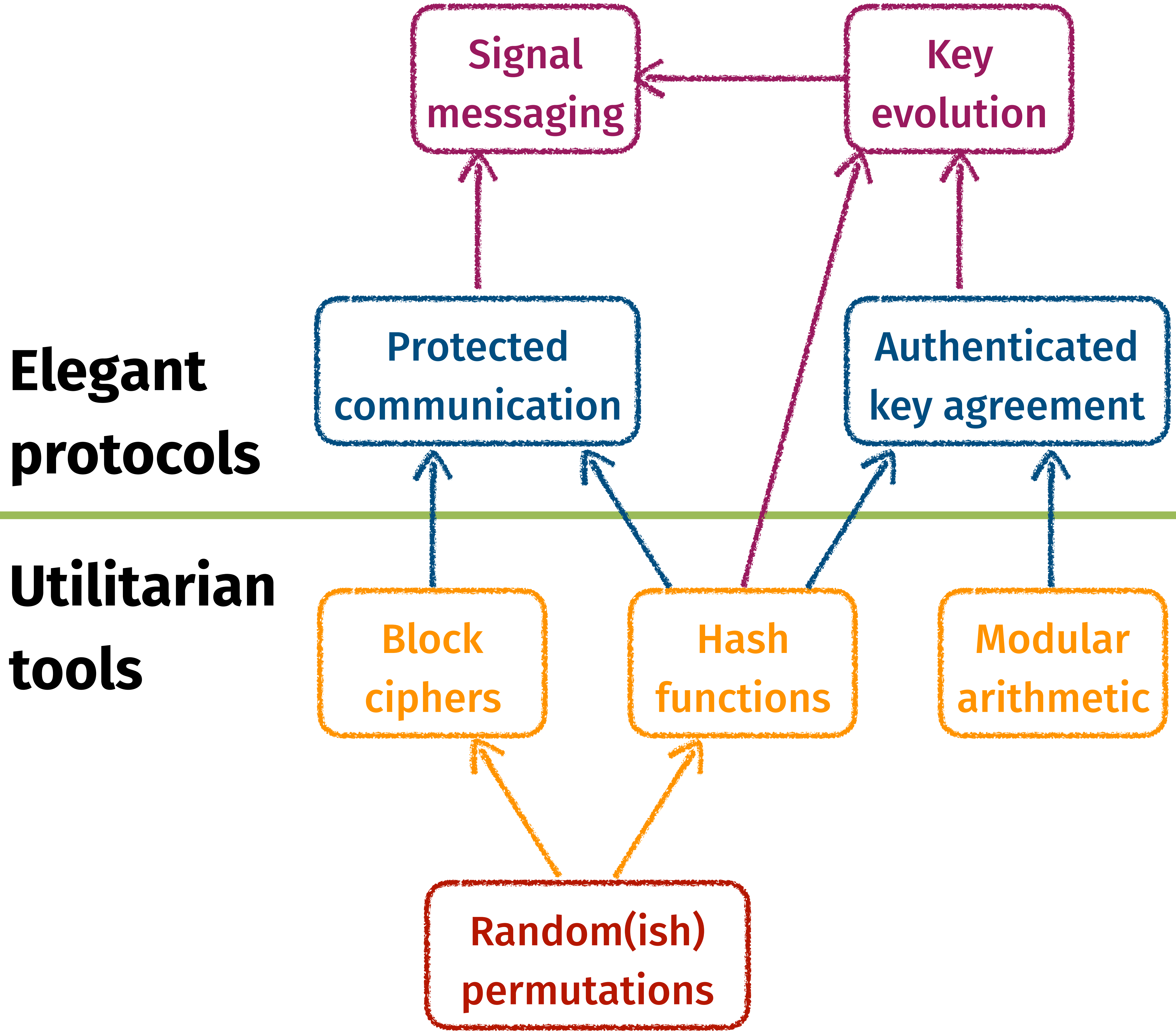
