Intro

When i go to a website I need to sign in and the most popular way is to use a password.

This sucks because the password gets passed to the server which can see it in the clear. The server then hashes it with salt and then looks up the data base and sees if that password matches.

Ideally we would switch to using some kind of signature based login. But that seems like its going to take some time.

if (hash(password, salt) is in password_database): login() else: pass

This sucks because the server gets the clear text password.

Why not do the hashing on the users side

A naive solution is to send the hash(password, salt)

to the server. But the problem is that the hashed password becomes the new password.

Use snarks to prove possession of the password.

We create a snark proof that checks

public hash(password, salt) public hash(password, blockhash)

The server checks the snark proofs is correct and that hash(password, salt) is in its password database.

if (hash(password, salt) is in password_database and snark.is_valid()): login() else: pass

Conclusion

Now the server does not need to ever see the users password and can still allow them to login. Previous logins cannot be replayed because we use an block hash as randomness.

We could maybe use this to make ETH brain wallets. But I am really worried about dictionary attacks as the salt will be public. Would need to check with someone who understands this more to see the kind of password that cannot be brute forced.