Grouper: A Group Finance Manager on Mobile devices Using Untrusted Servers

Department of Computer Science
Meng Li 201620728
Supervisor: Yasushi Shinjo
November 17th, 2016

1 Introduction

Conventional mobile applications are based on a client-server mode, which requires a central server for storing shared data. This means the users of the mobile applications must fully trust central servers and their application providers. Once a server is attacked by hackers, user information may be revealed because data is stored on only one server with cleartext. Finally, users may lose their data when service providers shut down their services.

Thus, we are going to implement a group finance manager application, Grouper, which is not relied on central servers. This means user data cannot be cracked easily and its service can be recovered after shutting down the old service. Recently, most popular way is using P2P(Peer to Peer) to transfer user data between devices. However, there is a obvious problem in such a no-server proposal. Data transfer can only be finished during two devices is online in a same time. Another problem is that the number of P2P connections will be increased fast as the number of users in a increases because each device can create connections to all of the other devices.

To address this problem without using P2P, we use multiple untrusted servers for synchronization. Data will divided to several pieces and uploaded to diverse servers. Each server can only save a piece of data temporarily, which means this piece will be deleted after a period time we specified. Those two rules explained above can ensure that user data cannot be cracked easily. In fact, we can regard untrusted servers as bridges between devices of group members. In addition, all devices of group members have saved a complete data set, data can be recovered even untrusted servers shut down. By this way, we think such a method can satisfy the requirements for an application which is running on untrusted servers, and not relied on central servers.

2 Methodology

We implement such a group finance manager application, Grouper, on mobile devices, which is not relied on central servers.

2.1 Group Finance Manager

The goal of a group finance manager is to develop an application that can record the income and expenditure of a small group. In Grouper, group members can create a record including money, classification, account, shop, remark and time. With data synchronization, group members can share their record with other group members, so that group income and expenditure information can be analyzed and shown to all members. Grouper using multiple untrusted servers to synchronize, can avoid defects of applications relied on central servers.

2.2 Synchronize with Multiple Untrusted Servers

We implement Grouper based on data synchronization by multiple untrusted servers rather than a single server. There are three principles in our proposal.

Firstly, data transfer by a server, but data does not save on this server permanently. Most current popular client-server applications store user data on several central servers, user data will not be deleted unless user deletes his account. Grouper use untrusted servers as a bridge for transferring data. Consider that a group include three members: Alice, Bob and Carol. Alice created a new record in her device, this record will be upload to untrusted servers, and the record on servers will eliminated after Bob and Carol synchronized it from servers. In other words, untrusted servers stores a record until it is be synchronized by all group members.

Secondly, server keep data temporarily. We define a period of time in which data can be saved in a server. In this paper, we set this period to 1 hour for our example situations. This means the record Alice uploaded to untrusted servers can only exist for 1 hour. After 1 hour since uploaded, this record will be deleted. Alice is required to resend this record to serves until all of other members has synchronized successfully. However, the longer keeping period means
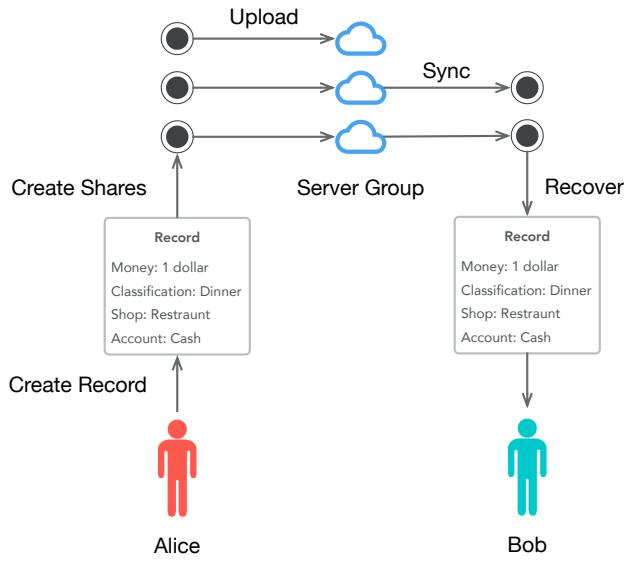
Figure 1: Flow of Synchronization

Figure 2: Sync Completely

the higher risk of data reveal. The most suitable period is influenced by the number of group members, security requirement, user expectations and others.