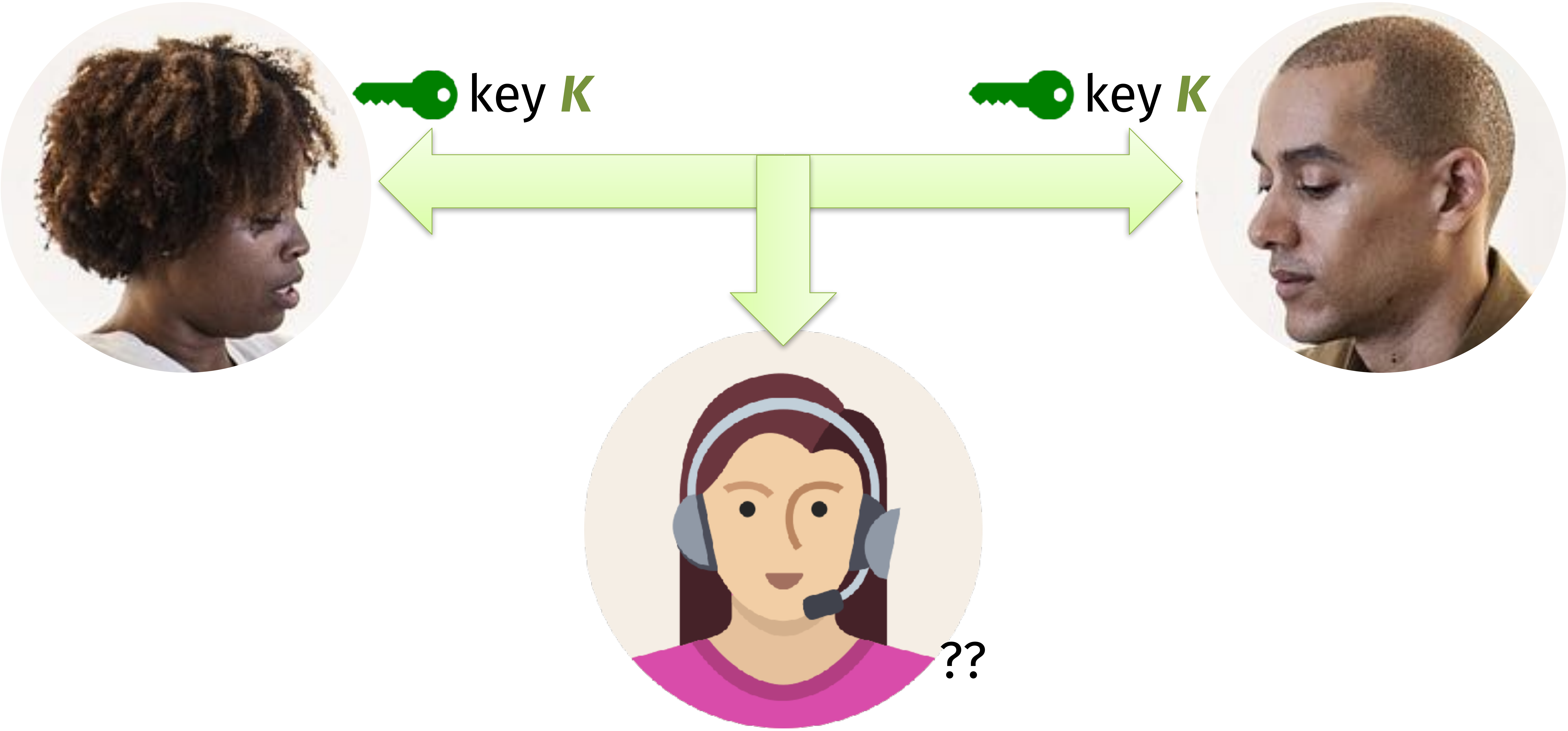
Lecture 17: Hashing, revisited

