

# Authentication

# Objectives

- Explain Identification and Authentication.
- Explain why passwords are terrible.
- Explain what makes a good hashing algorithm.
- Store a password *securely* using BCrypt.



Identification  
Who am I?  
^How do I prove it?

User name  
Password

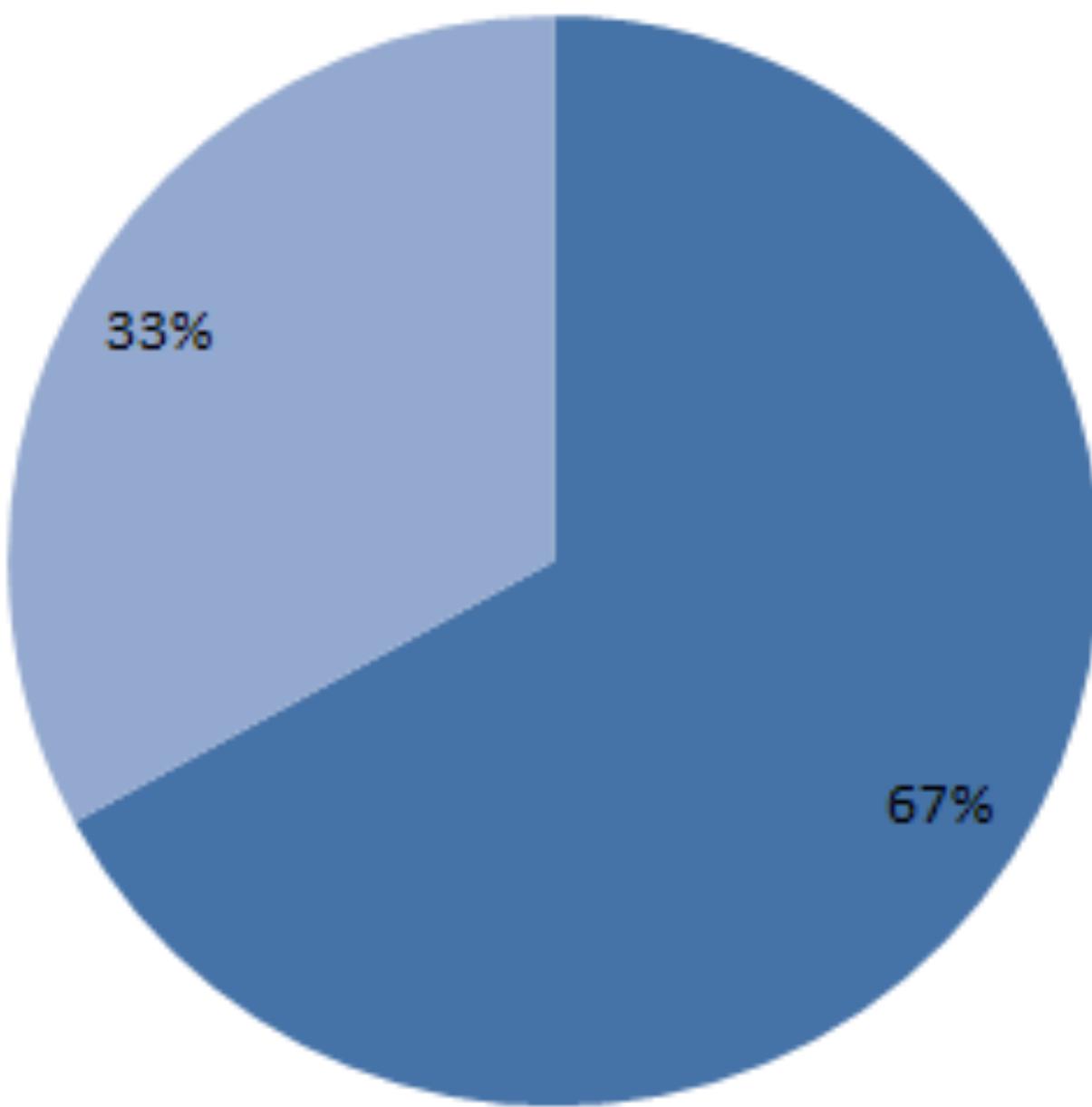
Cancel

Next

X

## Password reuse across Sony and Gawker

■ Identical password ■ Unique password



# THE TOP (WORST) PASSWORDS

RANK	PASSWORD
1.....	123456
2.....	PASSWORD
3.....	12345
4.....	12345678
5.....	QWERTY
6.....	123456789
7.....	1234
8.....	BASEBALL
9.....	DRAGON
10.....	FOOTBALL