Thirdly, servers do not know the content of data. Saving data temporarily cannot ensure data security, because servers know the cleartext of data in this temporary period. For this problem, developers often encrypt data before uploading to servers, which need a private key to decrypt data in other devices. In order to distribute the private key, users should share it by themselves. In this paper, we use secret sharing, which can create a number of shares and distributing them to a set of participants[1], to share data between user devices and untrusted servers.

Based on the three core principles introduced above, our synchronization flow can be described as Figure 1. At first, Alice add a record, the Grouper creates a number of shares by secret sharing and uploads those shares to untrusted servers. When Bob is online, Grouper in Bob's device will download shares from servers and recover the new record uploaded by Alice. In this situation, Grouper can recover the record after getting more than two shares. In this process, each server is separated from other servers, and cannot access to other servers. This means user data can not be recovered unless you have permission access to more than two servers. In our proposal, only group members have permission to access these untrusted servers.

2.3 Shamir's Secret Sharing

Secret sharing which can create a numbers of shares, plays an indispensable role in protecting user data from getting lost, destroyed, or into wrong hands. In this paper, we are going to use Shamir's secret sharing, a form of secret sharing proposed by Shamir and
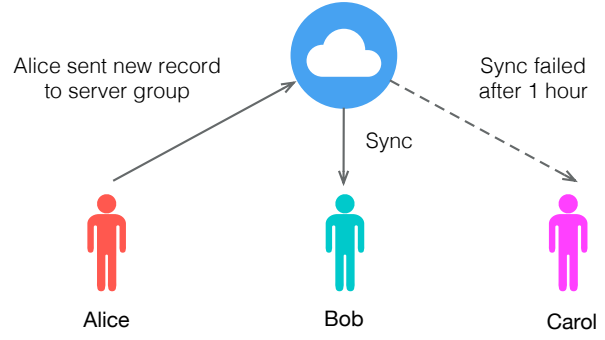
Blakley independently. In secret sharing scheme, a dealer securely shares a secret with a group of participants, by generating n shares using a cryptographic function[1]. At least k or more participants can reconstruct the secret, but $k-1$ or fewer participants can obtain nothing about the secret[2]. We describe this scheme as a function $f(k, n)$, n is the number of all shares, and k is the threshold to combine shares. A popular technique to implement threshold schemes uses polynomial interpolation ("Lagrange interpolation"). This method was invented by Shamir in 1979. Thus, we call it Shamir's Secret Sharing.

There are many implementations of Shamir's secret sharing by different programming languages. For developing an iOS app by Objective-C, we need an implementation without limitation of length, which supports Objective-C language and UTF-8 character set. c-SSS[3] is an implementation in original C code of Shamir's Secret Sharing by Fletcher T. Penney. c-SSS provide two main functions for generate shares and reconstructing them. Function *generate* can generate n shares by the string text with the threshold k. Function *extract* can recreate the text string after accessing to more than k shares.

2.4 Synchronize Completely

There is no doubt that Grouper should provide a reliable synchronization service. For Alice, once she created a new record in her device, all of other members in a group should synchronize this record, even if this record may be deleted by untrusted servers after 1 hour. We call this problem *Synchronize Completely*.

Figure 2 described the situation data cannot be synchronized completely. Alice sent a new record to untrusted servers in 10:00 AM and Bob synchronized it successfully before 11:00 AM. Carol forgot to synchronize this record before 11:00 AM. She cannot synchronize after 11:00 AM because servers has delete the record in 11:00 AM. To solve this problem, Alice is required to resend her new record until Carol synchronize it successfully. However, by what way can Alice get the information that all of other members in her group have synchronized. In other words, it

Column	Data Type	Introduction
sid	String	Physical unique identifier
content	String	JSON String of object
object	String	Name of object
count	Integer16	Count of sync time
resend	Boolean	Necessity of resending data
sendtime	Date	Data send time

Table 1: Columns of sender table

is an unreasonable requests for Alice to resend her record indefinitely. It is necessary for us to find a efficient way for notifying Alice that all members have synchronized successfully.

