Quantitative Cyber-Security

Colorado State University

Yashwant K Malaiya

CS559

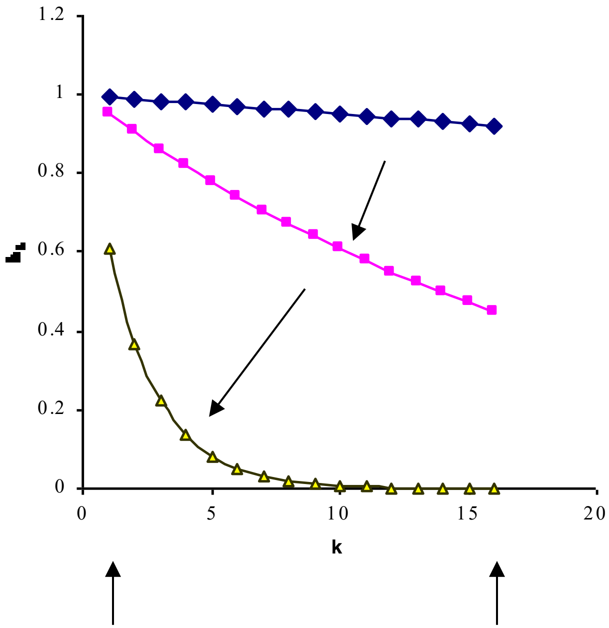
L18

CSU Cybersecurity Center
Computer Science Dept

Detectability Profile

- To test a potential fault, it needs to be triggered and error needs to be sensed.
- Some faults are easy to test, some are very hard to test.
- As testing and debugging progresses, the remaining faults are the ones that are harder to find.

- Corner cases: at extreme values for multiple variables.



Hard to test Low hanging fruit

Colorado State University

Software Reliability Growth models

- Time-based models:
  - Defect discovery rate = f(calendar time)
  - Cumulative number of defects discovered = f(calendar time)
    - Exponential and Logarithmic models
- Coverage based models
  - Cumulative number of defects discovered =f(coverage achieved)

Term Project

All submissions should follow the 2-column *format for IEEE conference papers*.

- Proposal and sources: Oct 10
- Semi-final report: Nov 7
  - It should indicate that you have finished at least two-thirds of the work. It should include an abstract, discussion of background literature, a summary of the investigations/findings, any refinements of the proposal objectives as a result of the past study, what the final report will contain and the applicable references.
  - Technical details, equations/tables/plots/screen-shots
  - You must be aware of the current trends in research/industry.
- Slides: Due Nov 18
- Ten-minute oral presentation Nov 19-Dec 8

Quantitative Cyber-Security

Colorado State University

Yashwant K Malaiya

CS559

Paswords

CSU Cybersecurity Center
Computer Science Dept

Authentication

- Authentication: the process of verifying an actor's identity
- Needed for security of systems
  - Permissions, capabilities, and access control are all contingent upon knowing the identity of the actor
- Parameterized as a username and a user's proprietary information
  - The proprietary information attempts to limit unauthorized access

Types of proprietary info

- Actors provide their proprietary information to login to a system
- Three classes of proprietary info:
  1. Something you know
     - Example: a password
  2. Something you have
     - Examples: a smart card or smart phone
  3. Something you are
     - Examples: fingerprint, voice scan, iris scan

Checking Passwords

- The system must validate passwords provided by users
- Thus, passwords must be stored somewhere
- Simple scheme: plain text (is this good?)

password.txt	
cbw	p4ssw0rd
sandi	i heart doggies
amislove	93Gd9#jv*0x3N
bob	security

Problem: Password guessing

How easy it is to guess a password?

