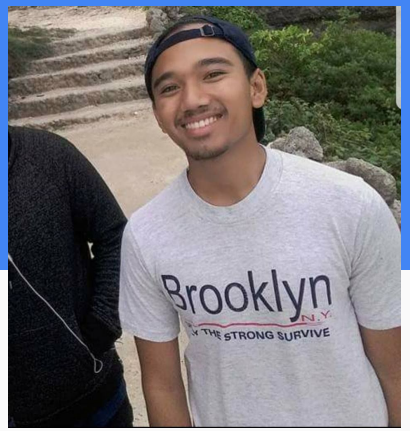
DuoDolo

Security is not an afterthought when it comes to design

The team

Trained by Prof. Aliasgari



Ryan “Ry” Valdriz

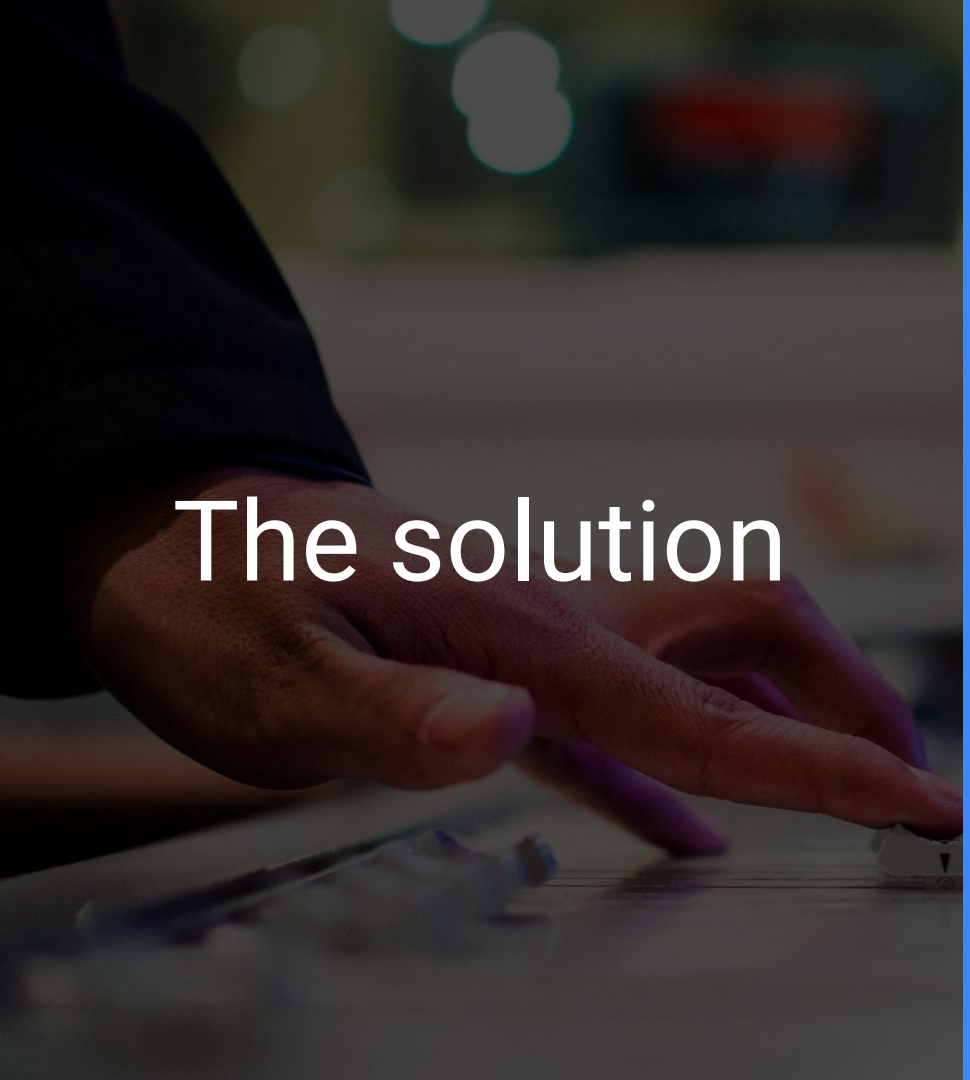
Robert “The Robbit” Sanderson

Goal:
To create an end-to-end encrypted chat

The problem

Messaging Apps with no end-to-end encryption

- Confidentiality is at risk
  - Plaintext
  - Server can be an adversary as well
- Fragile integrity of the conversation
  - No message authentication

