Secure Social Media

A WEBAPP MADE WITH REACT, FIREBASE AND CRYPTOJS

Rvail Naveed - 17321983 | Advanced Telecommunications | 1-03-2020

https://github.com/rvailnaveed/securesocialmedia

Demo: https://youtu.be/KfQolgWm8tw

Introduction

The goal of this project was to create a secure social media application using encryption techniques. Only members of a secure group should have the ability to see each other's decrypted posts.

The technologies I used to accomplish this were:

- ReactJS

- CryptoJS for encryption + decryption of posts/messages

- Firebase Firestore for backend (storing of posts, users, groups)

Approach

Populating the Database

As this is a proof of concept, I populated the backend with static, generic posts.

Each of the 5 dummy users has 3 dummy posts. This was done to illustrate the effects of encryption and decryption and how posts are displayed depending on group inclusion.

You can also create new posts, which are added to the feed and pushed to the Firestore backend after being encrypted.

Most of the logic happens inside Home.js, with the other components being for UI to display the data coming from the backend.

The demo video can be referenced with the following explanations to see the project in action.

Encryption

When the user enters new text to be posted and presses the post button, this triggers a call-back to a function.

The function then does the following steps:

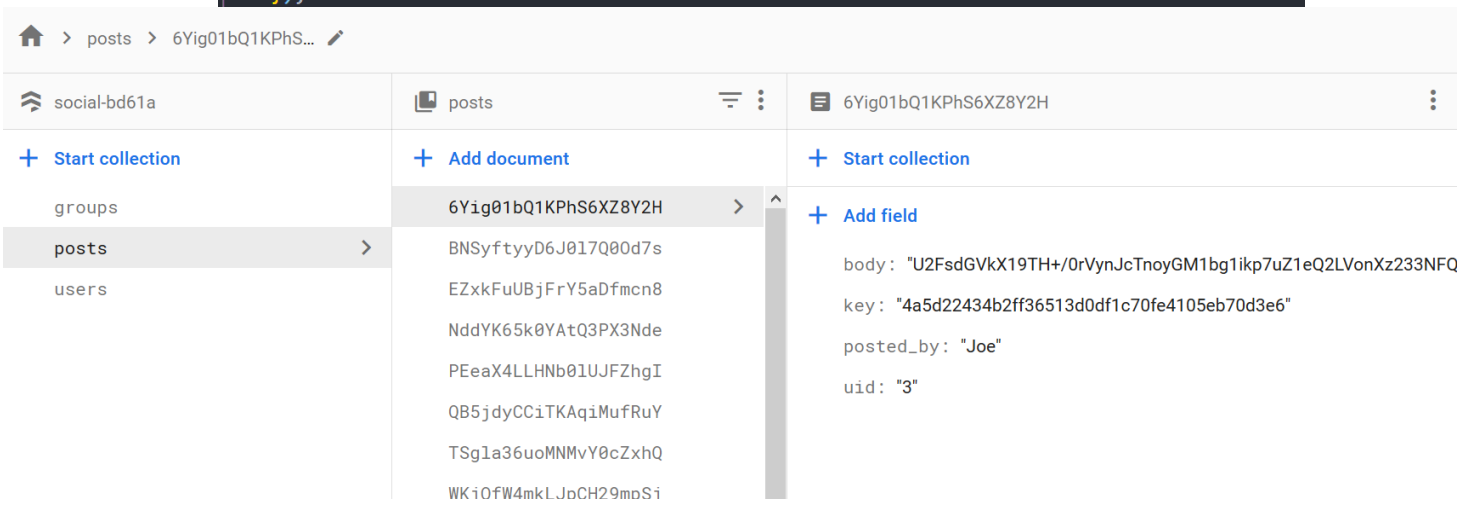
- Fetch specified text from box
- Generate a key for use with the AES encryption algorithm
- Encrypt the text using the key

- Create a post object that contains the key and the encrypted body
- Send to backend
- The posts store the key to decrypt them, so this also functions as a **key management system.**

```
// Generate key for new post
var current_date = (new Date()).valueOf().toString();
var random_string = Math.random().toString();
var key = Crypto.createHash('sha1').update(current_date + random_string).digest('hex');

var encrypted = CryptoJS.AES.encrypt(post, key).toString();

const db = firestore;
var postsRef = db.collection("posts");
postsRef.add({
    uid: 7,
    posted_by: "current_user",
    body: encrypted,
    key: key
});
```



