Protecting Your Users' Privacy

@_aijaz_

Protecting Your Users' Privacy

Password Data on Remote Servers

Two Simple Things

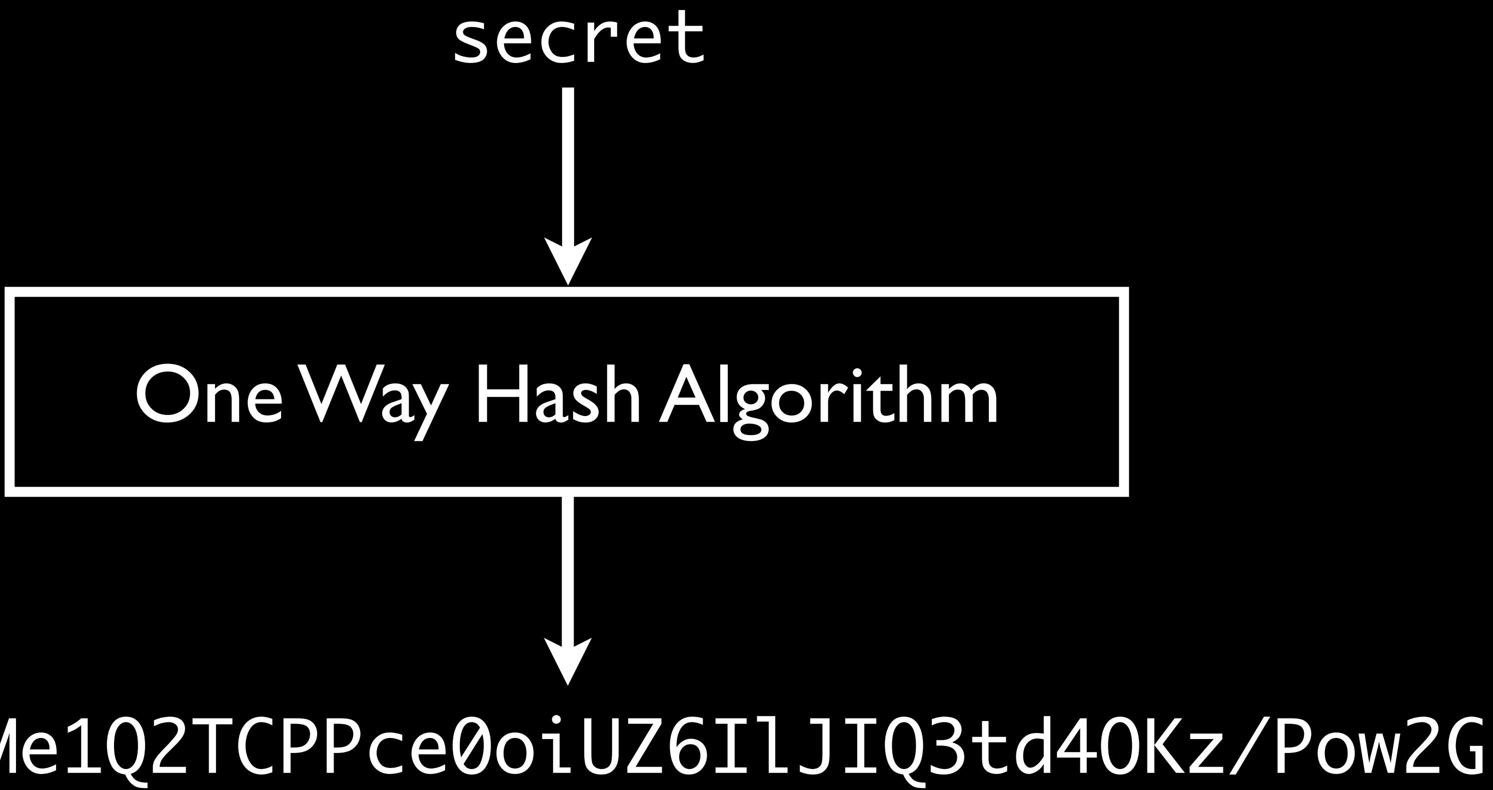
@_aijaz_

[redacted]

Survive

[redacted]

