Hashing Passwords

a.k.a. not being known as
“that company that leaked everyone’s passwords”

Storing passwords incorrectly is BAD BAD BAD!

The most common mistake I see with auth systems is storing a password improperly.

If you can decrypt the password, you are doing it wrong!

You are not hashing. You are encrypting.

These two are VERY different.

Your app should NEVER be able to recreate a password

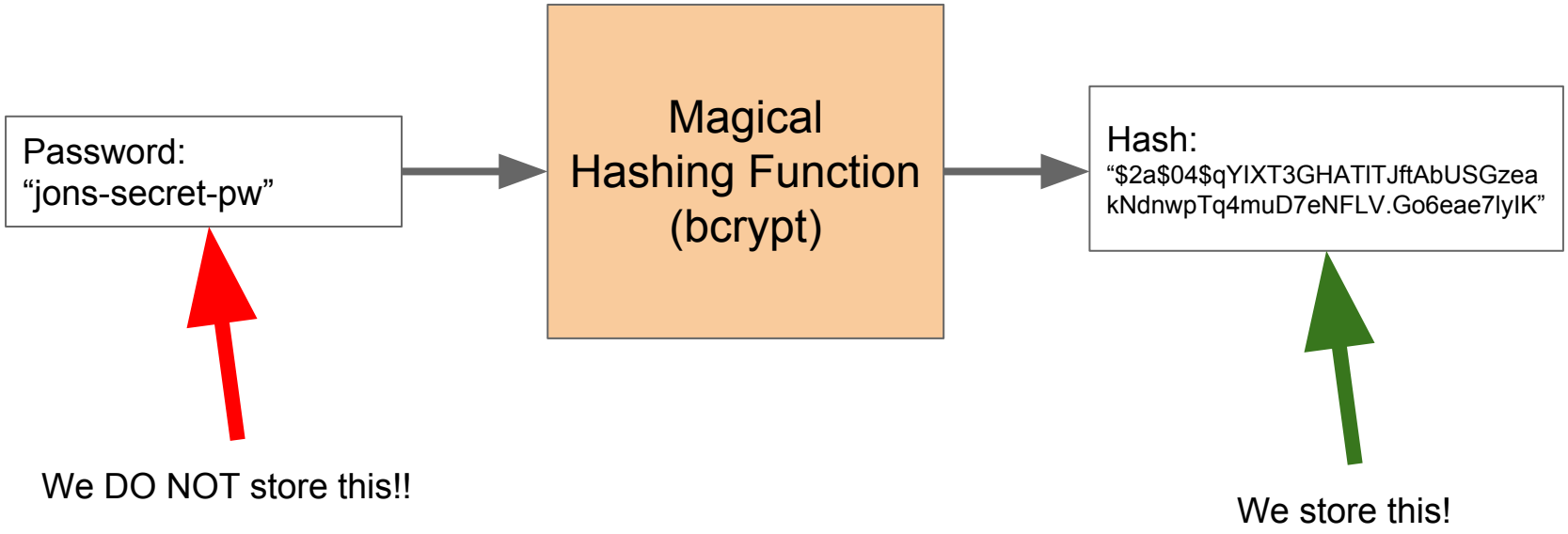
If you can recreate a password from data you store, an attacker who compromises your system could as well.

It simply isn’t secure, so DO NOT DO IT!

So how do we verify a user’s password when they log in?

Hashing functions baby!

Instead of storing a password, we store a hash.



Hashing functions baby!

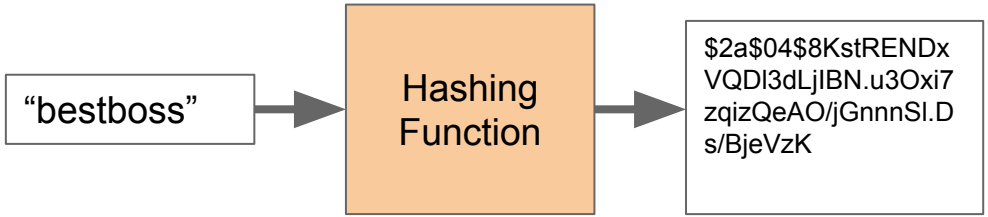
As a result, our database looks something like this:

ID	Email	PasswordHash
1231	jon@calhoun.io	$2a$04$qYIXT3GHATlTJftAbUSGzeakNdnwpTq4muD7eNFLV.Go6eae7lyIK
1232	michael@dunder.com	$2a$04$8KstRENDxVQDl3dLjIBN.u3Oxi7zqizQeAO/jGnnnSl.Ds/BjeVzK

Hashing functions baby!

