

Crypto Chat: An Application Encrypted with AES-CTR

<https://www.cryptoproject.site/>

Nowsha Islam, Harish Saravanakumar, Josh Yam

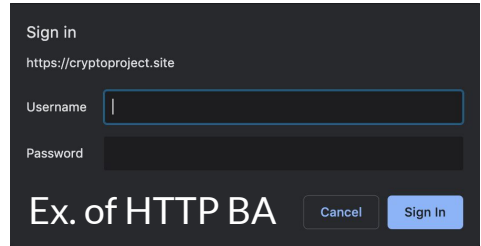
Contents

- Application Technology Overview
- Encryption Technology Overview
- Implementation
- Security
- Demonstration
- Other Considerations

Application Technology Overview

- We wanted users to be able to communicate in a classic global chat room
1. **Flask**
 - Python microframework (framework with few dependencies to external libraries)
 - Back-end application software for designing web applications
 2. **Pusher**
 - Hosted API that provides real-time bidirectional functionality using sockets
 - Hosts connection between client and server, allowing users to transmit chat messages
 3. **Database/SQLAlchemy**
 - Library allowing Python programs and databases to communicate
 - Used as an Object Relational Mapper (ORM), translating:
 - Python classes to database tables
 - Function calls to SQL statements

Encryption Technology Overview



Sign in
https://cryptoproject.site

Username

Password

Ex. of HTTP BA

1. HTTP Basic Access Authentication (BA)

- Method for a user to provide a username and password when making a request to the server

2. DNSSEC

- Traditionally, the Domain name system (DNS) uses unencrypted data for its records
- DNSSEC signs all the data sent on DNS records so DNS resolvers can verify their authenticity, ensuring users connect to the DNS records that belong to the real domain name (and not imposters)
- Utilizes public key infrastructure using a public and private key
 - DNS records are signed with the private key

3. HTTPS

- Secure extension of HTTP, encrypted using TLS (Transport Layer Security), also secures HTTP BA
- Provides authentication to the accessed website
- Guarantees privacy and integrity of exchanged data while in transit

4. AES-CBC

- Advanced form of block cipher encryption
- Each ciphertext block is dependent on all other blocks processed up to that point (drawback: sequential)
- IND-CPA secure

Security of Technologies

1. HTTP Basic Access Authentication (BA)

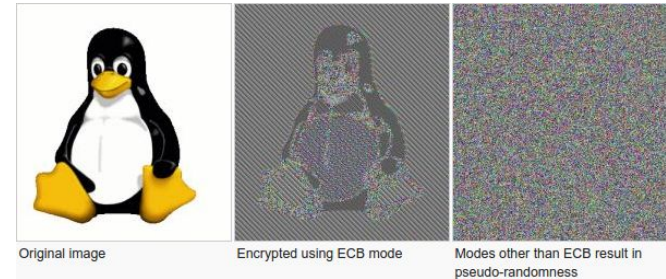
- Simplest technique for enforcing control to resources
 - Standard fields in the HTTP header; no handshakes required
- Credentials are sent over with no encryption (encoded in base64)
- No confidentiality
- HTTP itself does not provide a method for a server to instruct the client to log out the user (as a result, username and password will be cached)

2. HTTPS

- Protects against man-in-the-middle attacks because it makes it difficult for an attacker to obtain a valid certificate for a domain that is not controlled by the attacker
- This, in turn with the bidirectional encryption (as discussed before), protects against **eavesdropping** and **tampering** of the communication
- Has internal protection against replay attacks

3. AES-CBC

- Advanced form of block cipher encryption where each block is dependent on all other blocks processed up to that point
- IV-based encryption scheme, where the mode is, assuming a random IV, as secure as a probabilistic encryption scheme



Implementation of AES-CBC

- The server creates a (random) symmetric key and sends it to the client after it passes HTTP BA authentication via tls with https (asymmetric encryption)
- All further messages are encrypted & decrypted using this key
- If a user leaves the group, then a new random symmetric key is generated for every user on the site

More Specifically:

1. After logging in (passing the HTTP BA), the client receives a key from the server
2. Client converts input into a byte array, then encrypts it using AES. Sends data via \$.post method
3. Server receives the encrypted byte array via a flask HTTP POST request, and reads the form data
4. Username and encrypted message pairs gets logged onto the mysql database
5. Server uses the **pusher** api to send the encrypted array to other clients
6. Client decrypts message and converts it back to a string
7. Finally adds the code to the template

```
var key = {{arraykey}};  
window.alert('This is your key: ' + key);
```

```
$('#chat_btn').on('click', function() {  
  let plain_message = $('#chat_text').val();  
  let marray = string2Bin(plain_message);  
  let message = AES_Encrypt(marray, key);  
  console.log(message);  
  
  $.post('/message', { 'username' : username, 'message[]' : message }, function() {  
    $('#chat_text').val('');  
  });  
});
```

```
new_message=Message(username=username, message=str(message))  
db.session.add(new_message)  
db.session.commit()
```

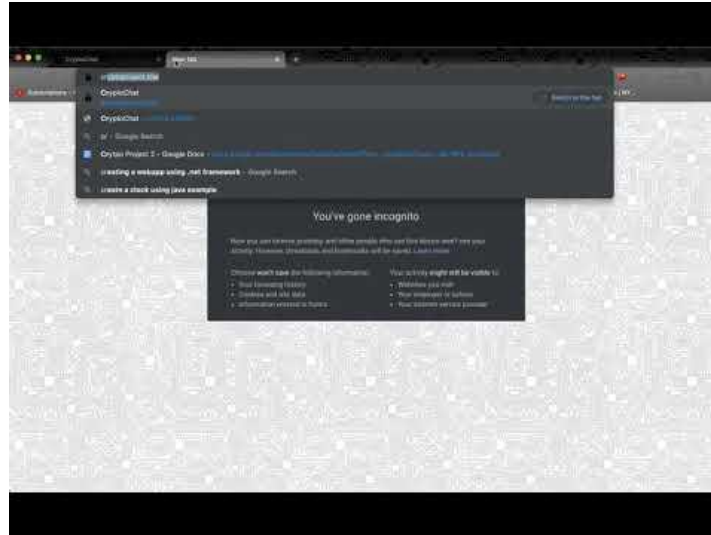
```
@app.route('/message', methods=['POST'])  
def message():  
    try:  
        username = request.form.get('username')  
        message = request.form.getlist('message[]')
```

```
pusher_client.trigger('chat-channel', 'new-message', {'username': username, 'message': message})
```

```
let message_template = `  
    <article class="media">  
        <div class="media-content">  
            <div class="content">  
                <p>  
                    <strong>${name}</strong>  
                    <br> ${message}  
                </p>  
            </div>  
        </div>  
    </article>`;   
$('#content').append(message_template);
```

```
channel.bind('new-message', function(data) {  
    let name = data.username;  
    //let message = data.message;  
    let en_message = data.message;  
    let bmessage = AES_Decrypt(en_message, key);  
    let message = bin2String(bmessage);
```

Demonstration



https://www.youtube.com/watch?v=VZ83Mqc_7CA&feature=youtu.be