- Lab 10 will be posted today, due *Wednesday* 4/24 at 11pm
  - You will be able to answer the first 2 questions after today's lecture
  - Question 3 will depend on Thursday's lecture
- Lab 11 will be posted Tuesday 4/23 and due Wednesday 5/1
- Reminder: my office hours have moved to Thursdays at 11am-1pm

Course roadmap

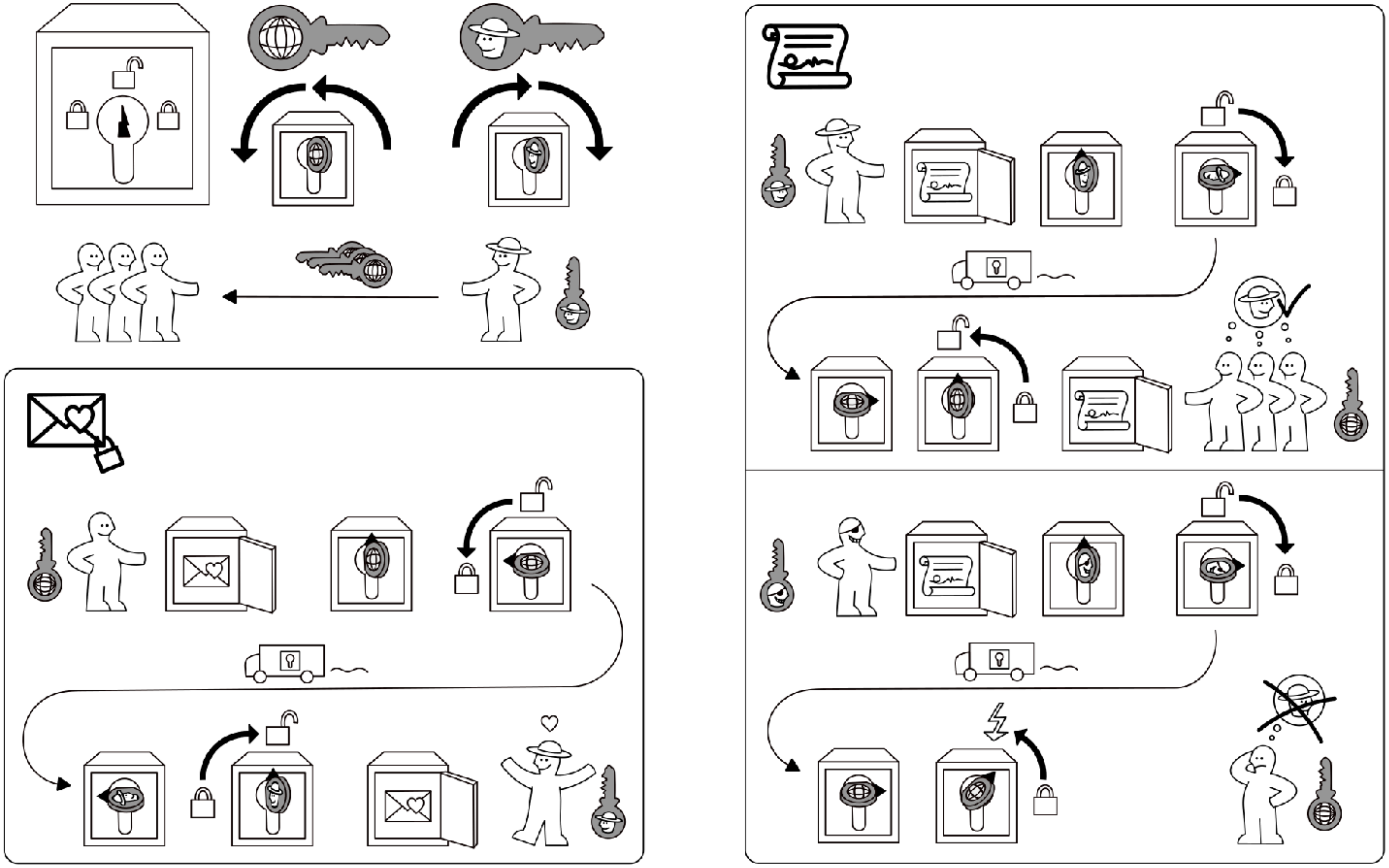


Part 3: Generate, exchange, evolve, and delete keys



