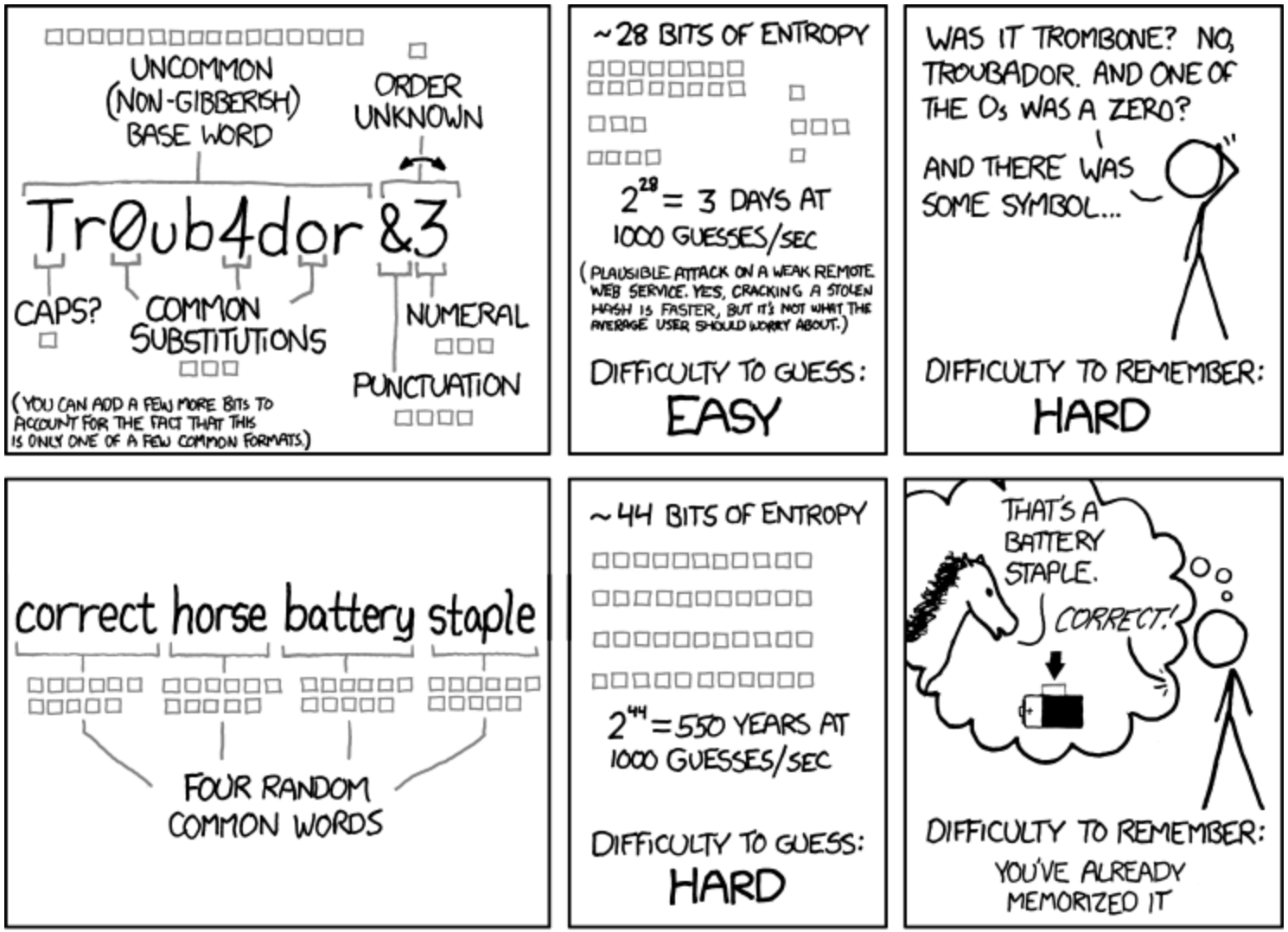
Security and Cryptography

Last year's security and privacy lecture focused on how you can be more secure as a computer *user*. This year, we will focus on security and cryptography concepts that are relevant in understanding tools covered earlier in this class, such as the use of hash functions in Git or key derivation functions and symmetric/asymmetric cryptosystems in SSH.

Entropy

- Measure of randomness
- Useful for determining password strength
- Measured in *bits*
- Example: A fair coin flip gives 1 bit of entropy



https://missing.csail.mit.edu/

Quantifying Entropy

- Entropy is `log_2(number of possibilities)`
- For a 6-sided die: ~2.58 bits of entropy
- Attacker knows the *model* of the password, not the randomness
- Online guessing: ~40 bits of entropy is good
- Offline guessing: 80 bits or more for stronger passwords

Hash functions

- Maps data of arbitrary size to a fixed size
- Example: SHA1

```
$ printf 'hello' | sha1sum
aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
```

Properties of Hash Functions

- Deterministic
- Non-invertible
- Target collision resistant
- Collision resistant

Note: SHA-1 is no longer considered strong. Use formal training for recommendations.

Applications of Hash Functions

- **Git**: For content-addressed storage
- **File Integrity**: Short summary of file contents for verification
- **Commitment Schemes**: Commit to a value, reveal it later

Key derivation functions (KDFs)

- Used for producing keys from passphrases
- Deliberately slow to thwart brute-force attacks

Applications

- Producing keys for cryptographic algorithms
- Storing login credentials with a salt

Symmetric cryptography

```
keygen() -> key

encrypt(plaintext, key) -> ciphertext
decrypt(ciphertext, key) -> plaintext
```

- Encrypting files for storage
- Example: AES

Asymmetric cryptography

```
keygen() -> (public key, private key)

encrypt(plaintext, public key) -> ciphertext
decrypt(ciphertext, private key) -> plaintext

sign(message, private key) -> signature
verify(message, signature, public key) -> bool
```

Applications

- PGP email encryption
- Private messaging with Signal or Keybase
- Signing software

Key distribution

- **Asymmetric-key cryptography** challenge: public key distribution
- **Solutions**:
  - Trust on first use
  - Web of trust
  - Social proof

Case studies

Password managers

- Use unique, high-entropy passwords
- Encrypted with a symmetric cipher
- Examples: KeePassXC, pass, 1Password

Two-factor authentication

- Combines something you know (passphrase) with something you have (e.g., YubiKey)

Full disk encryption

- Protect data if your laptop is stolen
- Use cryptsetup + LUKS, BitLocker, or FileVault

Private messaging

- Use Signal or Keybase for end-to-end security

SSH

- `ssh-keygen` generates an asymmetric keypair
- Public key stored in `.ssh/authorized_keys`
- Private key encrypted on disk
- Identity proven using asymmetric signatures

Resources

- Last year's notes
- Cryptographic Right Answers