2550 Intro to cybersecurity

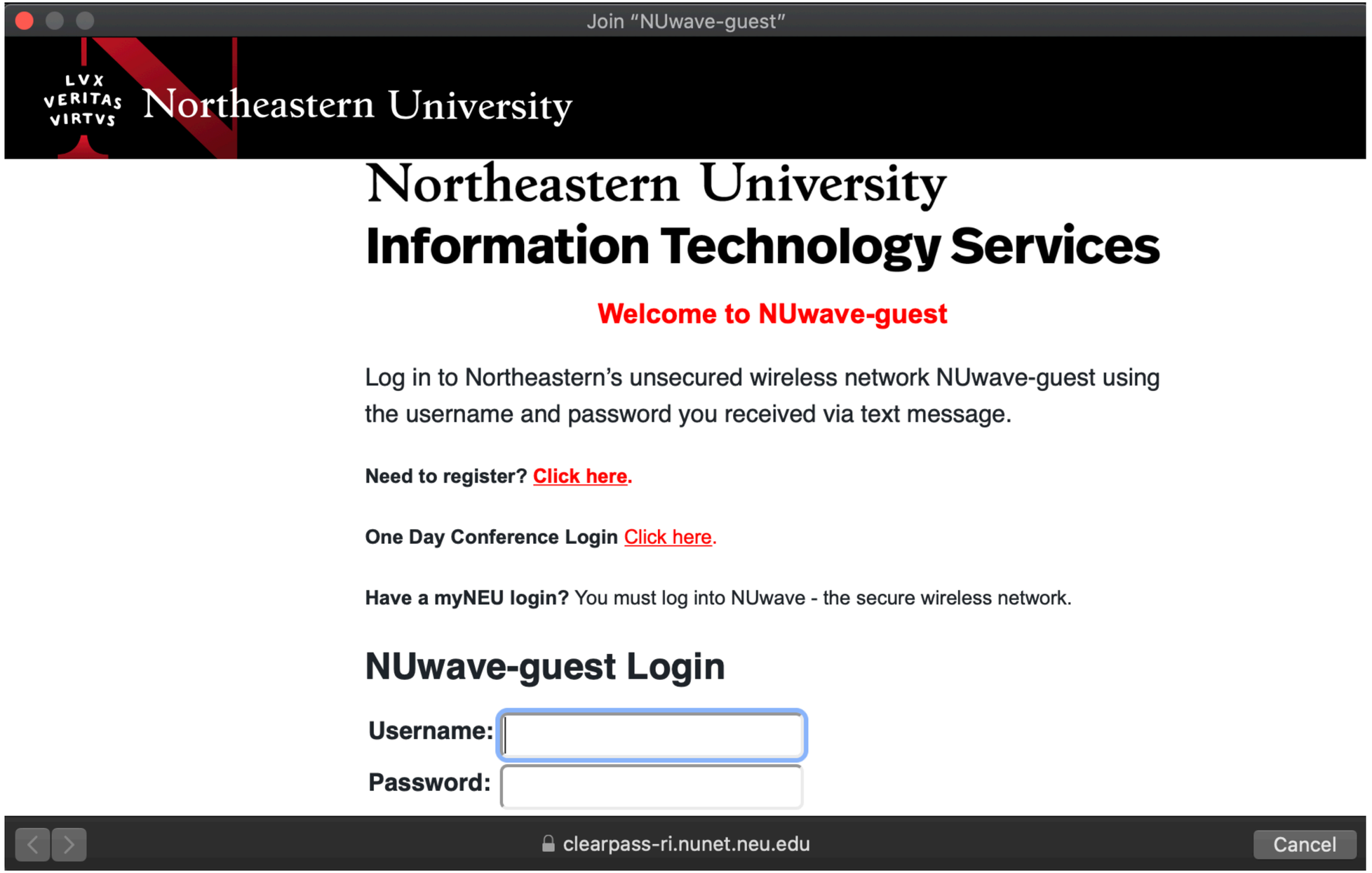
L2

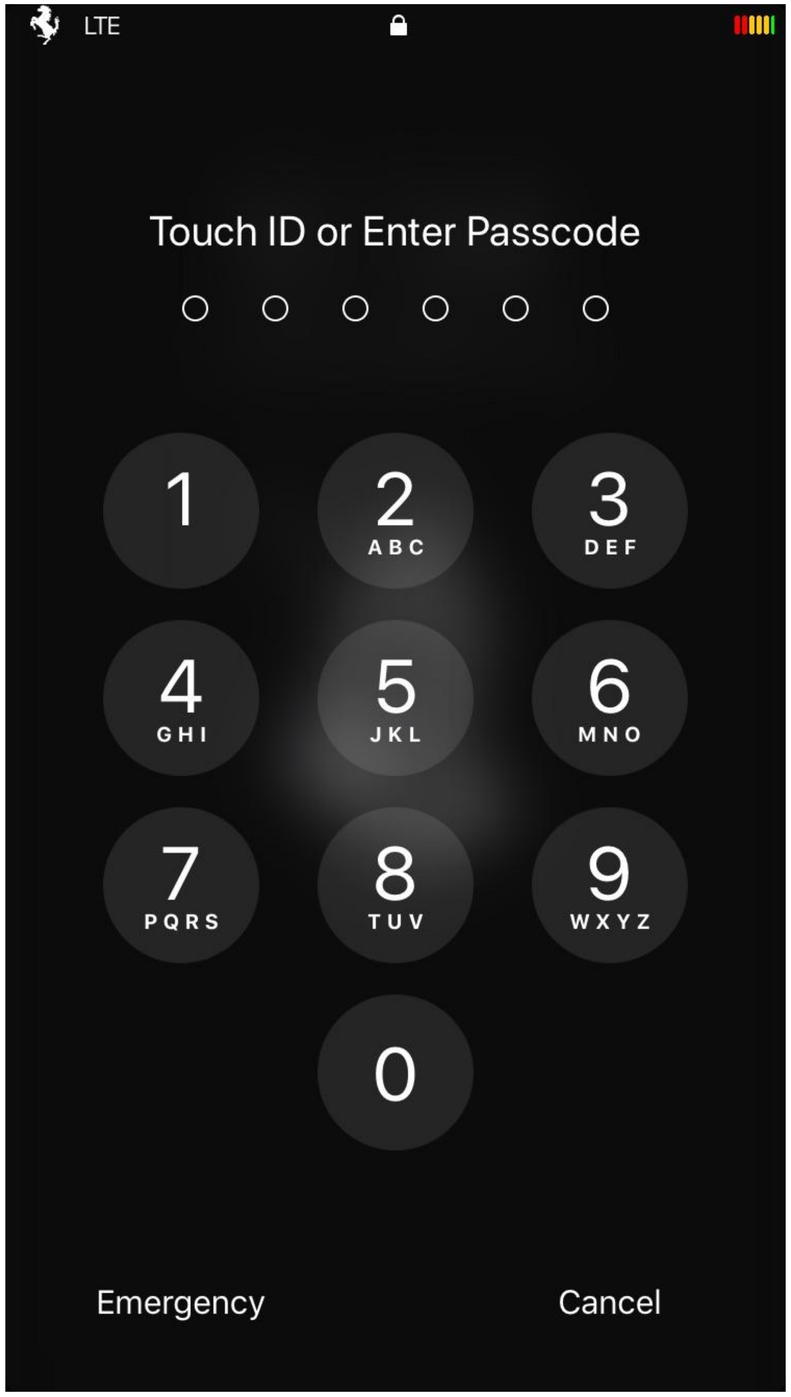
abhi shelat/Ran Cohen

What does it mean to attack a system?

What are our expectations?


abhi
Enter Password
Touch ID or Enter Password
Sleep
Restart
Shut Down


Join “NUwave-guest”
LVX VERITAS VIRTVS
Northeastern University
Northeastern University
Information Technology Services
Welcome to NUwave-guest
Log in to Northeastern’s unsecured wireless network NUwave-guest using the username and password you received via text message.
Need to register? Click here.
One Day Conference Login Click here.
Have a myNEU login? You must log into NUwave - the secure wireless network.
NUwave-guest Login
Username:
Password:
clearpass-ri.nunet.neu.edu
Cancel


LTE
Touch ID or Enter Passcode
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0
Emergency
Cancel

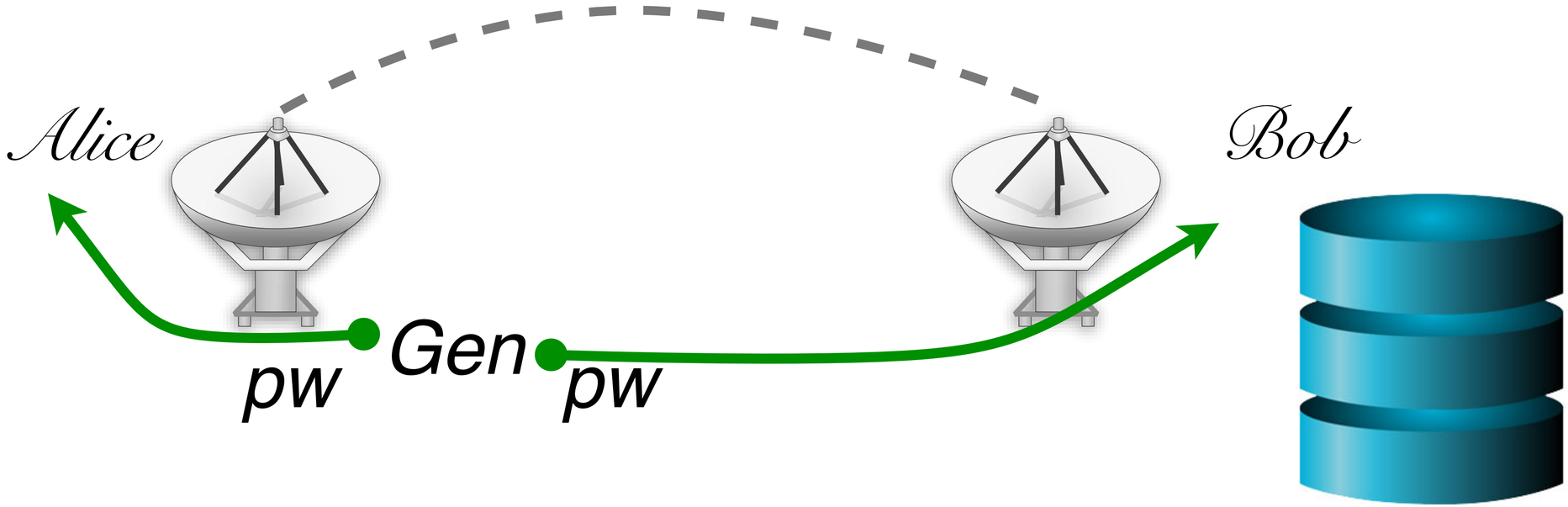
Authentication

- Authentication is the process of verifying an actor's identity
- Critical for security of systems
  - Permissions, capabilities, and access control are all contingent upon knowing the identity of the actor
- Typically parameterized as a username and a secret
  - The secret attempts to limit unauthorized access
- Desirable properties of secrets include being *unforgeable*, *unguessable*, and *revocable*

Passwords

Main problem:

