Recall: hash functions

Diffusion: small changes ⇒ large effects

All values should be equally likely

Should resist:

Collision attack: find X_1, X_2 s.t. $h(X_1) = h(X_2)$
Preimage attack: given $h(X_1)$, find X_2 s.t. $h(X_1) = h(X_2)$
2nd preimage attack: given X_1, find $X_2 \neq X_1$ s.t. $h(X_1) = h(X_2)$

Collision attack

Finding _____ that hash to the same value. When we get to digital signatures, we'll see that collision attacks can be quite important: if you can generate two messages with different meanings but the same hash, you can cause a lot of trouble! However, such attacks aren't so useful for password security.

Even with the strongest hash function, collisions are _____ _____ due to the birthday paradox. However, "easier" doesn't have to be "easy": if the hash output is large, you can still have a lot of work to do! $\sqrt{2^n}$ can still be a large number if n is big enough...

Preimage attack

Finding an input that hashes to the same value as a given hash. This could be the same input that was originally used to generate the hash or a different one.

Second preimage attack

Finding a _____ input that will hash to the same value as a given input. This is like a collision attack, but much harder: instead of generating lots of messages and finding two that hash to the same value, you have to find one that hashes to the same value _____.

MAC generalization

Can we MAC without a block cipher?

HMAC: hash-based message authentication code*

$$h\left((k \oplus p_o)||h((k \oplus p_i)||text)\right)$$

* Bellare, Canetti and Krawczyk, "Keying Hash Functions for Message Authentication", CRYPTO 1996, 1996. Standardized by NIST (FIPS 198-1) and the IETF (RFC 2104).

An HMAC uses a hash function *with* a key. This provides the same security properties as a block-cipher–based MAC, just with a different underlying cryptographic algorithm. HMACs are pretty popular in circumstances where you'd be doing a bunch of hashing anyway (e.g., Transport Layer Security cipher suites, which we'll talk about later).

Password hashing

Resisting *offline* dictionary attacks*

Rainbows† and salt

Iterative password hashing (KDFs)

* see, e.g., John the Ripper

† Oeschslin, "Making a Faster Cryptanalytic Time-Memory Trade-Off", CRYPTO 2003: Advances in Cryptology - CRYPTO 2003, 2003. DOI: 10.1007/978-3-540-45146-4_36.

We don't need *any* cryptography to resist an online dictionary attack. Protecting password databases is, instead, all about resisting _____, where an adversary has gained access to a password database and they want to get passwords from it. Without any cryptography, they can simply do a database lookup. With cryptography, however, we can make things much harder for them.

As a (very bad!) alternative to password hashing, check out this analysis of a major password database breach at Adobe.

Tools like GPUs are really good at parallel computation. Attackers can use them to try lots and lots of passwords concurrently to see if they can find the correct one (a bit like the Bombes in Bletchley Park!). _____ (KDFs) make life harder for an attacker by forcing computation to be _____. There is a cost for the user, too, but it's insignificant compared to the benefit of not having your password cracked when a business suffers a data breach!

What makes a good password?

Hard to guess

Complex?

Fundamentally, it should be hard for an attacker to guess a password.

Q: what's some common password guidance you've been given over the years?

We have ideas about what makes guessing harder: not using common words, maybe making passwords long, maybe using funny symbols. Some of these ideas are intuitive, others have been _____. But *why* would those things make it harder to guess a password?

Before talking about sensible password policies, we need to understand just a little bit of information theory. One way of describing hard-to-guess-ness — but one which is often misunderstood — is the information-theoretic concept of *entropy*.

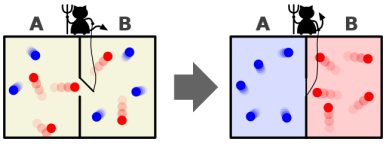
Entropy

A measure of information

- or disorder, or chaos...
- thermodynamics: Maxwell's demon
- units: *Shannons* (a.k.a., bits!)

The concept of entropy is much older than computing: entropy has been used for a long time in thermodynamics to describe the disordered-ness of systems. In a closed system, entropy is a monotonically non-decreasing quantity, i.e., left alone, a closed system will become less ordered and more chaotic. The "heat death of the universe" doesn't refer to things getting really hot, it refers to an increase of random motion to the point that there is no structure, no separation between hot and cold, so no useful work can be done.

Maxwell's demon refers to a theoretical idea that links thermodynamics and information theory. Normally, if you bring hot and cold things together, their temperature evens out. However, if you could control the interface between two chambers of gas such that you open the door for faster molecules going right and slower molecules going left, but close the door for other molecules, you could make lukewarm gas turn into hot gas in one chamber and cold in the other. Would this violate the second law of thermodynamics? No, because the _____ affects the entropy!



Measuring entropy

Shannon entropy:

$$H(\mathbf{X}) = -\sum_{i=0}^{n} P(x_i) \log_b P(x_i)$$

Hartley function:

$$H_0(\mathbf{X}) = \log_b |\mathbf{X}|$$

Guessing entropy, which is more directly relevant to password guessing, is a bit harder to calculate, but it is bounded by Shannon entropy. See: Massey, "Guessing and Entropy", *Proc. IEEE Int. Symp. on Info. Th.*, 1994.

Claude Shannon defined one way of measuring entropy, based on how _____ specific values in a distribution are. In this definition, a distribution that includes only equally-likely values will have higher entropy than a distribution that has a different distribution.