Decryption

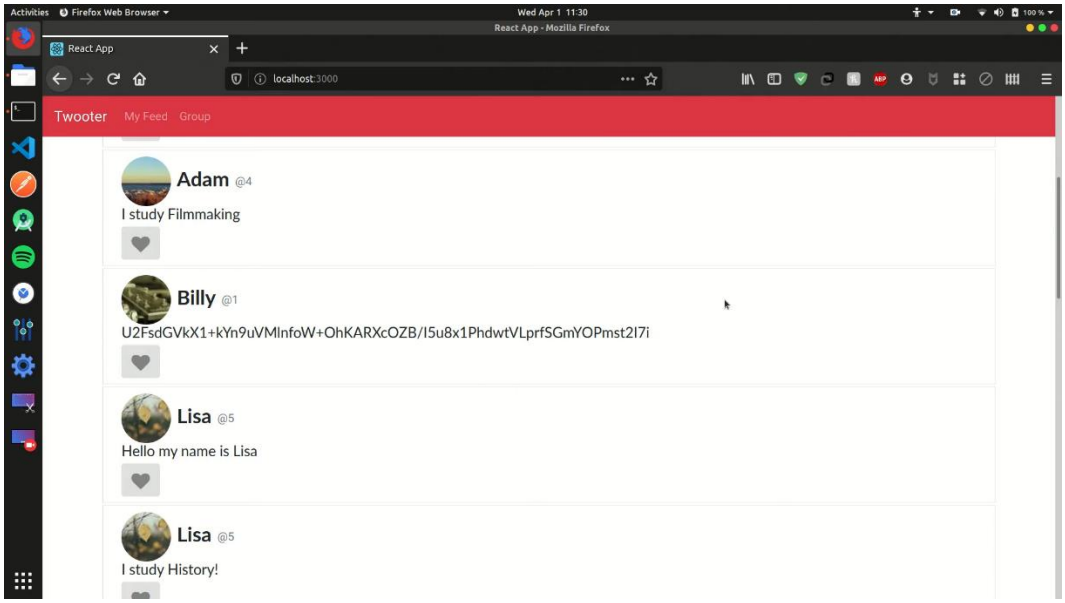
When the home screen component is being loaded, the following steps occur:

- Posts are fetched from the Firestore backend
- It is checked whether the user who made the post is part of the secure group. (Only members of a group can see each other's posts)
- If they are: the post is decrypted with the attached key and displayed in plain text.
- Otherwise, the encrypted post body is shown.

```
postsRef.get()
.then(snapshot => {
    snapshot.docs.forEach((post) => {
        var postData = post.data();
        var uid = postData.uid;
        //console.log(uid)
        var posted_by = postData.posted_by;
        var body = postData.body;
        // decrypt if part of group
        for(var i = 0; i < this.state.groupMembers.length; i++){
            console.log(typeof uid, typeof groupMembers[i].uid)
            if(uid == this.state.groupMembers[i].uid || posted_by === "current_user"){
                var key = postData.key
                body = CryptoJS.AES.decrypt(postData.body, key).toString(CryptoJS.enc.Latin1);
            }
        }

        if (uid === 7 && posted_by === "current_user"){
            uid = 'current_user';
            posted_by = 'You';
        }

        // post will contain decrypted or encrypted body depending on group inclusion
        posts.push({
            uid: uid,
            id: post.id,
            name: posted_by,
            body: body
        })
        return;
    })
    this.setState({allPosts: posts})
    this.loaded = true;
})
```



Billy is not part of the secure group, so his post is encrypted

Group Management

On the groups screen, the user can toggle which other users are part of his/her secure group. Only people within a group can see each other's decrypted posts, otherwise the encrypted text is shown.

