CipherChat

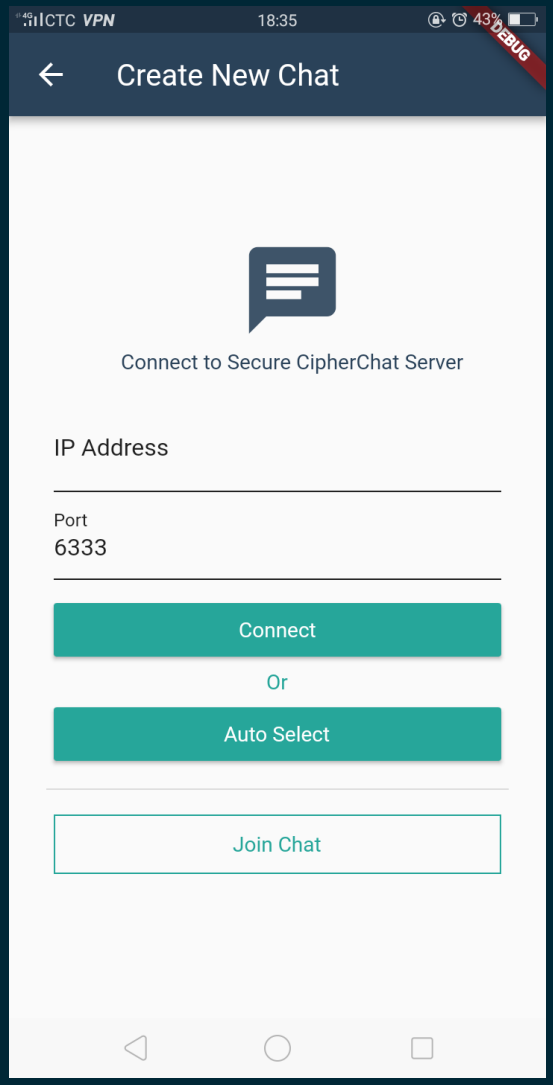
A Secure, Decentralized Chat Application

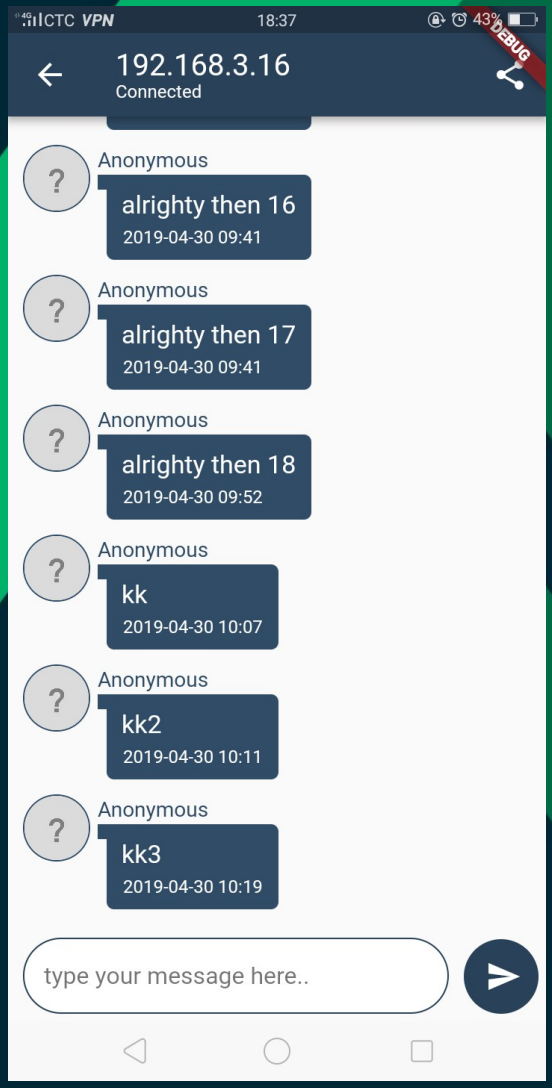
Group: K.O.D Inc
Members: Ottor Mills
Kenneth Anglin
Nicoy Smith
David Thomas