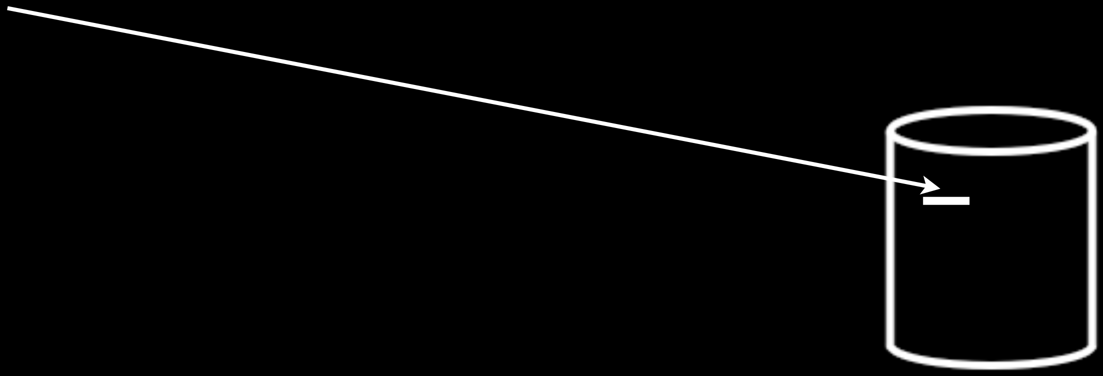
```
SELECT * FROM USERTABLE
WHERE login = ‘admin’
AND password = ‘secret’
```



```
secret
```

```
$2a$07$Me1Q2TCPPce0oiUZ6IlJIQ3td4OKz/Pow2G
```

```
$2a$07$Me1Q2TC6IlJIQ3td4OKz/Pow2G
```



```
SELECT * FROM USERTABLE
WHERE login = ‘admin’
AND password=crypt(‘secret’, password)
```

Modern databases support many kinds of hashes.

Salts

```
Login:  aijaz@example.com
Password: ‘hello’
Salt: 23ddkc
Hash: $2a$05$23ddkc20ker0998Q...

Login:  dave@example.com
Password: ‘hello’
Salt: Zff7dk
Hash: $2a$05$Zff7dkf93kNSqnzC...
```

Use Random Salts

```
aijaz@example.com | ab287efee2876aa...
dave@example.com  | ab287efee2876aa...
...
alice@example.com | ab287efee2876aa...
```

```
Target hash: $2a$05$23ddkc20ker09

'open!'    -> $2a$05$dhwefu23823ld  NO MATCH
'sesame!' -> $2a$05$39dk2sdkfu3el  NO MATCH
...
'friend' -> $2a$05$23ddkc20ker09  ENTER!
```

Guessing Passwords

Your System can throttle, lock out

Password files can be analyzed offline

1- Use bcrypt

```
$2a$05$d098b0fc4aZKYOHnlPZff7NSqnzCrPwn0yqnyi
```

2^5 iterations

```
$2a$10$8ib9C2Zk8dDAPE8Kim0ogE9su11SDyHGuv4YUC
```

2^{10} iterations - much more secure, much slower

You get to choose how many iterations you want.
The hash includes a random salt.

Increasing Iterations

```
$2a$05$d098b0fc4aZKYOHnlPZff7NSqnzCrPwn0yqnyi

password = cgi->fields->password
if (password is valid) {
  numIter = substr(hash, 4, 2);
  numIter++;
  newhash = bcrypt(password, numIter);
  replace hash with newhash in database
}

$2a$10$8ib9C2Zk8dDAPE8Kim0ogE9su11SDyHGuv4YUC
```

Secure Hashes are Slow

Algorithm	***Guessing Speed***
Unsalted SHA-1	15.5 Billion guesses/second
SHA512	11,000 guesses/second
bcrypt(10)	11 guesses/second
bcrypt(16)	**5.5 seconds/guess**
bcrypt(20)	87.5 seconds/guess

Reusing a Password

Facebook	User: bob@example.com Password: 'bob'
Twitter	User: bob@example.com Password: 'bob'
MegaBank	User: bob@example.com Password: 'bob'
Your Site	User: bob@example.com Password: 'bob'

Your Password File

```
alice@foo.com:$2a$16$873AB23783...
bob@example.com:$2a$16$23d98Q7K129S...
cathy@bar.com:$2a$16$12AB43BBCE...
```

A good guess for bob's password: 'bob'
Total time to crack: 5.5 seconds.

