreement of the shared key k_i requires exchanging some messages using the Diffie-Hellman algorithm [18].

3.1 Secret Key Agreement

Before presenting the secret key agreement algorithm in the PASS scheme, we would like to describe an easy algorithm first, then followed by arguing the use of the easy one may violate a design guideline. It works as follows: Using ECC, the server computes a point $Q_i = d_s D_i$ and the client i can compute the same point $Q_i = d_i D_s$ since $d_s D_i = d_s d_i G = d_i D_s$. The shared symmetric key k_i can just be the x-coordinate of Q_i. The security of this key agreement scheme is based on the hardness assumption of the elliptic curve Diffie-Hellman problem. Even though this is an efficient key agreement algorithm without any message being exchanged, the PASS scheme decided not to use it because of the violation of the secret isolation design guideline. The violation occurs if the server is compromised and the private key d_s is revealed, an adversary can then derive the secret key k_i of any client i.

Thus, the traditional Diffie-Hellman key agreement algorithm will be used to establish the shared keys in the PASS scheme:

1. Both the server and the client i choose a random number $r_s \in F_p$ and $r_i \in F_p$, respectively.

2. The server computes and sends a point $R_s = r_s G$ to the client. Similarly the client i computes and sends a point $R_i = r_i G$ to the server.

3. The server computes a point $Q_i = r_s R_i$ and the client can compute the same point $Q_i = r_i R_s$ since $r_s R_i = r_s r_i G = r_i R_s$. Both the server and the client permanently remove r_s and r_i from their storage after Q_i is computed.

4. The shared secret key k_i can then be the x-coordinate of the point Q_i.

Note that this process generates two extra messages. However, the key agreement is a one-time process and will be performed only when a customer subscribes as a new client.

3.2 Secret Shares Agreement

After k_i is agreed, the two secret shares SS_i and CS_i can then be generated separately by the server and the client i, respectively. In this secret share agreement algorithm, there is no extra message required. The algorithm works as follows:

1. Let a be the x-coordinate of the point $Q_i + D_i$, where D_i is the public key of the client i.

2. The server and the client i can construct a same polynomial

$$f(x) = k_i + ax \tag{3}$$

3. Both the server and the client i randomly choose their secret shares $SS_i = (x_1, f(x_1))$ and $CS_i = (x_2, f(x_2))$, respectively, where these two shares are points on the polynomial.

4. After the secret shares are chosen, both the server and the client remove Q_i and the polynomial from their storage.

5. In a later query, the client i presents his share CS_i to the server. Combining with its own share SS_i, the server is able to reconstruct the polynomial and thus the secret key k_i can be recovered.

4 Key Management

There are many secret keys and secret shares that the server needs to keep track of. How to manage these keys and other useful information in an integrated fashion is another important design of the PASS scheme.

For each client i, the server keeps a profile storing the related keys and secret share, along with some other information that can be used later for key derivation, client authentication or access control. Table 1 below shows the contents of a client i's profile.

Table 1: Cloud's server keeps a profile for each client i.

Client ID	$h(k_i)$	SS_i	D_i
Request Counter		Security Label	

Inside a profile, it does not store the secret key k_i in a clear form, but in a hashed form $h(k_i)$, where h is a cryptographic hash function. Inclusion of the hashed key is for the purpose of client authentication. Recall that not storing shared secret keys anywhere in the cloud is our major design choice to enhance data privacy. The server also needs to store its secret share SS_i for a later key derivation when the client i makes a query with his share CS_i attached. Note that after the query is processed, the server must discard both k_i and CS_i from storage to ensure a better data

privacy protection. It is also useful to keep the client's public key D_i in the profile. If the server would like to initiate a short but sensitive talk to the client i, it can use D_i to encrypt the message since the server does not have the shared secret key k_i at that moment. For the request counter field in the profile, initially it is set to 0. The counter increments by one each time the client i makes a request to the server. The purpose of this counter is to prevent replay attacks. Using this counter approach, the client needs to keep a counter on his side too. The authentication procedure will be described in Section 5. Finally the security label in a profile is used to implement mandatory access control. Section 6 has more discussion on this.