RANK	PASSWORD
11.....	1234567
12.....	MONKEY
13.....	LETMEIN
14.....	ABC123
15.....	111111
16.....	MUSTANG
17.....	ACCESS
18.....	SHADOW
19.....	MASTER
20.....	MICHAEL

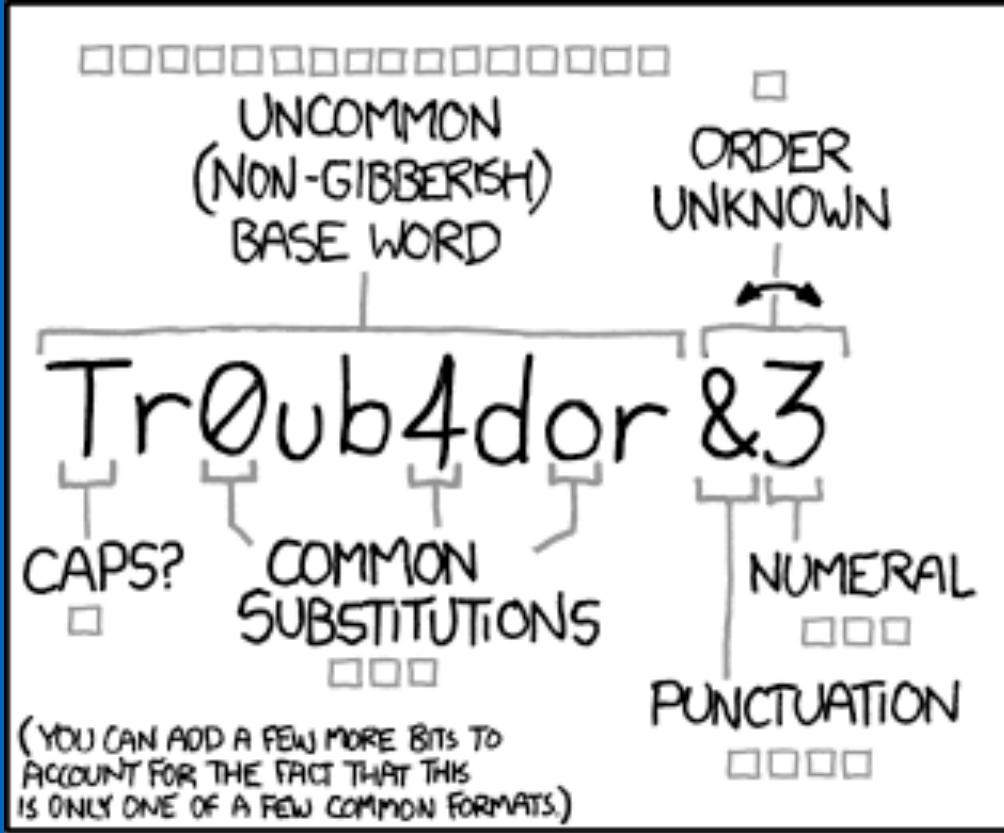
RANK	PASSWORD
21.....	SUPERMAN
22.....	696969
23.....	123123
24.....	BATMAN
25.....	TRUSTN01

Credit: SplashData

## Help: List of Password Rules

1. The password must be exactly 8 characters long.
2. It must contain at least one letter, one number, and one special character.
3. The only special characters allowed are: @ # \$
4. A special character must not be located in the first or last position.
5. Two of the same characters sitting next to each other are considered to be a "set." No "sets" are allowed.
6. Avoid using names, such as your name, user ID, or the name of your company or employer.
7. Other words that cannot be used are Texas, child, and the months of the year.
8. A new password cannot be too similar to the previous password.
  - a. Example: previous password - abc#1234, acceptable new password - acb\$1243
  - b. Characters in the first, second, and third positions cannot be identical. (abc\*\*\*\*\*)
  - c. Characters in the second, third, and fourth positions cannot be identical. (\*bc#\*\*\*\*)
  - d. Characters in the sixth, seventh, and eighth positions cannot be identical. (\*\*\*\*\*234)
9. A password can be changed voluntarily (no Help Desk assistance needed) once in a 15-day period. If needed, the Help Desk can reset the password at any time.
10. The previous 8 passwords cannot be reused.

