Cracking the Code of Secure Secrets:

Password Cracking Strategies

April 15, 2025

Outline

Cryptography Review

- What is a password hash, and how did we get to this point?

Things that Help PW Crackers

- What practices make it easier for password hashes to be broken?

Things that Hurt PW Crackers

- What practices make it harder for password hashes to be broken?

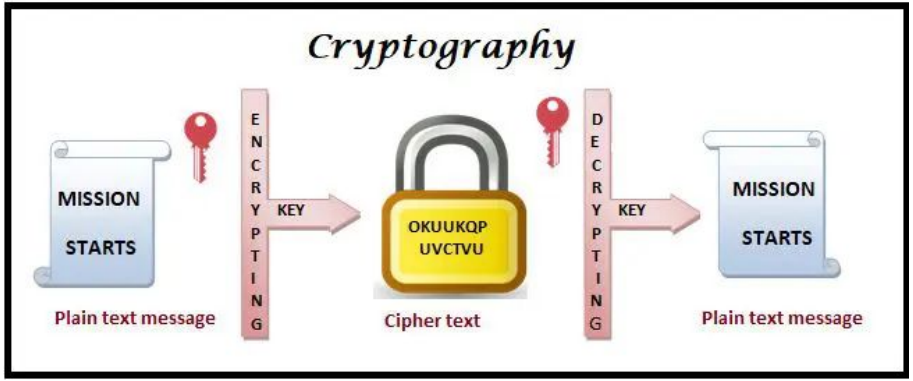
1. Cryptography Review

How do you stop people from reading your stuff?

How do you stop people from reading your stuff?

- Come up with a system to “hide” information from people not intended to receive it

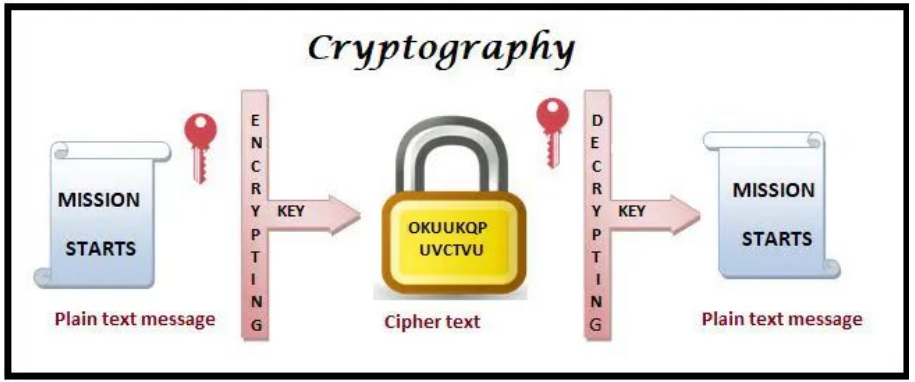
- **Plaintext** - the unencrypted information
- **Ciphertext** - the encrypted information
- **Cipher/Algorithm** - The algorithm/process used to encrypt/decrypt information



How do you stop people from reading your stuff?

Objectives of Cryptography

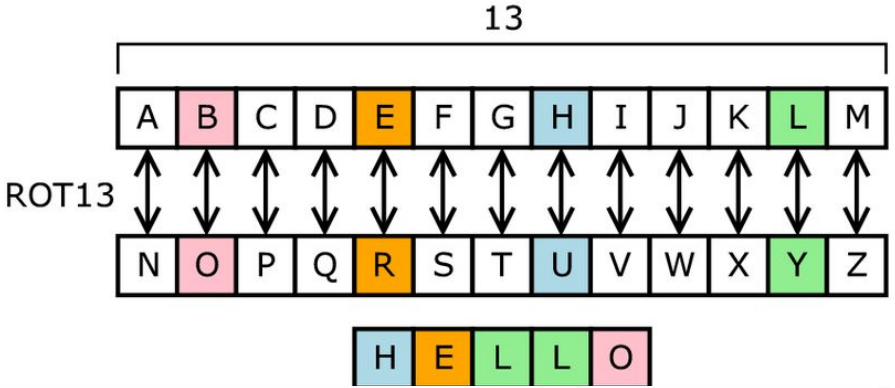
1) Hide the original message being sent (the main goal)
2) Hide additional information about the message that might hint towards its contents
    a) Think: is there anything in the message itself that tells me what it COULD be saying?



The Old Solution (Classical Ciphers)

Classical Cipher - Cryptography conducted mainly through pen and paper

- **Substitution Cipher** - Uses a system of replacing each letter/character with another one
- **Transposition Cipher** - Rearrange the letters in a way that hides the original message