PUBLIC KEY KRÜPTO

idea-instructions.com/public-key/

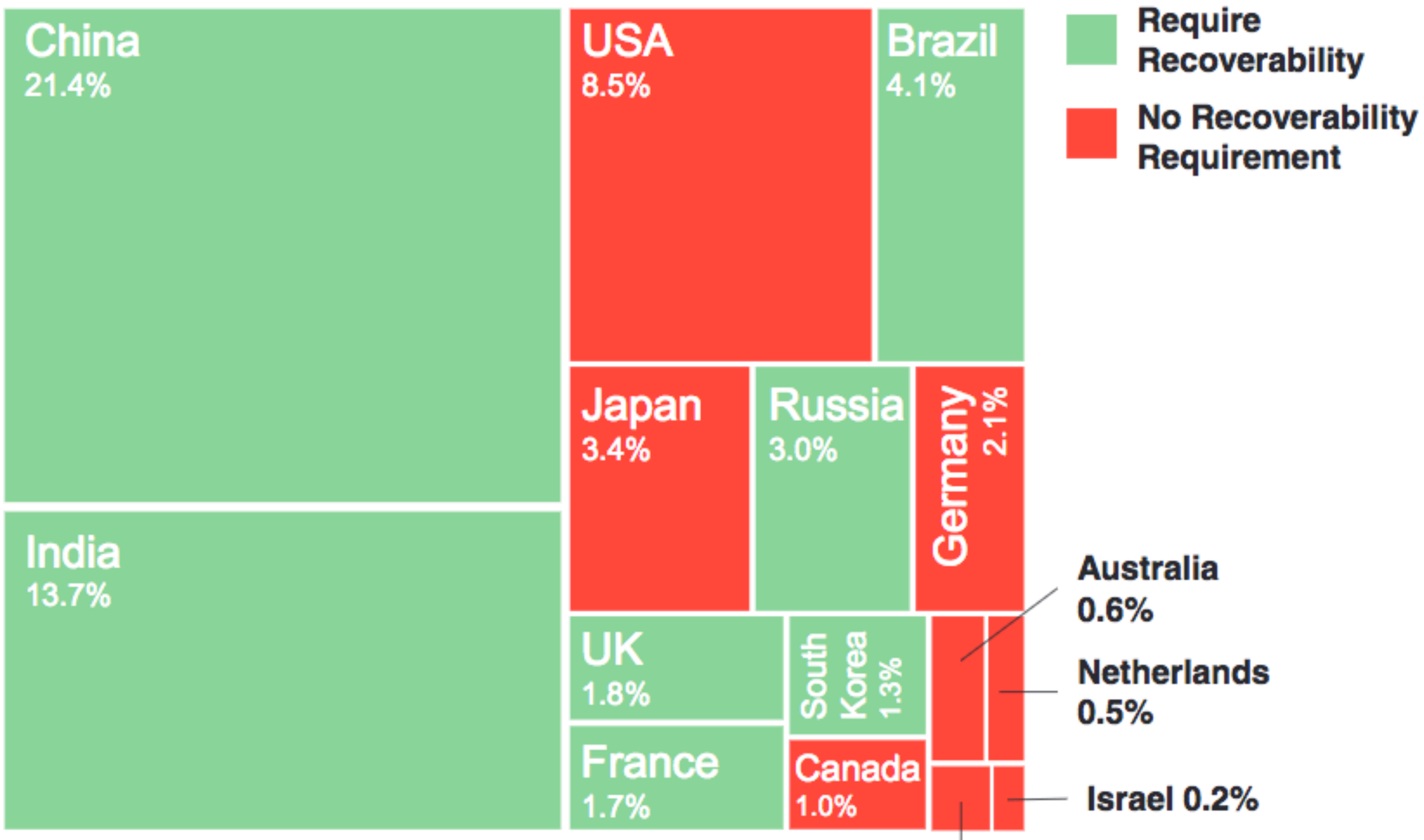


Randomness ⇒ Unpredictability ⇒ Secrecy

Part 4: Cryptography meets the law

- You can be compelled to provide access, subject to 4th + 5th Amendments
- Technology provider can be compelled to provide access

Figure 4.1. Share of Global Internet Users in Select Countries and Recoverable Encryption Requirements



