



CipherChat

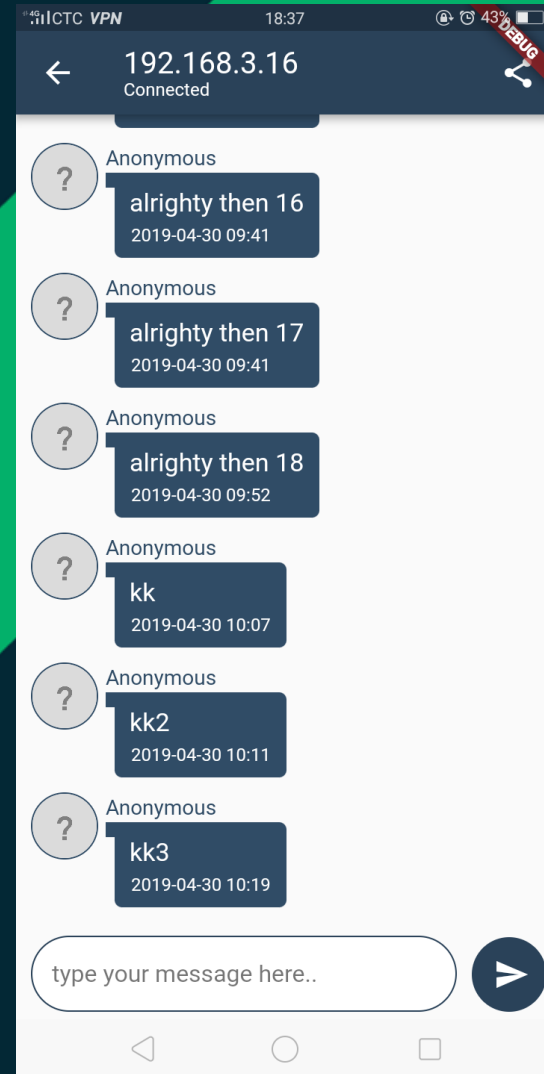
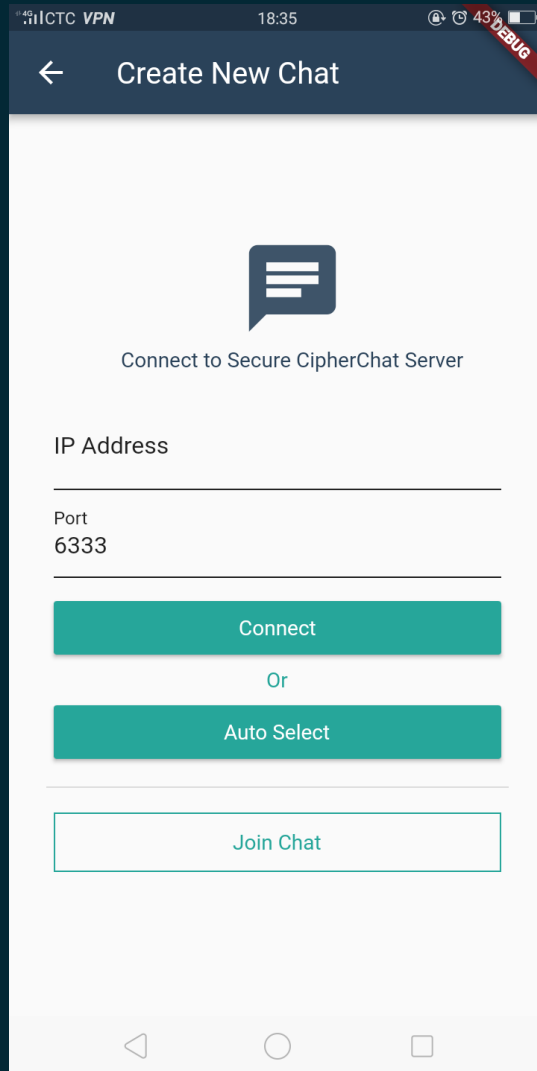
A Secure, Decentralized Chat Application

Group: K.O.D Inc
Members: Ottor Mills
Kenneth Anglin
Nicoy Smith
David Thomas

Motivation


In today's world the internet has become a hotbed of hacking plots at data harvesting. CipherChat was created based on the premise that everyone should be entitled to their privacy regardless of how ignorant that individual may be.