- If your keyboard has R= 95 unique characters,
- randomly constructing a password from that whole set, 12-character password, then L = 12.
- 95^{12}= 540,360,087,662,636,962,890,625 passwords

Entropy = $\log_2(R^L)$ = 78.9 bits assuming passwords are created randomly

Ascii is 8 bits. Thus about 2^{12x8}

- Non-randomness makes password guessing easier.
- Measures of password strength proposed and used

Password guessing at login? Can be defeated by

- Limited number of tries: 3-5
- Blocking attempts from unknown/suspected IP addresses

Colorado State University

Problem: Password File Theft

- Attackers often compromise systems
- They may be able to steal the password file
  - Linux: /etc/shadow
  - Windows: c:\windows\system32\config\sam
- If the passwords are plain text, what happens?
  - The attacker can now log-in as any user, including root/administrator
- **Passwords should never be stored in plain text**

Famous Password breaches

	Top 5	
Yahoo	2013; 2014	3 billion; 500 million
First American Financial Corp	2019	885 million
Facebook	2019	540 million
Marriott International	2018	500 million
Friend Finder Networks	2016	412.2 million

According a Verizon Data Breach Investigations Report,

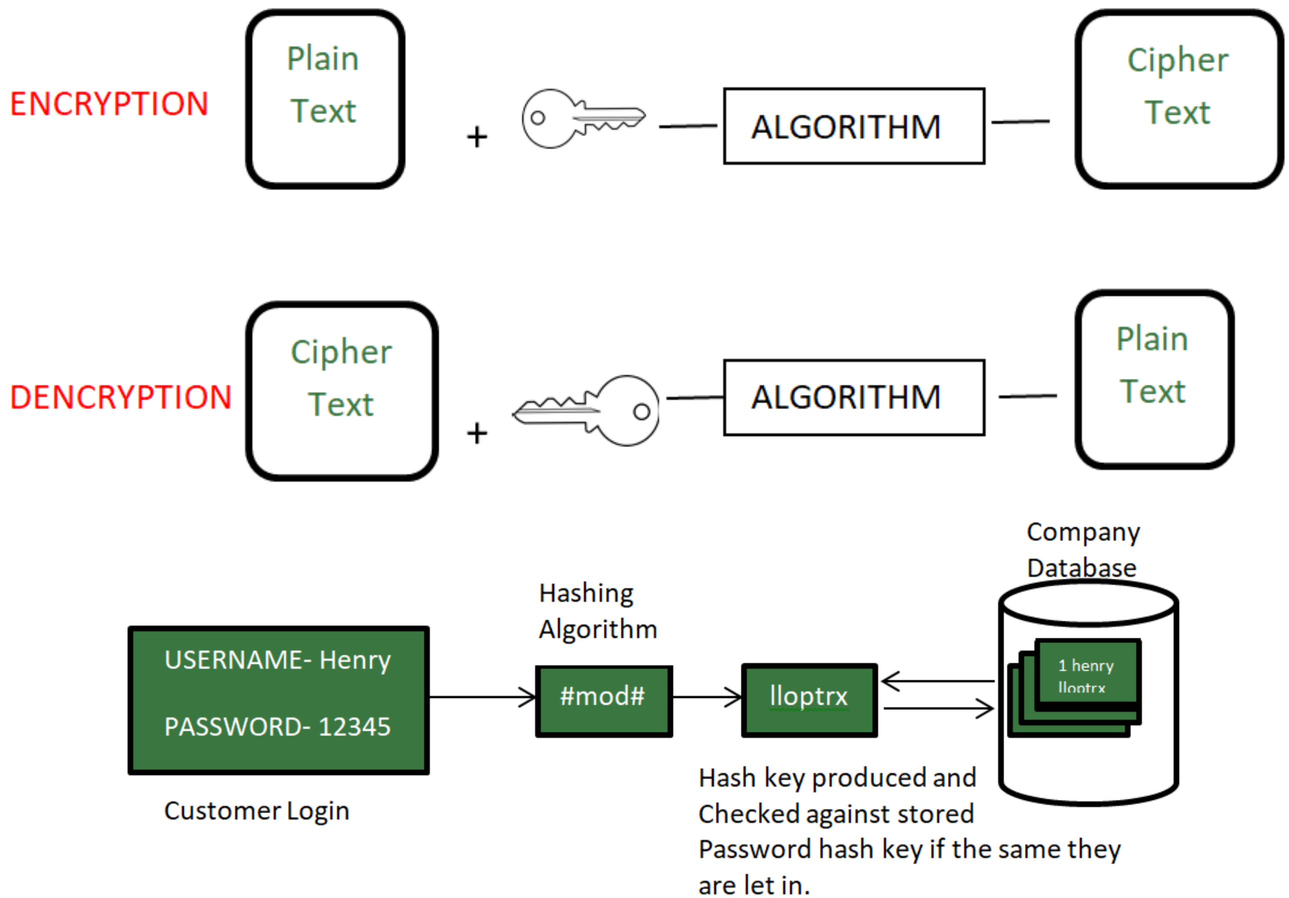
- over 70% of employees reuse passwords at work.
- “81% of hacking-related breaches leveraged either stolen and/or weak passwords.”

