California State University Long Beach
-
SecChat.me Messenger
-
CECS 478 Introduction to Computer Security

David Krahe

David Schemm

TwoDavidsOneComputer

CECS 478

Table of Contents

DEMO

Design Phase

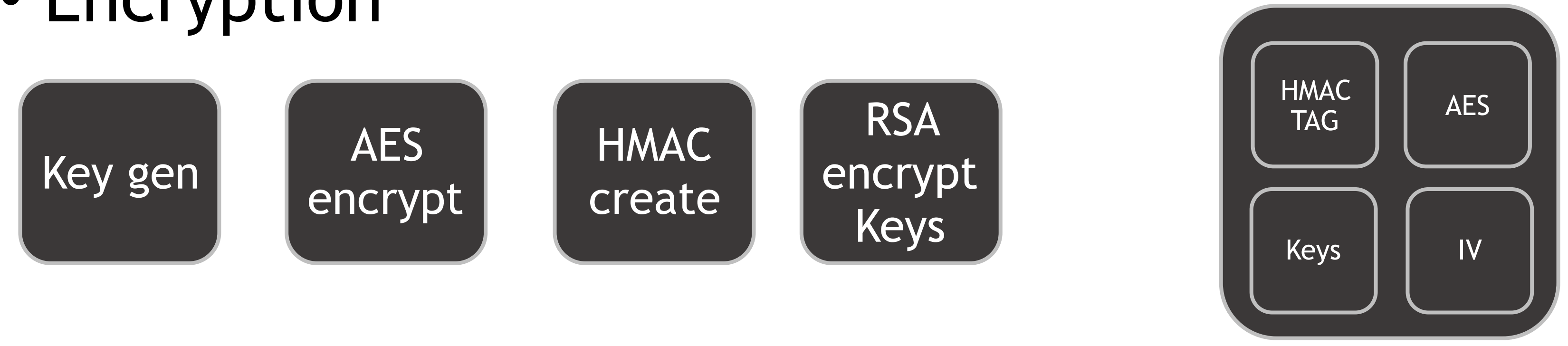
- Identify Assets, Stakeholders, Adversaries
- Define Attack surfaces
- Research frameworks and libraries
- Prototype solution
- Analyze based on design requirements

Implementation
Client side – Encapsulation/Decapsulation

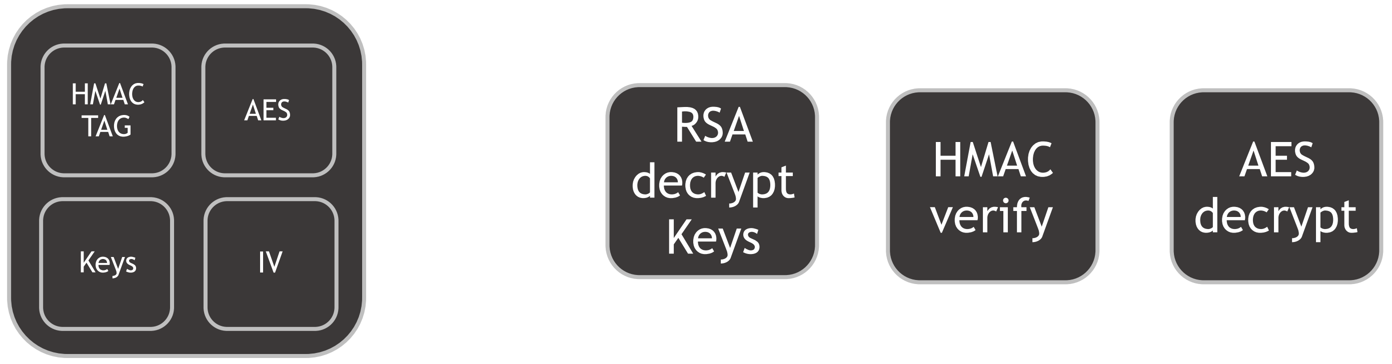
- Key/IV generation -> RSA (OAEP only!!)
  - Entropy pool
  - 2 Keys - 256 bit
- AES Encryption/Decryption
  - PKCS#7 padding
- HMAC
  - Encrypt then HMAC
  - SHA 256
- Javascript library node-forge