Plain text: MEET ME AFTER THE TOGA PARTY

Row 1: M M T H G R

Row 2: E T E F E T E O A A T

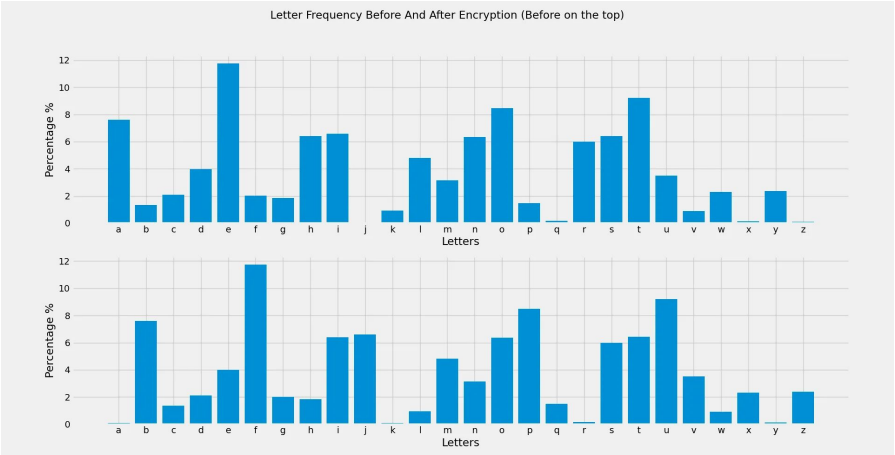
Row 3: E A R T P Y

Cipher Text: MMTHGR ETEFETEOAAT EARTPY

So Why Do Classic Ciphers Suck?

Cryptanalysis - Study of a cipher to find weaknesses that allow it to be decoded by an attacker

- Frequency Analysis - check how often letters show up in encrypted text vs in plain English
- Known-plaintext Attacks - Use “cribs” to figure out what plaintext encrypts to a specific ciphertext
- Good for by-hand encryption, but defeatable with statistics **and** computers!



Solution: Complex Algorithms!

- Algorithms no longer feasible to do by hand
- Mathematical functions, careful analysis to ensure these algorithms are **designed and implemented** to not leak information!

RSA Algorithm

Key Generation

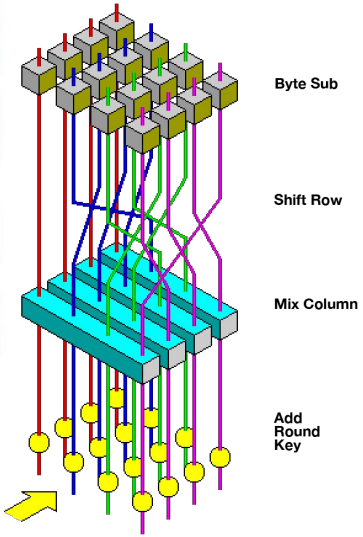
Select p,q	p and q both prime; p ≠ q
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n),e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e,n\}$
Private key	$KR = \{d,n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption

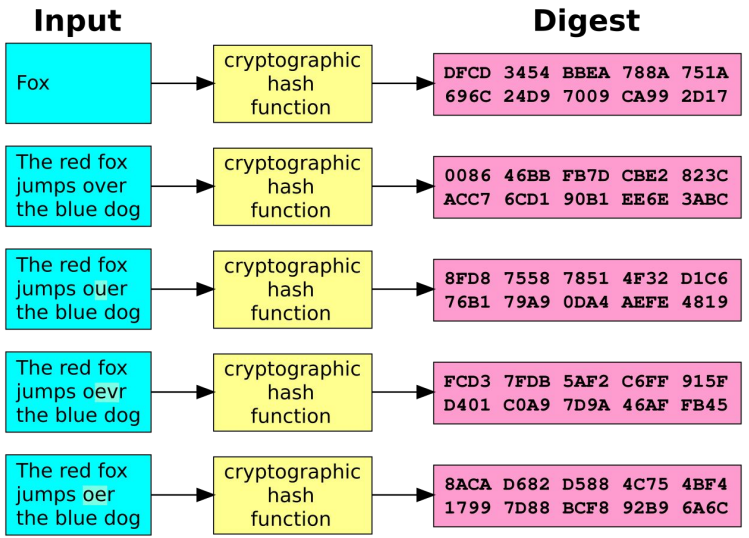
Plaintext:	C
Ciphertext:	$M = C^d \pmod n$



Hash Functions

Hashing

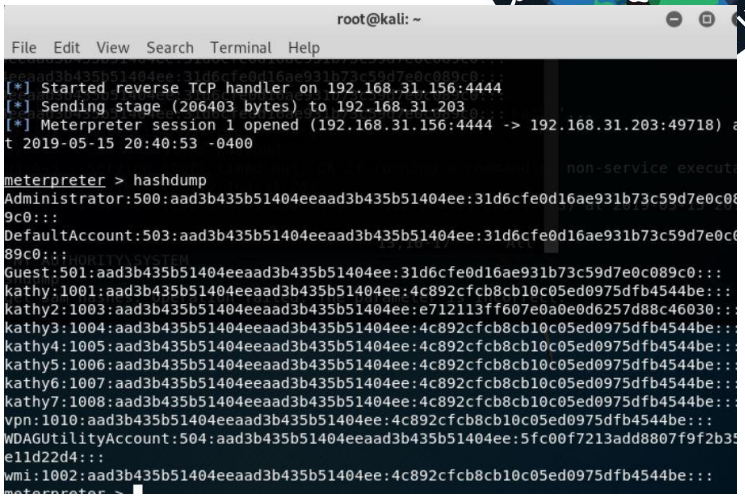
- Produces a “fingerprint” for a given stream of bits that can’t be used to recover those bits!
  - Think of it like cooking; you can’t recover the raw ingredients from the final product!
- Used to “hash” passwords for secure storage, check if files are corrupted, etc.



Hashing and Passwords

“Secure” Password Storage

- How do you store a password without keeping the actual password in memory?
  - Proves authentication: if a user’s password hash matches, they have the correct password!
  - Secures your password storage: if your hashes leak, not exactly losing the password



Guess and Check

- If it's not practical to "reverse" the hash algorithm, then why don't we just guess it?