Passwords



Google

Create your Google Account

First name

Last name

Username @gmail.com

You can use letters, numbers & periods

Use my current email address instead

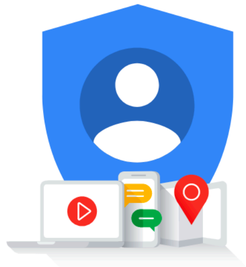
Password

Confirm

Use 8 or more characters with a mix of letters, numbers & symbols

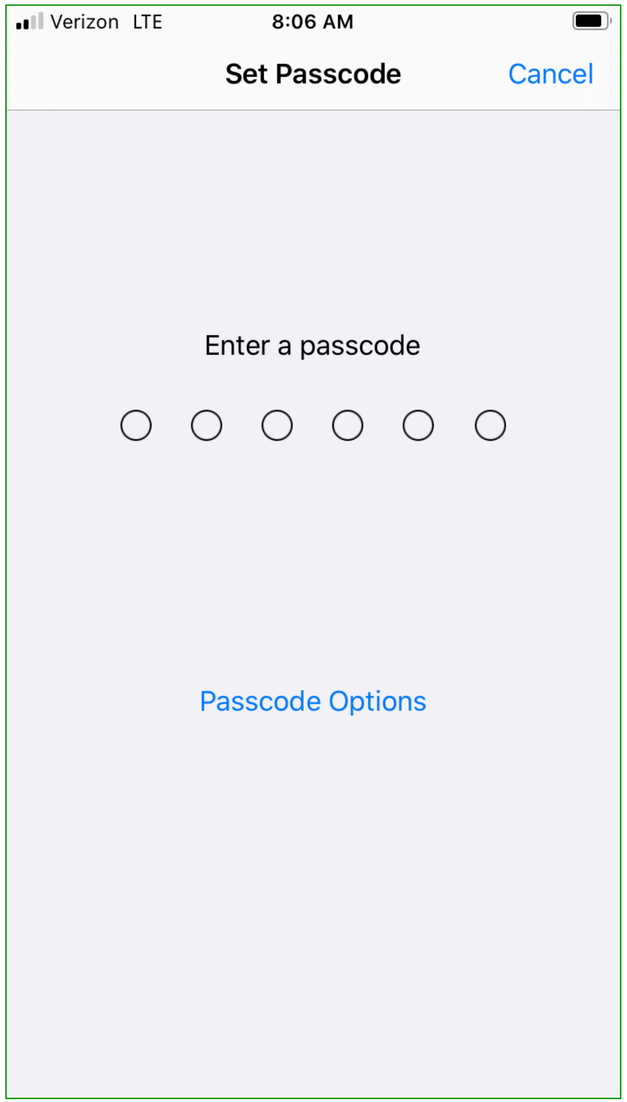
Sign in instead

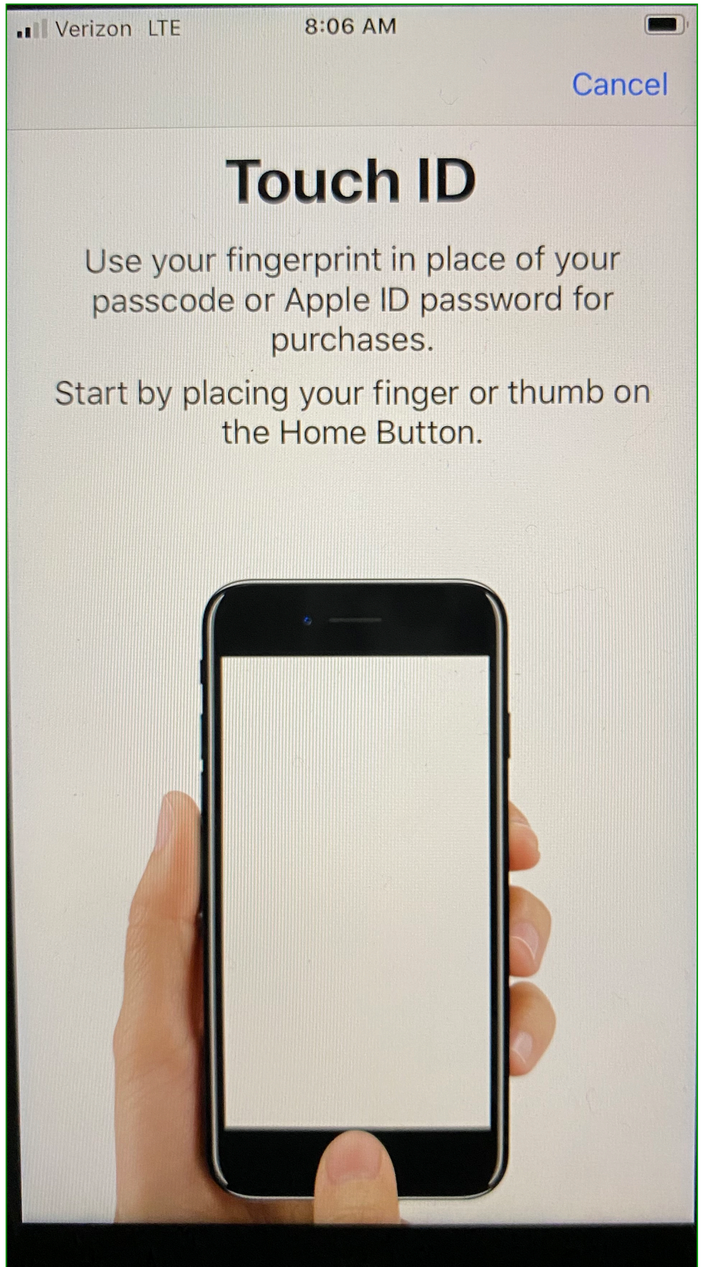
Next

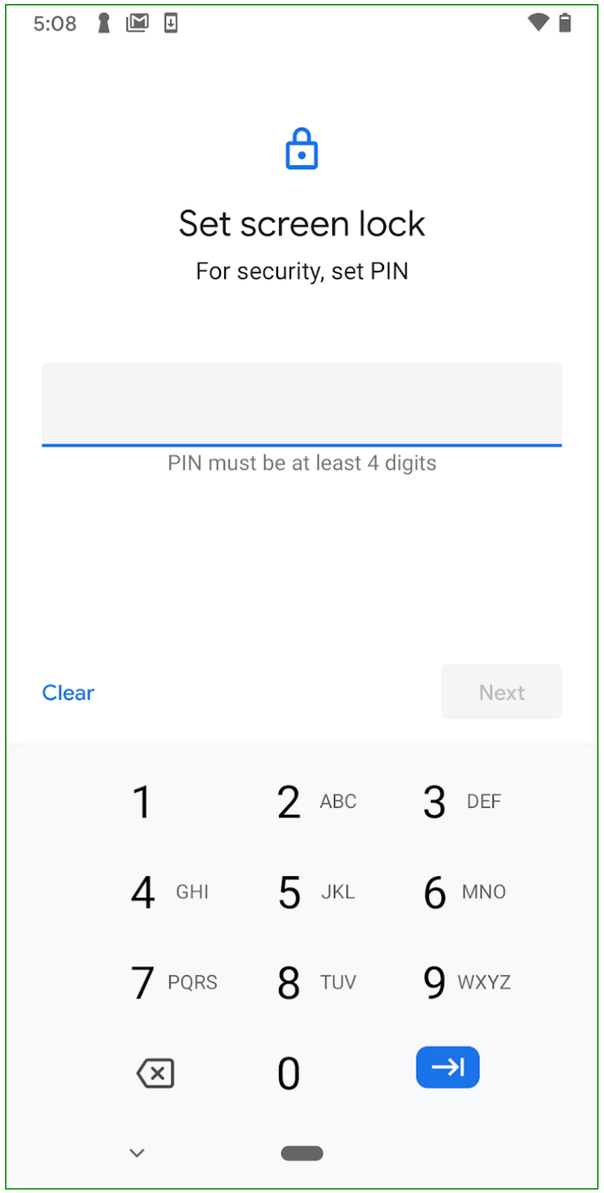


One account. All of Google working for you.

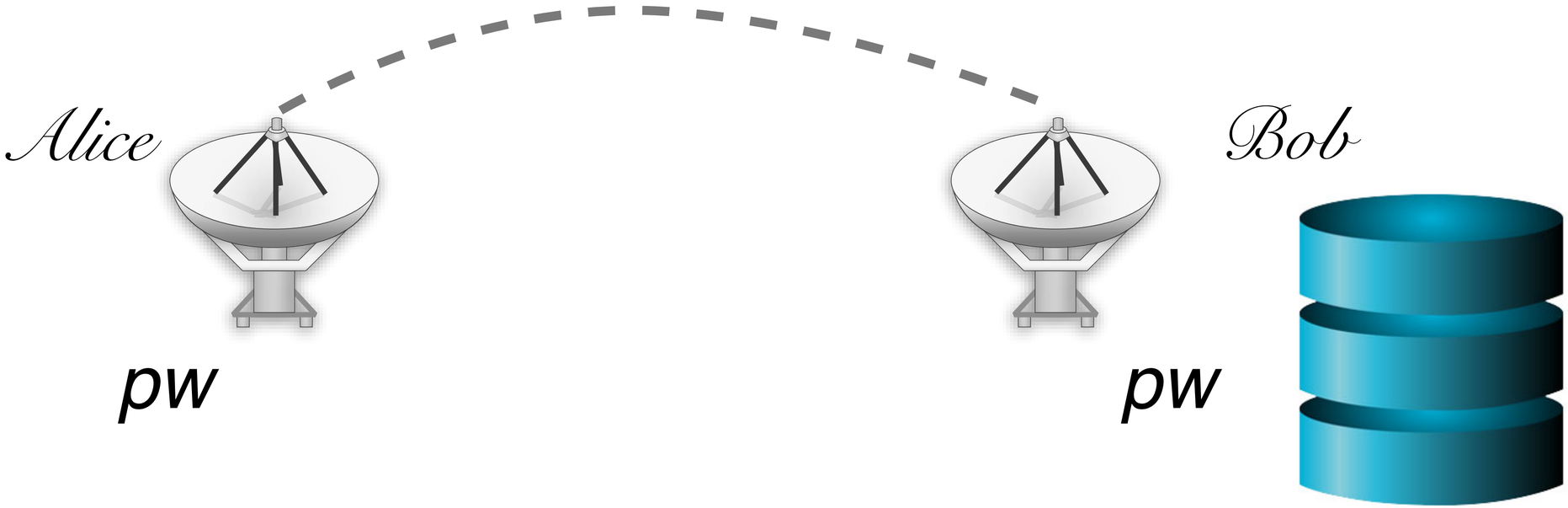
PIN setup


Verizon LTE
8:06 AM
Set Passcode
Cancel
Enter a passcode
Passcode Options

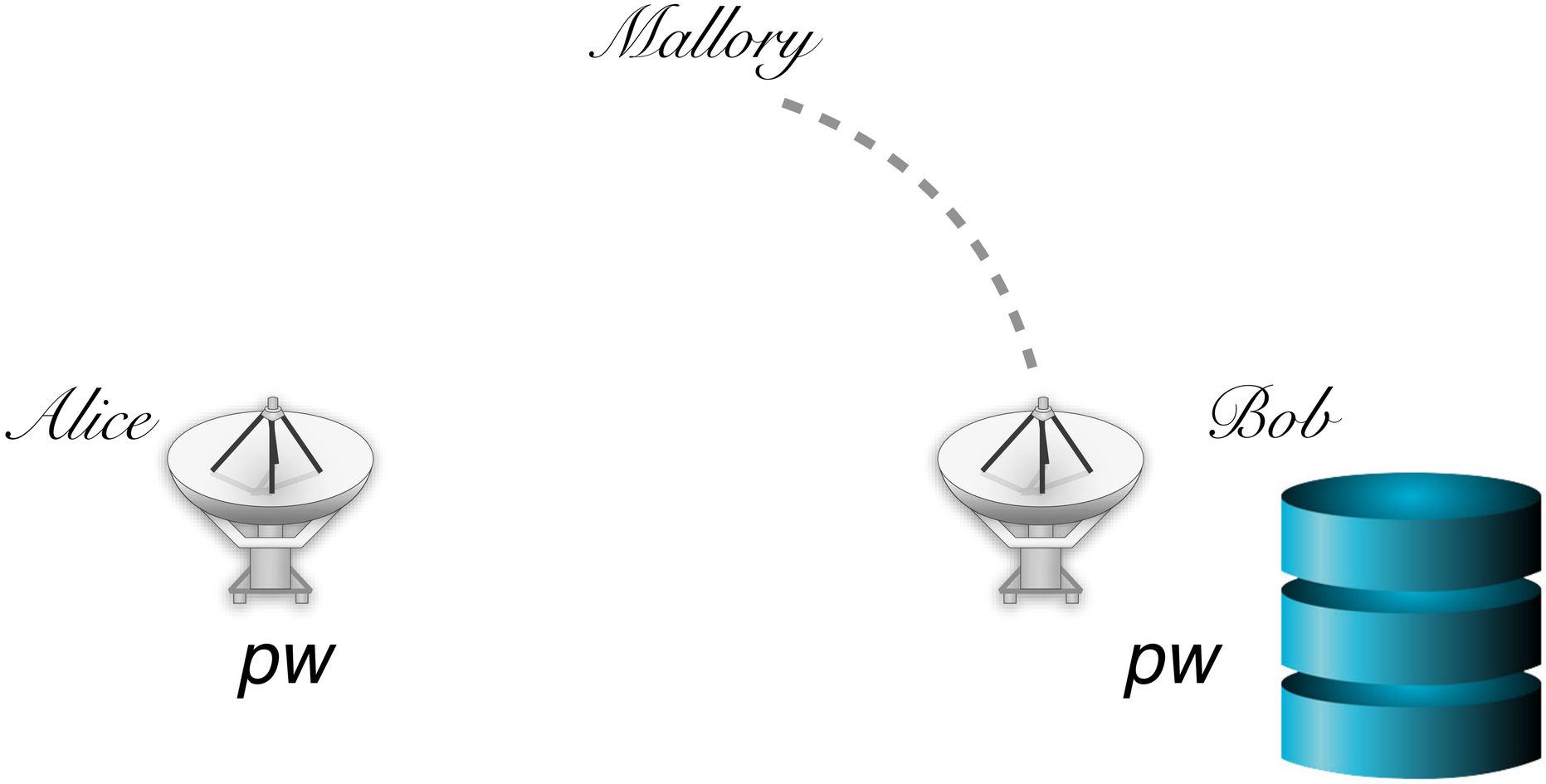

Verizon LTE
8:06 AM
Cancel
Touch ID
Use your fingerprint in place of your passcode or Apple ID password for purchases.
Start by placing your finger or thumb on the Home Button.


5:08
Set screen lock
For security, set PIN
PIN must be at least 4 digits
Clear
Next
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0

Passwords: Alice always succeeds



Passwords: Others do not succeed



Natural authenticators

Operating Systems

R. Stockton Gaines
Editor

Password Security: A Case History

Robert Morris and Ken Thompson
Bell Laboratories

This paper describes the history of the design of the password security scheme on a remotely accessed time-sharing system. The present design was the result of countering observed attempts to penetrate the system. The result is a compromise between extreme security and ease of use.

Key Words and Phrases: operating systems, passwords, computer security

CR Categories: 2.41, 4.35

Communications of the ACM

November 1979
Volume 22
Number 11

"The UNIX system was first implemented with a password file that contained the actual passwords of all the users, and for that reason the password file had to be heavily protected against being either read or written. Although historically, this had been the technique used for remote-access systems, it was completely unsatisfactory for several reasons."

Checking Passwords

- System must validate passwords provided by users
- Thus, passwords must be *stored* somewhere
- Basic storage: plain text

password.txt	
Alice	p4ssw0rd
Eve	i heart doggies
Charlie	93Gd9#jv*0x3N
bob	security

Attacks against the Password Model

Mallory

Bob

{username: pwd}

password.txt	
Alice	p4ssw0rd
Eve	i heart doggies
Charlie	93Gd9#jv*0x3N
bob	security

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?

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**

RockYou Hack: From Bad To Worse

Nik Cubrilovic

@nikcub / 2:42 am EST • December 15, 2009

Comment

Earlier today news spread that social application site RockYou had suffered a data breached that resulted in the exposure of over 32 Million user accounts. To compound the severity of the security breach, it was found that **RockYou** are storing all user account data in plain text in their database, exposing all that information to attackers. RockYou have yet to inform users of the breach, and their blog is eerily silent – but the details of the security breach are going from bad to worse.