Now when Michael logs in, he gives us a password like “bestboss”

We then hash that password:

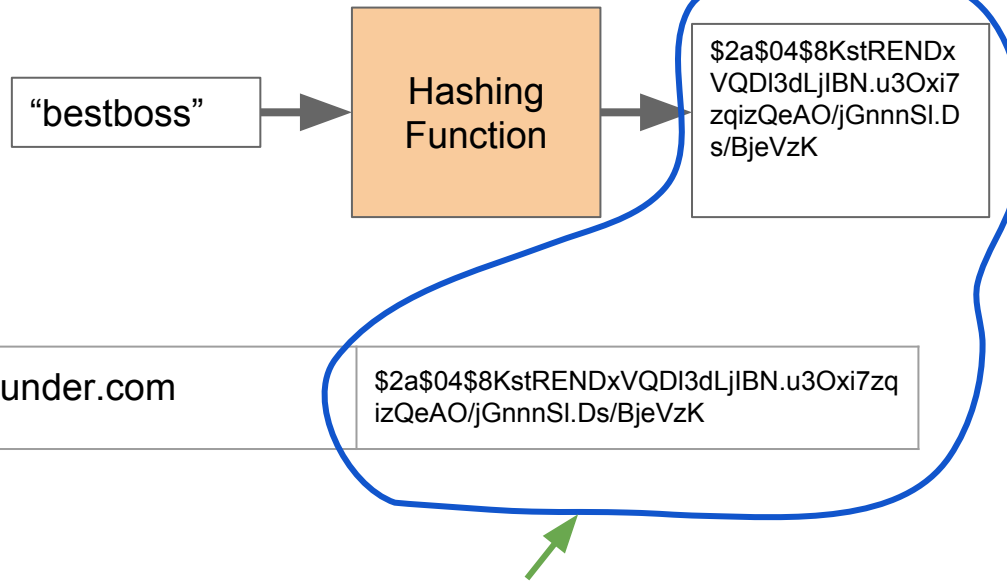


Then we look up michael in our database:

1232	michael@dunder.com	$2a$04$8KstRENDxVQDl3dLjIBN.u3Oxi7zqizQeAO/jGnnnSl.Ds/BjeVzK

Hashing functions baby!

Then we compare those two hashes to see if they match.



We store hashes, not passwords

Instead of storing a password, we store a hash.

We can compare hashes to see if a user is logging in with the correct password.

We don’t store raw passwords! Success!

But can’t someone just reverse those hashes?

Hashes cannot be reversed

Imagine you gave me your name, and I returned you a color.

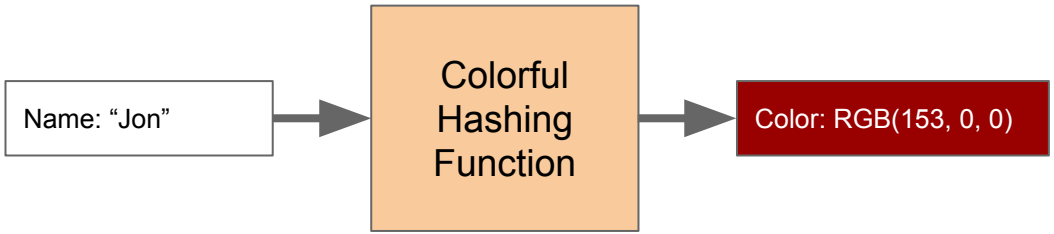
Hashes cannot be reversed

The color you get back for each name is always the same.

Each name will get a different color (sorta).

But the most important part - **there is no way to go from a color to a name.**

We cannot say “What name has the color `RGB(153, 0, 0)`?”



Why aren't hashing functions reversible?

Imagine our function worked by giving each letter an RGB value and then adding those together. (results % 256)

“J” = rgb(50, 0, 0)
“o” = rgb(3, 0, 0)
“n” = rgb(100, 0, 0)

So our result is rgb(50 + 3 + 100, 0, 0)

Why aren't hashing functions reversible?

But we could have just as easily created the same color with the string “JJoJ” or “oJJJ” or even “noJ”

“J” = rgb(50, 0, 0)
“o” = rgb(3, 0, 0)
“n” = rgb(100, 0, 0)

JJoJ = rgb(50 + 50 + 3 + 50, 0, 0) = rgb(153, 0, 0)
oJJJ = rgb(3 + 50 + 50 + 50, 0, 0) = rgb(153, 0, 0)
noJ = rgb(100 + 3 + 50, 0, 0)   = rgb(153, 0, 0)

In practice this is much more complex

Our color hashing function was a very simple example

Password hashing is like this, but way more complicated

Eg - Password hashing functions are much harder to find multiple keys that hash to the same value

You can generally assume that two passwords won’t hash to the same hash value with a password hashing function

Encryption IS NOT hashing!!

At some point you may learn about encryption functions like AES

You may think they would be a good fit for passwords. YOU ARE WRONG!!!!

Encryption is reversible via decryption, so it isn’t suitable for raw passwords.

Now what MIGHT help (eg Dropbox does this) is to encrypt password hashes before storing them.

Up Next...

Coding time!

We will update the User type to add the Password and PasswordHash fields

We will then update our UserService to hash a password before creating a user