We are going to design a counter in clients as the solution to *Sync Completely* problem. Intuitively, this counter which can calculate the number of times other clients synchronized, works on the sender to decide resending record or not. The condition that sender device need not resend record can be described as equation 1.

$$\sum_{i=1}^{n} s_i = (m-1) \cdot k \tag{1}$$

In equation 1, we suppose that there are n untrusted servers , they are server 1, server 2,..., server n. We can also look at n from a different perspective, it is first parameter, the number of all shares, in the scheme $f(k,n)$ of Shamir's secret sharing. Each server saves a share, so n servers saves n shares. Each server should know how many times the share has been downloaded by clients. Thus, we use s_i to represent the number of times in server i. The sum of s_i is calculated in sender client. In the left of equation 1, m represents the number of group members, and k is the threshold in scheme $f(k,n)$. Obviously, a client can reconstruct data after getting k shares. If $m-1$ clients have reconstructed data, $(m-1) \cdot k$ shares is downloaded. That means clients except sender has synchronized successfully, so sender need not resend.

To implement such a counter, restriction about security and connection should be considered. On the one hand, servers cannot communicate with each other. Actually, one server of the group does not know the others exist in the group. On the other hand, clients cannot communicate with each other. Clients do know IP address or domain name of other device, so that they cannot send message with P2P connections. We decide to design a sync table described as figure 3 on both servers and client. In client, we call it sender table, while in server, we call it transfer table.

Columns of sender table is listed in table 1. Here, sid, content, and count of sync times are main columns. Sid is an unique identifier created by SQLite database in client, it is same as those in servers; content saves the JSON string converted by new record object; count saves the sync times of this object. In server, transfer table also has these 3 main columns
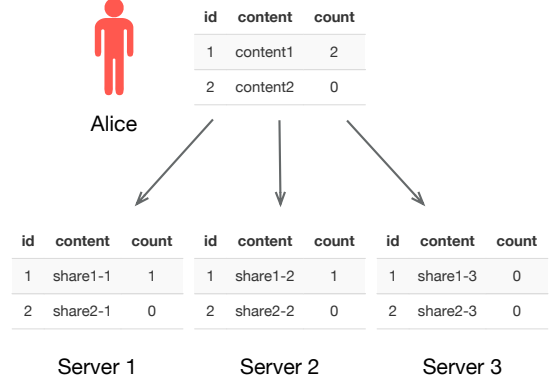

Figure 3: Sync Table

with different meaning. Sid is equals to which in client in order to point out the row corresponding to which in client; content saves one of the shares generated by secret sharing scheme; count saves the content sync times in server. Other auxiliary columns in transfer table will not be introduced here. Suppose the secret sharing scheme is $f(2,3)$, and only Bob have synchronized Alice's new record within prescribed time-limit. In figure 3, the column corresponding to Alice's new record, whose id is 1, is equals to those columns whose ids are 1 in server 1 and server 2. This is to say Bob downloaded two shares from server 1 and server 2. Client of Alice will know that by HTTP response from server 1 and server 2 when it resend data. It will send for one more time, because it's count is 2, which is less than 4, the threshold Alice need not resend.

3 Architecture

Grouper includes an iOS app for clients and a web service in server.

3.1 Architecture in Client

In Methodology, we have introduced how to sharing a string with other deices via multiple untrusted servers. In this paragraph, let us talk about persistent store and data synchronization before architecture. With untrusted servers, Grouper have to storage all data on mobile devices with an object-oriented way. Grouper use Core Data[4], a native iOS framework to manage the model layer objects, which provides generalized and automated solutions to common tasks associated with object life cycle and object graph management, including persistence, to storage and operate data as object. Sync[5] by Elvis Nuez, a modern swift JSON synchronization framework to Core Data, can help we complete data synchronization easily in Grouper by parsing a JSON response and getting it into Core Data. With Sync based on Core Date, we can concentrate on sharing with untrusted servers rather than data storage and synchronization.
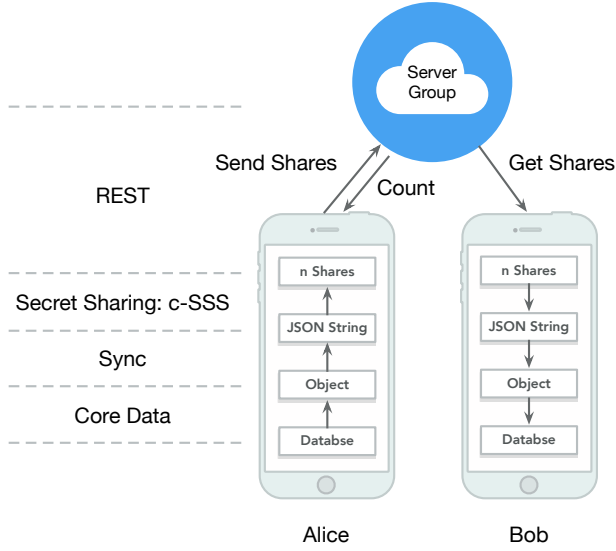
Figure 4: Architecture