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)

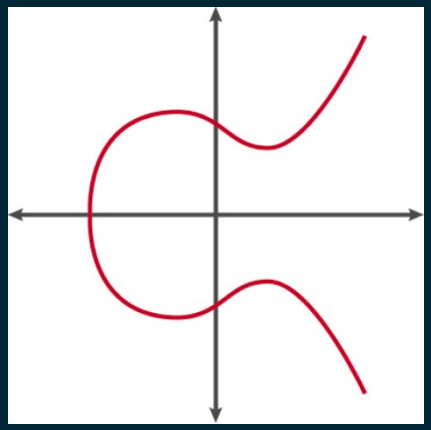
```
You, 2 days ago | 1 author (You)
class GroupsTable{
    public tableName:TableName = new TableName("groups");
    public groupId:TableColumn = new TableColumn(this.tableName.getTableName(), "gid", columnTypes.integer, true);
    public joinKey:TableColumn = new TableColumn(this.tableName.getTableName(), "joinKey", columnTypes.varchar, false);
    public timestamp:TableColumn = new TableColumn(this.tableName.getTableName(), "ts", columnTypes.timestamp, false);
}

You, 2 days 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, a few seconds ago | 1 author (You)
class MessagesTable{
    public tableName:TableName = new TableName("messages");
    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, 2 days 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)

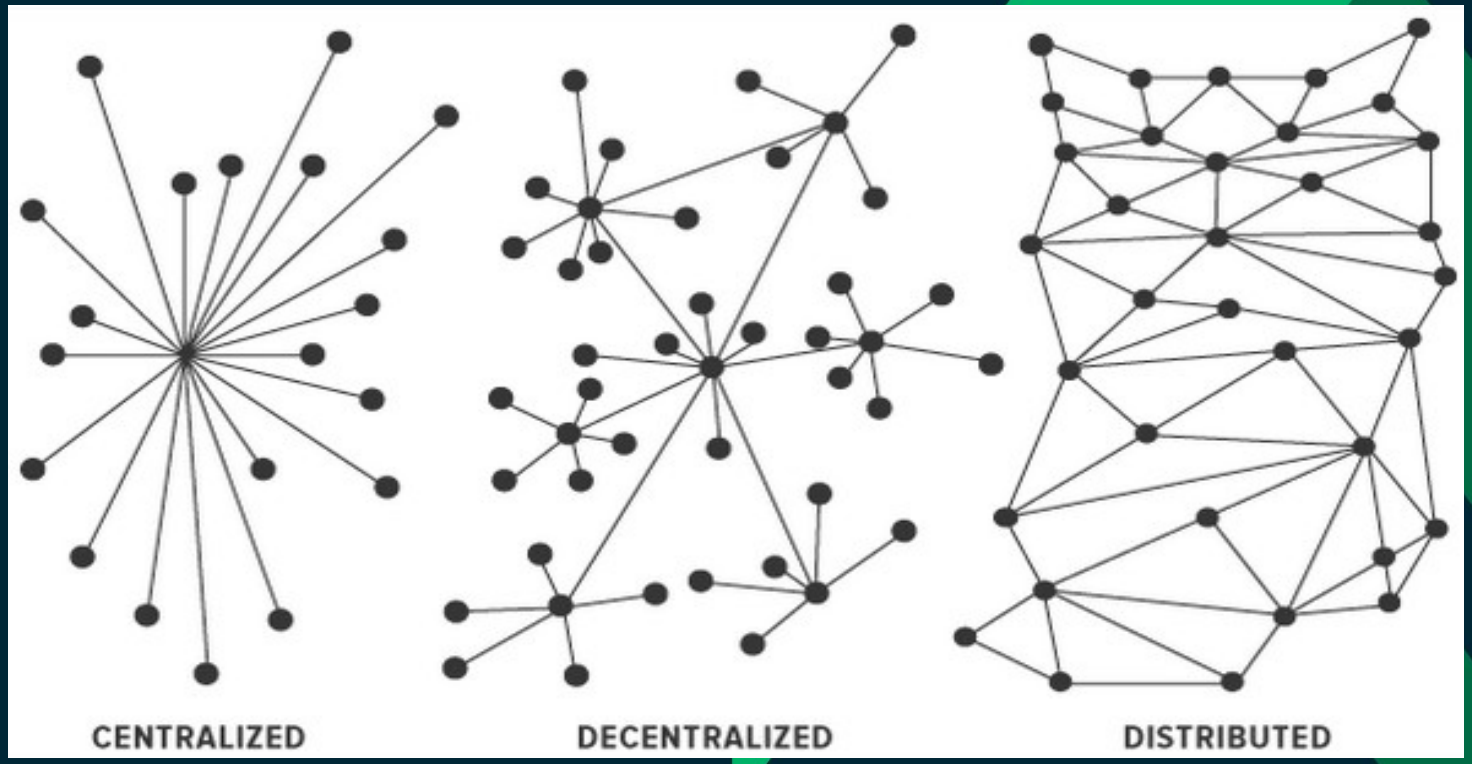
Chosen Elliptic Curve



SECP256k1

Used by government agencies and many cryptocurrencies due to special properties

Why Decentralization?



Decentralization creates a network which naturally more stable and accessible. Using digital signatures, the authenticity of messages can be verified

Elliptic Curve Diffie Hellman

Bob	Eve	Alice
		
Bob picks private key β	$y^2 = x^3 + ax + b$	Alice picks private key α
$1 \leq \beta \leq n-1$	p	$1 \leq \alpha \leq n-1$
Computes	a	Computes
$B = \beta G \%p$	b	$A = \alpha G \%p$
Receives	G	Receives
$A = (x_A, y_A)$	n	$B = (x_B, y_B)$
Computes	h	Computes
$P = \beta\alpha G \%p$	A	$P = \alpha\beta G \%p$
	B	
	$P = ?$	

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 Vulnurability

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

Stress Tester

