Client Sign-Up Request

```
POST /api/signup
Host:
www.https.example.com
content-type:
application-json
******—body—-*****
{"username":"billy",
"password":"myeasypas
sword"}
```

plain-text username:'billy'
and password:'myeasypassword'

https connection (encrypted)

#1 input password:
'myeasypassword'

#2 Output: hashed password: :
'dh9elsj9dj49djla65sde'

bcrypt.
hash()

#3 Create user with hashed
password stored in
'password'
property of user

one way hash function

two way symmetric encrypt/decrypt

crypto.random
Bytes()

user{**username**: 'billy',
password:
'dh9elsj9dj49djla65sde',

findHash:'sjdh83ldjd937
dodw7yd63uehd6t3ih'
}

#4 generate random string
with no external input from user
and store it in the user's findHash property

#6 use jwt .sign() to encrypt
user's findHash using symmetrical
encryption algorithm

#5 findHash is a randomly generated
string using node.crypto.randomBytes()
and stored in plaintext in user's
'findHash' property in db

JSON Web Token((jwt)

server cypher
key
(APP_SECRET)

#7 token is the encrypted
findHash value

user's findHash value is
a **unique** random string
generated by server

unique user Token

#8 token passed to client

Client

unique user Token

#9 client app request
to server with token

JSON Web Token(jwt)

server cypher
key
(APP_SECRET)

decrypted token

#10a use jwt .verify() to decrypt
users token

Compare
#10c jwt.verify()

user{**username**: 'billy',
password:
'dh9elsj9dj49djla65sde',

findHash:'sjdh83ldjd937
dodw7yd63uehd6t3ih'
}

#10b get the user's findHash from the db and
compare it to decrypted user's token. If the
two strings === the user is authenticated