Important note: _____

The Hartley function can be used to compute Shannon entropy _____ _____.

Estimating password entropy

Eight random alphanumeric characters:

$$H_0(\mathbf{X}) = \log_b |\mathbf{X}| = \log_2 \left|36^8\right| = 41.6 \text{ Sh (bits)}$$

Four diceware* words + two numbers:

$$H_0(\mathbf{X}) = \log_2 \left|\left(6^4\right)^4 \times 10^2\right| = 4 \times \log_2 \left|6^4\right| + \log_2 \left|10^2\right| = 48.0 \text{ Sh}$$

But people don't choose random passwords!

---

* See EFF instructions plus Bonneau, "Deep Dive: EFF's New Wordlists for Random Passphrases", EFF, 2016.

In this example, we're using 26 possible letters plus 10 digits, so a total of 36 symbols that can be used. Eight of these symbols means that there are 36^8 possible passwords that can be formed with this scheme; if they are all _____, we can use the Hartley function to calculate the entropy of this _____.

These calculations show how we can work with distributions that contain multiple components, e.g., multiple words or multiple classes of characters. It's worth noting, however, that the entropy of the distribution of passwords derived from just four diceware words would be:

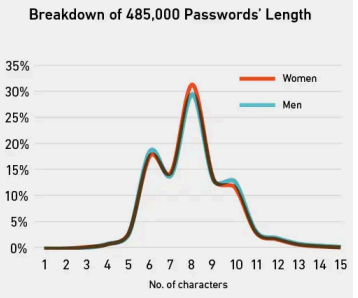
$$H_0(\mathbf{X}) = \log_2 \left|\left(6^4\right)^4\right| = 4 \times \log_2 \left|6^4\right| = 41.4 \text{ Sh}$$

So, four diceware words will stand up to a brute-force attack about as well as a random eight-character alphanumeric password. But which will be easier for a human to remember?

Calculating the entropy of random distributions isn't so hard. Unfortunately, however, people typically don't use randomly-generated passwords (although they should!).

Actual password entropy

People don't choose random passwords!



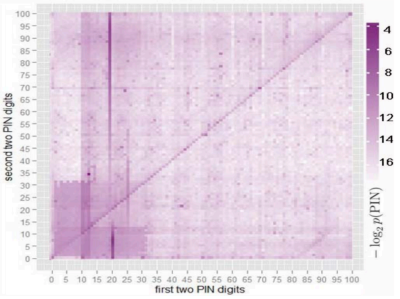
Source: Unmasked: What 10 million passwords reveal about the people who choose them

Actual password entropy

People don't choose random passwords!

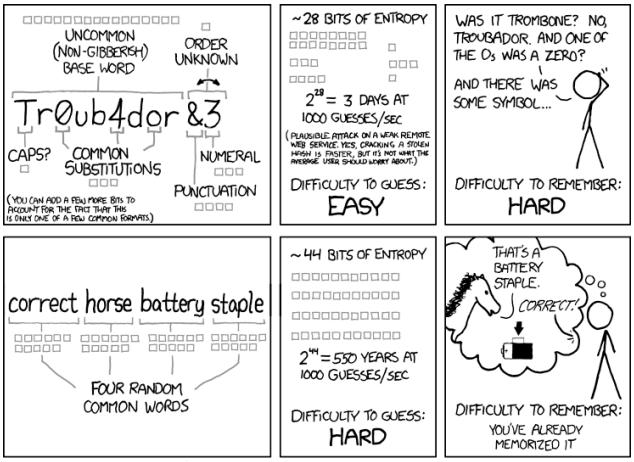
Or even random PINs!

- diagonal line: repetition
- lots of 19xx and even 20xx



Wang et al., "Understanding Human-Chosen PINs: Characteristics, Distribution and Security", in *ASIA CCS '17*, 2017. DOI: 10.1145/3052973.3053031.

This graphic was taken from PINs that were found as a subset of the password in the original RockYou data set. For those curious about "RockYou 2021": here's a nice writeup.



Source: XKCD 936

XKCD, although it's a web comic, often has spot-on analysis. Comedy and satire can make a point in a punchier way than a prosey explanation, and this is no exception!

Entropy vs password strength

- Shannon, Hartley entropies have clear definitions
- Other entropies: min-entropy, guessing entropy...
- Entropy a measure of a *distribution*, not a single password
- Can estimate password entropy **assuming random selection**
- One website with one breach... haveibeenpwned.com

What can we learn from this?

If we assume that _____, we can compute the entropy of _____. You should note that's exactly how I've phrased one of the questions in Assignment 2!

We can learn things about actual user password choices, etc., from password databases that have been leaked. We can also learn whether or not particular passwords have been compromised!

TODO: show https://pics.me.me/create-a-new-account-spacemoses1337-password-is-being-used-by-53149196.png

If a password has been leaked in the clear, _____ _____! Perhaps they weren't hashing, perhaps they weren't salting, perhaps they were allowing terrible password hints, but whatever it was, somebody should lose their Internet License!

More password guidance

(Modern) NIST guidance	Common guidance
No hints	Allow "rhymes with assword"
No KBA	Non-password passwords
Screen for compromises	???
Careful about 2FA	SMS FTW

... which will be the topic of our next lecture (2FA)