2 - Hash the email

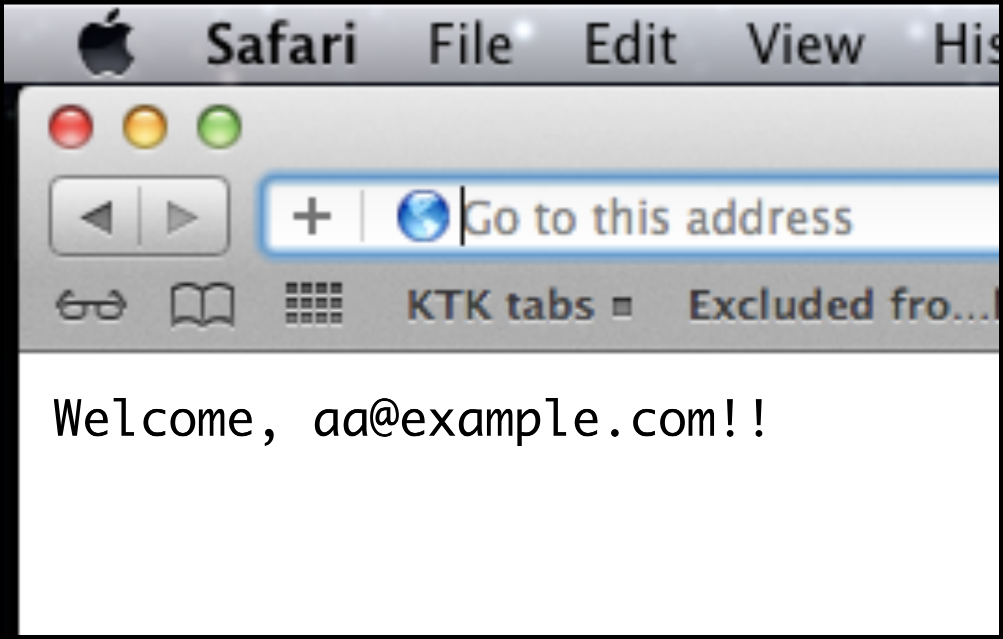
User: bob@example.com
Password: ‘bob’

In the database:
User: $2a$05$d098b0fc....
Password: $2a$05$23d98Q...

Without a user name, a password is worthless

Using a Hashed Email

- ✦ Normalize the email
  - ✦ remove leading and trailing spaces
  - ✦ convert to lower case
- ✦Guaranteed to get same hash every time
- ✦ Hash the normalized email
- ✦ Use that hash when adding a user or querying on login.



```
Welcome, aa@example.com!!
```