User Interface



Tools Used

The following languages contributed to completion of this project

1. TypeScript (Server)
 2. JavaScript (Server & Client)
 3. Dart (Client)
 4. SQL (Server & Client)
 5. Bash Scripting Language (Server)
- 

Database Schema

MYSQL Database schema (TypeScript)

You, 10 hours ago | 1 author (You)

```
class GroupsTable{
  public tableName:TableName = new TableName("groups");
  public groupId:TableColumn = new TableColumn(this.tableName.getTableName(), "gid", columnTypes.integer, true);
  public joinKey:TableColumn = new TableColumn(this.tableName.getTableName(), "joinKey", columnTypes.varchar, false);
  public timestamp:TableColumn = new TableColumn(this.tableName.getTableName(), "ts", columnTypes.timestamp, false);
}
```

You, 10 hours ago | 1 author (You)

```
class ParticipantsTable{
  public tableName:TableName = new TableName("participants");
  public participantId:TableColumn = new TableColumn(this.tableName.getTableName(), "pid", columnTypes.integer, true);
  public groupId:TableColumn = new TableColumn(this.tableName.getTableName(), "gid", columnTypes.integer, false);
  public username:TableColumn = new TableColumn(this.tableName.getTableName(), "username", columnTypes.varchar, false);
  public publicKey:TableColumn = new TableColumn(this.tableName.getTableName(), "publicKey", columnTypes.varchar, false);
  public publicKey2:TableColumn = new TableColumn(this.tableName.getTableName(), "publicKey2", columnTypes.varchar, false);
  public timestamp:TableColumn = new TableColumn(this.tableName.getTableName(), "ts", columnTypes.timestamp, false);
}
```

You, 10 hours ago | 1 author (You)

```
class MessagesTable{
  public tableName:TableName = new TableName("participants");
  public messageId:TableColumn = new TableColumn(this.tableName.getTableName(), "mid", columnTypes.integer, true);
  public groupId:TableColumn = new TableColumn(this.tableName.getTableName(), "gid", columnTypes.integer, false);
  public participantId:TableColumn = new TableColumn(this.tableName.getTableName(), "pid", columnTypes.integer, false);
  public message:TableColumn = new TableColumn(this.tableName.getTableName(), "message", columnTypes.varchar, false);
  public timestamp:TableColumn = new TableColumn(this.tableName.getTableName(), "ts", columnTypes.timestamp, false);
}
```

You, 10 hours ago | 1 author (You)

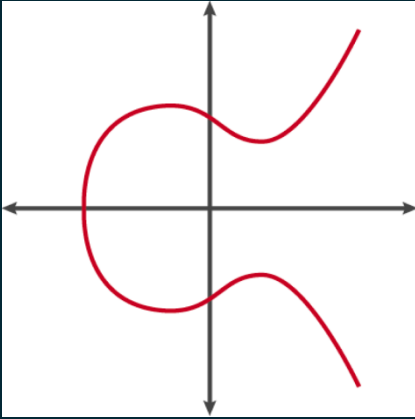
```
class CompositeKeysTable{
  public tableName:TableName = new TableName("compositeKeys");
  public compositeKeyId:TableColumn = new TableColumn(this.tableName.getTableName(), "cpid", columnTypes.integer, true);
  public messageId:TableColumn = new TableColumn(this.tableName.getTableName(), "mid", columnTypes.integer, false);
  public groupId:TableColumn = new TableColumn(this.tableName.getTableName(), "gid", columnTypes.integer, false);
  public participantId:TableColumn = new TableColumn(this.tableName.getTableName(), "pid", columnTypes.integer, false);
  public compositeKey:TableColumn = new TableColumn(this.tableName.getTableName(), "compositeKey", columnTypes.varchar, false);
  public timestamp:TableColumn = new TableColumn(this.tableName.getTableName(), "ts", columnTypes.timestamp, false);
}
```