Colorado State University

Problem: Password File Theft

- **Attackers often compromise systems**
- They are often able to steal the password file
  - Linux: /etc/shadow
  - Windows: c:\windows\system32\config\sam
- If the passwords are plain text, what happens?
  - The attacker can now log-in as any user, including root/administrator
- Thus Passwords should never be stored in plain text, but using ..

Encryption vs Hashing



- Impossible to reconstruct password from hash

Hashed Passwords

- Key idea: store encrypted versions of passwords
  - Use one-way cryptographic hash functions
  - Examples: md5, sha1, sha256, sha512
- Cryptographic hash function transform input data into scrambled output data
  - Deterministic: hash(A) = hash(A)
  - High entropy:
    - md5(‘security’) = e91e6348157868de9dd8b25c81aebfb9
    - md5(‘security1’) = 8632c375e9eba096df51844a5a43ae93
    - md5(‘Security’) = 2fae32629d4ef4fc6341f1751b405e45
  - Collision resistant
    - Locating A’ such that hash(A) = hash(A’) takes a long time
    - Example: 2^{21} tries for md5

Md5: 128 bit

Ack: Northeastern U

Colorado State University

16

Hashed Password Example

User: cbw

md5('p4ssw0rd') =
2a9d119df47ff993b662a8ef36f9ea20

md5('2a9d119df47ff993b662a8ef36f9ea20')
= b35596ed3f0d5134739292faa04f7ca3

hashed_passw xt	
cbw	2a9d119df47ff993b662a8ef36f9ea20
sandi	23eb06699da16a3ee5003e5f4636e79f
amislove	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Attacking Password Hashes