Data UserAccount [32603388]

==========

1|jennaplanerunner@hotmail.com|mek*****|myspace|0|bebo.com

2|phdlance@gmail.com|mek*****|myspace|1|

3|jennaplanerunner@gmail.com|mek*****|myspace|0|

5|teamsmackage@gmail.com|pro*****|myspace|1|

6|ayul@email.com|kha*****|myspace|1|tagged.com

7|guera_n_negro@yahoo.com|emi*****|myspace|0|

8|beyootifulgirl@aol.com|hol*****|myspace|1|

9|keh2oo8@yahoo.com|cai*****|myspace|1|

10|mawabiru@yahoo.com|pur*****|myspace|1|

11|jodygold@gmail.com|att*****|myspace|1|

12|aryan_dedboy@yahoo.com|iri*****|myspace|0|

13|moe_joe_25@yahoo.com|725*****|myspace|1|

14|xxxnothingbutme@aol.com|1th*****|myspace|0|

15|meandcj069@yahoo.com|too*****|myspace|0|

16|stacey_chim@hotmail.com|cxn*****|myspace|1|

17|barne1en@cmich.edu|ilo*****|myspace|1|

18|reo154@hotmail.com|ecu*****|myspace|1|

19|natapappaslie@yahoo.com|tor*****|myspace|0|

20|ypiogirl@aol.com|tob*****|myspace|1|

21|brittanyleigh864@hotmail.com|bet*****|myspace|1|myspace.com

22|topenga68@aol.com|che*****|myspace|0|

23|marie603412@yahoo.com|cat*****|myspace|0|

24|mellowchick41@aol.com|chu*****|myspace|0|

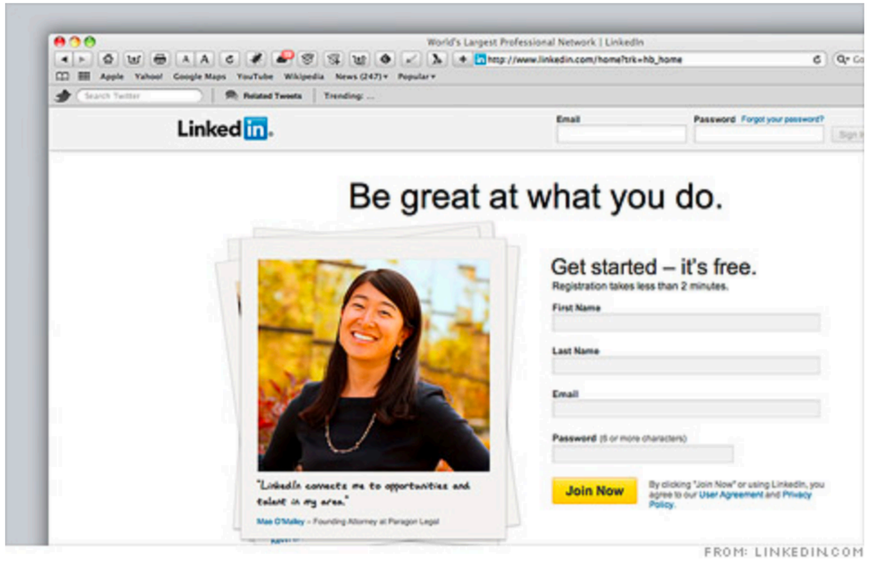
Pwd breaches

CNN Money

THE CYBERCRIME ECONOMY

More than 6 million LinkedIn passwords stolen

By David Goldman @CNNMoneyTech June 7, 2012: 9:34 AM ET



Researchers say a stash of what appear to be LinkedIn passwords were protected by a weak security scheme.

NEW YORK (CNNMoney) -- Russian hackers released a giant list of passwords this week, and on Wednesday security researchers identified their likely source: business social networking site LinkedIn.

Operating Systems
R. Stockton Gaines Editor

Password Security: A Case History

Robert Morris and Ken Thompson
Bell Laboratories

This paper describes the history of the design of the password security scheme on a remotely accessed time-sharing system. The present design was the result of countering observed attempts to penetrate the system. The result is a compromise between extreme security and ease of use.

Key Words and Phrases: operating systems, passwords, computer security
CR Categories: 2.41, 4.35

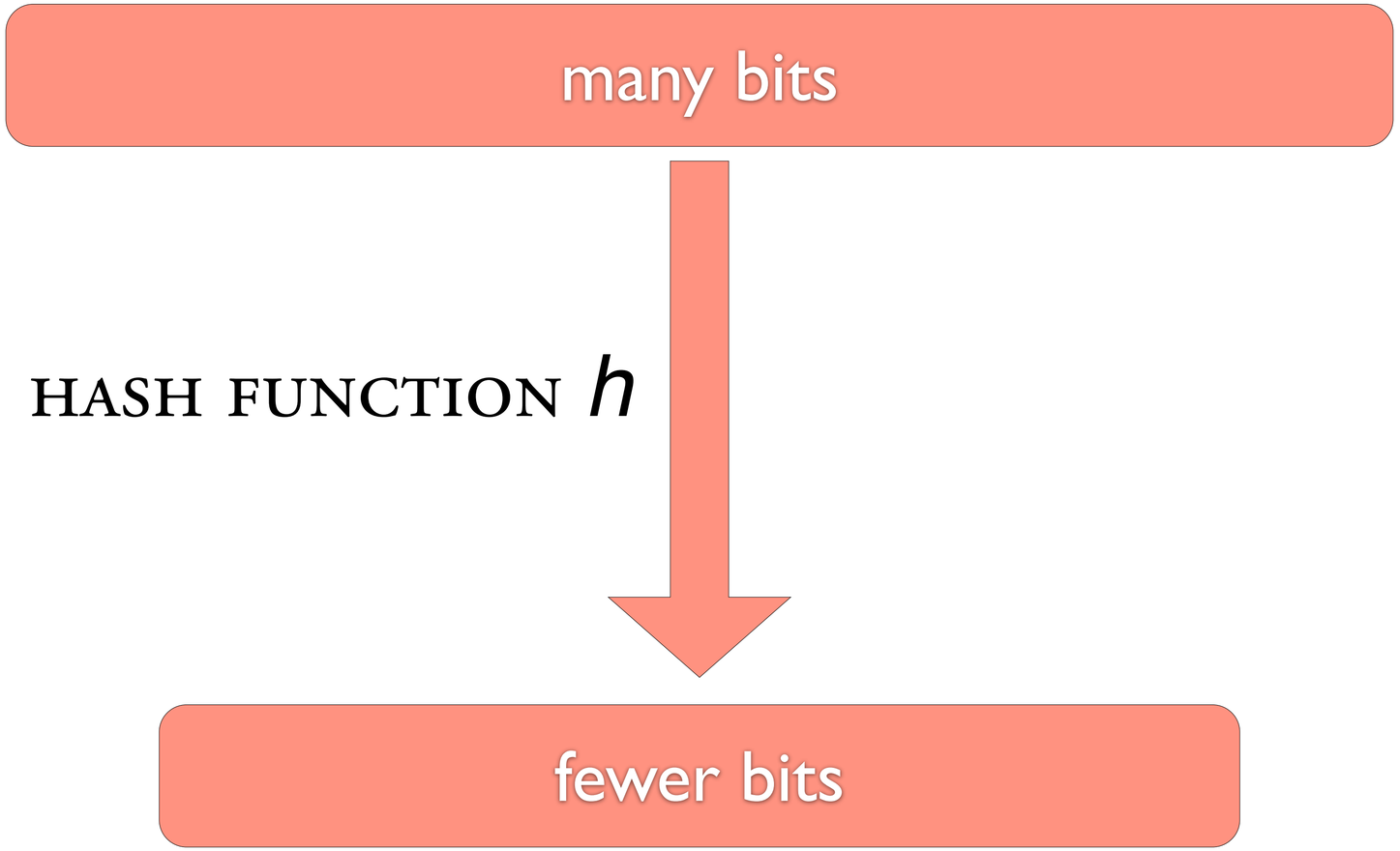
Communications of the ACM
November 1979
Volume 22
Number 11

"The obvious solution is to arrange that the passwords not appear in the system at all, and it is not difficult to decide that this can be done by encrypting each user's password, putting only the encrypted form in the pass- word file, and throwing away his original password (the one that he typed in). When the user later tries to log in to the system, the password that he types is encrypted and compared with the encrypted version in the password file. If the two match, his login attempt is accepted."

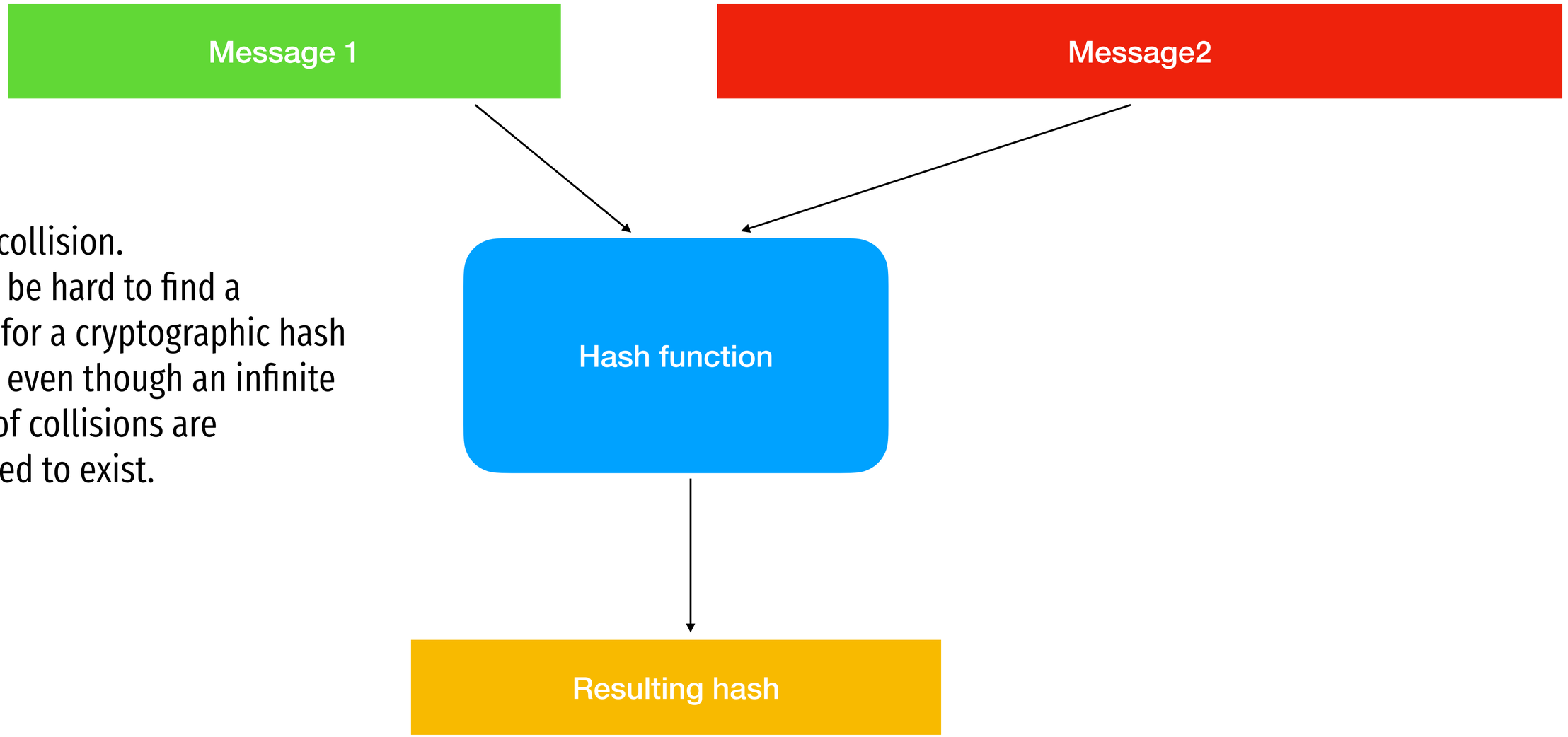
Hashed Passwords

- Key idea: store “hashed” versions of passwords
  - Use one-way cryptographic hash functions
  - Examples: MD5, SHA1, SHA256, SHA512, bcrypt, PBKDF2, scrypt

Goal of a hash function



Goal of a hash function: Collision resistance



This is a collision.
It should be hard to find a collision for a cryptographic hash function, even though an infinite number of collisions are guaranteed to exist.

MD5 is a broken hash function

```
abhi18: abhi$ md5 -s security
MD5 ("security") = e91e6348157868de9dd8b25c81aebfb9

abhi18: abhi$ md5 -s Security
MD5 ("Security") = 2fae32629d4ef4fc6341f1751b405e45


abhi18: abhi$ md5 -s Security1
MD5 ("Security1") = 8d01bda744a7a6392d3393e0ece561e8
```

```
abhi18: abhi$ echo -n "security" | shasum
8eec7bc461808e0b8a28783d0bec1a3a22eb0821  -


abhi18: abhi$ echo -n "security" | shasum -a 256
5d2d3ceb7abe552344276d47d36a8175b7aeb250a9bf0bf00e850cd23ecf2e43  -
```

Hashed Passwords

- Key idea: store “hashed” versions of passwords
  - Use one-way cryptographic hash functions
  - Examples: MD5, SHA1, SHA256, SHA512, bcrypt, PBKDF2, scrypt
- 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 (hopefully)
    - Example: 2^{21} tries for md5

Note:MD5 is broken

Hashed Password Example

User: Charlie

hashed_password.txt	
charlie	2a9d119df47ff993b662a8ef36f9ea20
greta	23eb06699da16a3ee5003e5f4636e79f
alice	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Note:MD5 is broken

Hashed Password Example

User: Charlie

MD5('p4ssw0rd') =
2a9d119df47ff993b662a8ef36f9ea20

hashed_passwo txt

charlie	2a9d119df47ff993b662a8ef36f9ea20
greta	23eb06699da16a3ee5003e5f4636e79f
alice	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Note:MD5 is broken

Hashed Password Example

User: Charlie

MD5(‘p4ssw0rd’) =
2a9d119df47ff993b662a8ef36f9ea20

hashed_password.txt	
charlie	2a9d119df47ff993b662a8ef36f9ea20
greta	23eb06699da16a3ee5003e5f4636e79f
alice	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Note:MD5 is broken

Hashed Password Example

User: Charlie

MD5(‘p4ssw0rd’) =
2a9d119df47ff993b662a8ef36f9ea20

MD5(‘2a9d119df47ff993b662a8ef36f9ea20’)
= b35596ed3f0d5134739292faa04f7ca3

hashed_password.txt	
charlie	2a9d119df47ff993b662a8ef36f9ea20
greta	23eb06699da16a3ee5003e5f4636e79f
alice	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Note:MD5 is broken

Hashed Password Example

User: Charlie

MD5(‘p4ssw0rd’) =
2a9d119df47ff993b662a8ef36f9ea20

MD5(‘2a9d119df47ff993b662a8ef36f9ea20’)
= b35596ed3f0d5134739292faa04f7ca3

hashed_passw[illegible]xt	
charlie	2a9d119df47ff993b662a8ef36f9ea20
greta	23eb06699da16a3ee5003e5f4636e79f
alice	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Attacking Password Hashes

- Recall: cryptographic hashes are collision resistant
  - Locating A’ such that hash(A) = hash(A’) takes a long time (hopefully)
- Are hashed password secure from cracking?