Adding members:

- Switch toggle for user on
- This creates a new user object to add to the firebase backend group
- Posted to firebase
- Next time the feed loads, it will reflect the changes by now showing the decrypted post of the user who made it which was previously encrypted.

```
// a new member has been added to firestore backend
groupRef.update({
    "members": Firebase.firestore.FieldValue.arrayUnion({"name": this.props.userInfo.name, "uid": this.props.userInfo.id})
})
```

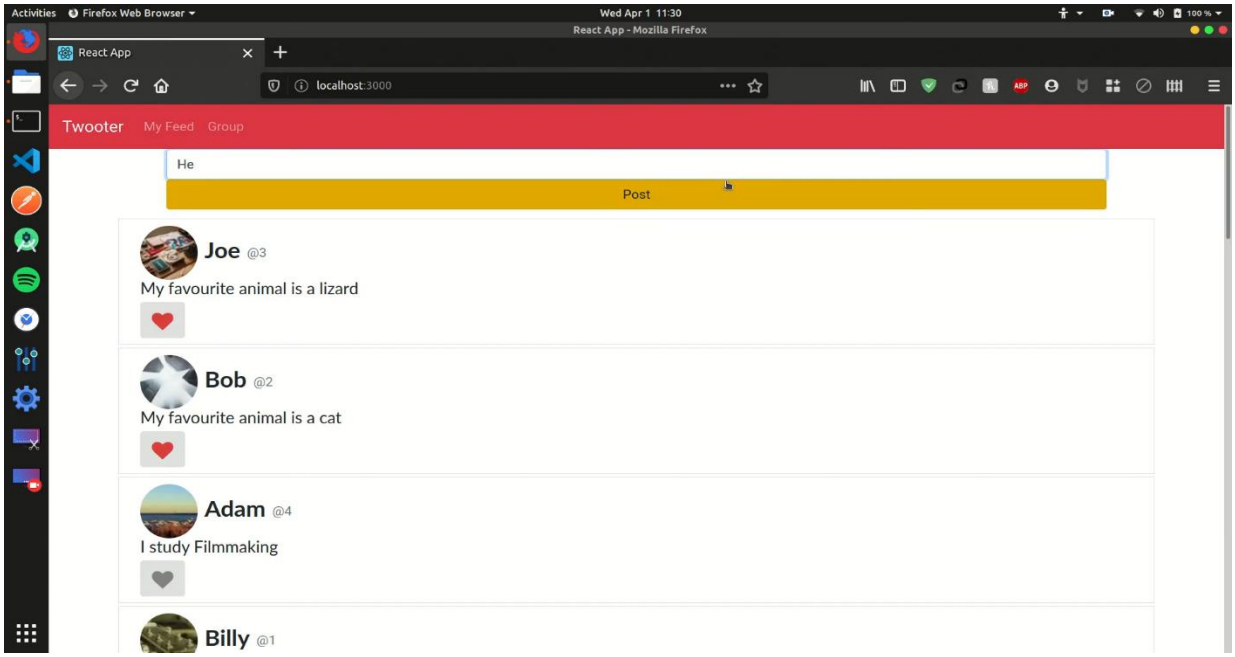
Removing members:

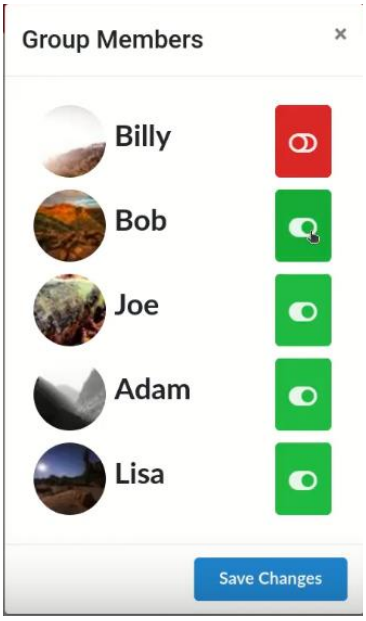
- Switch toggle off
- Removes member from group in firebase backend
- On next feed load, the removed users feed will be encrypted

```
// the member has been removed from group, so we reflect this in the backend
groupRef.update({
    "members": Firebase.firestore.FieldValue.arrayRemove({"name": this.props.userInfo.name, "uid": this.props.userInfo.id})
})
```