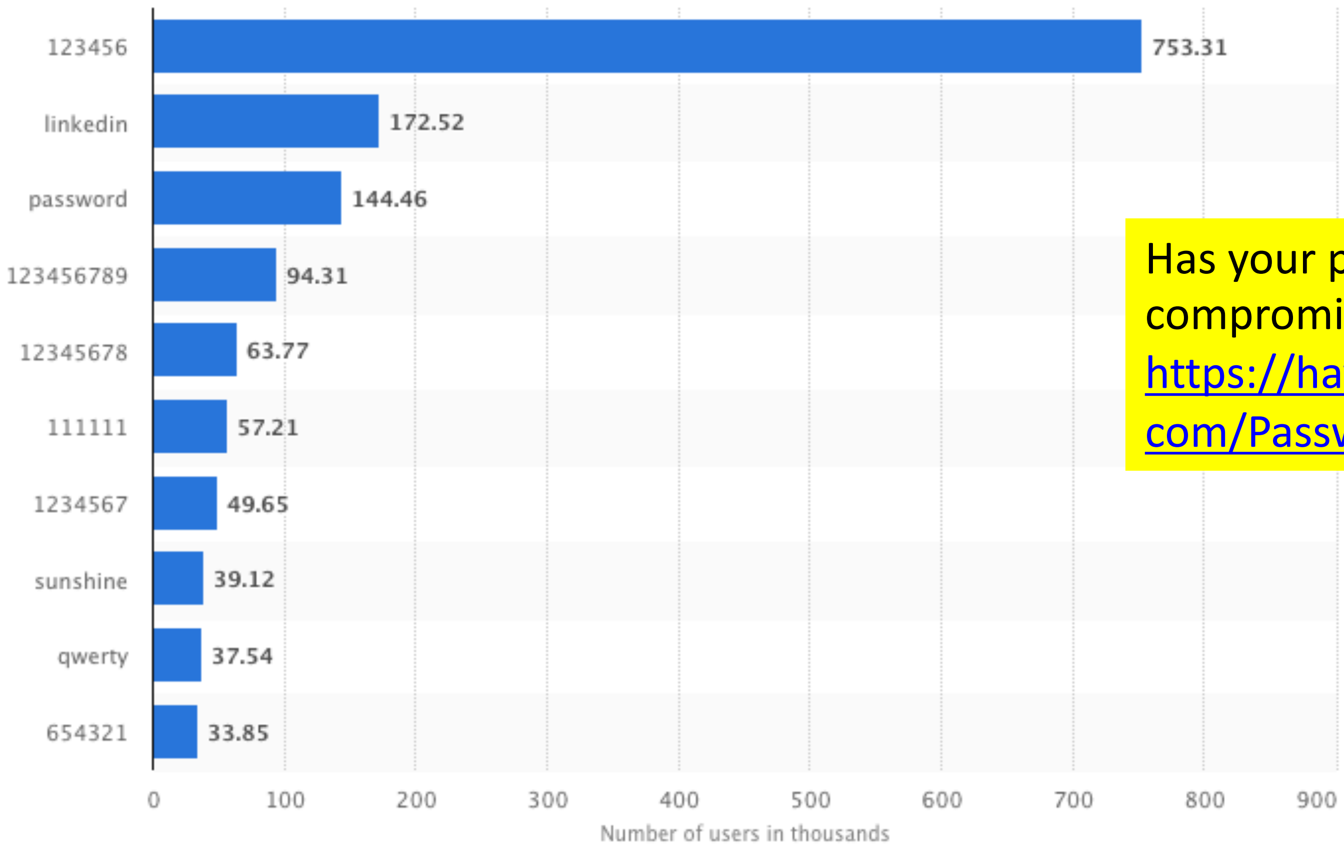
- Problem: users choose poor passwords
  - Most common passwords: 123456, password
  - Username: cbw, Password: cbw
  - Common password patterns
- Weak passwords enable dictionary attacks

Default passwords (*password*, *default*, *admin*, *guest etc*) if not changed can be a security hazard.

