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

Meng Li, 201620728
Supervisor: Yasushi Shinjo

November 17th, 2016

1 Introduction

Conventional mobile applications are built based on a client-server mode. This requires central servers for storing shared data. The users of such mobile applications must fully trust central servers and their application providers. Once this server is compromised by hackers, user information may be revealed because data is often stored on the server in cleartext. Users may lose their data when service providers shut down their services.

A popular approach to address the central server problem is using P2P(Peer to Peer) to transfer user data between devices. However, there is an obvious problem in such a P2P approach. Data transfer can only be finished during two devices are online at the same time. Another problem is that the number of P2P connections becomes large fast as the number of users increases.

We are implementing a mobile application that is not relied on trusted central servers. Concretely, we are developing a group finance manager application. We call it Grouper. Unlike P2P systems, we use multiple untrusted servers for data transfer. Data is divided into several pieces and uploaded to diverse servers. Each server can only keep a piece of data temporarily. A piece will be deleted after a period of time. Those two methods ensure that user data cannot be cracked easily. In addition, all devices of group members keep a complete data set, and data can be recovered even untrusted servers shut down.

2 Design

We design a group finance manager application, Grouper, on mobile devices. This does not rely on trusted central servers.

2.1 Group Finance Manager

In Grouper, group members generate records about money, classifications, accounts, shops, remark and times in their devices. For example, Alice records that she ate breakfast (classification) in the center canteen (shop) at 8:00 AM (time) on November 1st, 2016, and she paid her breakfast by credit card (account). These items from existing ones are created by users as they want. They can choose an item when they want to add a new record.

With data synchronization, they can also share their records with other group members. Therefore, group income and expenditure information are analyzed and shown to all members. Grouper uses multiple untrusted servers to synchronize and avoids relying on trusted central servers. To create a group in Grouper, the owner of this group needs to register and set access keys in untrusted servers. The owner passes the access keys to group members by a face-to-face way.

2.2 Shamir's Secret Sharing

In Grouper, secret sharing plays an indispensable role in protecting user data from getting lost or destroyed. In a secret sharing scheme, a dealer securely shares a secret with a group of shares by generating n shares using a cryptographic function[1]. At least k or more shares can reconstruct the secret, but $k-1$ or fewer participants can obtain nothing about the secret[2]. We describe this scheme as a function $f(k,n)$, where n is the number of all shares, and k is the threshold to combine shares. We use Shamir's Secret Sharing that is a popular technique to implement threshold schemes.

2.3 Data Synchronization Using Multiple Untrusted Servers

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

Firstly, a server transfers data as similar to a router, but does not keep it permanently. Most current popular client-server applications store their user data on several central servers, and the user's data will not be deleted unless the user deletes his account. Grouper uses untrusted servers as a bridge for transferring data. Consider that a group includes three members: Alice, Bob and Carol. Alice creates a new record in her device, this record is uploaded to untrusted

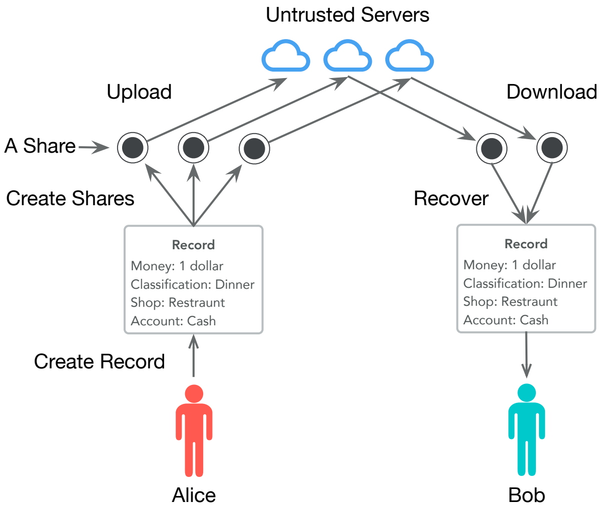


Figure 1: Flow of synchronization.

servers, and the record on servers will be deleted after Bob and Carol download it from servers.

Secondly, a server keeps data temporarily. We define a period of time in which data can be kept in a server. In this paper, we set this period to 1 hour for our example situations. The record Alice uploaded to untrusted servers can only exist for 1 hour. After 1 hour since uploaded, this record will be deleted. Alice requires to upload this record again to serves until all of other members have downloaded successfully. The longer keeping period means the higher risk of data reveal. We will find the suitable periods is as the number of group members, security requirement and others.

Thirdly, servers do not know the cleartext of data. Keeping 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, and this needs a decryption key to decrypt data in other devices. In order to distribute the decryption key, users should share it by themselves. In Grouper, we do not encrypt data but we use a secret sharing scheme as described in Section 2.2. Servers do not know the cleartext of a share generated by a secret sharing scheme.

Figure 1 describes our synchronization flow based on these three principles. At first, Alice adds a record and Grouper creates three shares by a secret sharing scheme where n is three and k is two. Next, Grouper uploads those shares to multiple untrusted servers. When Bob is online, Grouper in Bob's device downloads two shares from servers and recovers the new record created 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 these servers cannot recover user data because they do not have permission to access other untrusted servers. In our proposal, only group members have permission to access these untrusted servers.