Cryptography

CipherChat is Made Secure through the implementation of the following cryptographic technologies:

- Hypertext Transfer Protocol Secure
- Elliptic Curve Diffie Hellman Key Exchange Protocol
- Elliptic Curve Digital Signature Algorithm
- Advanced Encryption Standard (256 bit)

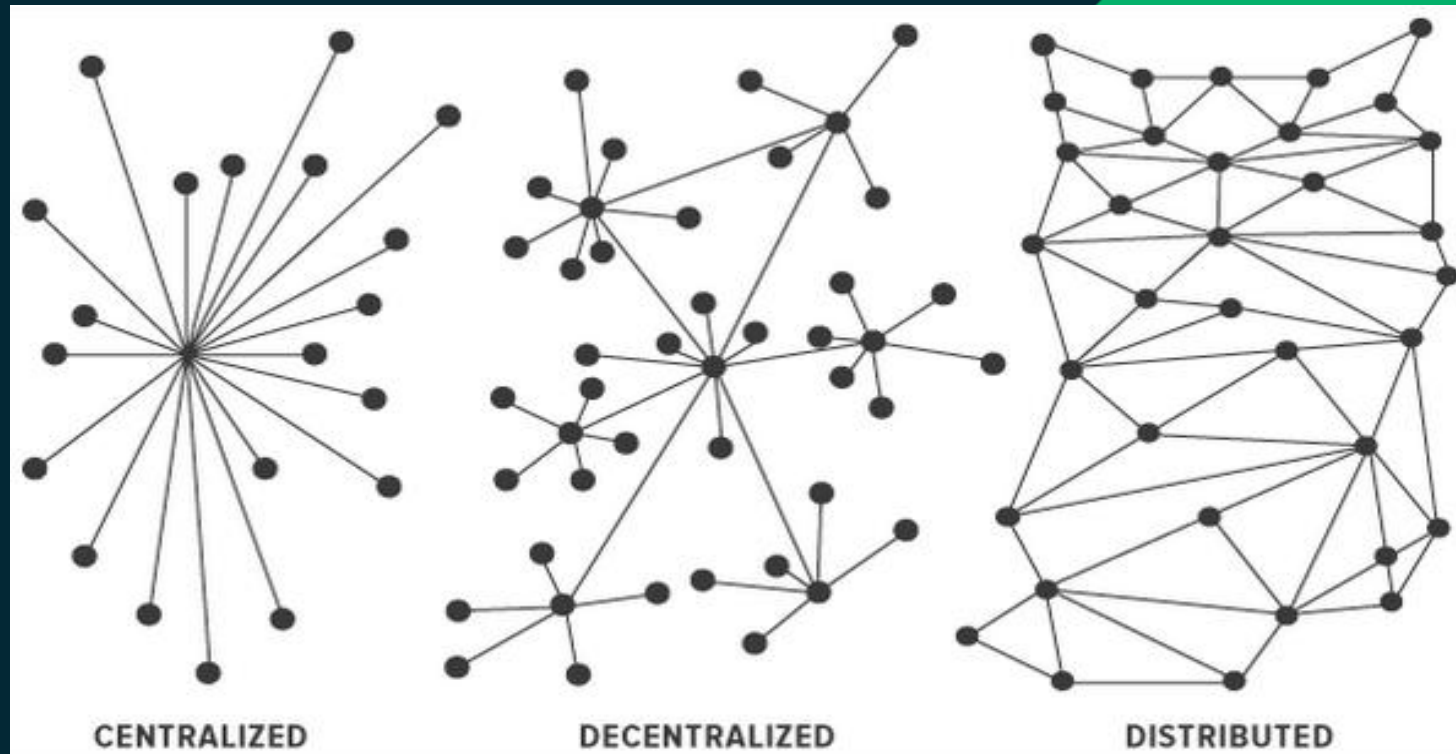
Chosen Elliptic Curve



SECP256k1

Used by government agencies and many cryptocurrencies due to special properties

Why Decentralization?