18

Most Common passwords

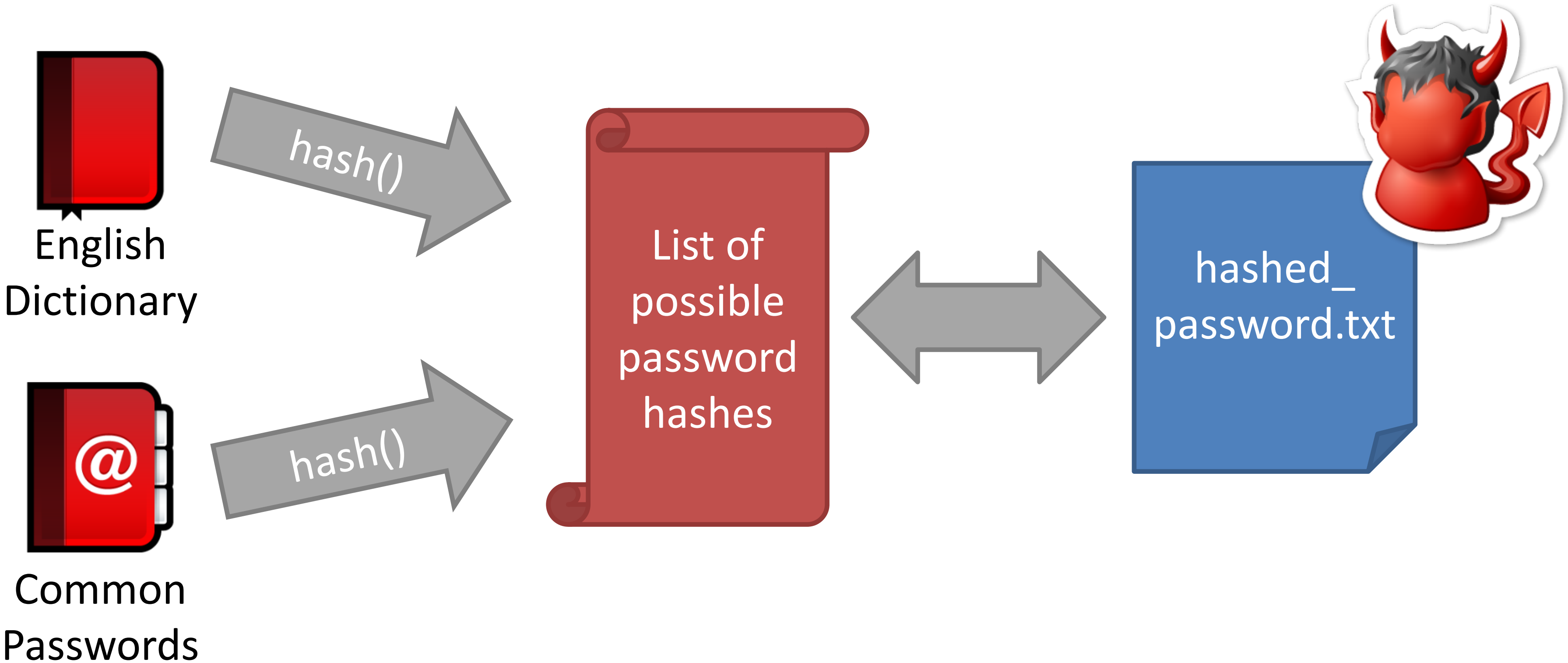
- **Most common passwords unscrambled from the 2012 leaked LinkedIn.com dataset as of 2016** *(in 1,000s)*



Has your password been compromised? https://haveibeenpwned.com/Passwords

https://www.statista.com/statistics/271098/most-common-passwords/

Dictionary Attacks



- Common for 60-70% of hashed passwords to be cracked in <24 hours

Hardening Password Hashes

- Key problem: cryptographic hashes are deterministic
  - hash('p4ssw0rd') = hash('p4ssw0rd')
  - This enables attackers to build lists of hashes
- Solution: make each password hash unique
  - Add a **salt** to each password before hashing
  - hash(salt + password) = password hash
  - Each user has a *unique, random* salt
  - Salts can be stores in plain text

