Privacy Patterns in Public Clouds

Sashank Dara [*]
Security Technologies Group, Cisco Systems, Bangalore
email: krishna.sashank@gmail.com

January 25, 2014

Abstract

Internet users typically consume a wide range of cloud services. On the flip side in all of the cloud services users have to compromise on the privacy of their data. Typical cloud service provider has access to entire user data.

Conventional cryptographic techniques successfully achieve *communication privacy* and *storage privacy*. Further special techniques like Fully Homomorphic Encryption(FHE) schemes are required for achieving *computational privacy*(i.e privacy of data being processed at an untrusted server). FHE schemes are not yet efficient and practical but proven to be possible.

In this paper the patterns of *privacy* concerns in public clouds are identified, deployments of various cryptographic techniques are provided.

Keywords Cloud security and privacy, Computational Privacy, Storage Privacy, Security Patterns, Fully Homomorphic Encryption

1 Introduction

Cloud computing is well defined and standardized [4].The principles defining the essential characteristics, delivery methods and deployment models are now well laid and widely accepted. Cloud computing use cases are well known and documented in great deal by cloud use cases community [2]. Privacy concerns in cloud computing deployments are

Communication privacy : Privacy of data while in transit from client to cloud server.

[*]The author is also a part time Masters student at International Institute of Information Technology-Bangalore specializing in the area of cloud security and cryptography

Storage privacy : Privacy of archival data at cloud server. Usually backups, archives etc that are not often processed once stored.

Computational privacy : Privacy of data being processed at an untrusted cloud server.

Conventional cryptographic techniques like AES, RSA translate the plain text to random cipher text and they provide communication privacy and storage privacy. Further to achieve *computational privacy* one needs cryptographic techniques that allow arbitrary functions to be evaluated on encrypted data. Fully Homomorphic Encryption (FHE) schemes [3] ensure computational privacy of data by allowing one to perform computations on cipher text.

The pattern language defined in this paper helps identify patterns, possible solutions for privacy concerns in public clouds. Although homomorphic encryption schemes are highly impractical, we assume such schemes exist for the sake of completeness of solutions. A very informal introduction to encryption schemes used in this paper is provided in the appendix.

2 Patterns

A pattern language for cryptographic protocols has been formulated and many patterns were identified in the past [1]. Below are few patterns that address *privacy* concerns in different public cloud deployments.

2.1 Unshared Secret

2.1.1 Context

A consumer wants to benefit from a provider's cloud service by exporting data but is concerned about the privacy of said data.

2.1.2 Problem

How can we ensure *privacy* of exported data ? More specifically, how can a consumer safely transport her data, ensure privacy of such data while being processed or stored at a provider's server?

2.1.3 Forces

The forces that need to be considered when choosing to use this pattern are as follows

- **Privacy Concerns** Consumer wants to benefit from provider but concerned about *privacy*

- **Compromise** Unauthorized access of consumer's encrypted data by other consumers or other providers should not compromise the overall confidentiality and *computational privacy.*

2.1.4 Solution

Consumer generates secret key of symmetric encryption and pair of public, private keys of encryption schemes. Consumer encrypts the data using secret key and encryption algorithm of symmetric encryption scheme. Both consumer and provider authenticate with each other over SSL/TLS protocols that internally use combination of public key and symmetric encryption schemes. Consumer exports the encrypted data to provider after successful authentication. The choice of exact symmetric encryption scheme depends on the application. Traditional techniques like AES provide strong storage privacy. Futuristic applications that need computational privacy may encrypt the data using FHE schemes but perform operations using *evaluate* method. Consumer decrypts the data/results using *secret key* locally.

2.1.5 Consequences

Benefits

- **Privacy Concern** SSL/TLS protocols using the public key encryption schemes guarantee the *privacy* of data in transit. Usage of strong symmetric key encryption schemes like AES ensure storage privacy. In situations where computational privacy is needed futuristic homomorphic encryption schemes can be used to encrypt the data.

- **Compromise** Encryption schemes guarantee that accidental or intentional sharing of data with other consumer's or provider's would not leak anything.

Liability

- **Compromise** Encryption schemes ensure compromise of data will not leak anything but the compromise of the *secret keys* itself may compromise the entire system. This would be true of any cryptographic system and adequate measures need to be taken for safe guarding the keys.