- It's how "password cracking tools" work:
  - Given an encrypted hash, guess the password that goes with it

```
Session..........: hashcat
Status...........: Cracked
Hash.Name........: SolarWinds Serv-U
Hash.Target......: e983672a03adcc9767b24584338eb378:00
Time.Started.....: Sun May 23 11:43:13 2021 (1 sec)
Time.Estimated...: Sun May 23 11:43:14 2021 (0 secs)
Guess.Mask.......: ?a?a?a?a?a?at [7]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 24620.9 MH/s (32.19ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Recovered........: 1/1 (100.00%) Digests
Progress.........: 31606272000/735091890625 (4.30%)
Rejected.........: 0/31606272000 (0.00%)
Restore.Point....: 0/857375 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:35840-36864 Iteration:0-1024
Candidates.#1....: 4{,erat -> cyr ~}t
Hardware.Mon.#1..: Temp: 62c Fan: 31% Util:100% Core:1920MHz Mem:7000MHz Bus:16
```

Length of Password	Combinations	Time to Crack (yrs)	Time to Crack (s)
4	456976	0.0	0.000228488
5	11881376	0.0	0.005940688
6	308915776	0.0	0.154457888
7	8031810176	0.0	4.015905088
8	208827064576	0.0	104.4135323
9	5429503678976	0.0	2714.751839
10	141167095653376	0.0	70583.54783
11	3670344486987780	0.1	1835172.243
12	95428956661682200	1.5	47714478.33
13	2481152873203740000	39.3	1240576437
14	64509974703297200000	1022.8	32254987352
15	1677259342285730000000	26592.8	8.3863E+11
16	436087428994289000000000	691412.1	2.18044E+13
17	11338273153851500000000000	17976714.2	5.66914E+14
18	2947951020001390000000000000	467394568.1	1.47398E+16

2.

How To Help Password Crackers

What can be done to make password hashes less secure?

Tools: Hash Crackers

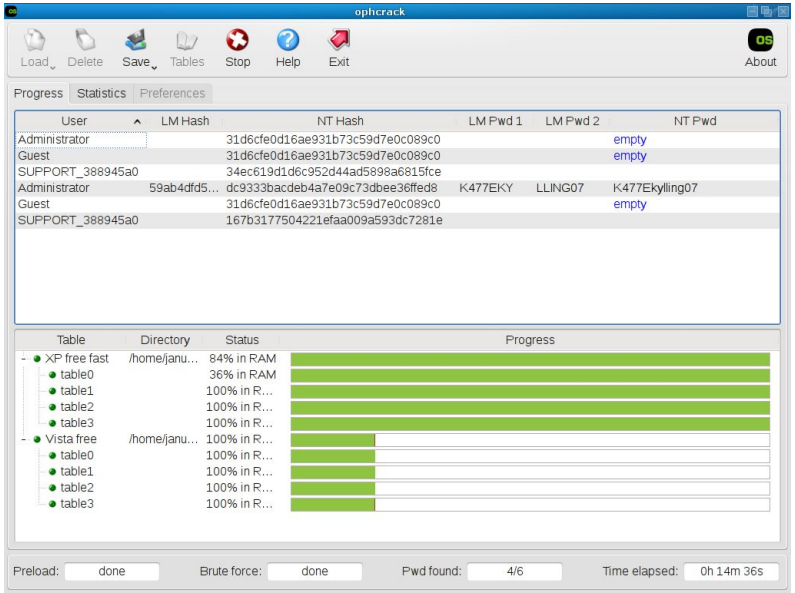
- Software that brute-forces hashes until it finds a match
- Optimizes hash cracking by splitting workload between CPUs and GPUs
    - Some unique features between programs, but functionally similar
- EX: Hashcat, John the Ripper


```
5f6b883d42910065a60fbdbfbfa27caa:Spring2018

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1000 (NTLM)
Hash.Target......: 5f6b883d42910065a60fbdbfbfa27caa
Time.Started.....: Fri May 27 14:29:42 2022 (0 secs)
Time.Estimated...: Fri May 27 14:29:42 2022 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.......: File (seasons.txt), Left Side
Guess.Mod........: Mask (20?d?d) [4], Right Side
Guess.Queue.Base.: 1/1 (100.00%)
Guess.Queue.Mod..: 1/1 (100.00%)
Speed.#1.........:    30204 H/s (0.07ms) @ Accel:64 Lo
Recovered.Total..: 1/1 (100.00%) Digests
Progress.........: 400/400 (100.00%)
Rejected.........: 0/400 (0.00%)
```

Tools: Rainbow Crackers

- Software that loads large tables of pre-computed hashes to try and find a match
- Trade memory for computation power compared to hash crackers
- EX: Ophcrack, RainbowCrack