Attacking Password Hashes

- Recall: cryptographic hashes are collision resistant
  - Locating A' such that hash(A) = hash(A') takes a long time (hopefully)
- Are hashed password secure from cracking?
  - **No!**
- Problem: users choose poor passwords
  - Most common passwords: 123456, password
  - Username: cbw, Password: cbw
- Weak passwords enable dictionary attacks

The authors have conducted experiments to try to determine typical users' habits in the choice of passwords when no constraint is put on their choice. The results were disappointing, except to the bad guy. In a collection of 3,289 passwords gathered from many users over a long period of time,

15 were a single ASCII character;
72 were strings of two ASCII characters;
464 were strings of three ASCII characters;
477 were strings of four alphamerics;
706 were five letters, all upper-case or all lower-case;
605 were six letters, all lower-case.

An additional 492 passwords appeared in various available dictionaries, name lists, and the like. A total of 2,831 or 86 percent of this sample of passwords fell into one of these classes.

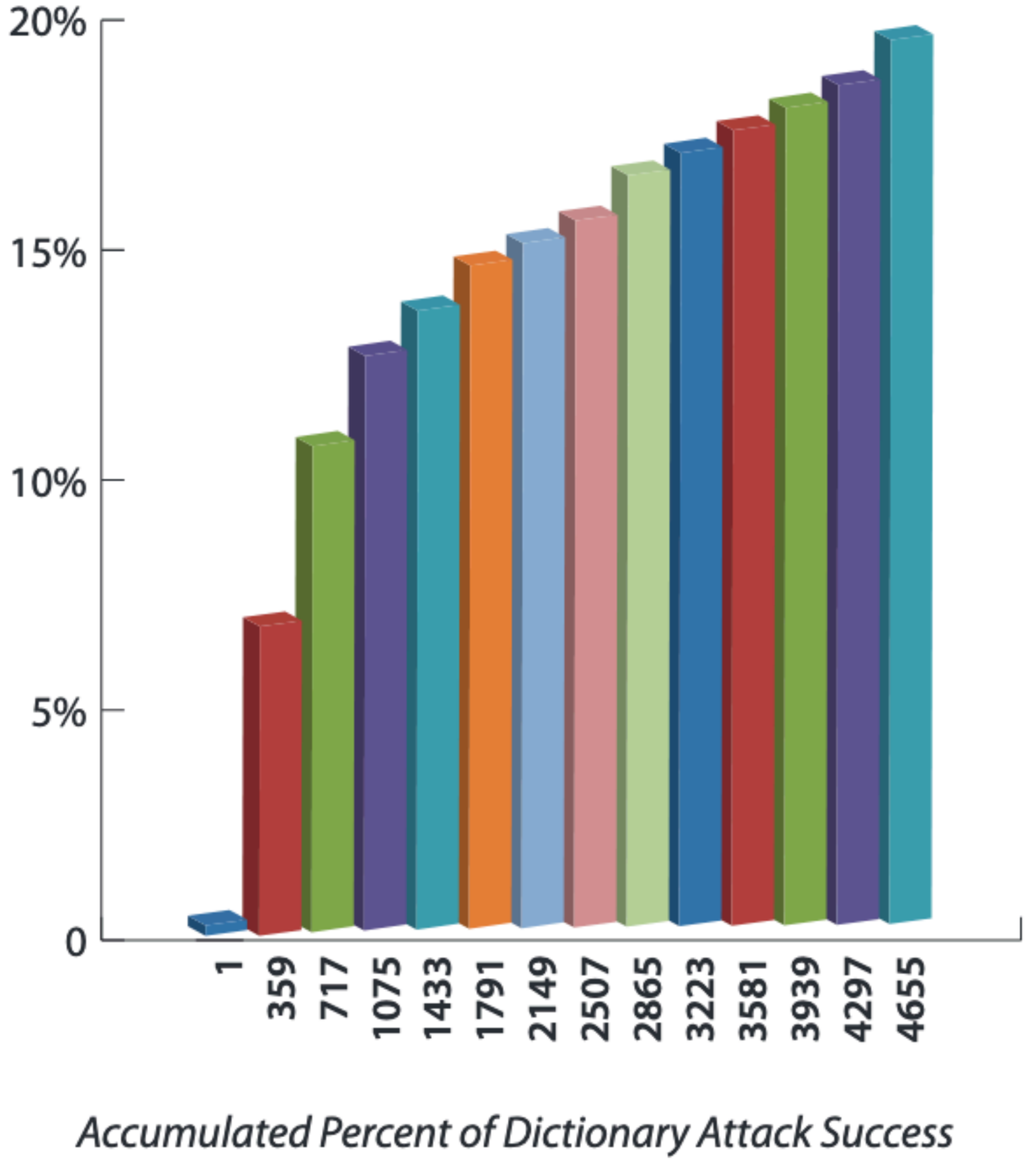
From Rockyou breach

Rank	Password	Number of Users with Password (Absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Rank	Password	Number of Users with Password (Absolute)
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856

Password Popularity—Top 20

https://www.imperva.com/docs/gated/WP_Consumer_Password_Worst_Practices.pdf


20%
15%
10%
5%
0
1
359
717
1075
1433
1791
2149
2507
2865
3223
3581
3939
4297
4655
Accumulated Percent of Dictionary Attack Success

Most Common Passwords

Rank	2013	2014
1	123456	123456
2	password	password
3	12345678	12345
4	qwerty	12345678
5	abc123	qwerty
6	123456789	123456789
7	111111	1234
8	1234567	baseball
9	iloveyou	dragon
10	adobe123	football

2012: 6.5 million hashes leaked onto Internet 90% cracked in 2 weeks

2016: 177.5 million more hashes leaked 98% cracked in 1 week

2012 Linkedin Breach had 117 Million Emails and Passwords Stolen, Not 6.5M

May 18, 2016

Long time users of Linkedin users may very well need to change their passwords once more

Related Posts

Web Skimming Attack on Blue Bear Affects School Admin

by Paul Ducklin

One month ago today, we wrote about Adobe's giant data breach.

As far as anyone knew, including Adobe, it affected about 3,000,000 customer records, which made it sound pretty bad right from the start.

But worse was to come, as recent updates to the story bumped the number of affected customers to a whopping 38,000,000.

We took Adobe to task for a lack of clarity in its breach notification.

OUR COMPLAINT

One of our complaints was that Adobe said that it had lost *encrypted* passwords, when we thought the company ought to have said that it had lost

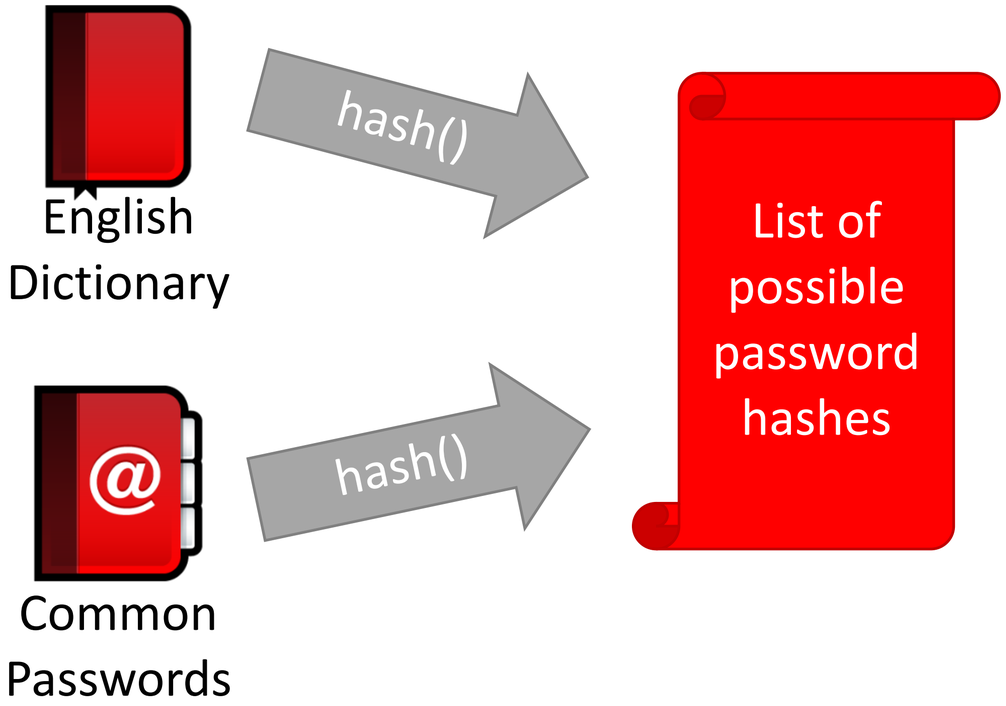
Adobe password data		Password hint	
110edf2294fb8bf4	->	numbers 123456	❶ 123456
110edf2294fb8bf4	->	==123456	
110edf2294fb8bf4	->	c'est "123456"	
8fda7e1f0b56593f e2a311ba09ab4707	->	numbers	❷ 12345678
8fda7e1f0b56593f e2a311ba09ab4707	->	1-8	
8fda7e1f0b56593f e2a311ba09ab4707	->	8digit	
2fca9b003de39778 e2a311ba09ab4707	->	the password is password	❸ password
2fca9b003de39778 e2a311ba09ab4707	->	password	
2fca9b003de39778 e2a311ba09ab4707	->	rhymes with assword	
e5d8efed9088db0b	->	q w e r t y	❹ qwerty
e5d8efed9088db0b	->	ytrewq tagurpidi	
e5d8efed9088db0b	->	6 long qwert	
ecba98cca55eabc2	->	sixxone	❺ 111111
ecba98cca55eabc2	->	1*6	
ecba98cca55eabc2	->	sixones	

