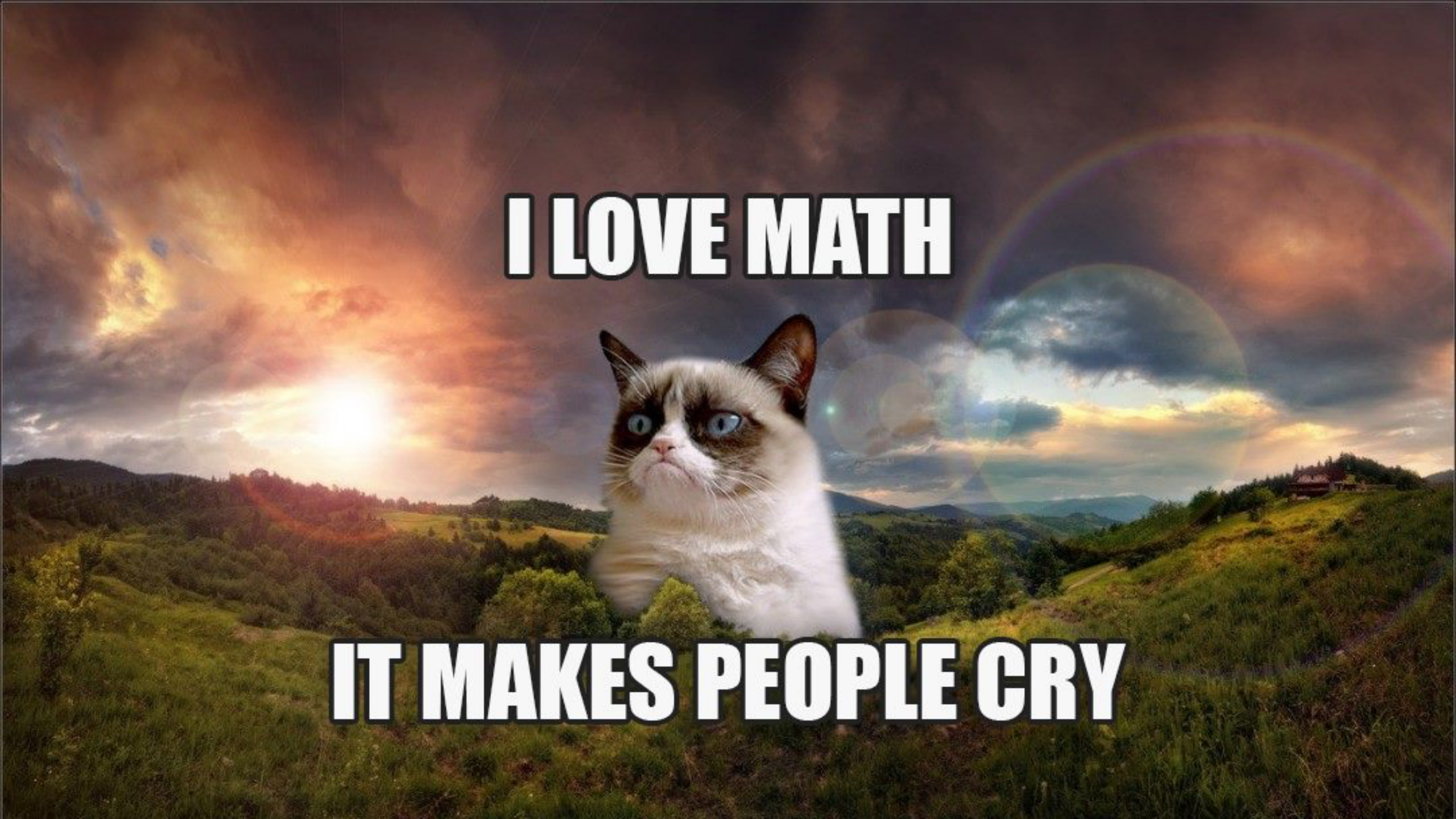
JavaScript
fwdays

Password Hashing

Emil Bay
CommodiTrader

Technical Founder, CommodiTrader

@emilbayes 🇩🇰

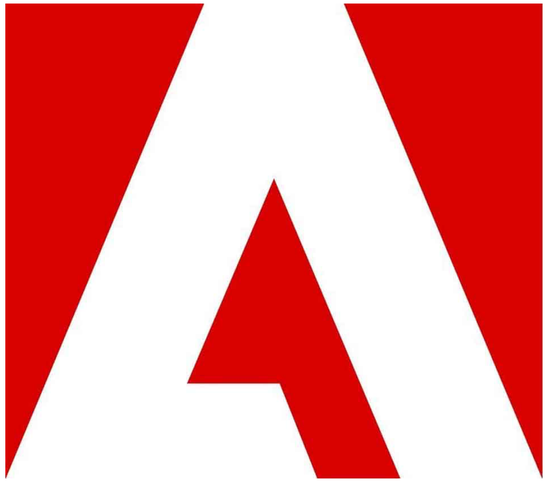

I LOVE MATH
IT MAKES PEOPLE CRY

Passwords

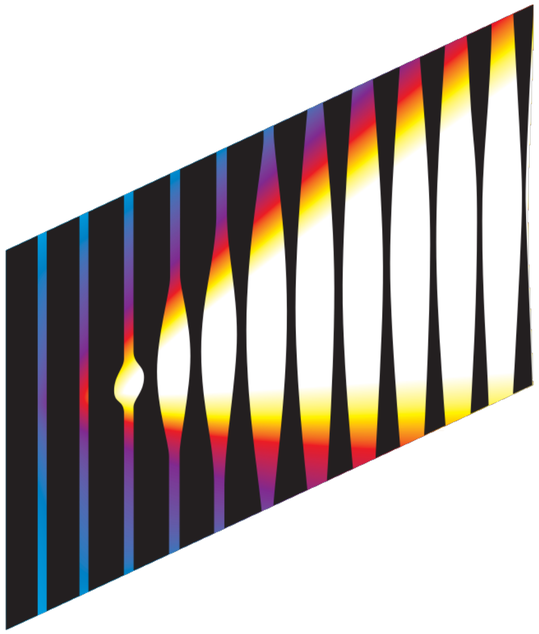
Proves your identity

1. No security holes
2. Unique passwords
3. Safer storage

1. No security holes
2. Unique passwords
3. Safer storage

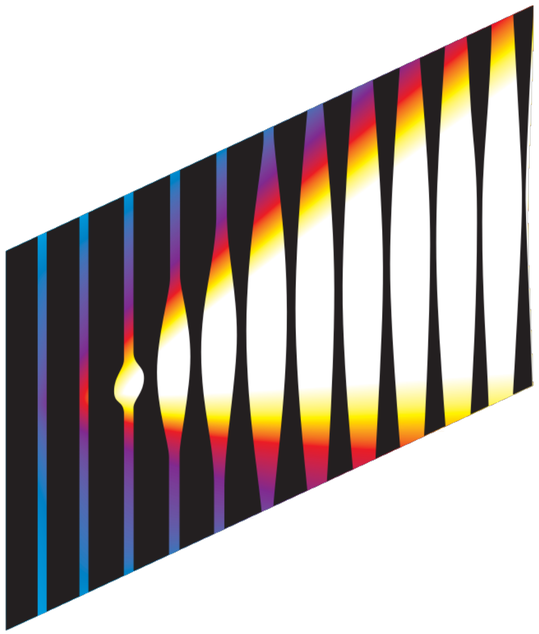


Bell



Johannes Bader, hacker_two, CC BY 2.0

Plaintext



SONY
PICTURES

Bell

Problems:

Plaintext!

Solution:

Obscure password

Hash

1. Deterministic
2. Pre-image resitant (one-way)
3. Second pre-image resitant (voluntary collisions)
4. Collision resistance (involuentary collisions)

$$\{0,1\}^* \mapsto \{0,1\}^n$$

~~MD5~~

~~SHA-1~~

SHA-2 (SHA-256 & SHA-512)

Blake2

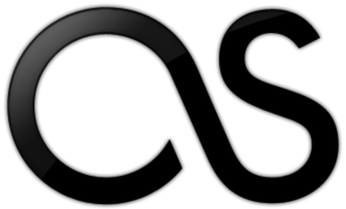
SHA-3 (Keccak)

bcrypt, scrypt

Argon2

```
var crypto = require('crypto')

var hash = crypto.createHash('md5')
  .update(password)
  .digest()
```



```
var words = //...

var rainbowTable = words.map(function (word) {
  return crypto.createHash('md5')
    .update(word)
    .digest()
})

var idx = indexOf(hash, rainbowTable)
var plaintext = words[idx]
```

Problems:

Rainbow Tables