2.1.6 Example Use cases

- An end user subscribes for an online file storage like DropBox or an on line photo sharing service like Flickr. A specialized client from the provider or browser is provisioned with a secret key. This key could be generated from a user supplied password. The client encrypts the files before they are uploaded into DropBox or Flickr.

- A small enterprise stores their encrypted sales records in cloud using CipherCloud's encryption gateway before being uploaded into Force.com
- An enterprise subscribes for SaaS application by a provider like customer relation management, enterprise resource planning etc .

2.2 Shared Secret

2.2.1 Context

In many cloud services consumer wants to work with multiple devices. The devices can be branch offices of an enterprise consumer or multiple personal gadgets of an individual consumer (say smart phone, tablet, laptop etc.). The consumer's devices usually have complete access privileges on the entire data. In addition to Unshared Secret pattern, Data is synced/shared by multiple devices. These devices need not be necessarily owned by the consumer it could be trusted device of a partner.

2.2.2 Problem

How can a consumer sync/share his encrypted data stored at a provider with multiple trusted devices ?

2.2.3 Forces

Forces that need to be considered when choosing to use this pattern are as follows

- **Device Management** Consumer shares data but with trusted devices.
- **Accountability** All the devices have same privileges but they should be accountable for their operations.
- **Individual Keys** Devices will have their own *public, private* key pairs and *secret keys* but encrypting with their respective *secret keys* would not result in encrypted data that can be operated upon collaboratively.

2.2.4 Solution

When a consumer wants to share/sync his data (stored at a provider) with his devices, she shares the *secret* key with them. If they secret key is generated by a password, then consumer provides the same password with all of his devices. All the devices authenticate themselves using their respective *public-private* key pairs. The data is encrypted with *secret key* (that is shared across all consumer's) before uploading to provider. If the cloud application is for storage then usage of conventional symmetric encryption schemes like AES would suffice. In a futuristic application, where computational privacy is required, operations requested by various devices can be done using *evaluate* method of FHE scheme by provider. Since the data is encrypted under same *secret* key by everyone the operations can be performed coherently.

2.2.5 Related pattern

While in Unshared Secret pattern we solved the problem when a single consumer utilizes provider's services, in this Shared secret pattern the problem when multiple trusted devices all having equal privileges is solved.

Benefits

- **Device Management** provider can authenticate consumers devices based on their respective *Public Key*.
- **Accountability** Since the consumers devices are identified while authentication, accountability of the operations can be maintained.
- **Individual Keys** Although consumers devices have their own key pairs, the actual encryption of data is done by common *secret key*

2.2.6 Example Use cases

In the following cases *secret key* can be shared among consumers trusted devices.

- An enterprise has its branch offices located at various different locations. All the branches have equal rights over data and applications with provider. For example, large enterprise having branch offices geographically located may purchase multiple such cloud gateways from CipherCloud (or similar provider). All the gateways can be provisioned with same shared secret key but with distinct public-private key pairs. Thus different branches can collaborate on their share data.
- An end user shares his personal data stored at provider with few of his trusted devices. For example, consumer may want sync his data stored at DropBox with his smart phone, tablet and a laptop. All the devices may have equal rights over common data. So the client application of all the end users may be provisioned by same secret key used for encryption.

2.3 Constrained Secret

2.3.1 Context

A much richer context is where consumer *authorizes* partner consumer's with different access control privileges. In many of real world scenarios, consumer would *authorize* partners to obtain results of only certain operations.

2.3.2 Problem

How can a consumer exercise granular Access Control for its partners on the encrypted data stored with provider while performing operations?

2.3.3 Forces

Forces that need to be considered when choosing to use this pattern are as follows

- **Partner Management** Partners are allowed but their addition, deletion and revocation etc should be controlled by consumer.
- **Authorization** Partner consumer's can do operations on data but confined to those they are *authorized* by consumer
- **Access Control** Consumer decides granular access control over data.