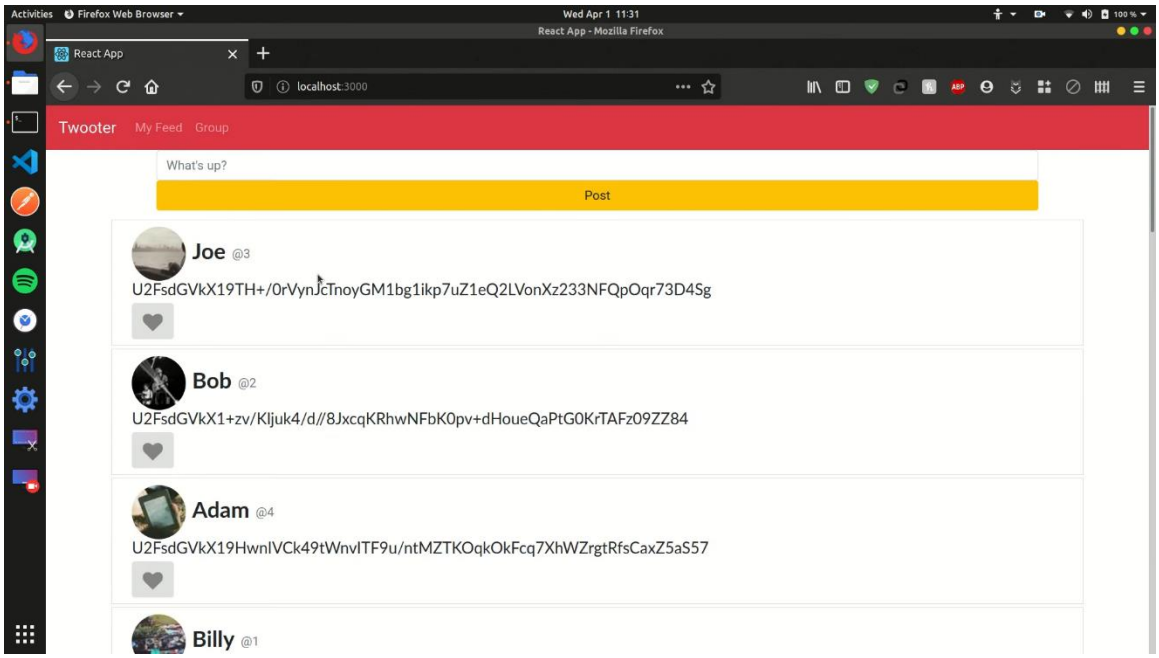
Example Interaction

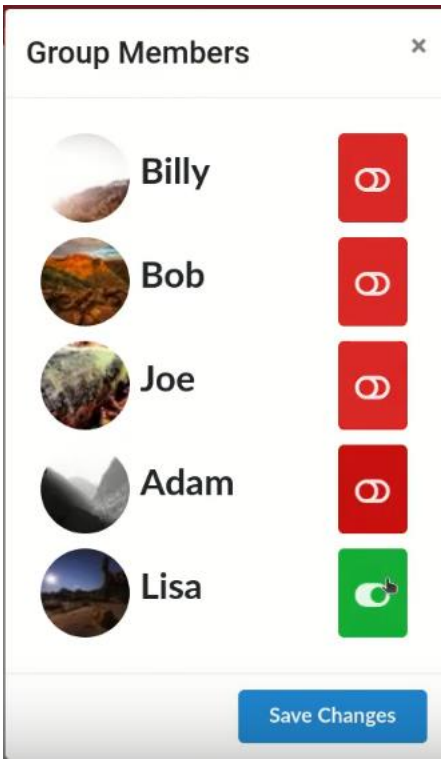
Current Setup:





Changing member access:





Conclusion

Overall, I was able to implement all the features specified in the brief. If I were to do this project again, I would choose a better method of encryption. Maybe an algorithm that makes the use of a public and private key setup. I would also like to add user context so that different users could log in and interact with each other, making my implementation more than a proof of concept and more like a real-world social media application.

The demo video for this project can be found on the first page of this report.

Appendix

It would be cumbersome to include all code here, so I have listed the main components links along with their function:

- Home.js – Where the logic for encryption/decryption resides, houses the posts feed.
- GroupMember.js – Has the logic for Groups (Inclusion, Exclusion, toggling group members)
- Group.js – Shows current status of each GroupMember above and allows toggling.
- Post.js – Component to display post content fetched from backend.
- Feed.js – Houses Post Components.
- UserFeed.js – Feed for new user posts.
- App.js – Wrapper component.

Link to full code repository on title page.