5 Client Authentication

Upon receiving a service request from a client, it is important for the server to authenticate the client. This authentication can be another defense of data privacy. That is, only allowing the authenticated client to access his own data.

Based on the design choices in other supporting components, the client authentication procedure in the PASS scheme is quite simple without the need of challenge and response handshakes, which are usually required in other typical authentication protocols.

The procedure starts by a client i sending a service request message with an authenticator

$$Client\ i \xrightarrow{ENC_{D_s}(client's\ counter||CS_i)} Server$$

where $ENC_{D_s}(x)$ stands for using the server's public key to encrypt x and the symbol $||$ means concatenation. Upon receiving the request, the server performs the following steps:

1. Use its own private key d_s to decrypt the received authenticator, i.e., $DEC_{d_s}(ENC_{D_s}(client's\ counter\ ||\ CS_i))$, and then retrieve the client's counter and the secret share CS_i.

2. Use the decrypted CS_i and the stored SS_i (in profile) to recover the secret key k_i using Shamir's secret sharing algorithm as described in Section 3.2, and then compute the hash $h(k_i)$.

3. If the computed hash value matches the stored hash value and the client's counter is the same as the server's counter, then the client is authenticated.

Using this request counter approach, the resulting authentication procedure is efficient. However, integrity protection of both counters is a must to prevent deny of service attacks. Including the client's counter inside the authenticator is to prevent replay attacks. A replay attack occurs if an adversary overhears an authenticator from a client's legitimate request and later uses the authenticator to impersonate the client to the server. With the request counter approach in the PASS scheme, an old authenticator will not have a matching counter to the one in the server.

If the authenticator contains only request counter, a milder replay attack could occur. Without CS_i inside the authenticator, an adversary may overhear the authenticator from a client i with a counter value c_i and later use this authenticator to impersonate another client j when his counter value $c_j = c_i$. Of course, the adversary needs to have the capability to keep track of the counter values of both clients i and j. Including the secret share CS_i inside the authenticator, the above minor replay attacks can also be detected.

6 Access Control

The access control in the PASS scheme plays another important defense for data privacy, especially for malicious employees inside the cloud. To strongly control accesses, the PASS scheme chooses the mandatory access control model, in which no access right delegation can be granted. To implement such access control in cloud computing, a good security labeling is essential.

A security label consists of two parts $(security\ level,\ \{categories\})$, which can be assigned to either data items or subjects. Assigning to a data item, the security level indicates the data's security sensitivity and the set of categories describes kinds of information of the data. While assigning to a subject, the security level is the subject's security clearance and the set of categories describes what kinds of data the subject has right to access. The U.S. DoD has defined four security levels for their applications: top secret, secret, confidential and unclassified. With assigned security labels, a subject is allowed to access a data item if his security clearance is higher than or equal to the data's security level and his set of categories is a super-set of those assigned to the data item. This model is well-known and mature. Thus, no further refinement to the model in the PASS scheme is conducted. However, in the context of cloud

computing, it requires to define appropriate security levels and the granularity of categories, as well as to carefully identify the subjects which may be involved in any cloud operation. All these variables should be considered as a whole so that the resulting security labeling is not complex but flexible enough for access control in the cloud.

To reduce the complexity, the PASS scheme only defines two security levels: secret and non-secret. Table 2 lists the identified subject types. All cloud implementations may require these top-level subject types. However, they can be further divided into subtypes. Different cloud applications could have different subtypes. Thus, each cloud implementation should define subtypes based on their own discretion. All query processes can be invoked by the query server only. They are all built-in functions and should not be altered in any circumstances unless the few top classified employees decide to update them to newer versions. Cloud processes are not designated to manipulate clients' data and they could be any process related to other operations in the cloud. Thus, these processes do not have any access right to clients' data.

Table 2: Different subject types in cloud computing.

Clients	Should be allowed to access their own data only.
Query Servers	May include authentication, query, and encryption servers
Query Processes	Processes that are allowed to access clients' data.
Cloud Servers	All other servers in the cloud that are not related to query processing.
Cloud Processes	Processes that are not allowed to access clients' data.
Classified Employee	Few top trusted managers, cloud engineers, and system administrators. Able to revoke and check the status of query processes.
Unclassified Employees	Unclassified employees in the cloud. No right to perform any operations to the query processing black box.