[Top of page](#)



~28 BITS OF ENTROPY

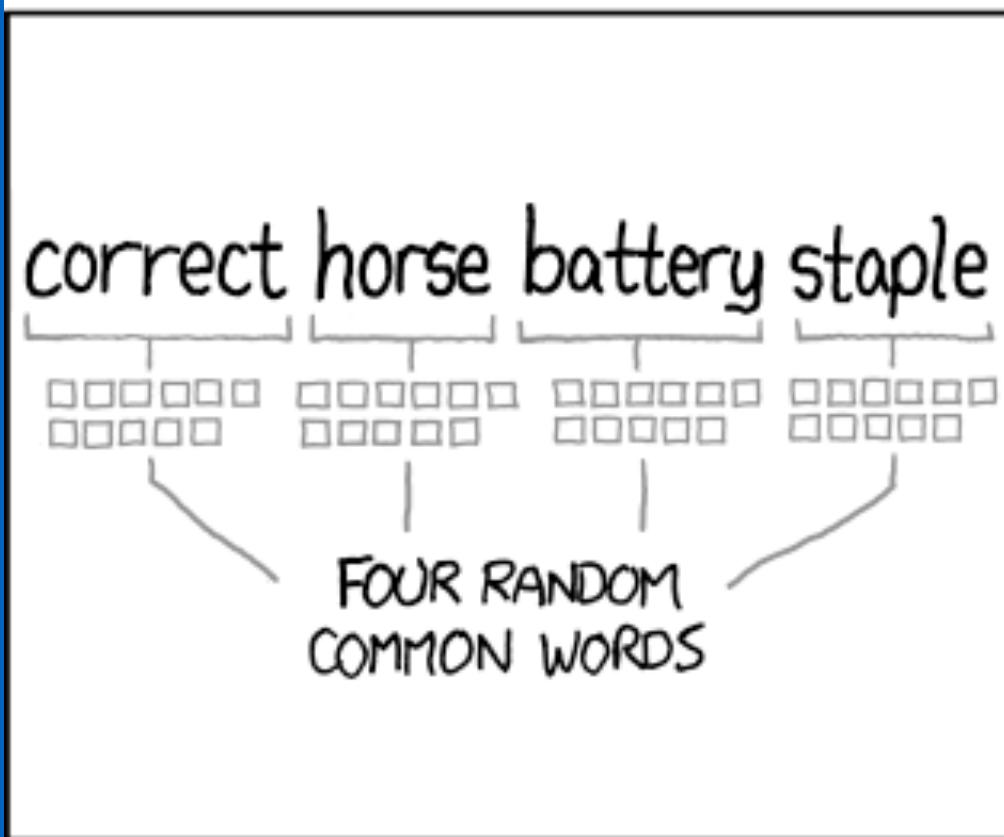
$2^{28} = 3$  DAYS AT 1000 GUESSES/SEC

(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)

DIFFICULTY TO GUESS: **EASY**

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?  
AND THERE WAS SOME SYMBOL...