Anti-Tip #1: Use a Fast Hash Algorithm

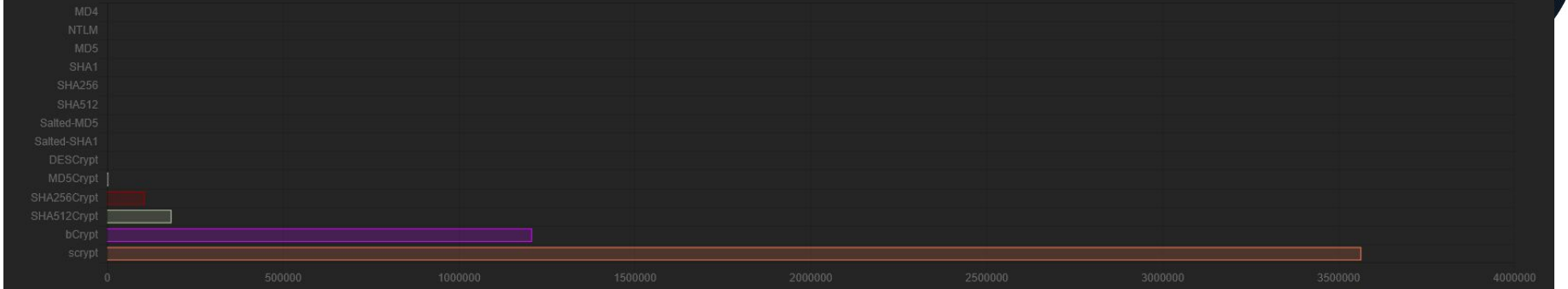
- Some hash algorithms are less computationally intense than others…
  - EX: MD5, NTLM, SHA
- But what's the difference between "legitimate" hashing and "illegitimate" hashing?
  - Authentication only does it once, cracking does it million of times!

Estimated Password Recovery Times — 1x Terahash Brutalis, 44x Terahash Inmanis (448x Nvidia RTX 2080)
Alphanumeric mask attack with Terahash Hashstack

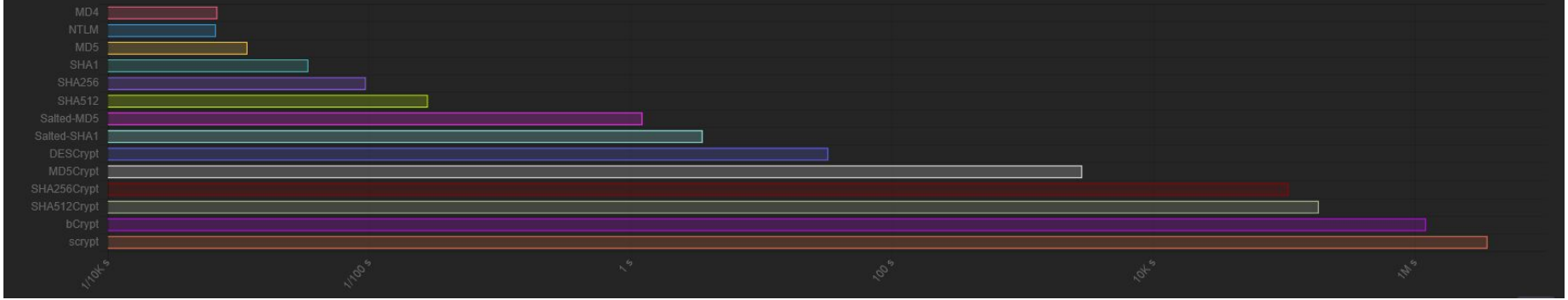
	Speed	Length 4	Length 5	Length 6	Length 7	Length 8	Length 9	Length 10	Length 11	Length 12	Length 13
NTLM	31.82 TH/s	Instant	Instant	Instant	Instant	Instant	7 mins 6 secs	7 hrs 19 mins	2 wks 4 days	3 yrs 2 mos	199 yrs 2 mos
MD5	17.77 TH/s	Instant	Instant	Instant	Instant	Instant	12 mins 42 secs	13 hrs 7 mins	1 mo 0 wk	5 yrs 9 mos	356 yrs 7 mos
NetNTLMv1 / NetNTLMv1+ESS	16.82 TH/s	Instant	Instant	Instant	Instant	Instant	13 mins 25 secs	13 hrs 51 mins	1 mo 0 wk	6 yrs 0 mo	376 yrs 10 mos
LM	15.81 TH/s	Instant	Instant	Instant	Instant						
SHA1	5.89 TH/s	Instant	Instant	Instant	Instant	Instant	38 mins 18 secs	1 day 15 hrs	3 mos 1 wk	17 yrs 4 mos	1.1 mil
SHA2-256	2.42 TH/s	Instant	Instant	Instant	Instant	1 min 31 secs	1 hr 33 mins	4 days 0 hr	8 mos 0 wk	42 yrs 2 mos	2.6 mil
NetNTLMv2	1.22 TH/s	Instant	Instant	Instant	Instant	3 mins 0 sec	3 hrs 5 mins	1 wk 0 day	1 yr 4 mos	83 yrs 10 mos	5.2 mil
SHA2-512	801.9 GH/s	Instant	Instant	Instant	Instant	4 mins 33 secs	4 hrs 41 mins	1 wk 5 days	2 yrs 0 mo	127 yrs 5 mos	7.9 mil
descrypt, DES (Unix), Traditional DES	647.59 GH/s	Instant	Instant	Instant	Instant	5 mins 38 secs	5 hrs 48 mins	2 wks 1 day	2 yrs 6 mos	157 yrs 10 mos	9.8 mil
Kerberos 5, etype 23, TGS-REP	206.97 GH/s	Instant	Instant	Instant	Instant	17 mins 35 secs	18 hrs 10 mins	1 mo 2 wks	7 yrs 11 mos	493 yrs 11 mos	30.6 mil
Kerberos 5, etype 23, AS-REQ Pre-Auth	206.78 GH/s	Instant	Instant	Instant	Instant	17 mins 36 secs	18 hrs 11 mins	1 mo 2 wks	7 yrs 11 mos	494 yrs 5 mos	30.7 mil
md5crypt, MD5 (Unix), Cisco-IOS 1 (MD5)	7.61 GH/s	Instant	Instant	Instant	7 mins 44 secs	7 hrs 58 mins	2 wks 6 days	3 yrs 5 mos	216 yrs 9 mos	13.4 mil	833.9 mil
LastPass + LastPass sniffed	1.78 GH/s	Instant	Instant	Instant	32 mins 54 secs	1 day 9 hrs	2 mos 3 wks	14 yrs 10 mos	924 yrs 0 mo	57.3 mil	3554 mil
macOS v10.8+ (PBKDF2-SHA512)	335.09 MH/s	Instant	Instant	2 mins 50 secs	2 hrs 55 mins	1 wk 0 day	1 yr 3 mos	79 yrs 4 mos	4.9 mil	305.3 mil	18926.3 mil
WPA-EAPOL-PBKDF2	277.23 MH/s					1 wk 2 days	1 yr 6 mos	95 yrs 11 mos	6 mil	369 mil	22876.6 mil
TrueCrypt RIPEMD160 + XTS 512 bit	211.78 MH/s	Instant	Instant	4 mins 29 secs	4 hrs 37 mins	1 wk 4 days	2 yrs 0 mo	125 yrs 7 mos	7.8 mil	483 mil	29947.1 mil
7-Zip	181.51 MH/s	Instant	Instant	5 mins 13 secs	5 hrs 23 mins	1 wk 6 days	2 yrs 4 mos	146 yrs 6 mos	9.1 mil	563.6 mil	34940.7 mil
sha512crypt 6, SHA512 (Unix)	119.46 MH/s	Instant	Instant	7 mins 56 secs	8 hrs 11 mins	3 wks 0 day	3 yrs 7 mos	222 yrs 7 mos	13.8 mil	856.3 mil	53090.5 mil
DPAPI masterkey file v1	47.23 MH/s	Instant	Instant	20 mins 3 secs	20 hrs 42 mins	1 mo 3 wks	9 yrs 0 mo	563 yrs 1 mo	34.9 mil	2165.7 mil	134271.5 mil
RAR5	28.15 MH/s	Instant	Instant	33 mins 39 secs	1 day 10 hrs	2 mos 4 wks	15 yrs 2 mos	944 yrs 11 mos	58.6 mil	3634.4 mil	225334 mil
DPAPI masterkey file v2	27.82 MH/s	Instant	Instant	34 mins 2 secs	1 day 11 hrs	2 mos 4 wks	15 yrs 5 mos	955 yrs 11 mos	59.3 mil	3676.7 mil	227953.7 mil
RAR3-hp	20.84 MH/s	Instant	Instant	45 mins 26 secs	1 day 22 hrs	3 mos 4 wks	20 yrs 6 mos	1.3 mil	79.2 mil	4907.7 mil	304274.7 mil
KeePass 1 (AES/Twofish) and KeePass 2 (AES)	17.8 MH/s	Instant	Instant	53 mins 12 secs	2 days 6 hrs	4 mos 2 wks	24 yrs 1 mo	1.5 mil	92.7 mil	5746.9 mil	356305.7 mil
bcrypt $2*$, Blowfish (Unix)	11.37 MH/s	Instant	1 min 21 secs	1 hr 23 mins	3 days 14 hrs	7 mos 1 wk	37 yrs 8 mos	2.3 mil	145.1 mil	8996 mil	557755.1 mil
Bitcoin/Litecoin wallet.dat	3.55 MH/s	Instant	4 mins 18 secs	4 hrs 26 mins	1 wk 4 days	1 yr 11 mos	120 yrs 8 mos	7.5 mil	464.2 mil	28782.1 mil	1784492.8 mil