21

Example Salted Hashes

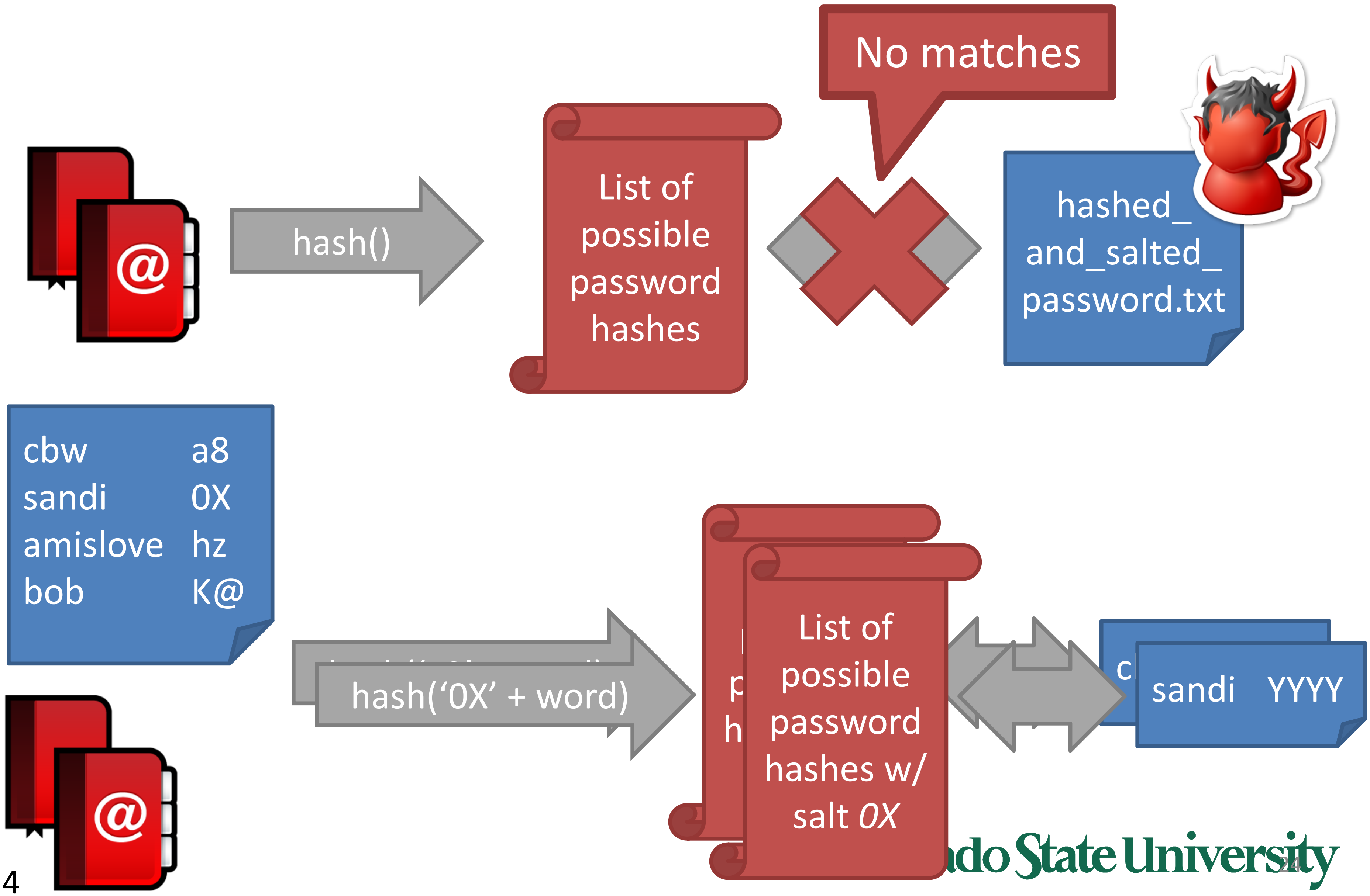
hashed_password.txt

cbw	2a9d119df47ff993b662a8ef36f9ea20
sandi	23eb06699da16a3ee5003e5f4636e79f
amislove	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

hashed_and_salted_password.txt

cbw	a8	af19c842f0c781ad726de7aba439b033
sandi	0X	67710c2c2797441efb8501f063d42fb6
amislove	hz	9d03e1f28d39ab373c59c7bb338d0095
bob	K@	479a6d9e59707af4bb2c618fed89c245

Attacking Salted Passwords



Breaking Hashed Passwords

- **Stored passwords should always be salted**
  - Forces the attacker to brute-force each password individually
- Problem: it is now possible to compute cryptographic hashes very quickly
  - GPU computing: hundreds of small CPU cores
  - nVidia GeForce GTX Titan Z: 5,760 cores
  - GPUs can be rented from the cloud very cheaply
    - 2x GPUs for $0.65 per hour (2014 prices)

Examples of Hashing Speed

- A modern x86 server can hash all possible 6 character long passwords in 3.5 hours
  - Upper and lowercase letters, numbers, symbols
  - $(26+26+10+32)^6$ = 690 billion combinations
- A modern GPU can do the same thing in 16 minutes
- Most users use (slightly permuted) dictionary words, no symbols
  - Predictability makes cracking much faster
  - Lowercase + numbers → $(26+10)^6$ = 2B combinations