DIFFICULTY TO REMEMBER: **HARD**



~44 BITS OF ENTROPY

$2^{44} = 550$  YEARS AT 1000 GUESSES/SEC

DIFFICULTY TO GUESS: **HARD**

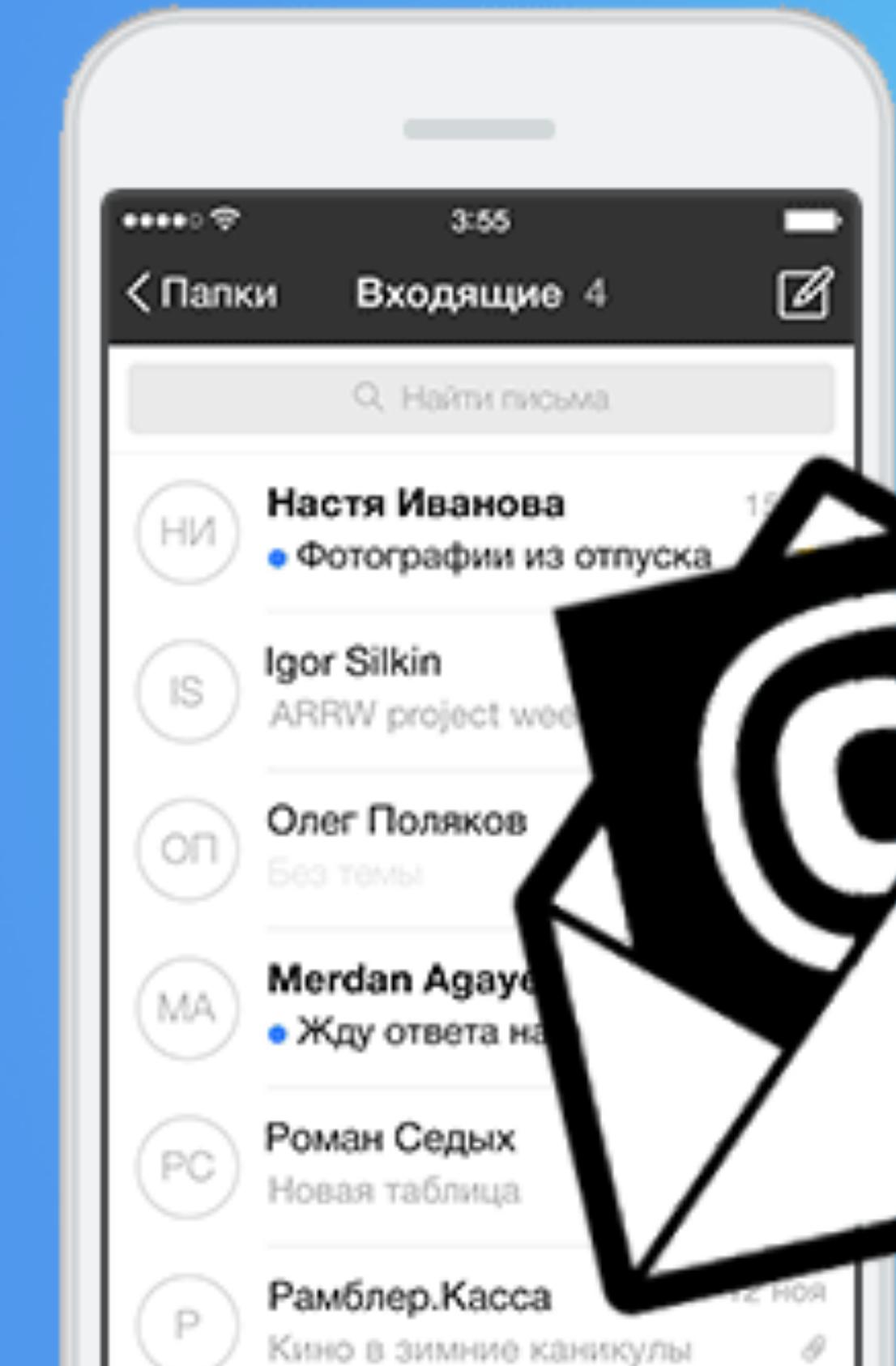
THAT'S A BATTERY STAPLE.  
CORRECT!

DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

# Storing Passwords

Plaintext



# Rambler.ru Hacked

# 100 Million Plaintext Passwords Leaked



# First Rule

Never store passwords in plain text.

# Second Rule

**Never store passwords in plain  
text!**

Always ask:

Do I actually need to store this?