Dictionary Attacks

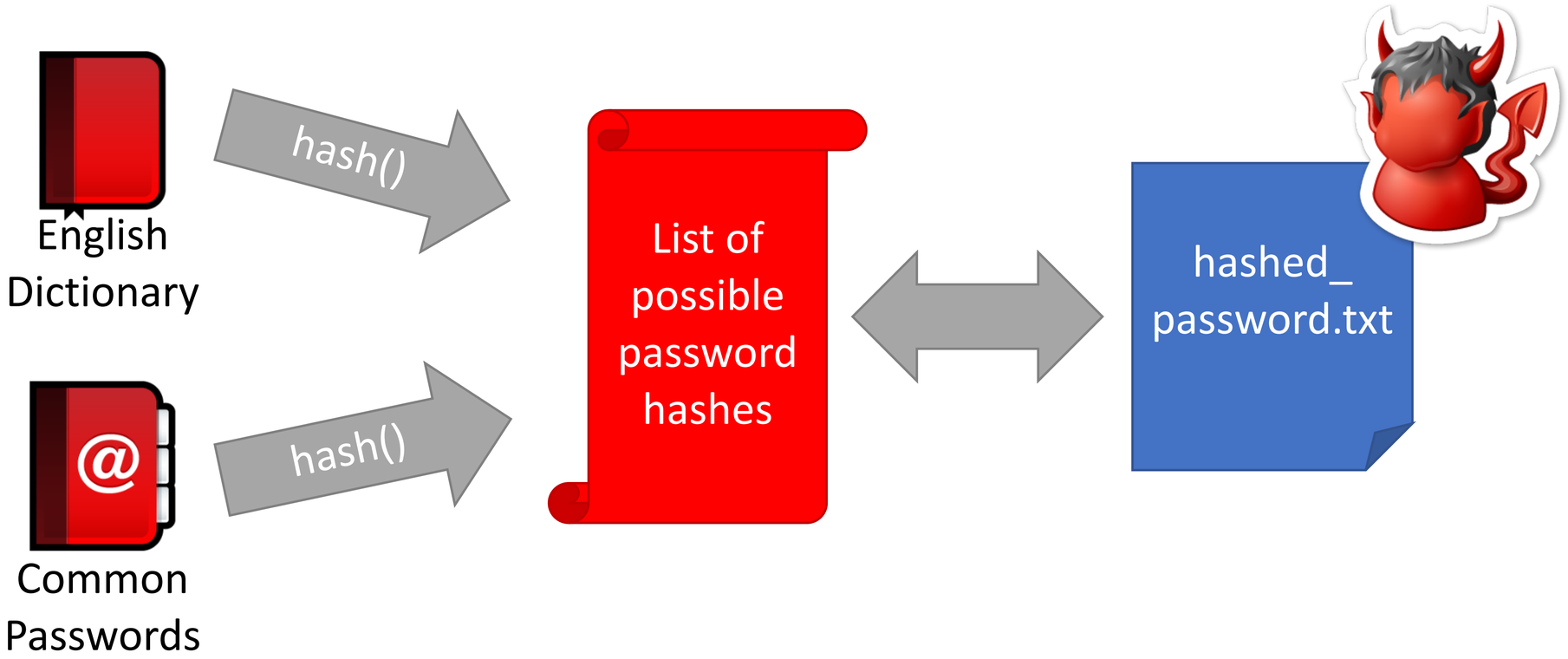
English Dictionary

Common Passwords

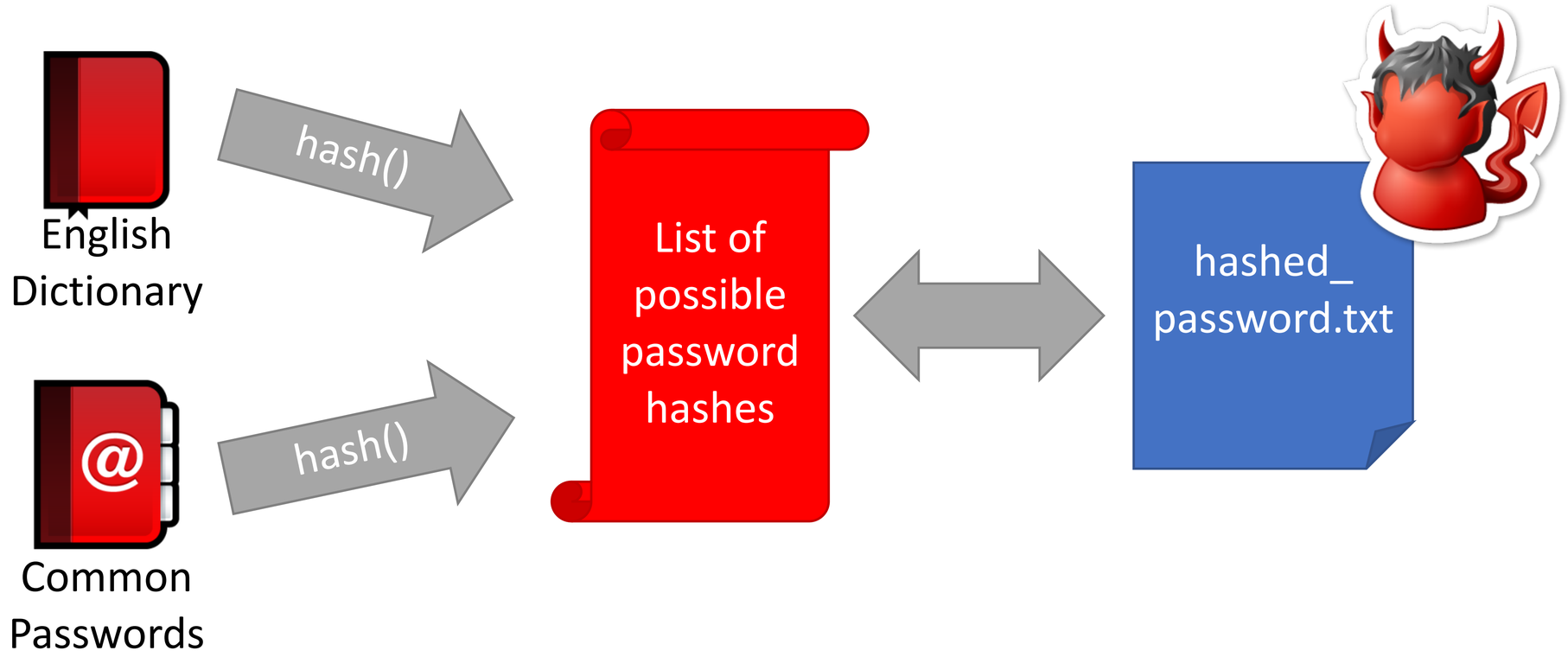
Dictionary Attacks



Dictionary Attacks

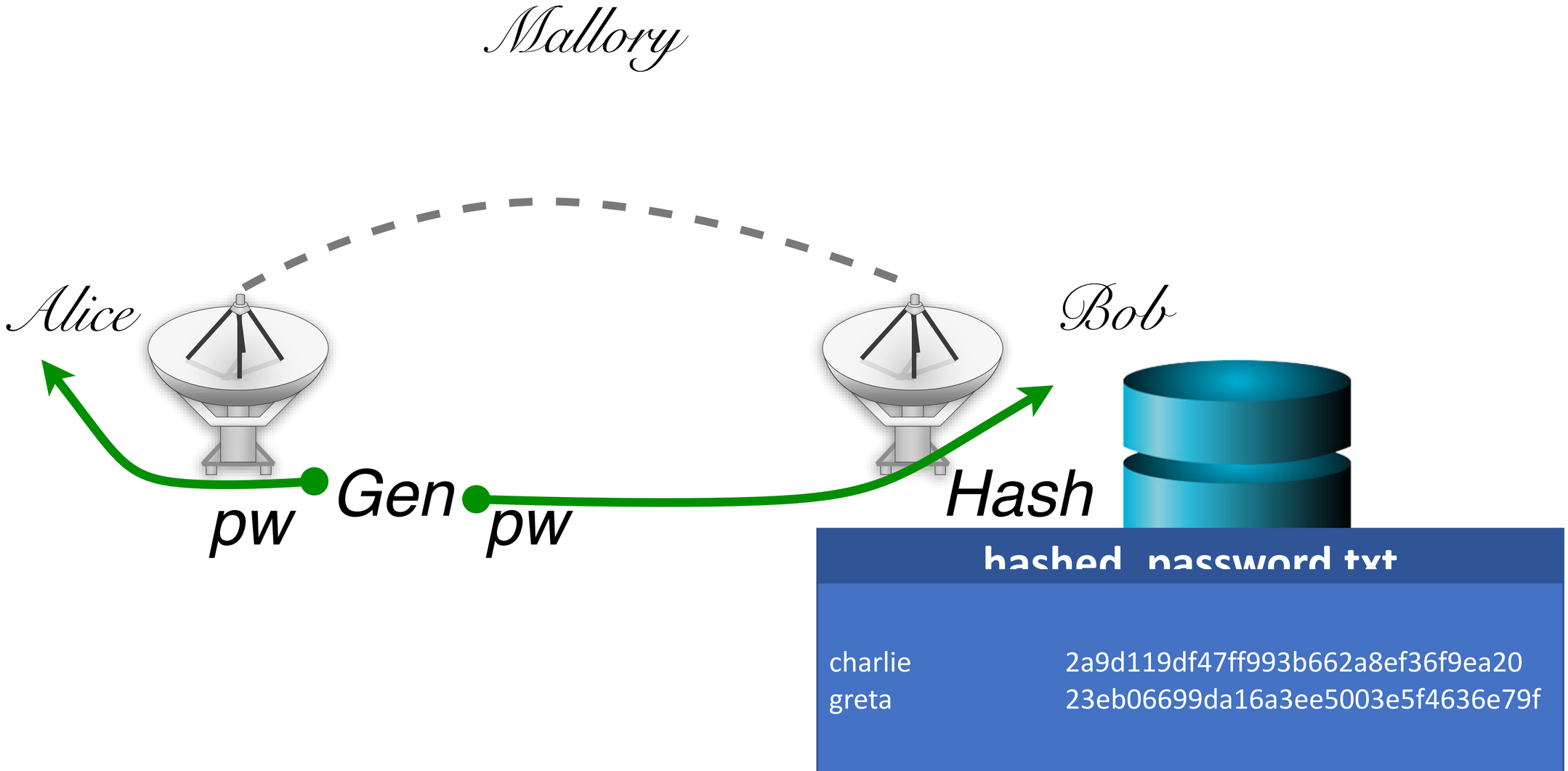


Dictionary Attacks

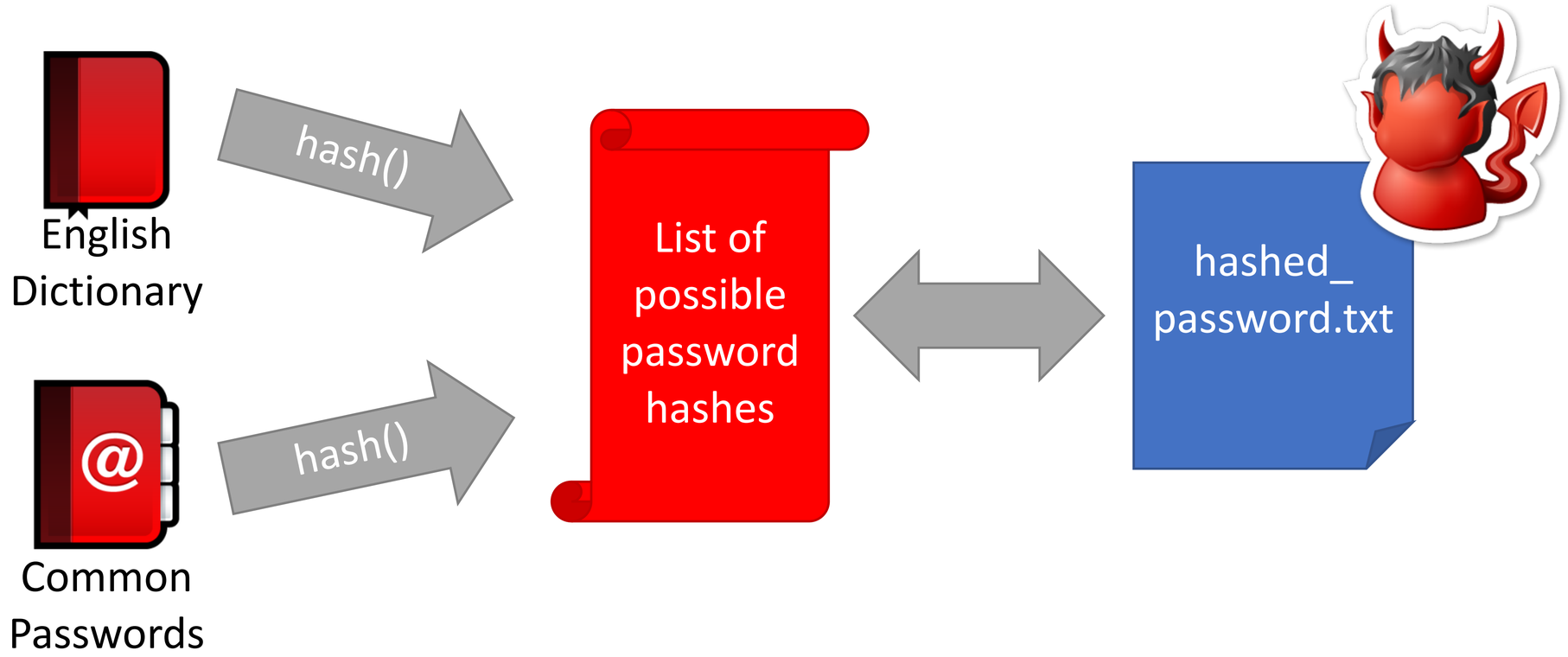


- Common for 60-70% of hashed passwords to be cracked in <24 hours

Attack 1


Mallory
Alice
Bob
pw
Gen
pw
Hash
hashed_password.txt
charlie 2a9d119df47ff993b662a8ef36f9ea20
greta 23eb06699da16a3ee5003e5f4636e79f

Dictionary Attacks



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

Brute force attack estimates

How big is the alphabet from which pwd are chosen?

Brute force attack estimates

How big is the alphabet from which pwd are chosen?

95 symbols

How long is a password?

Size of password domain:

Brute force attack estimates

Size of password domain: 95^8 `6,634,204,312,890,625`

Built by Edward Betts. Comments welcome: edward@4angle.com

Last updated: 07 February 2020.

3.5" internal drives

Price per TB	Price	Size	Drive
$18.75	$149.99	8TB	Seagate BarraCuda ST8000DM004 8TB 5400 RPM 256MB Cache SATA 6.0Gb/s 3.5" Internal Hard Drive Bare Drive
$22.25	$88.99	4TB	WD Blue 4TB Desktop Hard Disk Drive - 5400 RPM SATA 6Gb/s 64MB Cache 3.5 Inch - WD40EZRZ
$22.50	$89.99	4TB	Seagate BarraCuda ST4000DM004 4TB 5400 RPM 256MB Cache SATA 6.0Gb/s 3.5" Hard Drives Bare Drive - OEM
$22.52	$135.12	6TB	Seagate BarraCuda ST6000DM003 6TB 5400 RPM 256MB Cache SATA 6.0Gb/s 3.5" Internal Hard Drive Bare Drive
$23.33	$139.99	6TB	WD Blue 6TB Desktop Hard Disk Drive - 5400 RPM SATA 6Gb/s 256MB Cache 3.5 Inch - WD60EZAZ
$23.92	$334.88	14TB	Seagate Exos X16 ST14000NM001G 14TB 7200 RPM 256MB Cache SATA 6.0Gb/s 3.5" Hard Drives, 512E/4KN
$24.00	$71.99	3TB	WD Blue 3TB Desktop Hard Disk Drive - 5400 RPM SATA 6Gb/s 64MB Cache 3.5 Inch - WD30EZRZ
$24.06	$384.99	16TB	Seagate Exos 16TB Enterprise HDD X16 SATA 6Gb/s 512e/4Kn 7200 RPM 256MB Cache 3.5" Internal Hard Drive ST16000NM001G
$24.42	$292.99	12TB	Seagate 12TB HDD Exos X14 7200 RPM 512e/4Kn SATA 6Gb/s 256MB Cache 3.5-Inch Enterprise Hard Drive (ST12000NM0008)
$24.66	$73.99	3TB	Seagate BarraCuda ST3000DM007 3TB 5400 RPM 256MB Cache SATA 6.0Gb/s 3.5" Hard Drives
$25.00	$49.99	2TB	Seagate BarraCuda ST2000DM008 2TB 7200 RPM 256MB Cache SATA 6.0Gb/s 3.5" Hard Drive Bare Drive
$25.00	$99.99	4TB	Seagate IronWolf 4TB NAS Hard Drive 5900 RPM 64MB Cache SATA 6.0Gb/s 3.5" Internal Hard Drive ST4000VN008
$25.40	$253.99	10TB	Seagate Exos Enterprise Capacity 3.5" HDD 10TB (Helium) 7200 RPM SATA 6Gb/s 256MB Cache Hyperscale 512e Internal Hard Drive ST10000NM0016
$25.67	$307.99	12TB	Seagate Exos Enterprise Capacity ST12000NM0007 12TB 7200 RPM SATA 6Gb/s 256MB Enterprise Hard Drive (Helium & 3.5 inch)
$25.75	$102.99	4TB	WD Purple 4TB Surveillance Hard Disk Drive - 5400 RPM Class SATA 6Gb/s 64MB Cache 3.5 Inch WD40PURZ
$26.00	$103.99	4TB	Seagate SkyHawk 4TB Surveillance Hard Drive 64MB Cache SATA 6.0Gb/s 3.5" Internal Hard Drive ST4000VX007