Store it in the session

Now, only current sessions are at risk

Delete when the session expires

Encrypt it for additional safety

Uploading The Address Book

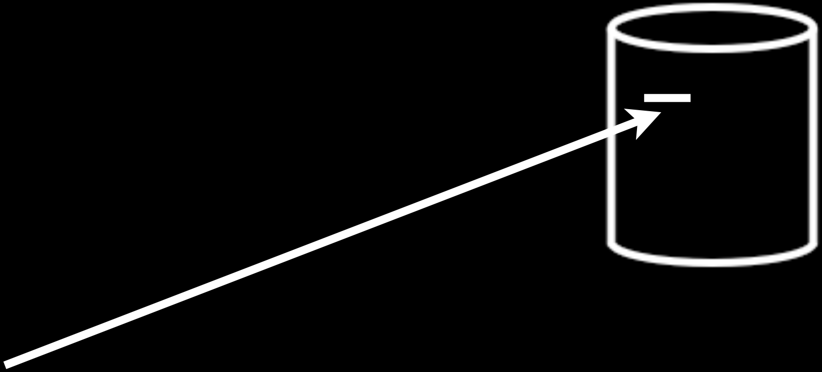
Never upload email addresses

Upload their hashes instead

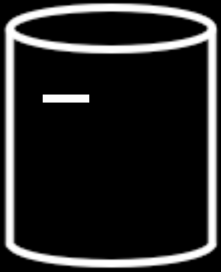
Only upload what you need

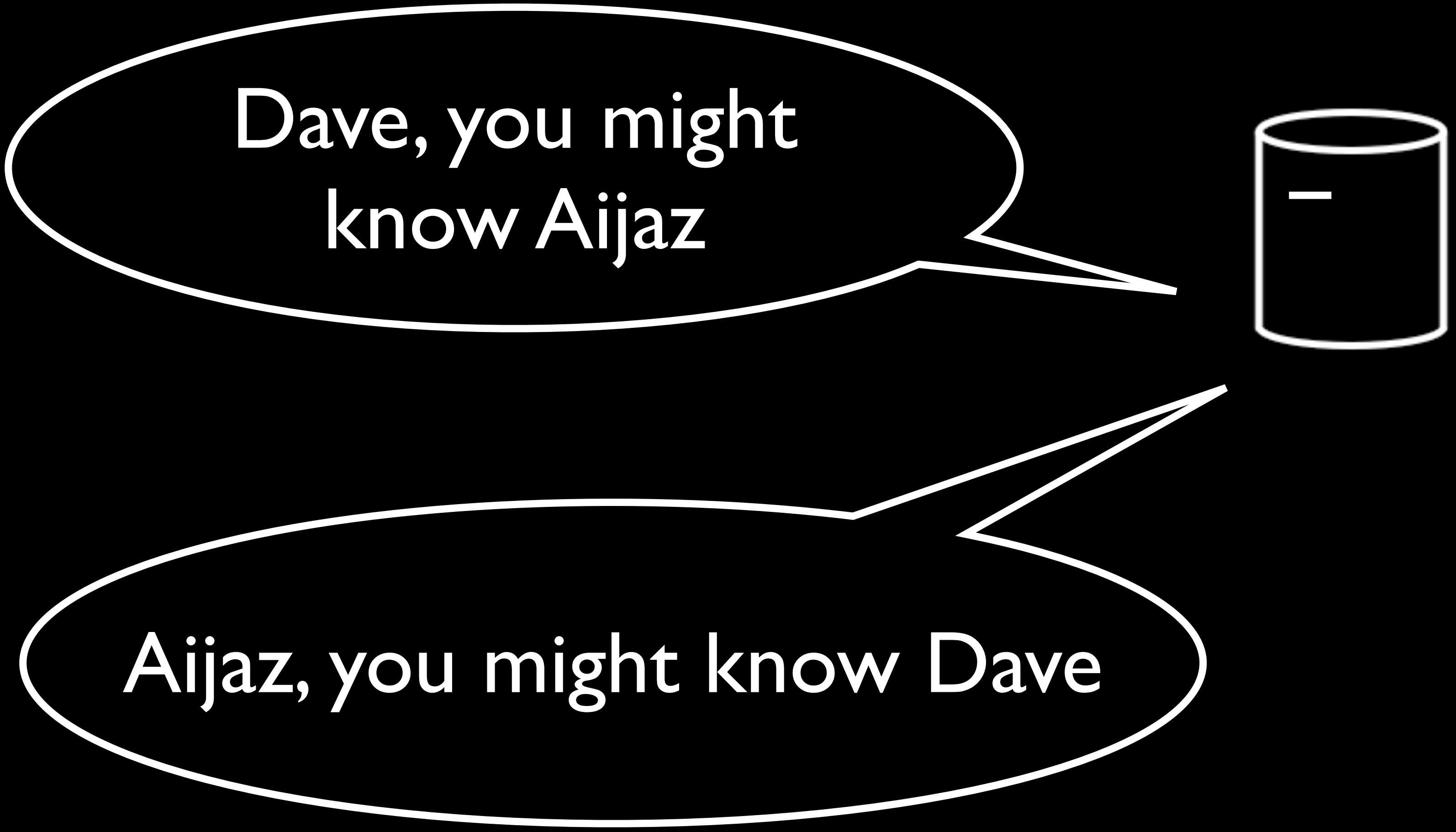
```
My Address Book:

Dave: dave@example.com
```

```
Aijaz’s Address Book:

Dave: $2a$05$qZHnl....
```



```
Later, Dave registers as dave@example.com

Insert user with email: $2a$05$qZHnl....
SELECT * FROM FRIENDS
WHERE EMAIL=’ $2a$05$qZHnl....’
```





You really didn’t need to upload the email

What to Remember

Hash passwords with bcrypt

Hash identifying data, too

Don’t store anything you don’t need

http://TheJoyOfHack.com/

@_aijaz_

Thank you!