Decentralization creates a system which naturally more reliable and allows average individuals to benefit, rather than a single entity.

Elliptic Curve Diffie Hellman

Bob



Bob picks private key β

$$1 \leq \beta \leq n-1$$

Computes

$$B = \beta G \% p$$

Receives

$$A = (x_A, y_A)$$

Computes

$$P = \beta \alpha G \% p$$

Eve



$$y^2 = x^3 + ax + b$$

p

a

b

G

n

h

A

B

$$P = ?$$

Alice



Alice picks private key α

$$1 \leq \alpha \leq n-1$$

Computes

$$A = \alpha G \% p$$

Receives

$$B = (x_B, y_B)$$

Computes

$$P = \alpha \beta G \% p$$

Method

1. Multiple clients connect to a CipherChat server
2. Clients exchange public keys using ECDH
3. Messages are encrypted using the generated symmetric key and AES
4. Messages are signed using ECDSA Signing
5. The encrypted message is sent to the server
6. The server verifies the authenticity of the message using the signature (ECDSA Verification)
7. Server saves the authenticated message
8. The other peer(s) sends requests to the server and receive the latest messages
9. The received messages are decrypted using the symmetric key and AES

Diffie Hellman Vulnerability

Although Diffie Hellman can defend against passive attackers it is vulnerable to Man in the Middle Attack. In this case the server may function as two or more separate peers, decrypting and re-encrypting messages before sending them to the intended recipient.

Current Solution

1. Host and use your own server

This vulnerability will addressed later in this presentation

Load Balanced Servers

```
const handler = function(req, res){  
  req.pipe(request({ url: servers[currentServer] + req.url })).pipe(res);  
  currentServer = (currentServer + 1) % servers.length;  
};
```

```
otto@ottor-HP-Notebook:~/Apps/Mobile-Application-Tech-Year-3-Semester-2/SWEN3004/SERVER$ ./status.sh  
info:    Forever processes running  
data:    uid  command  script          forever pid    id logfile          uptime  
data:    [0] UGtU  ./node  loadBalancer.js 11285  11294  /home/otto/.forever/UGtU.log 0:0:0:14.618  
data:    [1] gGik  ./node  server.js       11312  11340  /home/otto/.forever/gGik.log 0:0:0:11.547  
data:    [2] cP-w  ./node  server.js       11351  11372  /home/otto/.forever/cP-w.log 0:0:0:10.868  
data:    [3] pSha  ./node  server.js       11383  11396  /home/otto/.forever/pSha.log 0:0:0:10.128
```

CipherChat Servers are designed to be scalable. Multiple instances can be ran simultaneously on separate threads with each instance handling tasks asynchronously, maximizing efficiency

Open Source

Food for Thought

1. Why should anyone trust applications if they cannot prove for themselves that it is secure?
2. Applications should be secure by design and not by policy!

CipherChat is Open Source. By design, CipherChat is trustless if a known server is used for each conversation (else MITM vulnerability) and will evolve to be trustless regardless of which server is used.

<https://github.com/CipherChat/CipherChat/>

An abstract geometric design on the right side of the slide. It features a large, bright green chevron pointing upwards and to the right. Overlapping this and extending further right are several dark teal shapes, including another chevron and rectangular blocks, creating a layered, architectural effect.

Live Demonstration

Future Improvements

1. Use of websockets instead of polling
2. Restrict Public Key Exchange to face-to-face interaction only
3. Create more incentives for person to host their own CipherChat servers
4. Notifications
5. Make app multilingual
6. Allow for the parsing of more complex data such as images and videos



Thank you for your kind attention!

Any Question?