more

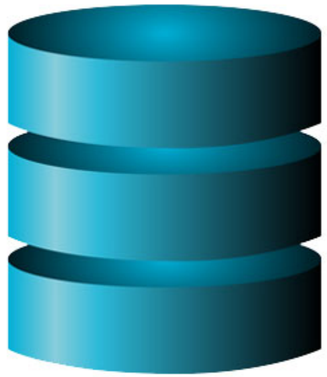
Attack 2: brute force attack

Mallory

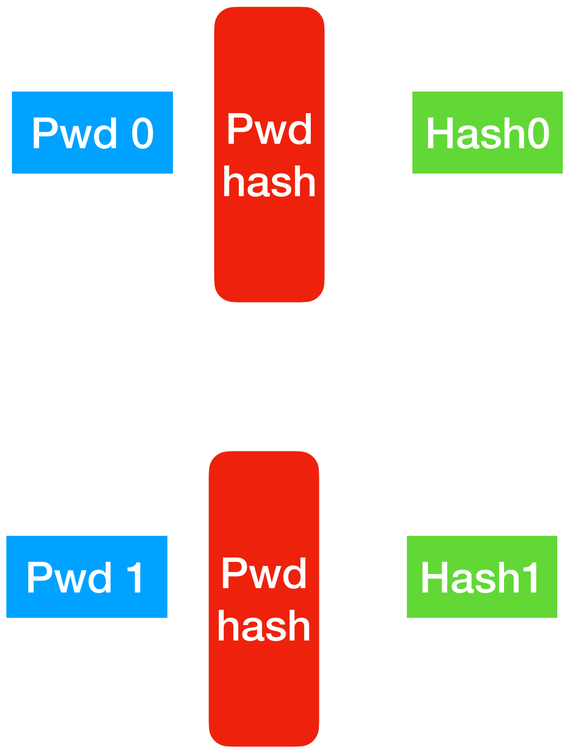
1. Buy storage system

2.

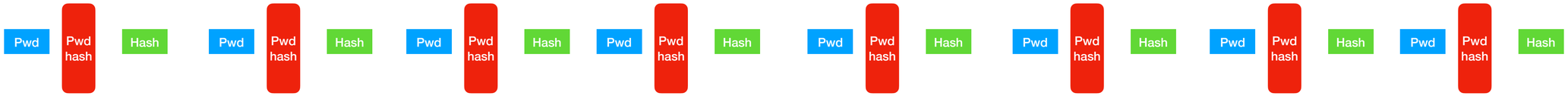
Bob



Classic Time-memory tradeoff



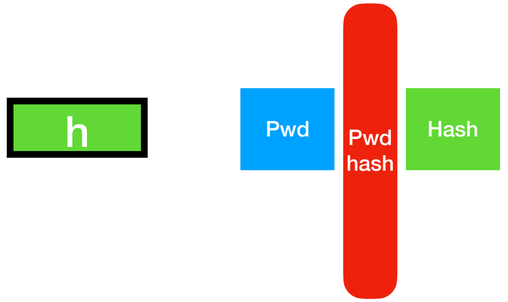
Classic Time-memory tradeoff

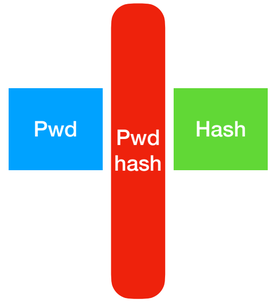


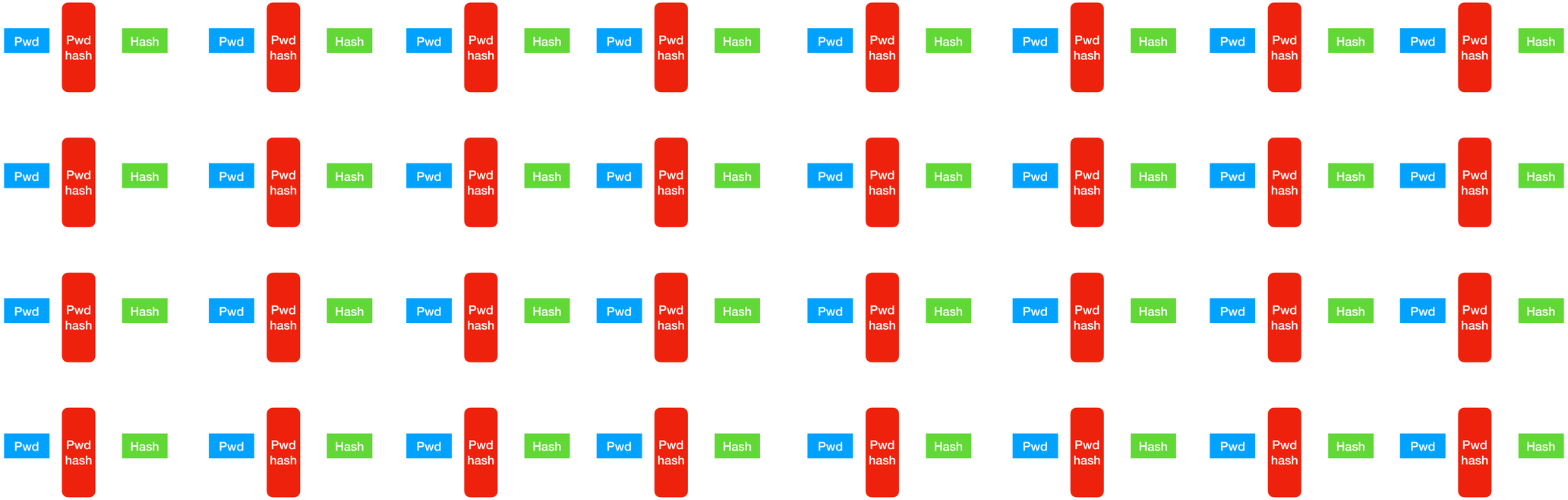
Given a hash [h] that you want to invert, you can:

h

Classic Time-memory tradeoff







SHA1 Rainbow Tables

Table ID	Charset	Plaintext Length	Key Space	Success Rate	Table Size	Files	Performance
sha1_ascii-32-95#1-7	ascii-32-95	1 to 7	70,576,641,626,495	99.9 %	52 GB 64 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_ascii-32-95#1-8	ascii-32-95	1 to 8	6,704,780,954,517,120	96.8 %	460 GB 576 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_mixalpha-numeric#1-8	mixalpha-numeric	1 to 8	221,919,451,578,090	99.9 %	127 GB 160 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_mixalpha-numeric#1-9	mixalpha-numeric	1 to 9	13,759,005,997,841,642	96.8 %	690 GB 864 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_loweralpha-numeric#1-9	loweralpha-numeric	1 to 9	104,461,669,716,084	99.9 %	65 GB 80 GB	Perfect Non-perfect	Perfect Non-perfect
sha1_loweralpha-numeric#1-10	loweralpha-numeric	1 to 10	3,760,620,109,779,060	96.8 %	316 GB 396 GB	Perfect Non-perfect	Perfect Non-perfect

https://project-rainbowcrack.com/table.htm

RainbowCrack Software Features

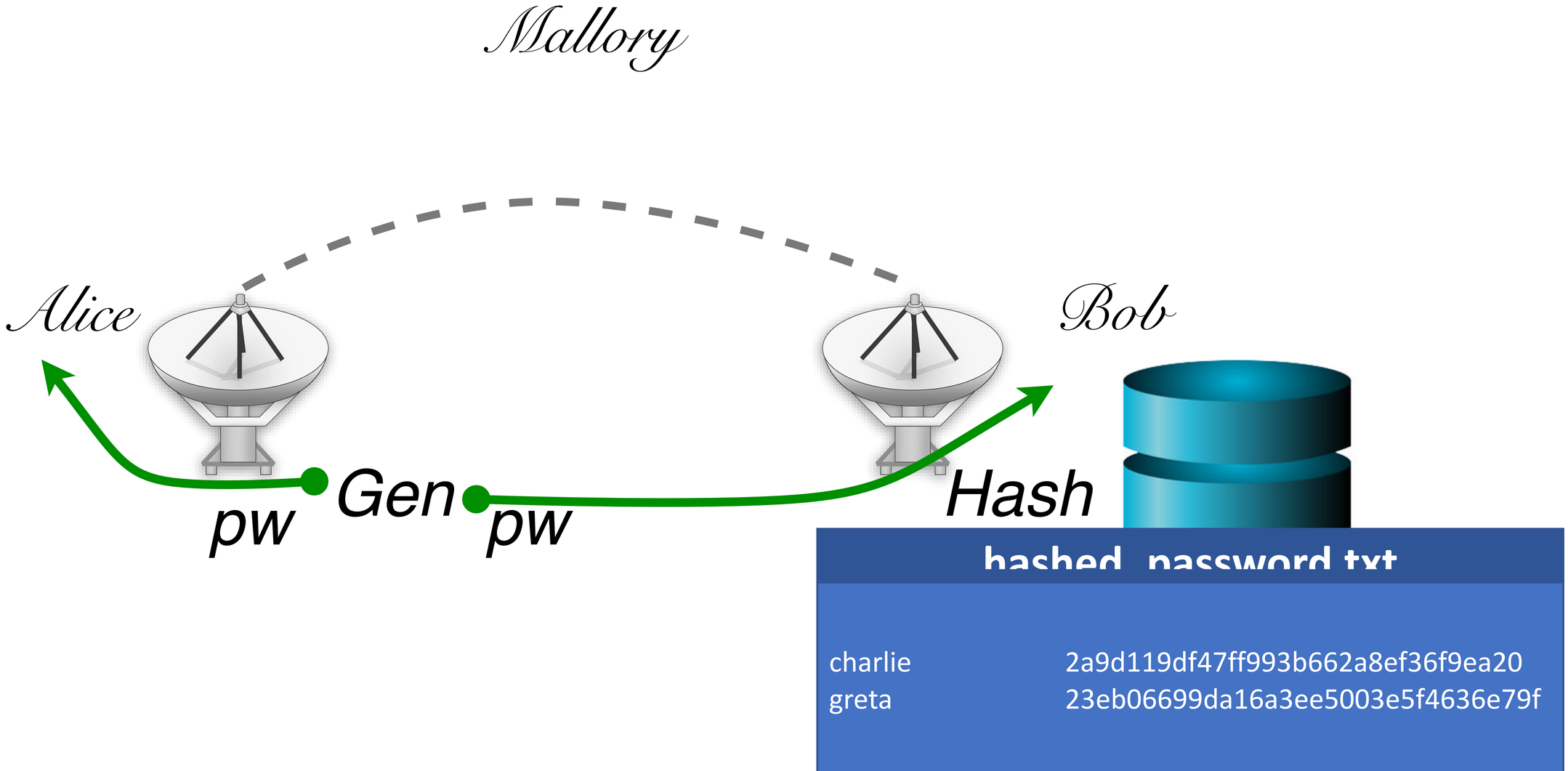
- High performance hash cracking on PC (> 10,000,000,000,000 plaintext tests per second)
- Optimized implementation of time-memory trade-off algorithm
- GPU acceleration with NVIDIA and AMD GPUs
- GPU acceleration with multiple GPUs
- Supports 64-bit Windows operating system
- Easy to use

- RainbowCrack 1.7 software
- One Seagate BarraCuda 6TB ST6000DM003 (SATA) hard drive containing rainbow tables and software
- License in USB dongle

The attack is highly effective

https://www.youtube.com/watch?v=TkMZJ3fTgrM

Attack 2: offline brute force



How to hamper offline brute force attacks?