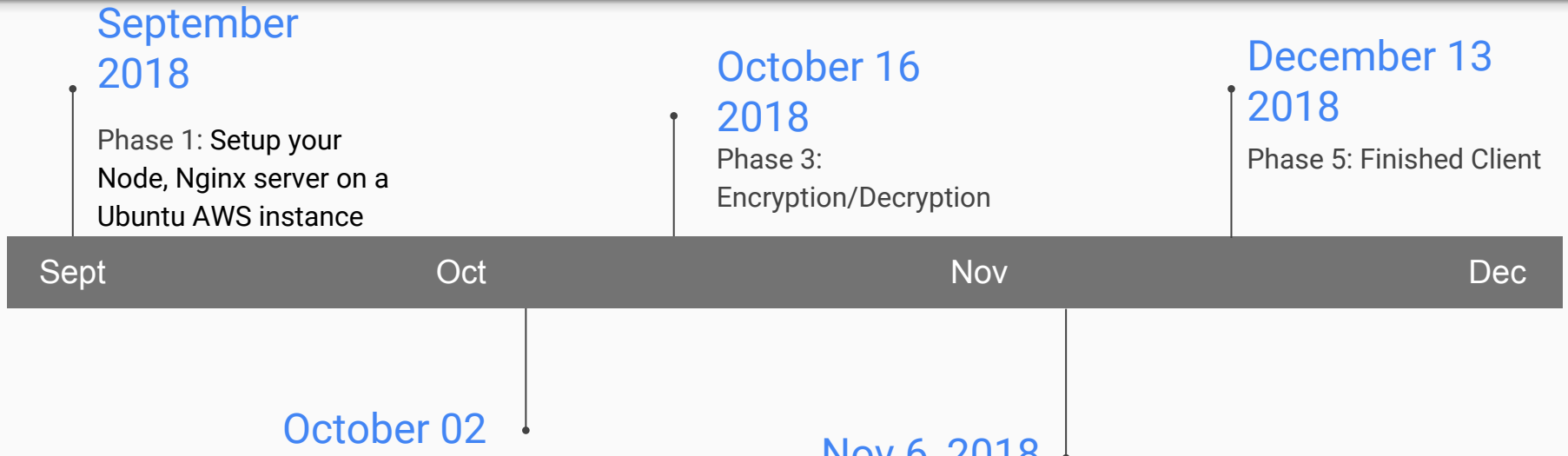


The solution

End-To-End encryption

- Preserve Confidentiality
- Only the people communicating can read the messages
- Prevents eavesdroppers from reading the messages without the keys
  - ISP
  - TELECOM PROVIDERS
  - Hackers(depends)

Milestones

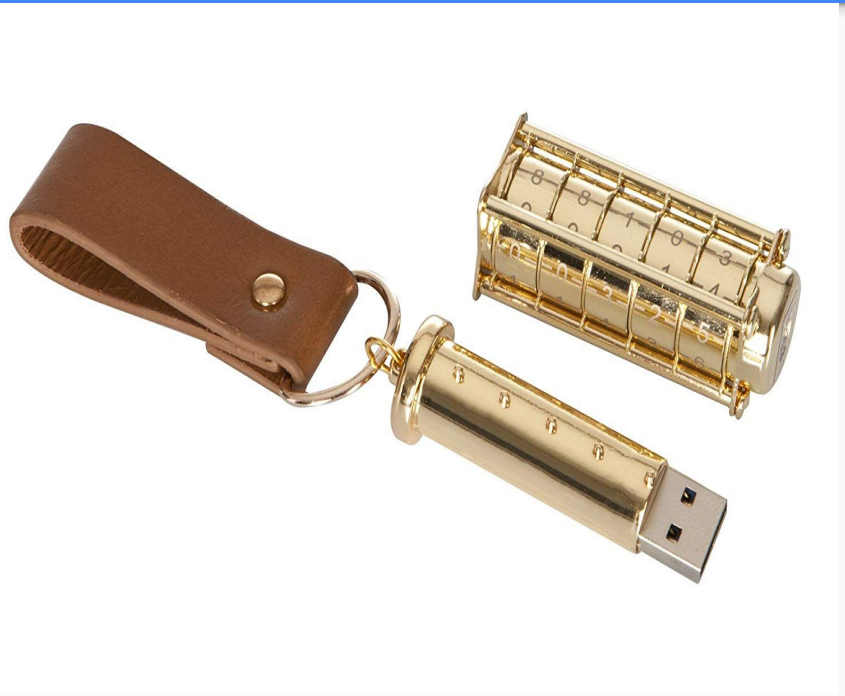

September 2018
Phase 1: Setup your Node, Nginx server on a Ubuntu AWS instance
October 16 2018
Phase 3: Encryption/Decryption
December 13 2018
Phase 5: Finished Client
Sept
Oct
Nov
Dec
October 02 2018
Phase 2: A simple HTTPS Server
Nov 6, 2018
Phase 4: JWT and RESTful Server

Attack Surfaces

- Man in the Middle
  - Encrypted messages from hitting send to hitting receive
- Compromised Database
  - Passwords are hashed and stored in the database
- Server DDoS
  - No access for anyone