What if there was a way to verify that a password was correct, without knowing the original password?

$$X_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



# Hashing Algorithms

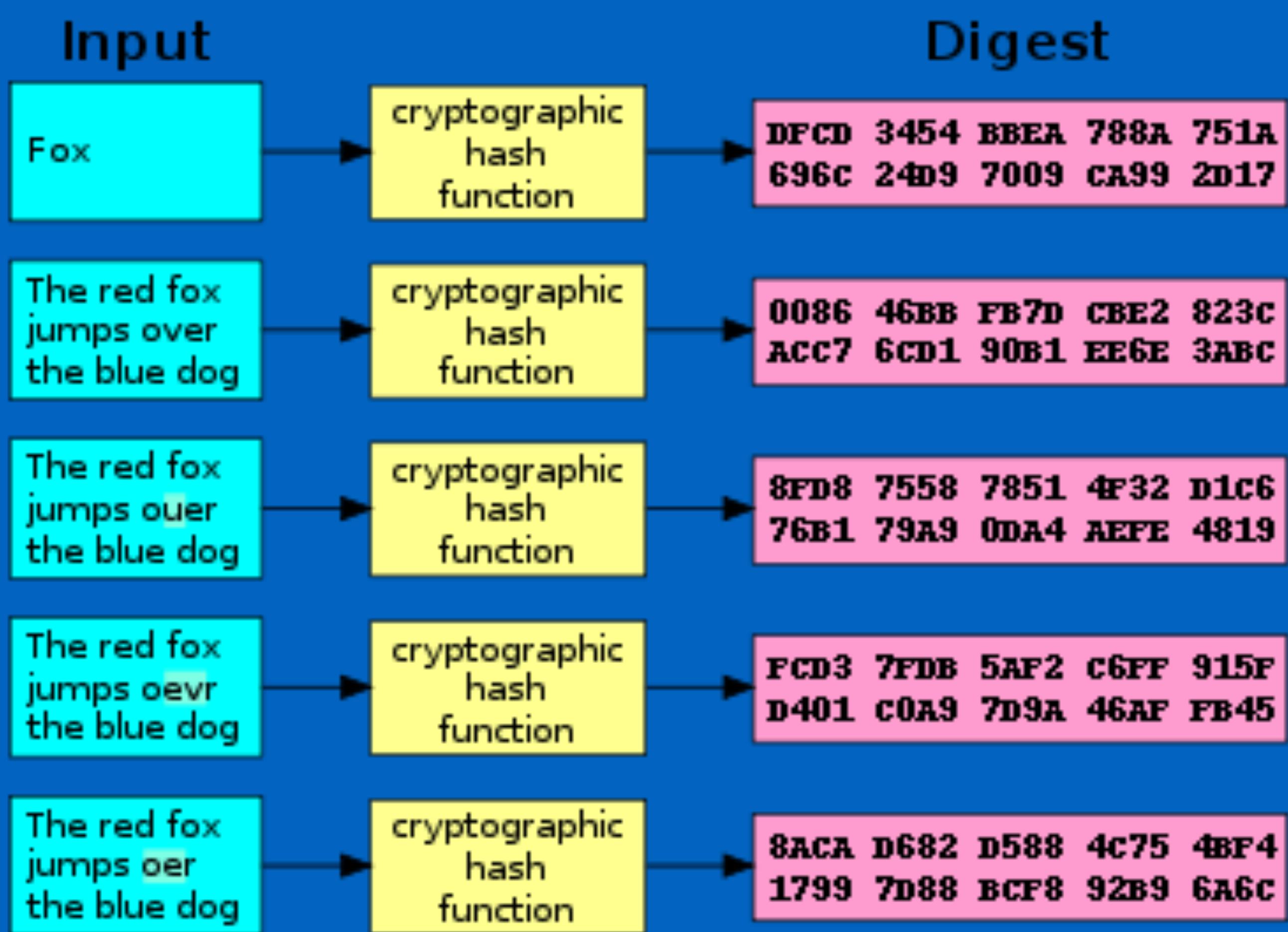


$$\begin{aligned} X &= 6 - 2Y \\ X + a &= b \\ f(x) &= \tan x \end{aligned}$$

$$f(x) = \sin x$$



**Password is one way.**



Hash needs to be the same size.