Mallory

hashed_password.txt	
charlie	2a9d119df47ff993b662a8ef36f9ea20
greta	23eb06699da16a3ee5003e5f4636e79f

Hardening Password Hashes

- Key problem: cryptographic hashes are deterministic
  - hash(‘p4ssw0rd’) = hash(‘p4ssw0rd’)
  - This enables attackers to build lists of hashes

Hardening Password Hashes

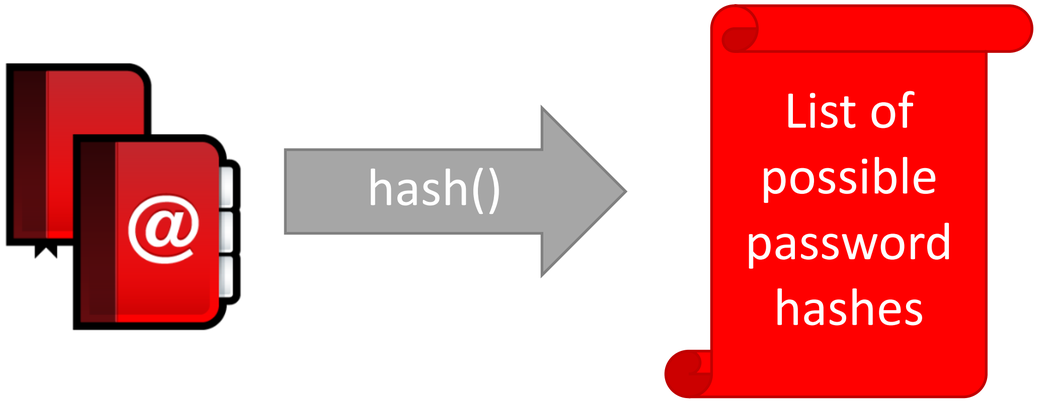
- Key problem: cryptographic hashes are deterministic
  - hash(‘p4ssw0rd’) = hash(‘p4ssw0rd’)
  - This enables attackers to build lists of hashes
- Solution: make each password hash unique
  - Add a random salt to each password before hashing
  - hash(salt + password) = password hash
  - Each user has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

Example Salted Hashes

hashed_password.txt	
cbw	2a9d119df47ff993b662a8ef36f9ea20
sandi	23eb06699da16a3ee5003e5f4636e79f
amislove	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

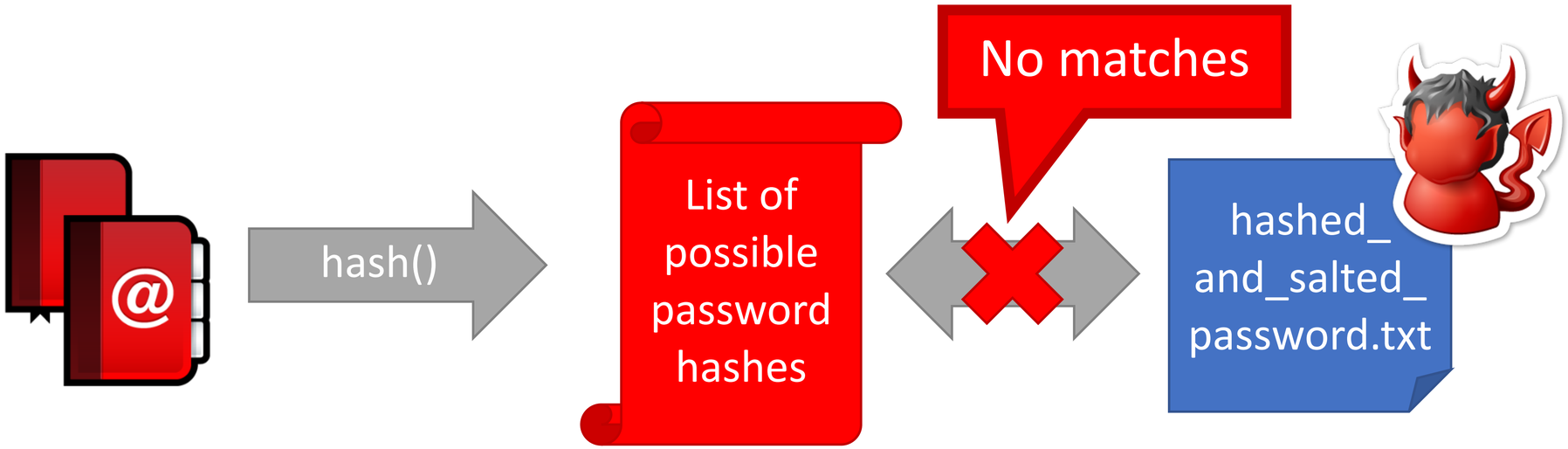
hashed_and_salted_password.txt		
cbw	a8	af19c842f0c781ad726de7aba439b033
sandi	0X	67710c2c2797441efb8501f063d42fb6
amislove	hz	9d03e1f28d39ab373c59c7bb338d0095
bob	K@	479a6d9e59707af4bb2c618fed89c245