Common Hashes and Cracking Speeds

The hash type used by the compromised site will have a massive impact on how long it would take the attacker to get through that attack. Here is a graph of (relatively, roughly) how many seconds it would take to complete that attack depending on what hash type was in use, with a standard graphics card:



Well, that's useless! The strongest hash types are so much slower that all the faster types just squash down to nothing. Let's try the same data again using a logarithmic X axis time scale. As the bars go left-to-right they are increasing by powers of 10:



Anti-Tip #2: Use a Predictable Password

- People tend to use things they can actually remember to make passwords
  - “password” and other variations
  - Easy number patterns i.e. “12345,” years, dates
  - **Actual words**

- Instead of guessing every possible combination, we can narrow it down A LOT

Probably not your first choice of password

	2019		2020		2021		2022	
	Password	Number of users	Password	Number of users	Password	Number of users	Password	Number of users
1	12345	2,812,220	123456	2,543,285	123456	103,170,552	password	4,929,113
2	123456	2,485,216	123456789	961,435	123456789	46,027,530	123456	1,523,537
3	123456789	1,052,268	picture1	371,612	12345	32,955,431	123456789	413,056
4	test1	993,756	password	360,467	qwerty	22,317,280	guest	376,417
5	password	830,846	12345678	322,187	password	20,958,297	qwerty	309,679
6	12345678	512,560	111111	230,507	12345678	14,745,771	12345678	284,946
7	zinch	483,443	123123	189,327	111111	13,354,149	111111	229,047
8	g_czechout	372,278	12345	188,268	123123	10,244,398	12345	188,602
9	asdf	359,520	1234567890	171,724	1234567890	9,646,621	col123456	140,505
10	qwerty	348,762	senha	167,728	1234567	9,396,813	123123	127,762
11	1234567890	329,341	1234567	165,909	qwerty123	8,933,334	1234567	110,279
12	1234567	261,610	qwerty	156,765	000000	8,377,094	1234	106,929

People Are Bad At Making Passwords

- Instead of guessing every combo, why don't we try lists of common passwords and combinations?