EASY



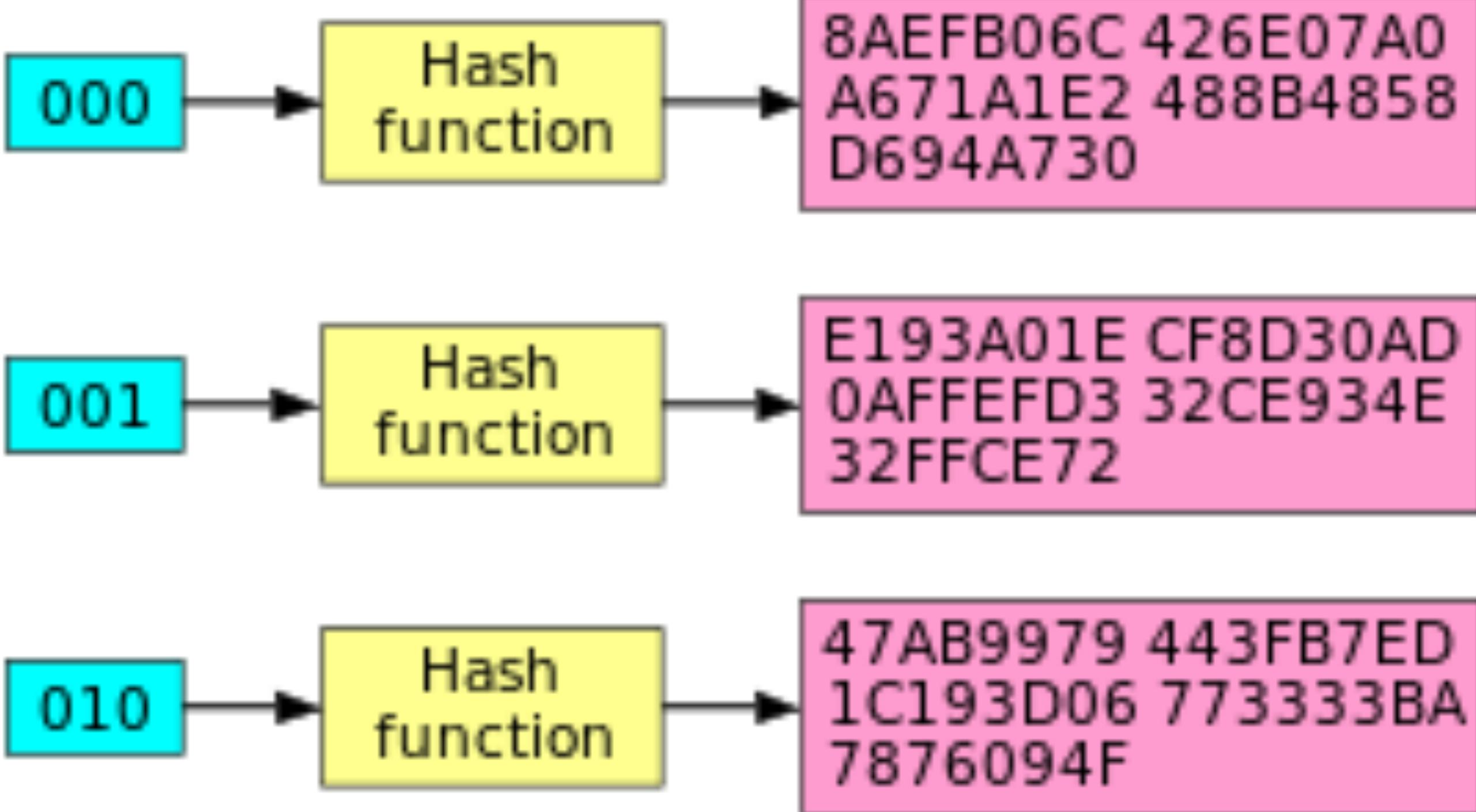


**No Collisions!**



# Input

# Hash sum



<http://www.miraclesalad.com/webtools/md5.php>

Are We Safe?