Next, what granularity of category is appropriate in cloud computing? Obviously, data belong to different clients should be in different categories. Let C_i denote a category for each client i. For security levels, it's clients' responsibility to specify secret or non-secret to each data item. Thus, client i's data could be either ($secret$, C_i) or ($nonsecret$, C_i). In addition, each client i's profile is secret and labeled as ($secret$, C_i). Data labeled secret needs to be encrypted all the time in the cloud. Finally, the front-end authentication server, and the back-end query server and encryption server are all assumed trusted with the highest label ($secret, query\_system$), where the category $query\_system \supset \{all\ C_i\}$. In order for these servers to be trusted, the cloud providers should put extra effort to secure them such as firewalls, intrusion detection, physical protection, or their mix. Only these query servers are able to derive encryption keys upon receiving clients' queries.

Another notable design for access control in the PASS scheme is to build a black box for query processing. This black box can only be accessed by some selected classified employees with the highest security label ($secret, query\_system$). All other unclassified employees, servers and processes that are not related to query processing should absolutely have no right to access anything inside the box.

7 The PASS Scheme

With the five security components, Figure 1 shows a diagram that gives an overall picture of the proposed PASS scheme. We walk through the diagram step by step. Step 1 to Step 4 describe how to store a client's data in the cloud. This process includes key agreement, data encryption, and profile creation/update. Step 5 through Step 12 list the sequence of operations for a query processing.

Key Agreement and Data Encryption Steps:

1. The client i and the Authentication Server (`AS`) perform the key agreement procedure as described in Section 3. This procedure can be performed either for a periodic secret key update or when the client i first subscribes to the cloud.
2. `AS` creates or updates the client i's profile to record the newly agreed key k_i and the secret share SS_i.
3. `AS` sends a data encryption request, along with k_i, to the Encryption Server (`ES`).
4. `ES` encrypts or re-encrypts client i's data using k_i.

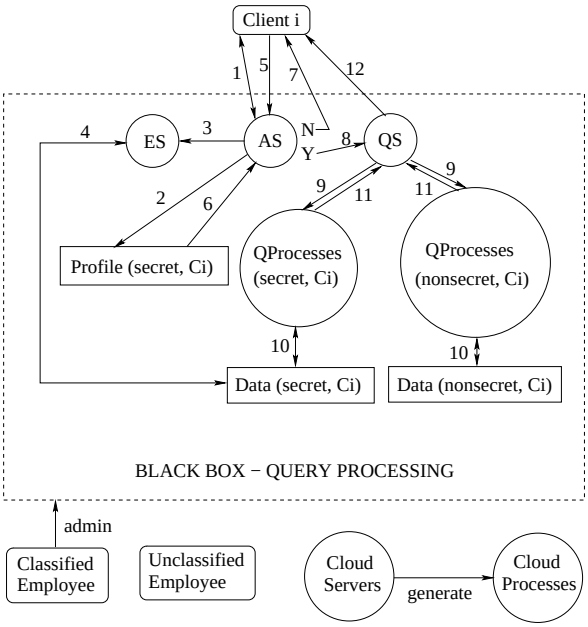

Figure 1: Query processing diagram, which is a black box to most employees in the cloud

Query Processing Steps:

5. Client i makes a Query Request `QR` to `AS`. An authenticator (see Section 5) is sent along with the request.

6. Upon receiving a `QR` from the client i, `AS` performs the authentication procedure described in Section 5 by first retrieving SS_i and $h(k_i)$ from the client i's profile. The shared key k_i will be derived in this step.

7. If the authentication fails, `AS` either sends back a deny message to the client i or simply discard the request. `AS` should forget (i.e., discard) k_i from its storage right away in this case.

8. If the authentication succeeds, `AS` forwards the query, along with k_i, to the Query Server (`QS`).

9. Depending on the query, `QS` invokes a query process, which could be either `QProcess` ($secret$, C_i) or `QProcess` ($nonsecret$, C_i), to perform operations requested in the query. A secret `QProcess` will get k_i from `QS`.