```
const command = function(botID){
  return `
output`+botID.toString()+`=$(curl -d \
"username=bot`+botID.toString()+`&\
publicKey=872648736492384792387498&\
publicKey2=08023480324983243242380 9&\
passphrase=" -X POST https://127.0.0.1:6333/newgroup -k); printf "Output from \
bot`+botID.toString()+`: "$output`+botID.toString()+`"\n';
`;
}

const intensityLevel = 10;
const pulseIntensity = 2;
//Endpoint stress test
(async () => {
  for(let x = 0; x < intensityLevel; x++){
    setInterval(() => {
      let pulseArr = [];
      for(let y = 0; y < pulseIntensity; y++){
        pulseArr.push(new Promise(function(resolve, reject){
          require("child_process").exec(command(x), function (error, stdout, stderror) {
            if (error)
              console.log('[ERROR]: ' + error);
            else
              console.log(stderror+"\n"+stdout);
            resolve();
          });
        }));
      }
      Promise.all(pulseArr);
    }, 600);
  }
})();
```

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 dependable 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/

Similar Software

Extensible Messaging and Presence Protocol (XMPP)

- Uses XML instead of JSON
- Developed in 1999
- Used today (eg. Whatsapp)

Demonstration

```
otto@ottor-HP-Notebook:~/Apps/Mobile-Application-Tech-Year-3-Semester-2/SWEN3004/SERVER$ ./start.sh
Launch in debugging mode? [y/N]
```

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
7. Implement a broadcast method to transfer messages to other CipherChat nodes
8. Research into DDOS protection mechanisms
9. Migration from JavaScript to TypeScript

Thank you for your kind attention!

Any Question?