Attacking Salted Passwords

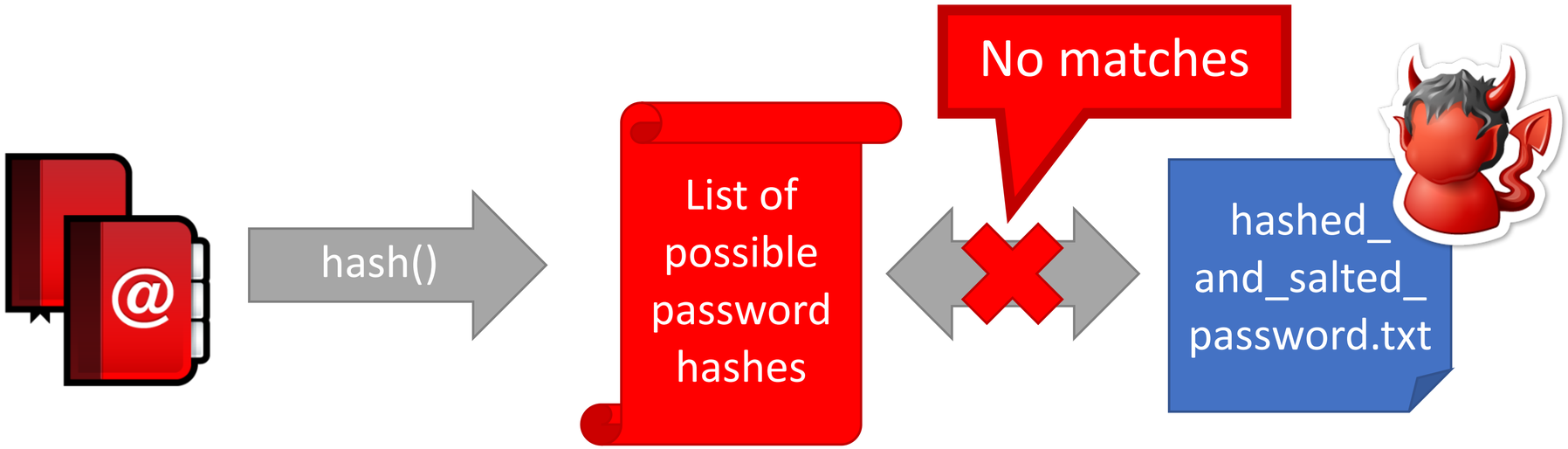


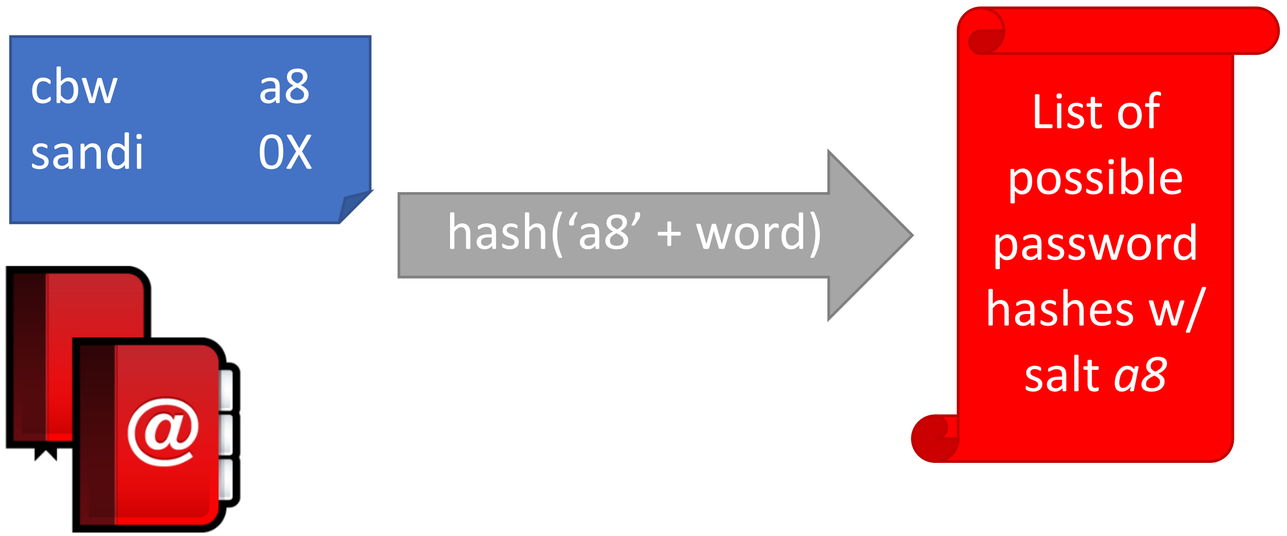
Attacking Salted Passwords

Attacking Salted Passwords

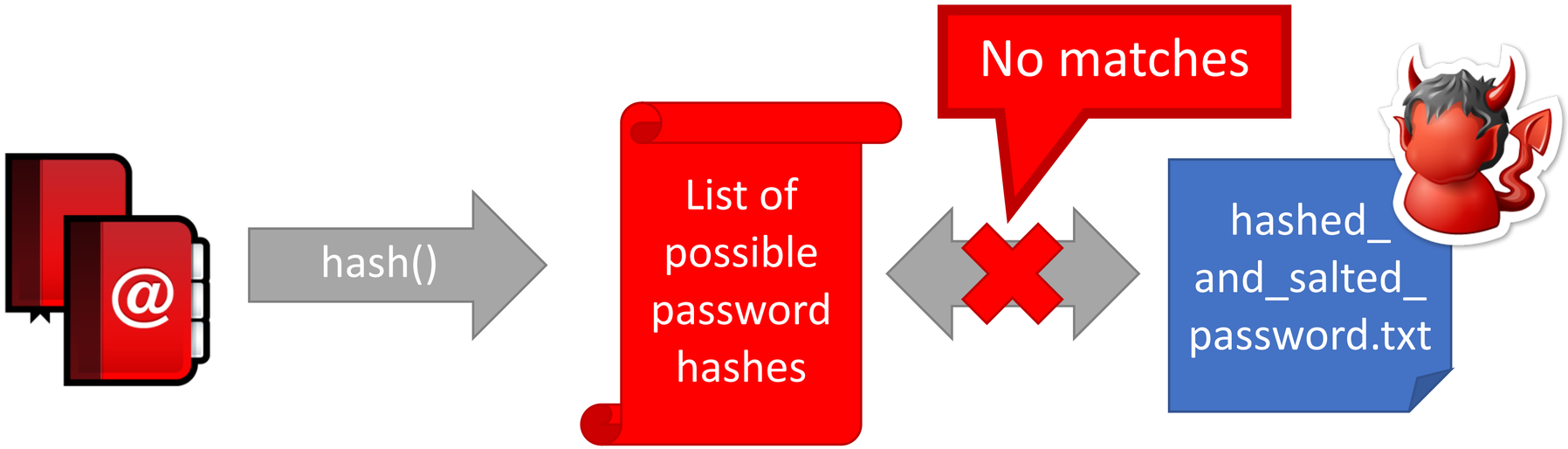


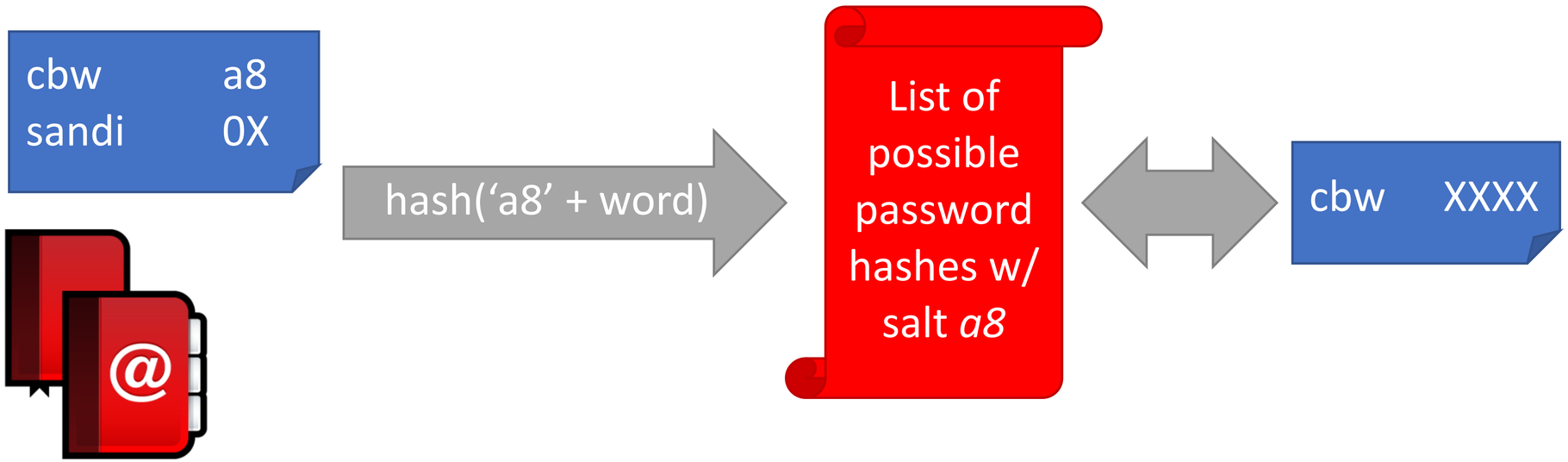
Attacking Salted Passwords



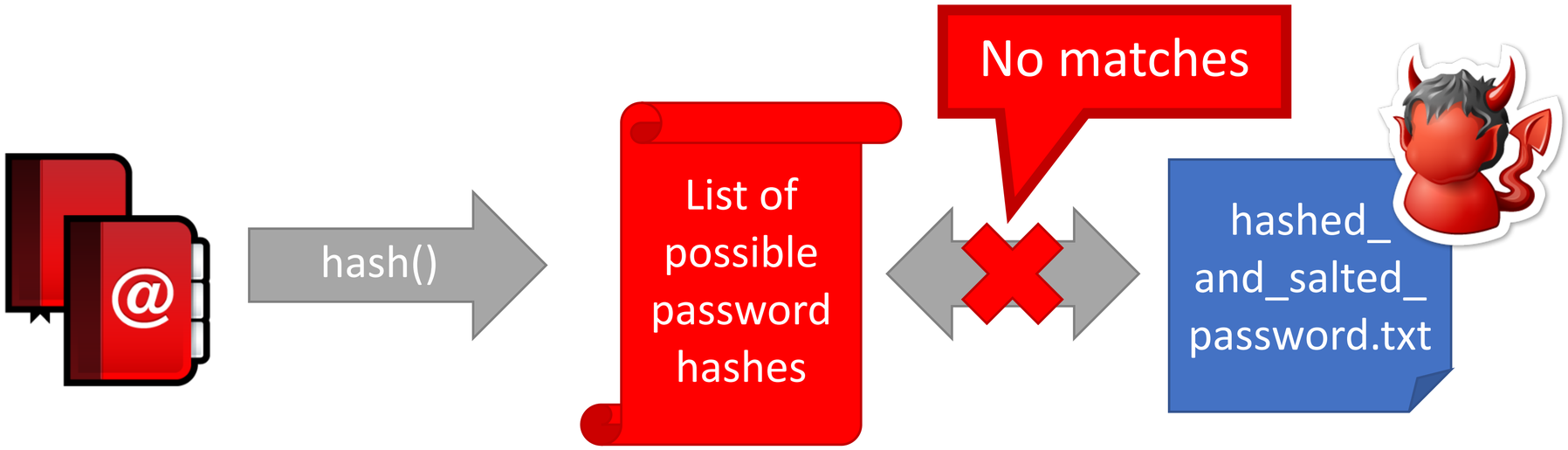


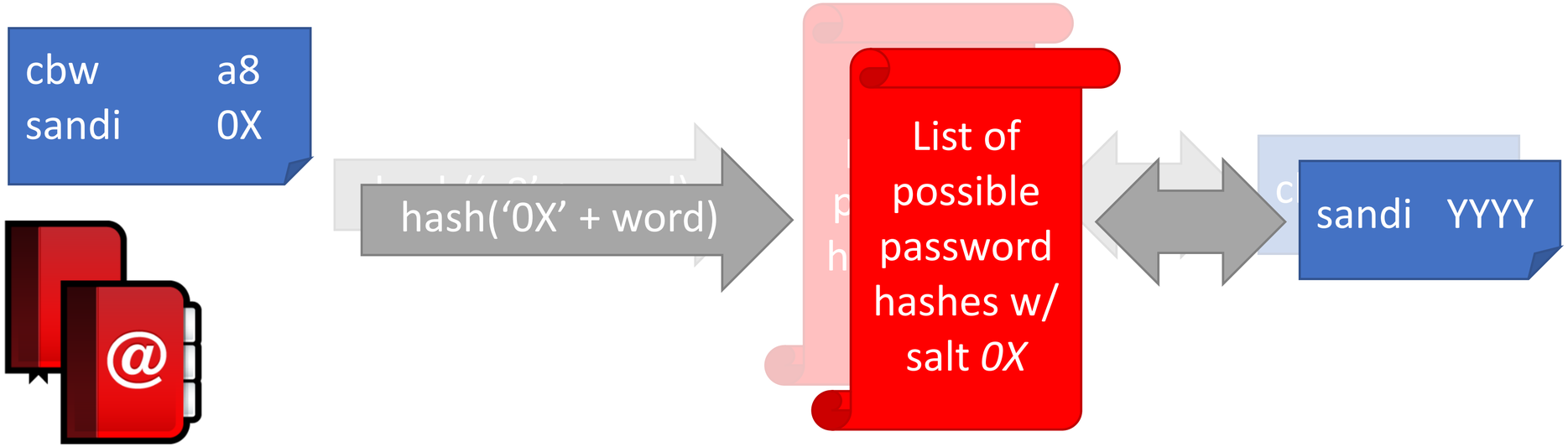
Attacking Salted Passwords





Attacking Salted Passwords





Breaking Hashed Passwords

- **Stored passwords should always be salted**
  - Forces the attacker to brute-force each password individually

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 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
    - $0.9 per hour (2018 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

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

Examples of Hashing Speed

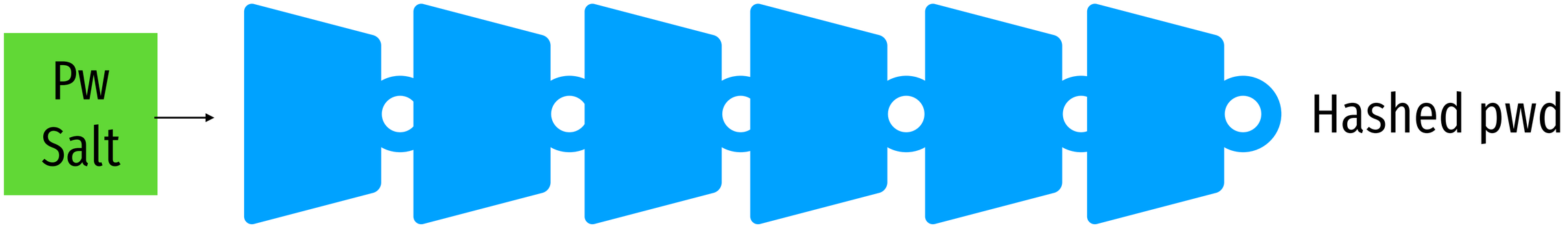
- A modern x86 server can hash all possible 6 character long passwords in 3.5 hours
  - Upper and lowercase letters, numbers, symbols
  - $(26+26+10+32)^6$ = 690 billion combinations
- A modern GPU can do the same thing in 16 minutes
- Most users use (slightly permuted) dictionary words, no symbols
  - 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
- Old solution: hash the password multiple times
  - Known as key stretching
  - Example: *crypt* used 25 rounds of DES
- New solution: use hash functions that are designed to be **slow**
  - Examples: bcrypt, PBKDF2, scrypt
  - 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

Slow hash movement

Iterated hash function {x times}



bcrypt Example

- Python example; install the *bcrypt* package

```
[cbw@localhost ~] 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

Best practices so far:

Dealing With Breaches

Dealing With Breaches

- Suppose you build an extremely secure password storage system
  - All passwords are salted and hashed by a high-work factor function
- It is still possible for a dedicated attacker to steal and crack passwords
  - Given enough time and money, anything is possible
  - E.g. The NSA
- Question: is there a principled way to detect password breaches?

Honeywords

- Key idea: store multiple salted/hashed passwords for each user
  - As usual, users create a single password and use it to login
  - User is unaware that additional honeywords are stored with their account

Honeywords

- Key idea: store multiple salted/hashed passwords for each user
  - As usual, users create a single password and use it to login
  - User is unaware that additional honeywords are stored with their account
- Implement a honeyserver that stores the index of the correct password for each user
  - Honeyserver is logically and physically separate from the password database
  - Silently checks that users are logging in with true passwords, not honeywords

Honeywords

- Key idea: store multiple salted/hashed passwords for each user
  - As usual, users create a single password and use it to login
  - User is unaware that additional honeywords are stored with their account
- Implement a honeyserver that stores the index of the correct password for each user
  - Honeyserver is logically and physically separate from the password database
  - Silently checks that users are logging in with true passwords, not honeywords
- What happens after a data breach?
  - Attacker dumps the user/password database…
  - But the attacker doesn’t know which passwords are honeywords
  - Attacker cracks all passwords and uses them to login to accounts
  - If the attacker logs-in with a honeyword, the honeyserver raises an alert!

Honeywords example

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fI	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

Honeywords example

Bob

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fI	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

Honeywords example

Bob

SHA512(“fl” | “p4ssW0rd”) → bHDJ8l

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fl	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

Honeywords example

Bob

SHA512(“fl” | “p4ssW0rd”) → bHDJ8l

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fl	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

Honeywords example

Bob

SHA512(“fl” | “p4ssW0rd”) → bHDJ8l

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fl	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

Honeywords example

Bob

SHA512(“fl” | “p4ssW0rd”) → bHDJ8l

Cracked Passwords

User	PW 1	PW 2	PW 3
Bob	123456	p4ssW0rd	Turtles!
sandi	puppies	iloveyou	blizzard
Alice	coff33	3spr3ss0	qwerty

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fl	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

Honeywords example

Bob

SHA512(“fl” | “p4ssW0rd”) → bHDJ8l

Cracked Passwords

User	PW 1	PW 2	PW 3
Bob	123456	p4ssW0rd	Turtles!
sandi	puppies	iloveyou	blizzard
Alice	coff33	3spr3ss0	qwerty

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
Bob	aB	y4DvF7	fl	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
Alice	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

User	Index
Bob	2
sandi	3
Alice	1

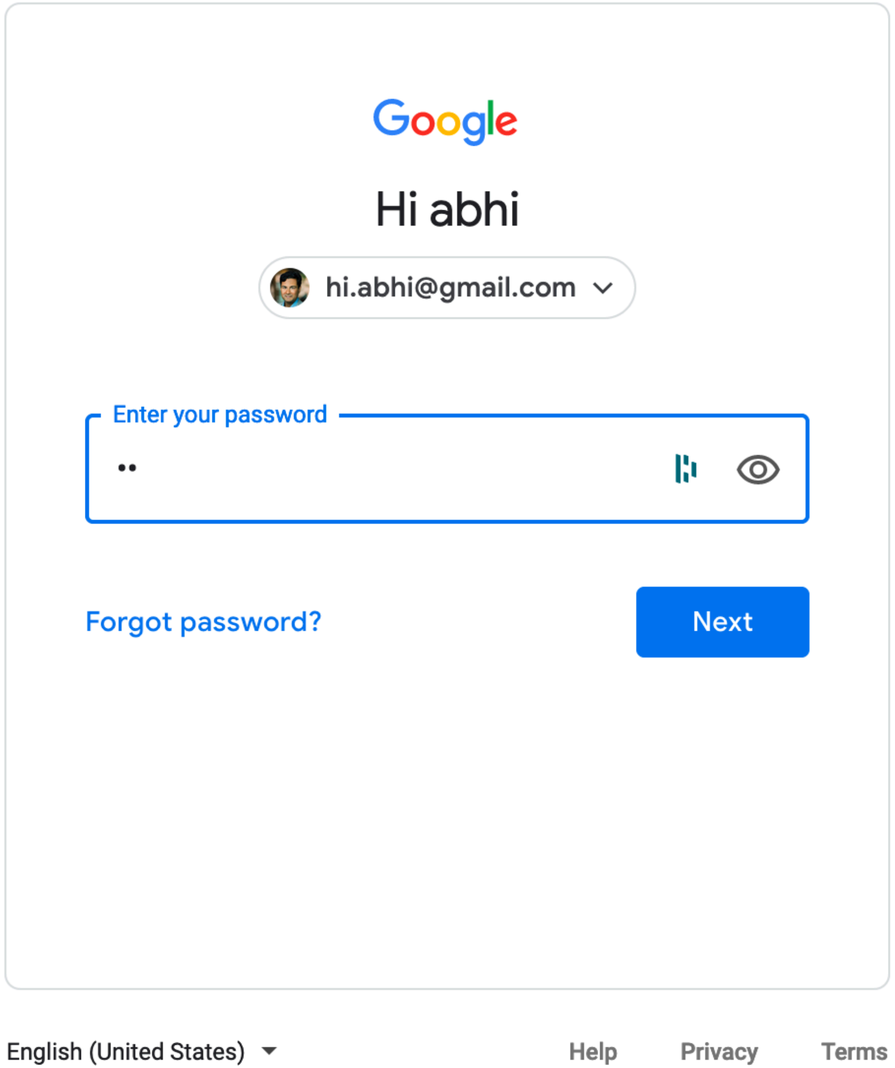
Multiple layers of storage

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**
4. **Implement honeywords to detect breaches**

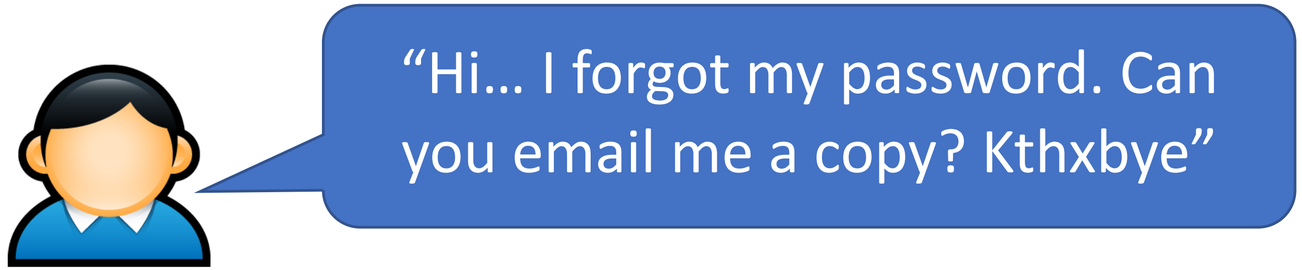
- These rules apply to any system that needs to authenticate users
  - Operating systems, websites, etc.

Still one problem?



Password Recovery/Reset

- Problem: hashed passwords cannot be recovered (hopefully)



- 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

Cracking Password Reset

- Typical implementations use Knowledge Based Authentication (KBA)
  - What was your mother's maiden name?
  - What was your prior street address?
  - Where did you go to elementary school

Cracking Password Reset

- Typical implementations use Knowledge Based Authentication (KBA)
  - What was your mother's maiden name?
  - What was your prior street address?
  - Where did you go to elementary school
- Problems?

Cracking Password Reset

- Typical implementations use Knowledge Based Authentication (KBA)
  - What was your mother's maiden name?
  - What was your prior street address?
  - Where did you go to elementary school
- Problems?
  - This information is widely available to anyone
  - Publicly accessible social network profiles
  - Background-check services like Spokeo

Cracking Password Reset

- Typical implementations use Knowledge Based Authentication (KBA)
  - What was your mother's maiden name?
  - What was your prior street address?
  - Where did you go to elementary school
- Problems?
  - This information is widely available to anyone
  - Publicly accessible social network profiles
  - Background-check services like Spokeo
- Experts recommend that services not use KBA
  - When asked, users should generate random answers to these questions