c-SSS by Fletcher T. Penney can generate shares from a JSON string saved on sync table. Grouper develop with Objective-C language, which is compatible with original C code. Grouper will generate shares and send them by REST API to servers until this record need not to resend. Response of REST API contains the count of sync times in servers, for calculating the sum and deciding wether resend or not.

3.2 Necessity of Having a Web Service

Many applications with synchronization is based on commercial cloud services like Amazon S3, Google Cloud or iCloud. It is necessary that Grouper has its own web service running on individual server rather than using commercial cloud services directly for following reasons:

- Data Storage: untrusted servers save share generated by secret sharing.
- Provisionality: web service ensure that share will be delete within prescriptive time.
- Security: only group members who has access key can download share from server. In other word, web service plays a role of access control.
- Counter: web service palys a role of counter to calculate sync times in server.

4 Related Work

DepSky[6] is a system that storage encrypted data on servers and run application logic on the client-side[7]. DepSky provide a storage service that improves the availability and confidentiality provided by commercial storage cloud services. *Cloud-of-Clouds* is the core concept in DepSky. It represents that DepSky is a virtual storage cloud, which is accessed by its users by invoking operations in several individual severs. DepSky saves encrypted data in commercial storage cloud services and do application logic in individual servers. In Grouper, untrusted servers undertake responsibility of temporarily data storage and message delivery with server-side computation. Thus, Grouper cannot only use commercial storage cloud services.

Mylar[8] stores sensitive data encrypted on the server, and decrypts that data only in users browsers. Developer should uses Mylars API to encrypt a regular (non-encrypted) web application, and users decrypt data by browser extension. Like Grouper, application in Mylar can control how user data is shared[7]; unlike Grouper, Mylar build its system on a browser with browser extension while Grouper use mobile devices as client.

Of course, there are many applications or frameworks synchronize with untrusted network and servers. Compared to them, Grouper use secret sharing and temporary data storage with untrusted servers to protect user data from malicious attacking.

5 Future Work

An application running in client has been developed now. In future, we are going to develop a web service running on multiple untrusted servers to manage data temporary storage and provide REST API for mobile devices. This web service developed by Java EE should be install by group owner, in order that nobody except the group owner know all the IP address of servers. Thus, security of Grouper can be guaranteed.

6 Conclusion

This paper introduced Grouper, a group finance manager, using secret sharing and data synchronization on mobile devices. Grouper is synchronized by multiple untrusted servers, each server of which does not know existence of others. Each server only saves one of the shares generated by secret sharing temporarily, to ensure that user data cannot be cracked easily.

References

[1] Guillaume Smith, Roksana Boreli, and Mohamed Ali Kaafar. A layered secret sharing scheme for automated profile sharing in OSN groups. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 487–499. Springer, 2013.

[2] Liao-Jun Pang and Yu-Min Wang. A new (t, n) multi-secret sharing scheme based on shamirs secret sharing. *Applied Mathematics and Computation*, 167(2):840–848, 2005.

[3] Fletcher T. Penney. c-SSS, an implementation of shamir's secret sharing.

[4] Apple Inc. Core data programming guide. https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html.

[5] Elvis Nu nez. Sync.

[6] Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. Depsky: dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, 9(4):12, 2013.

[7] Frank Wang, James Mickens, Nickolai Zeldovich, and Vinod Vaikuntanathan. Sieve: cryptographically enforced access control for user data in untrusted clouds. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 611–626, 2016.

[8] Raluca Ada Popa, Emily Stark, Steven Valdez, Jonas Helfer, Nickolai Zeldovich, and Hari Balakrishnan. Building web applications on top of encrypted data using mylar. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 157–172, 2014.