Hardening Salted Passwords

- Problem: typical hashing algorithms are too fast
  - Enables GPUs to brute-force passwords
- Solution: use hash functions that are designed to be **slow**
  - Examples: bcrypt, scrypt, PBKDF2
  - These algorithms include a work factor that increases the time complexity of the calculation
  - scrypt also requires a large amount of memory to compute, further complicating brute-force attacks

bcrypt Example

- Python example; install the *bcrypt* package

```
[cbw@ativ9 ~] python
>>> import bcrypt
>>> password = "my super secret password"
>>> fast_hashed = bcrypt.hashpw(password, bcrypt.gensalt(0))
>>> slow_hashed = bcrypt.hashpw(password, bcrypt.gensalt(12))
>>> pw_from_user = raw_input("Enter your password:")
>>> if bcrypt.hashpw(pw_from_user, slow_hashed) == slow_hashed:
…       print "It matches! You may enter the system"
…   else:
…       print "No match. You may not proceed"
```

Work factor

Colorado State University

Password Storage Summary

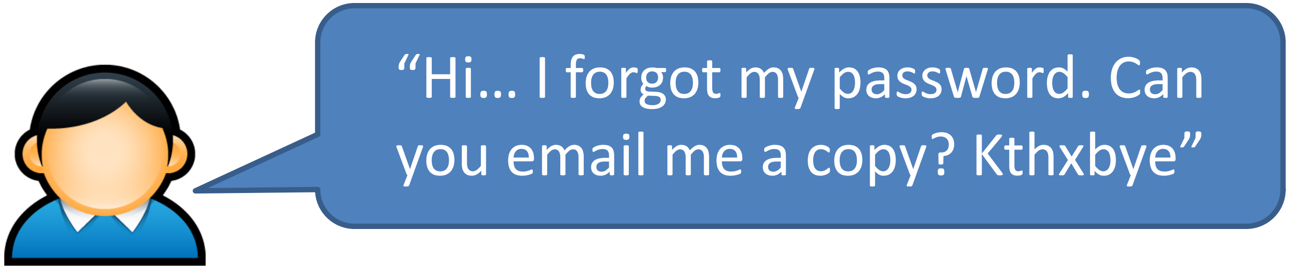
1. **Never store passwords in plain text**
2. **Always salt and hash passwords before storing them**
3. **Use hash functions with a high work factor**

- These rules apply to any system that needs to authenticate users
  - Operating systems, websites, etc.