Implementation
Client side – Encapsulation/Decapsulation

- Encryption

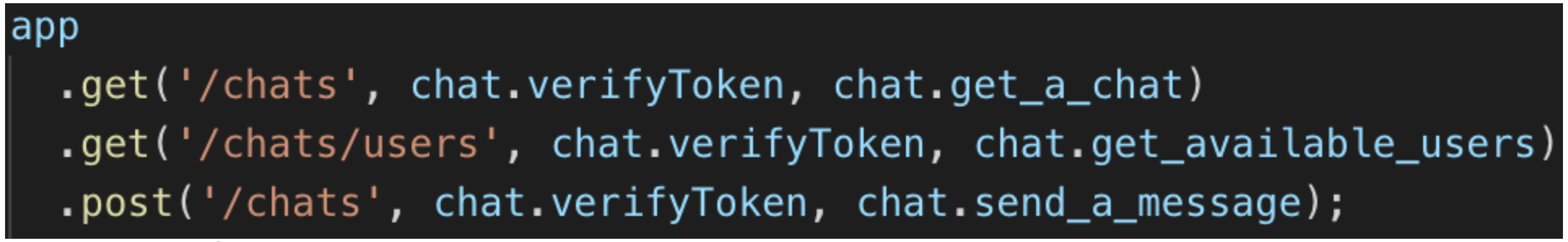


- Decryption



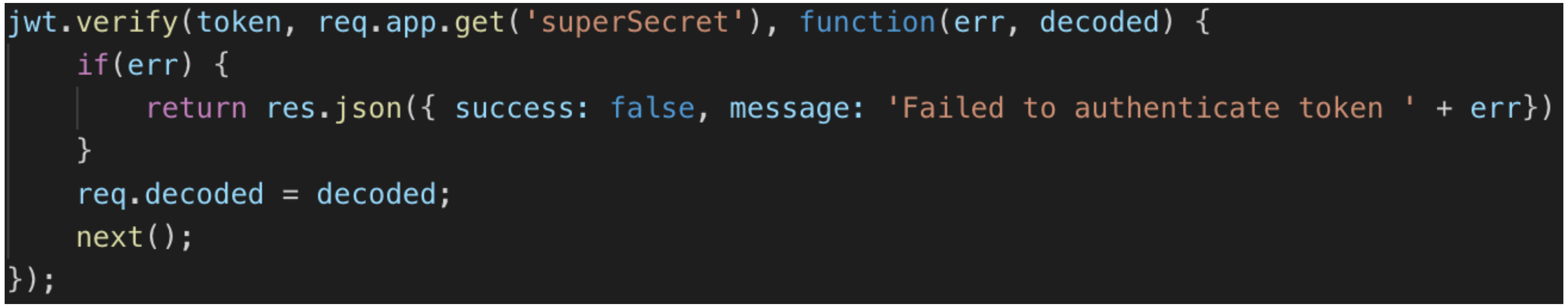
Implementation
Server side – Get/Post/JWT

- JWT
  - Chat routes



```
app
  .get('/chats', chat.verifyToken, chat.get_a_chat)
  .get('/chats/users', chat.verifyToken, chat.get_available_users)
  .post('/chats', chat.verifyToken, chat.send_a_message);
```

  - Token verification



```
jwt.verify(token, req.app.get('superSecret'), function(err, decoded) {
    if(err) {
        return res.json({ success: false, message: 'Failed to authenticate token ' + err})
    }
    req.decoded = decoded;
    next();
});
```

Problems

- Nginx Reverse proxy not forwarding to node service
- http open and accessible
    - ➡ http port 80 blocked for AWS
- database model too simplistic
- performance
    - ➡ Redux state management

Conclusion: Think First

- Data model for chats and user
  - User/chat model too simplistic
  - Define database requirements/interfaces in the beginning!

➡ Clear Design before starting to program!

Thank you for your attention

-The End-