- rockyou.txt - The most well known password wordlist!
  - 32 MILLION unencrypted passwords breached
  - In every Kali installation
  - (Most cyber competition challenges will have you use this)

rockyou

```
jason@kali:/usr/share/wordlists$ ls -l
total 188756
lrwxrwxrwx 1 root root        26 Jul 22 23:33 amass → /usr/sh
lrwxrwxrwx 1 root root        25 Jul 22 23:33 dirb → /usr/sha
lrwxrwxrwx 1 root root        30 Jul 22 23:33 dirbuster → /us
lrwxrwxrwx 1 root root        41 Jul 22 23:33 fasttrack.txt →
lrwxrwxrwx 1 root root        45 Jul 22 23:33 fern-wifi → /us
lrwxrwxrwx 1 root root        28 Jul 22 23:33 john.lst → /usr
lrwxrwxrwx 1 root root        27 Jul 22 23:33 legion → /usr/s
lrwxrwxrwx 1 root root        46 Jul 22 23:33 metasploit → /u
ts
lrwxrwxrwx 1 root root        41 Jul 22 23:33 nmap.lst → /usr
-rw-r--r-- 1 root root 139921507 Jul 17  2019 rockyou.txt
-rw-r--r-- 1 root root  53357329 May 12  2023 rockyou.txt.gz
lrwxrwxrwx 1 root root        39 Jul 22 23:33 sqlmap.txt → /u
lrwxrwxrwx 1 root root        25 Jul 22 23:33 wfuzz → /usr/sh
lrwxrwxrwx 1 root root        37 Jul 22 23:33 wifite.txt → /u
jason@kali:/usr/share/wordlists$
```

People Are Bad At Making Passwords

Wordlist Sources

OK, what if rockyou didn't work?

- MORE Wordlists!
  - Crackstation (15 **GB!**)
    - https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm
  - Hashmob (good for wordlists and for sample hashlists!)
    - https://hashmob.net/resources/hashmob
  - Default Password Wordlists!
    - https://www.google.com/search?q=default+password+wordlists

Power Up your Wordlists!

Rules Engine

- Nearly every trick to change a password to be “more secure” has been thought of already!
- Feed “rules” into crackers to change/combine wordlists with modifications!

EX: OneRuleToRuleThemAll (on Github)

Generate your Own Lists!

- (In CTF-land) Do passwords follow a theme? Use tools to scrape the web for words you can use!
  - EX: CeWL
- (In the real world) Know your environment! Words associated with the company, local sports teams, etc.

Wordlist Only


```
Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 0 (MD5)
Hash.Target......: ..\wordlists\onlinetrade_ru.hashes
Time.Started.....: Tue Apr 15 13:04:59 2025 (1 min, 0 secs)
Time.Estimated...: Tue Apr 15 13:05:59 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (..\wordlists\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   179.0 kH/s (17.25ms) @ Accel:1024 Loops:1 Thr:64 Vec:1
Speed.#3.........:    68875 H/s (5.31ms) @ Accel:2048 Loops:1 Thr:32 Vec:1
Speed.#*.........:   247.9 kH/s
Recovered........: 149747/1100746 (13.60%) Digests (total), 16632/1100746 (1.51%) Digests (new)
Remaining........: 950999 (86.40%) Digests
Recovered/Time...: CUR:N/A,N/A,N/A AVG:N/A,N/A,N/A (Min,Hour,Day)
Progress.........: 14344384/14344384 (100.00%)
Rejected.........: 0/14344384 (0.00%)
Restore.Point....: 13349290/14344384 (93.06%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Restore.Sub.#3...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[313233333231766970657 2] -> $HEX[042a0337c2a156616d6f732103]
Candidates.#3....: 1712841-2 -> 123321vivi
Hardware.Mon.#1..: Temp: 53c Util:  0% Core: 210MHz Mem: 810MHz Bus:8
Hardware.Mon.#3..: Temp:  0c Fan:  0% Util:  0% Core: 400MHz Mem: 800MHz Bus:16

Started: Tue Apr 15 13:04:45 2025
Stopped: Tue Apr 15 13:06:00 2025
PS C:\Users\nucle\CTF\CSCDevelopment\hashcat-6.2.6> S
```