Colorado State University

Password Recovery/Reset

- Problem: hashed passwords cannot be recovered



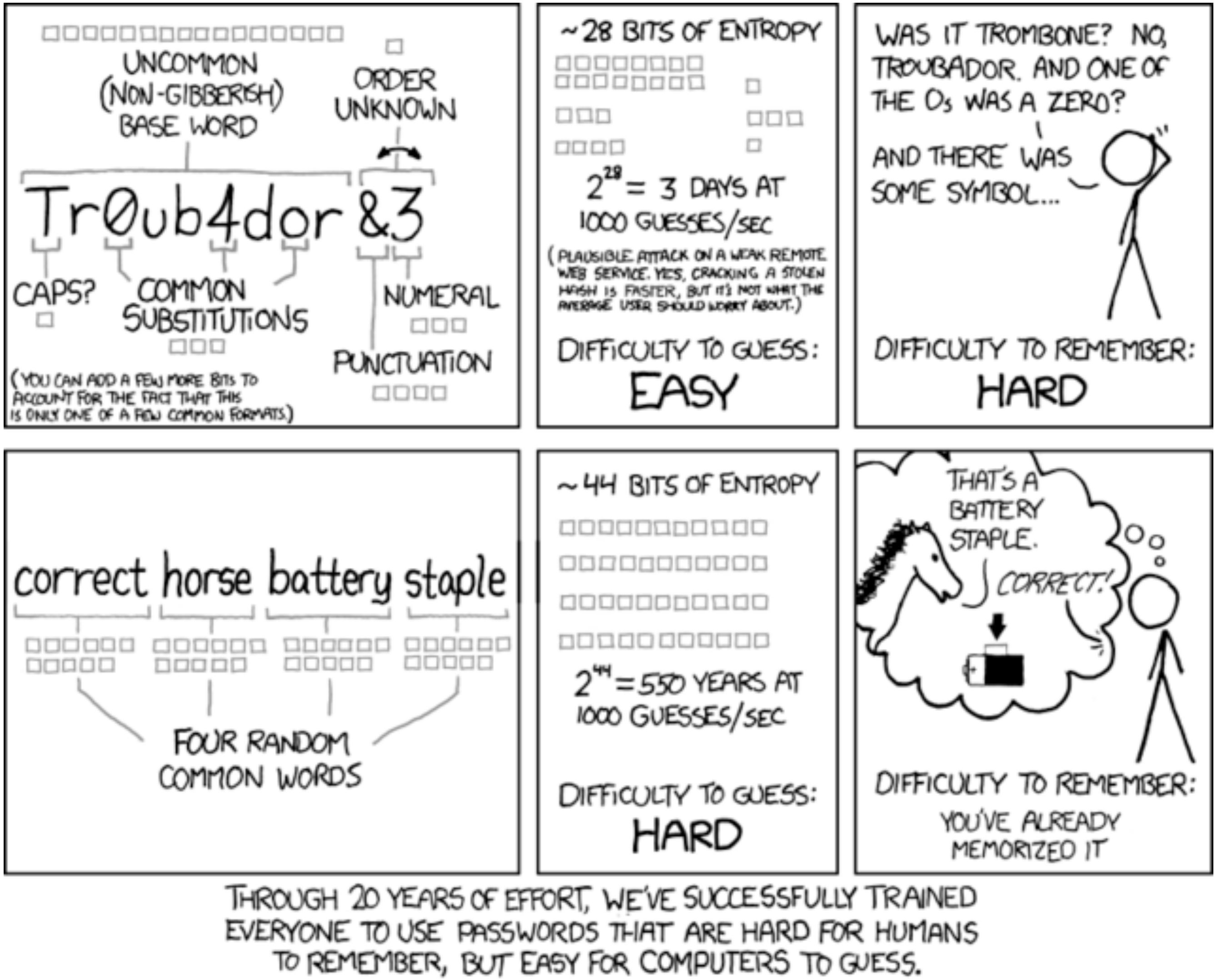
- This is why systems typically implement password reset
  - Use out-of-band info to authenticate the user
  - Overwrite hash(old_pw) with hash(new_pw)
- Be careful: its possible to crack password reset

Password crackers

Forgotten passwords

- Too many passwords to remember
- “Strong” passwords can be hard to remember
- Traditional approach: user physically requests password reset
  - Using phone numbers or email addresses on record
  - Showing IDs
- Danger: fraudulently obtaining password using social engineering. May represent the weakest link in the password system.

Good passwords are bad



Security Questions

- Security Questions are used to authenticate when
  - Suspicious attempts
  - Forgotten passwords
- 33-39% may be guessed by family members. Friends or those with access to personal information
- 20% of users could not remember their own answers.
- Possible solution: Multiple questions with a minimum threshold of right answers

Multifactor Authentication

- Smartphone with number xxx-xxx-xxxx: one in 10^{10}
  - About 33 bits of entropy
- Fingerprints might be unique. However information my be lost when 25-80 minutiae are used for comparison. Uniqueness still being researched.
- Face recognition: 97.25% accuracy?

Password managers

- Can record username, password, form information etc.

for automatic filling.

  - Locally on a device
  - On the web

- Can generate good passwords
- The master password may be kept locally. If you forget it, you may have to extract it yourself.
- Some browsers may include password management capabilities
- Can protect against keyloggers

Disadvantages:

  - May have vulnerabilities
  - May be blocked by some websites