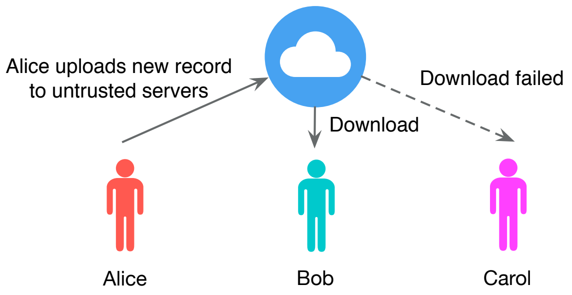


Figure 2: Unreliable synchronization.

2.4 Reliable Synchronization

Grouper should provide a reliable synchronization service. For example, once Alice, who is a user in a group, creates 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 a period of time. We call this problem *Reliable Synchronization.*

Figure 2 describes an unreliable situation that data cannot be synchronized. Alice sends a new record to untrusted servers at 10:00 AM and Bob downloads it successfully at 10:30 AM. Carol becomes online at 11.30 AM and fail to download shares of this record because they has been deleted by servers. To solve this problem, Alice should upload her new record again until Carol downloads it successfully. However, it is hard for Alice to get the information that all of other members in her group have downloaded. We must find an efficient way for notifying Alice that all members have synchronized successfully.

We must consider a further problem that Carol, who is offline, can attack this group by being lazy. Alice, the record creator, has to upload shares to untrusted servers indefinitely. We are addressing this problem now by holding a list group members in untrusted servers.

2.5 Creating a Group

A group in Grouper is created by its owner. For example, Alice is a group owner and she registers her group in multiple untrusted server by submitting her own user information, group ID and group name. Here, group ID and group name are assigned by Alice and they should be same in all untrusted servers of this group. In each untrusted server, its Web service creates this new group successfully and returns a master key including the highest privilege for Alice.

Figure 3 shows the procedure to add a new member to a group. At first, when Bob wants to join Alice's group, he meets Alice by a face-to-face way. After establishing a connection between Alice's and

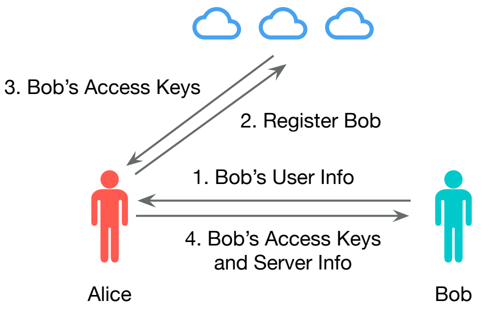


Figure 3: Adding a new member.

Bob's devices, Bob's device sends his user information to Alice's device at first. Alice creates a new user called Bob in the three untrusted servers of her group. Servers return three access keys for Bob to Alice. At last, Alice sends access keys and user information to Bob, so that he can access those untrusted servers.

3 Implementation

Grouper consists of clients that run an iOS application and multiple untrusted servers that run a Web service. In Section 2, we have introduced how to share strings with other deices via multiple untrusted servers. In this section, we describe persistent store and data synchronization. Grouper stores all data on mobile devices with an object-oriented way.

3.1 iOS Application

Grouper uses Core Data[3], a native iOS framework to manage the model layer objects. Core Data provides generalized and automated solutions to common tasks associated with object life cycle and object graph management, including persistence. Sync[4] is a modern JSON synchronization framework for Core Data, and performs data synchronization by parsing JSON strings. With Sync based on Core Data, we can concentrate on sharing with untrusted servers rather than data storage and synchronization.

In this paper, we use c-SSS[5], an implementation of Shamir's Secret Sharing in the C language. This implementation supports UTF-8 character set without limitation of length. It provides two main functions: *generate*, which generates n shares by a string text with the threshold k, and *extract*, which recreates the text string from more than k shares.

c-SSS generates shares from a JSON string. There shares are stored as an entity in Core Data. Grouper generates shares and upload them by a REST API to servers.

Figure 4 describes the architecture of the iOS application running on clients. Grouper stores all data on mobile devices in an object-oriented way. Grouper uses Core Data to store and operate data as objects. *NSManagedObject+HYPPropertyMapper* is a part of the Sync framework and creates JSON string from objects. REST API provided by web service in untrusted servers transfers data between clients and servers. User information and server information are transferred between clients using the Multipeer Connectivity framework.