Attack against the onlinetrade.ru breach (on HashMob) **just using rockyou.txt**

13.6% and no others found

Wordlist + Rules

```
Session..........: hashcat
Status...........: Quit
Hash.Mode........: 0 (MD5)
Hash.Target......: ..\wordlists\onlinetrade_ru.hashes
Time.Started.....: Tue Apr 15 13:08:00 2025 (7 mins, 2 secs)
Time.Estimated...: Wed Apr 16 01:42:39 2025 (12 hours, 27 mins)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (..\wordlists\rockyou.txt)
Guess.Mod........: Rules (..\wordlists\OneRuleToRuleThemStill.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  8556.3 kH/s (66.31ms) @ Accel:64 Loops:64 Thr:64 Vec:1
Speed.#3.........:  6784.6 kH/s (18.60ms) @ Accel:128 Loops:128 Thr:32 Vec:1
Speed.#*.........: 15340.9 kH/s
Recovered........: 255867/1100746 (23.24%) Digests (total), 106120/1100746 (9.64%) Digests (new)
Remaining........: 844879 (76.76%) Digests
Recovered/Time...: CUR:12169,N/A,N/A AVG:15062.01,N/A,N/A (Min,Hour,Day)
Progress.........: 6305382400/694469006976 (0.91%)
Rejected.........: 0/6305382400 (0.00%)
Restore.Point....: 90112/14344384 (0.63%)
Restore.Sub.#1...: Salt:0 Amplifier:5248-5312 Iteration:0-64
Restore.Sub.#3...: Salt:0 Amplifier:16640-16768 Iteration:0-128
Candidate.Engine.: Device Generator
Candidates.#1....: 98bitak -> Blutnote
Candidates.#3....: 0225ed79 -> zacharykatiekatie
Hardware.Mon.#1..: Temp: 51c Util:  0% Core: 210MHz Mem: 405MHz Bus:8
Hardware.Mon.#3..: Temp:  0c Fan:  0% Util:  0% Core: 400MHz Mem: 800MHz Bus:16

Started: Tue Apr 15 13:07:46 2025
Stopped: Tue Apr 15 13:15:04 2025
PS C:\Users\nucle\CTF\CSCDevelopment\hashcat-6.2.6>
```

Attack against the onlinetrade.ru breach (on HashMob) **using rules + rockyou**

23.24% recovered **and still going!** (my computer is burning up)

Sample Hashcat Usage

Sample dictionary (wordlist only) attack w/ Hashcat

hashcat -m (hash type) *-a 0* (hash file) (wordlist) *--show*

Sample Mask attack (brute-force) using a 6-char PW

hashcat -m (hash type) *-a 3* (hash file) *?a?a?a?a?a?a --show*

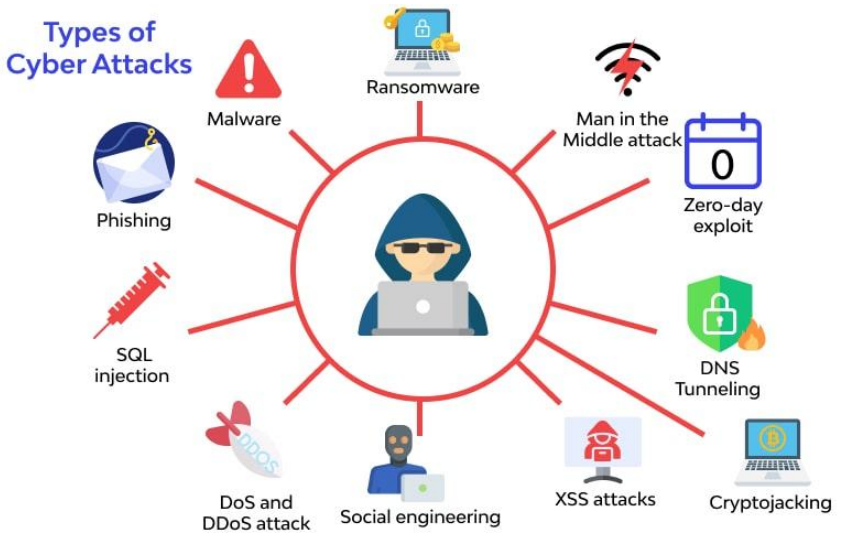
3.

How To Hurt Password Crackers

What can be done to make password hashes more secure?

Disclaimer

- Most attackers won't bother to crack your password
  - Doesn't mean you shouldn't try…
- Goal: Don't be the weak link!



Tip #1: Slower Password Algorithms

- Not all hashing algorithms created equal!
- Slower algorithms exponentially increase the power an attacker needs to break your hash

To sum up our recommendations:

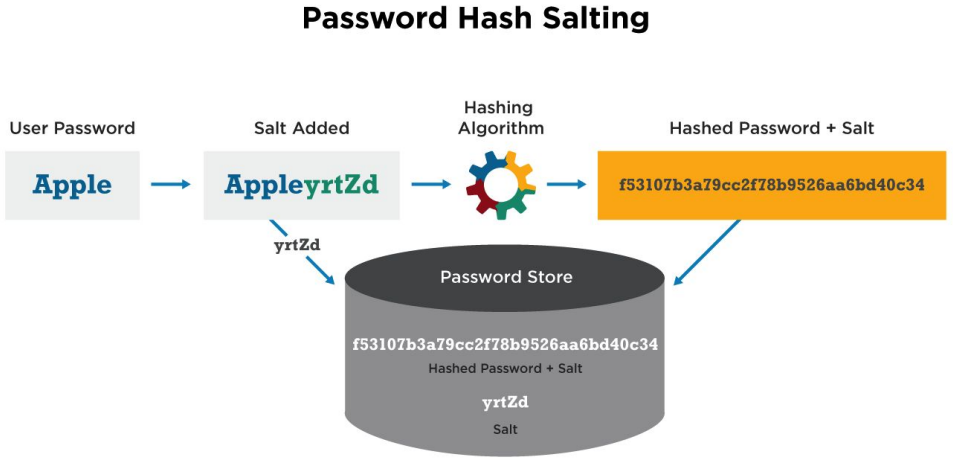
- Use Argon2id with a minimum configuration of 19 MiB of memory, an iteration count of 2, and 1 degree of parallelism.
- If Argon2id is not available, use scrypt with a minimum CPU/memory cost parameter of (2^17), a minimum block size of 8 (1024 bytes), and a parallelization parameter of 1.
- For legacy systems using bcrypt, use a work factor of 10 or more and with a password limit of 72 bytes.
- If FIPS-140 compliance is required, use PBKDF2 with a work factor of 600,000 or more and set with an internal hash function of HMAC-SHA-256.
- Consider using a pepper to provide additional defense in depth (though alone, it provides no additional secure characteristics).