e10adc3949ba59abbe56e057f20f883e  
<http://md5cracker.org/>

-::TYPE	-::HASH	-::PASS	-::STATUS	-::TIME	-::SUBMITTED
md5	7e89bcc6151b24992a255cd665d4aa16		waiting	0:0:46	2006-11-11 10:45:31
md5	0696eeaff05bf2105b0bcf6d93ac73a0		waiting	0:0:47	2006-11-11 10:45:30
md5	db549b9d18aabe8ad07aa3d9338d441c		waiting	0:1:38	2006-11-11 10:44:39
md5	70c9ecbd2512460fa861de25fb3d7c6e		waiting	0:24:8	2006-11-11 10:22:09
md5	c32cf089d464d3ed1a3af347ae208188		processing3	0:25:6	2006-11-11 10:21:11
md5	c6fe5851aff10a64e8a52e82b323304f		processing3	0:46:29	2006-11-11 09:59:48
md5	a79c879d28c5c8a4707d52bbbaa57607f	12050	cracked	0:45:41	2006-11-11 09:51:43
md5	a79e1c64d27737e3f959a6a56b41c650		processing3	0:57:18	2006-11-11 09:48:59
md5	2ef5b8b0eee93568a1126bb923664057		processing3	0:57:36	2006-11-11 09:48:41
md5	e53cc072934b25e45dc273c6c342556d		processing3	0:58:7	2006-11-11 09:48:10
md5	d38ad0e58c9525343f492161b87400a1	htmlldb	cracked	0:58:23	2006-11-11 09:44:01
md5	d926dbaeb7fac97612ec219f7f172610		processing3	1:4:30	2006-11-11 09:41:47
md5	fcf2483ced17683085849877134fd50c		processing3	1:6:32	2006-11-11 09:39:45
md5	377a8f80271a6f920df0e4aa84d1029a	bombi	cracked	0:43:12	2006-11-11 09:38:26
md5	85d95e2ad51bfcd5d6d352486fbe2769	pupsi	cracked	1:8:2	2006-11-11 09:28:25
md5	96bc2c727049b5dce27bd8b9e8b264bf		processing3	1:19:6	2006-11-11 09:27:11
md5	8aa12bbde69504ba86b942726b4d7623		notfound	1:18:15	2006-11-11 09:02:54
md5	5ce1d809749963448767622e0ca8169f	28264451	cracked	0:48:15	2006-11-11 09:02:35

Rainbow Tables/Precomputed Lookup Tables.

# Third Rule

# Never just store hashes!



**Salt Those Passwords**  
**Demo Salt**  
**Who would do this?**

As you may or may not recall, given how much time has passed, **hackers broke into LinkedIn's network back in 2012**, stole some 6.5 million encrypted passwords, and posted them onto a Russian hacker forum. Because the **passwords were stored** as unsalted SHA-1 hashes, hundreds of thousands were quickly cracked.

This time, the breach is said to [come from MySpace](#), and the number of passwords claimed is an eye-popping 427 million.

Apparently, there are only 360 million users on the list, but some accounts have more than one password listed, for reasons that aren't explained.

Once again, the passwords allegedly exposed in this breach were simple, [unsalted](#) SHA-1 hashes, vulnerable to just the same sort of high-speed *try 'em all* attack as in the LinkedIn breach of 2012.



# Slowness



# Easy Solution

# BCrypt

# BCrypt gives us

1. Good cryptographic hashing algorithm.
2. Automatically generated salts.
3. Control over the computational complexity.

From @garthk, on a 2GHz core you can roughly expect:

```
rounds=8 : ~40 hashes/sec
rounds=9 : ~20 hashes/sec
rounds=10: ~10 hashes/sec
rounds=11: ~5 hashes/sec
rounds=12: 2-3 hashes/sec
rounds=13: ~1 sec/hash
rounds=14: ~1.5 sec/hash
rounds=15: ~3 sec/hash
rounds=25: ~1 hour/hash
rounds=31: 2-3 days/hash
```

The value you submit there is not just the number of rounds that the module will go through to hash your data. The module will use the value you enter and go through  $2^{\text{rounds}}$  iterations of processing.

# BCrypt - How It Works

`bcrypt.hashSync(req.body.password, 8)`

returns

`$2a$08$.jEA1FSMX0SGQhLXrKkN9.k9gyad8xN6r76Yq0IzoA318fhoqUp7a`

To Compare

`bcrypt.compareSync(myPlaintextPassword, hash);`

**Version/Cost (SaltRounds)/Salt/Hash**

# Objectives

- Explain Identification and Authentication.
- Explain why passwords are terrible.
- Explain what makes a good hashing algorithm.
- Store a password *securely* using BCrypt.

# Questions?