10. The invoked `QProcess` may need to access some client i's data. Based on the security labels defined in Section 6, `QProcess` ($nonsecret$, C_i) is only allowed to access data labeled ($nonsecret$, C_i), whereas `QProcess` ($secret$, C_i) is able to access data either labeled ($nonsecret$, C_i) or labeled ($secret$, C_i).

11. After the `QProcess` finishes processing, it returns a Query Result `QRT` to `QS`. This `QRT` should have the same security label as the `QProcess`. In case the `QRT` is secret, it will be encrypted before being returned to `QS`.

12. `QS` returns either encrypted `QRT` ($secret$, C_i) or not encrypted `QRT` ($nonsecret$, C_i) to the client i. Finally, all servers must discard k_i from their memory.

Note that data encryption/decryption may be performed by either the `ES` or a secret `QProcess`. `ES` is responsible to encrypt the whole database of a client i at the time a new shared key k_i is agreed, whereas each secret `QProcess` performs encryption/decryption to only related data in a query processing.

8 Conclusion

This paper presents a PASS scheme in cloud computing, which aims at protecting data privacy for clients. The scheme consists of five components as described from Section 2 to Section 6. Some designs of the scheme are innovative to the cloud computing field such as not storing data encryption keys anywhere in the cloud by secret sharing, using ECC for key agreement, using request counters and secret shares for authentication, forming a black box for query processing, as well as defining some top-level subject types in the cloud. A sequence of walk through steps for both the initial key agreement and a later query processing are also provided to show how these five components work together.

9 References

[1] G. Boss, P. Malladi, D. Quan, L. Legregni and H. Hall, "Cloud Computing", *IBM Corporation*, 2007.

[2] "Application Architecture for Cloud Computing", *rPath Inc.*, 2008.

[3] "Securing Microsoft's Cloud Infrastructure", *Microsoft Global Foundation Services*, 2009.

[4] "Cisco Cloud Computing - Data Center Strategy, Architecture, and Solutions - Point of view

White Paper for U.S. Public Sector", *Cisco Systems, Inc.*, 2009.

[5] S. Bennett, M. Bhuller and R. Covington, "An Oracle White Paper in Enterprise Architecture - Architectural Strategies for Cloud Computing", *Oracle Corporation*, 2009.

[6] "Amazon Web Services Launches - Amazon S3", *amazon.com*, 2006.

[7] C. Wang, Q. Wang, K. Ren and W. Lou, "Ensuring Data Storage Security in Could Computing", *17th International Workshop on Quality of Service*, 2009.

[8] V.D. Cunsolo, S. Distefano, A. Puliafito and M.L. Scarpa, "Achieving Information Security in Network Computing Systems", *8th IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009.

[9] J. Harauz, L.M. Kaufman and B. Potter, "Data Security in the World of Cloud Computing", *IEEE Security and Privacy*, 2009.

[10] C. Wang, Q. Wang, K. Ren and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Could Computing", *Proceedings of IEEE INFOCOM*, 2010.

[11] B. Grobauer, T. Walloschek and E. Stocker, "Understanding Cloud-Computing Vulnerabilities", *IEEE Security and Privacy*, 2010.

[12] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, 21(2), 1978.

[13] I. Black, G. Seroussi and N. Smart, "Elliptic Curves in Cryptography", *Cambridge University Press*, 1999.

[14] "An Elliptic Curve Cryptography (ECC) Primer, why ECC is the next generation of public key cryptography", *The Certicom 'Catch the Curve' White Paper Series*, 2004.

[15] "Recommended Set of Advanced Cryptography Algorithms - Suite B", *National Security Agency*, 2005.

[16] "Announcing the Advanced Encryption Standard (AES)", *Federal Information Processing Standards Publication 197*, 2001.

[17] A. Shamir, "How to Share a Secret", *Communications of the ACM*, 22(11), 1979.

[18] W. Diffie and M.E. Hellman, "New Directions in Cryptography", *IEEE Transaction on Information Theory*, IT-22, 1976.