OWASP recommendations for password storage

Tip #2: Salt your Passwords!

Password Salt

- A random value appended to the end of your password before it's hashed
- Not a defense against brute-force attacks, but prevents the use of rainbow tables



Tip #3: General Password Tips

Best Solution: Password Manager

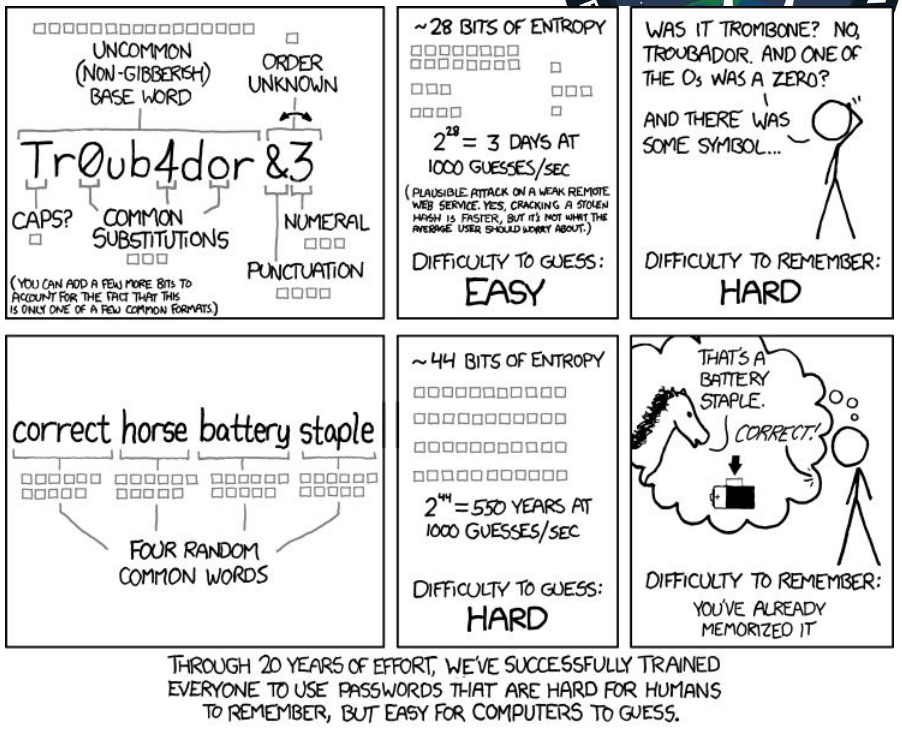
- A lot of discourse on this topic…
  - Not perfect, but it's good considering the tradeoffs
- A number of us officers use BitWarden; take that as you will

Tip #3: General Password Tips

Generating Secure Passwords

- At minimum, don't use the really easy-to-guess passwords
- Classic "mangling" rules aren't as cost-effective anymore for password complexity
- Check if your PW has been breached before!

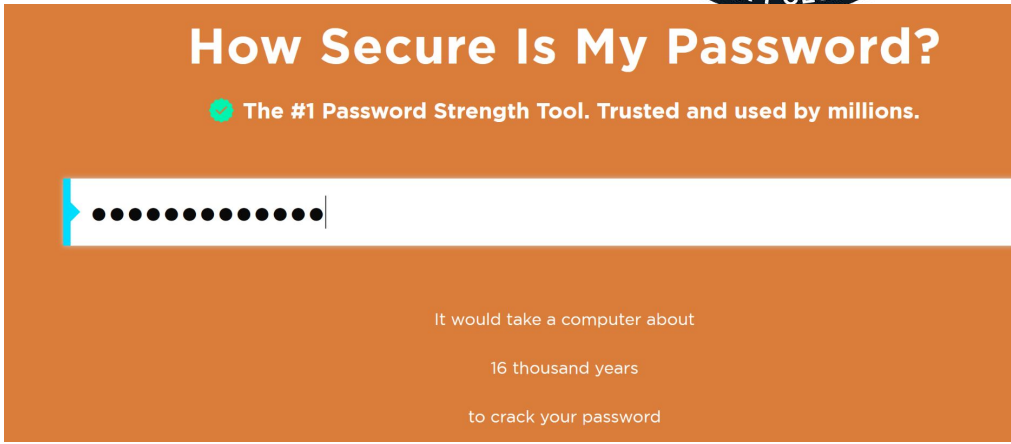
https://haveibeenpwned.com/



Tips to Secure Your Passwords

Should I use a Passphrase?

- Works, if the dictionary words are truly random!
- “To-be-or-not-to-be” is actually really guessable…



Long password, but not random enough!

Review

Cryptography Review

- Password hash - a one-way function that forces attackers to guess the password instead of “breaking” encryption

Things that Help PW Crackers

- Weak hash types speed up cracking, but the biggest culprit is weak password practices!
- Doing just the “bare minimum” for good passwords

Things that Hurt PW Crackers

- Using strong hash functions and salting slows down crackers!
- Use password managers, or a long, random password

Thanks!

Any questions?