2.3.4 Solution

The solution is similar to that of SHARED SECRET Pattern. In addition to that, consumer defines a specialized *access control list (ACL)* of privileges to be given to various partner consumer's and shares the ACL with provider. No specific assumption on how the ACL should look like is made. It depends highly on the application. For example, the ACL can be as simple a privacy restrictions list on social network engine like Facebook. The ACL can also be similar to unix styled access control list for complex applications. Provider grants access to partner consumer's as per the *ACL*.

In special situations, where additional privacy is required, the ACL itself can be encrypted using FHE schemes to prevent any leakage of information .Granular access control in its plain form might *leak* some meta information of data as it has details specific to data and operations in finer detail. For example a *resize* operation might *leak* that data is an image file.

2.3.5 Related pattern

In SHARED SECRET pattern we solved the problem when multiple devices of consumers having equal privileges. In this current pattern consumers have different access privileges and such privileges can be controlled too.

2.3.6 Consequences

Benefits

- **Partner Management** consumer can control the addition, deletion or revocation of partner consumer's. consumer can simply resend the encrypted *ACL* to provider to enforce new privileges of partner consumer's.

- **Authorization** consumer can ensure authorization of partners.
- **Access Control** consumer can ensure fine granular access control with usage of access control lists.

2.3.7 Example Use cases

- A user may provide access privileges at granular file level while using DropBox. All the files owned by the user are marked as Owner. After encrypting the files with his secret key and uploading his files to DropBox, user also configures a list of permissions on who (from his contact list) can read, write or download those files. Similar features are available even in Google's Drive. User can have a shared password (to generate shared secret key) for his contacts to decrypt the files.
- Patient health records can be encrypted and stored at a provider. Hospitals can take authorization from patients on how their data should be accessed. Hospitals can then export the encrypted data to a provider. Patients can decide access control on their data, based on who they are for example insurance, doctors, researchers etc. provider can be completely blind about the data being processed
- Private Social Networking can be achieved on Public Internet . Each user can *authorize* on who (friends, family, everyone etc) can see their data (like photos, blogs etc) and also what operations (add, delete, tag) can be done without the platform ever knowing it.

3 Conclusions

Three different patterns for cloud privacy are identified in this paper. These three patterns address privacy concerns in gamut of use cases of cloud computing identified by the community [2].

4 Acknowledgments

I would like to thank Cisco Systems for supporting this work. I would also like to thank my research advisor Dr V.N. Muralidhara for all the initial discussions in coming up with the solutions for the patterns identified. I would also like to thank reviewers Andreas Fiesser, Anand Krishnaswamy for giving very insightful suggestions while shepherding.

References

[1] A. Braga, C. Rubira, and R. Dahab. Tropyc: A pattern language for cryptographic software. 1999.

[2] community. Cloud computing use cases. *A White Paper Produced by the Cloud Computing Use cases*, 2010. `http://bit.ly/gjxdL7`.

[3] C. Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009.

[4] P. Mell and T. Grance. The nist definition of cloud computing, special publication 800-145. *US Department of Commerce, Gaithersburg, MD*, 2011.

A Appendix

Below are the schemes that are used/referred in this paper[1].

Symmetric Key Encryption In a Symmetric Key Encryption (SKE) scheme there is only one Secret key. A message encrypted/decrypted with the same key. SKE has three algorithms

KeyGen Generates a secret key.

Encrypt Used for encryption of plain text to cipher text

Decrypt Used for decryption of cipher text to plain text

Public Key Encryption In a Public Key Encryption (PKE) scheme there are two keys namely Public Key and Private Key. Public Key is made known to everyone and Private Key is kept secret with the owner. A message encrypted with either of these keys cannot be decrypted with the same key, the other key in the pair should be used. PKE has three algorithms

KeyGen Generates pairs of Public and Private Keys

Encrypt Used for encryption of plain text to cipher text

Decrypt Used for decryption of cipher text to plain text

Fully Homomorphic Encryption A Fully Homomorphic Encryption (FHE) scheme can be based on symmetric key encryption or public key encryption schemes with an additional algorithm. Note that current popular theoretical constructions of these schemes are based public key encryption schemes. For sake of simplicity, we assume existence of homomorphic encryption schemes that are symmetric in nature for the purpose of this paper.

Evaluate Used for evaluating any arbitrary function on cipher text using the public key

[1]Formal definitions of the schemes is beyond the scope of this paper.