Solution:

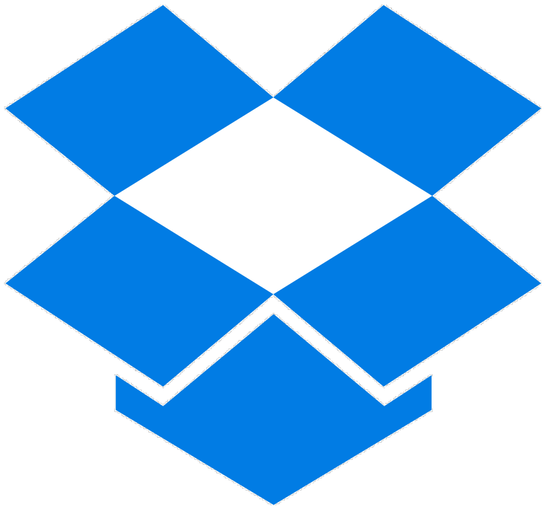
Make identical passwords unique hashes

Salted Hash

Makes precomputation impractical
aka Rainbow Tables

```
var crypto = require('crypto')

var salt = crypto.randomBytes(64)
var hash = crypto.createHash('md5')
  .update(salt)
  .update(':')
  .update(password)
  .digest()
```



Problems:

Still too efficient

Solution:

Expend more resources

Iterated Hash

NIST compliant

```
var crypto = require('crypto')

var salt = crypto.randomBytes(128)
var iter = 10000
var hashLen = 512
var algo = 'sha512'

crypto.pbkdf2(password, salt, iter, hashLen, algo, function (err, hash) {
  // ...
})
```

Problems:

Very easy to parallelize

Solution:

Make even more expensive!

KDF

*Key Derivation Function
Purposefully slow
Uses lots of memory and computation

```
npm install secure-password
```

Argon2

Winner of PHC

Problems:

- Blocks event loop

Solution:

- Use async API

```
var pwd = require('secure-password')()

var userPassword = Buffer.from('my-password')
pwd.hash(userPassword, callback)
```

```
var sodium = require('sodium-native')
```

```
var sodium = require('sodium-universal')
```

```
var SecurePassword = require('secure-password')
```

```
var SecurePassword = require('secure-password')
```