Key Exchange

- One way we are exchanging keys between clients
- USB stick with a mechanical combination lock
  - Public key would be contained on the stick
  - Special Edition Cryptex USB Flash Drive 64 GB, USB 3.0, Gold Edition
    - $69.99 Prime Free one-day
- Less secure alternative
  - Key server
    - https://keyserver.pgp.com/vkd/GetWelcomeScreen.event
    - Possible loss of authenticity



How it works(sender)

Step 1

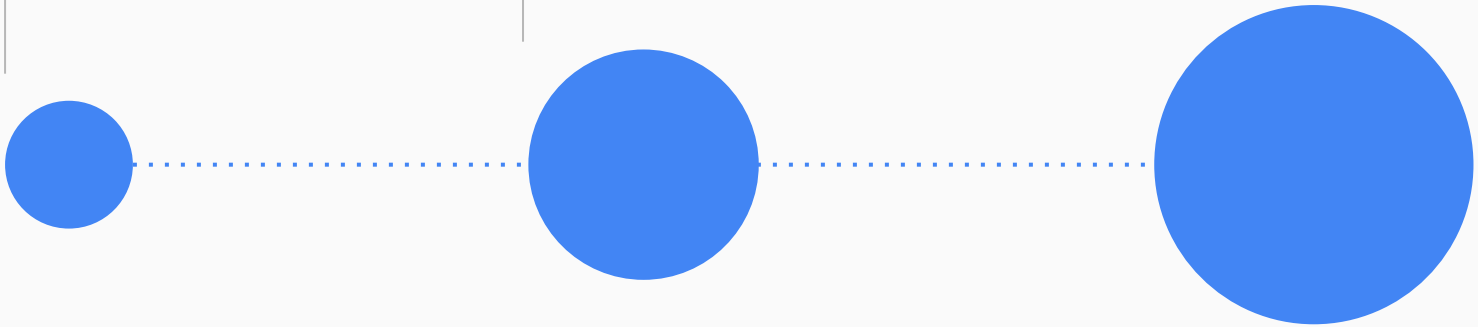
- User logs in
- Zero-Proof authentication
- User gains jwt

Step 2

- User inputs message
- Message is encrypted with receiver's public key
- Encrypted message is hashed

Step 3

- Message is sent to server
- Server stores the message in our mongodb atlas database



How it works(receiver)

Step 1

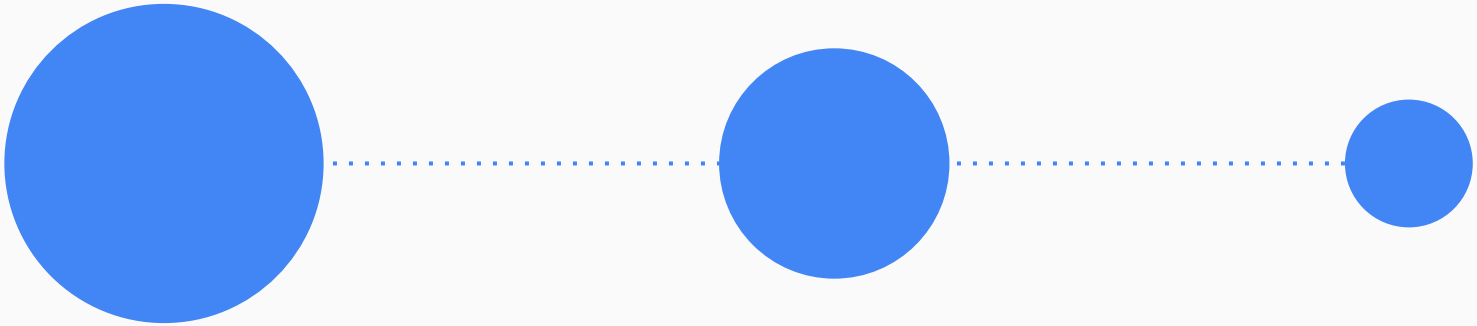
- Request to receive messages sent to from the server/database
- Get message

Step 2

- Check integrity of the message

Step 3

- Decrypt message with the users private key
- Ponder on how to reply




The Technology:

- AWS
  - Server
- Node.js
  - Server code
- Python
  - Client Application
- SSL
- Encryption
  - AES
  - RSA
- MongoDB
- HMAC

Shortcomings

References

- https://medium.freecodecamp.org/securing-node-js-restful-apis-with-json-web-tokens-9f811a92bb52
- https://hackernoon.com/restful-api-design-with-node-js-26ccf66eab09
- https://www.pythonforbeginners.com/requests/using-requests-in-python
- https://pycryptodome.readthedocs.io/en/latest/index.html
- https://www.zeolearn.com/magazine/designing-a-rest-api-with-nodejs-and-mongodb-atlas

DEMONSTRATION