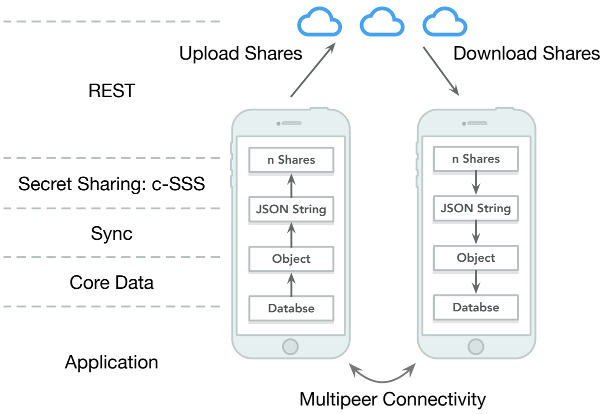
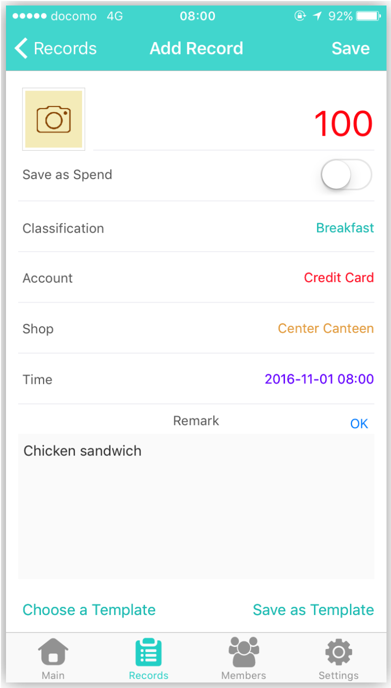


Figure 4: Architecture of the iOS application.



100

Save as Spend

Classification Breakfast

Account Credit Card

Shop Center Canteen

Time 2016-11-01 08:00

Remark

Chicken sandwich

Figure 5: A screen shoot of adding a record.

Figure 5 is a screen shoot of the view *Add Record*. In this view, users add a new record including amount of money, classification, account, a shop, time and remark in this view.

3.2 Web Service

Many applications with synchronization use commercial cloud services like Amazon S3, Google Cloud and others. Grouper needs its own Web service rather

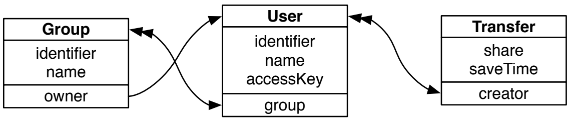


Figure 6: Entities in the Web service.

than using commercial general cloud services for following reasons:

- The Web service supports reliable synchronization.
- The Web service ensures that shares are deleted after a prescriptive time.
- The Web service allows only group members who have access keys to download shares.

Our Web service runs on the Tomcat server that is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. We use the Spring Web model-view-controller (MVC) framework to create our RESTful API. Like in the iOS client, we use Hibernate, an open source Java persistence framework, to save and operate object in Web service. Hibernate enables developers to more easily write applications whose data outlives the application process.

There are three entities including *Group*, *User* and *Transfer* in our Web service. Figure 6 describes the attributes of these entities and relationship between them. The relationship between *Transfer* and *User* and the relationship bwtween *User* and *Group* are *many-to-one*. The *Group* entity has the unique identifier, group name and its owner. The *User* entity has basic information of a user, access key for this user, and group of this user. The *Transfer* entity has the share generated with a secret sharing scheme, the time when the user uploads it and its creator. For each user, there is a unique access key for him in an untrusted server. This ensures that only he can remove or modify what he uploaded to this server. For a group, one of a user is its owner who has a highest privilege of this group.

4 Related Work

DepSky[6] is a system that stores encrypted data on servers and runs application logic on the client-side. DepSky provides a storage service that improves the availability and confidentiality by using commercial storage services. *Cloud-of-Clouds* is the core concept in DepSky. It represents that DepSky is a virtual storage, and its users invoke operations in several individual severs. DepSky keeps encrypted data in commercial storage 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.

Mylar[7] stores encrypted and sensitive data on a server, and decrypts this data only in users browsers. Developers of Mylar use its API to encrypt a regular(non-encrypted) Web application, and users decrypt data by a browser extension. Like in Grouper, applications in Mylar can control how user data is shared. Mylar builds its system on a browser with extension while Grouper uses mobile devices.

There are many applications and frameworks that use untrusted network and servers. Compared to them, Grouper uses secret sharing and temporary data storage with untrusted servers to protect user data from malicious attacking. Using secret sharing is more convenient and faster than using data encryption because clients need not to distribute decryption keys and perform encryption computation.

5 Conclusion

This paper introduces Grouper, a group finance manager that synchronizes data among mobile devices with multiple untrusted servers. Grouper uses secret sharing and temporary data storage services. Grouper consists of clients that run an iOS application and multiple untrusted servers that run a Web service. Each server of Grouper does not know the others and keeps one piece of the shares generated by secret sharing temporarily to perform user data cannot be cracked easily. We connect clients and multiple untrusted servers by REST API. We are developing a robust protocol of reliable synchronization using multiple untrusted servers.

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] Apple Inc. Core Data Programming Guide, Guides and Sample Code. `https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/`.

[4] Elvis Nunez. Sync, modern Swift JSON synchronization to Core Data. `https://github.com/SyncDB/Sync`.

[5] Fletcher T. Penney. c-SSS, an implementation of Shamir's secret sharing. `https://github.com/fletcher/c-sss`.

[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] 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.