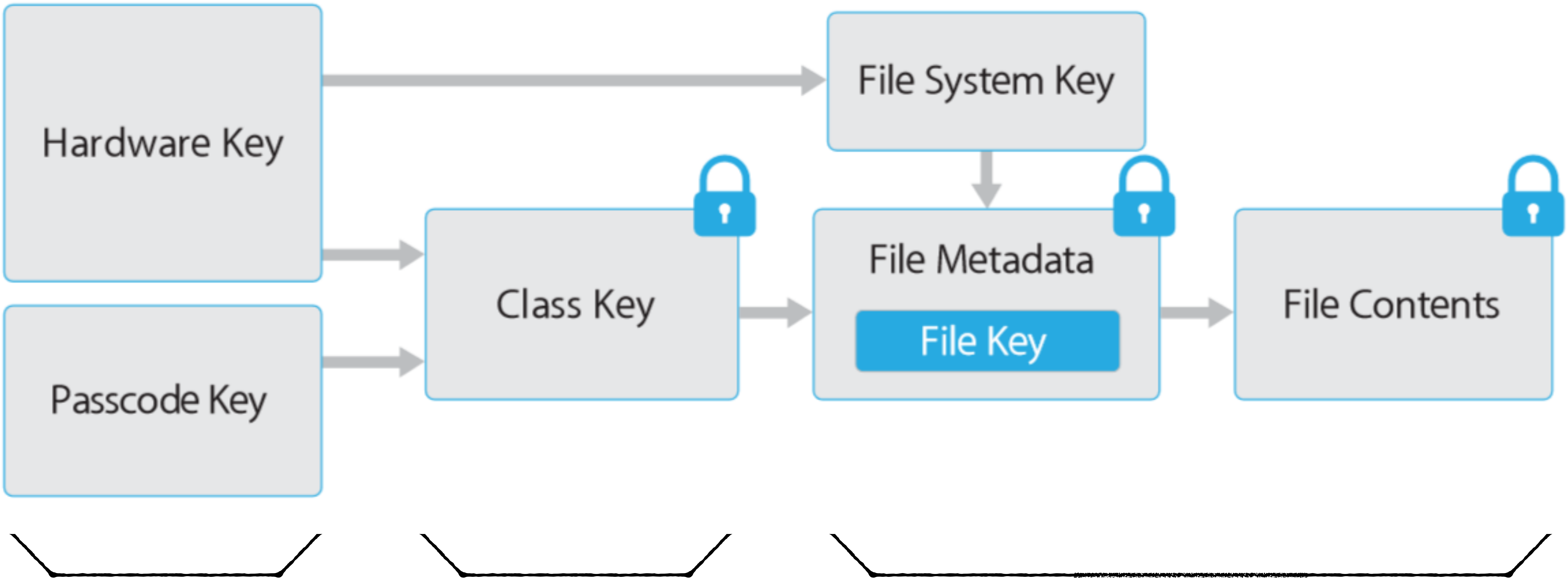
Source: csis.org/analysis/effect-encryption-lawful-access-communications-and-data

Auguste Kerckhoffs' principles to protect communication

1. The system must be practically, if not mathematically, *indecipherable*
2. It should *not require secrecy*, and it should not be a problem if it falls into enemy hands
3. It must be possible to communicate and ***remember the key*** without using written notes, and correspondents must be able to *change or modify it at will*
4. It must be applicable to telegraph communications
5. It must be portable, and should not require several persons to handle or operate
6. Lastly, given the circumstances in which it is to be used, the system must be ***easy to use*** and should not be stressful to use or require its users to know and comply with a long list of rules

Source: *La Militaire*, 1883

Reminder from Part 1: Data at rest protection



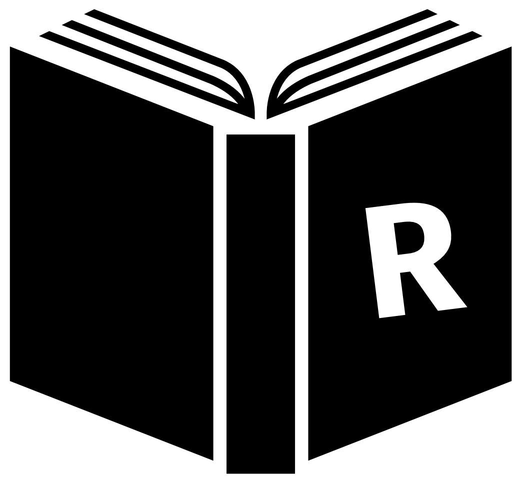
Generate master key from things that you know, are, and have (rather than storing it)

Derive keys that live only for limited time

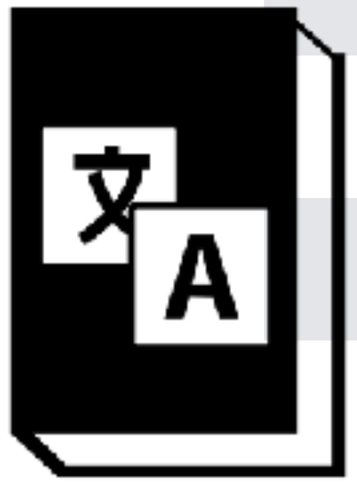
Use these to wrap a unique key for each file

Hash function = 1 public codebook

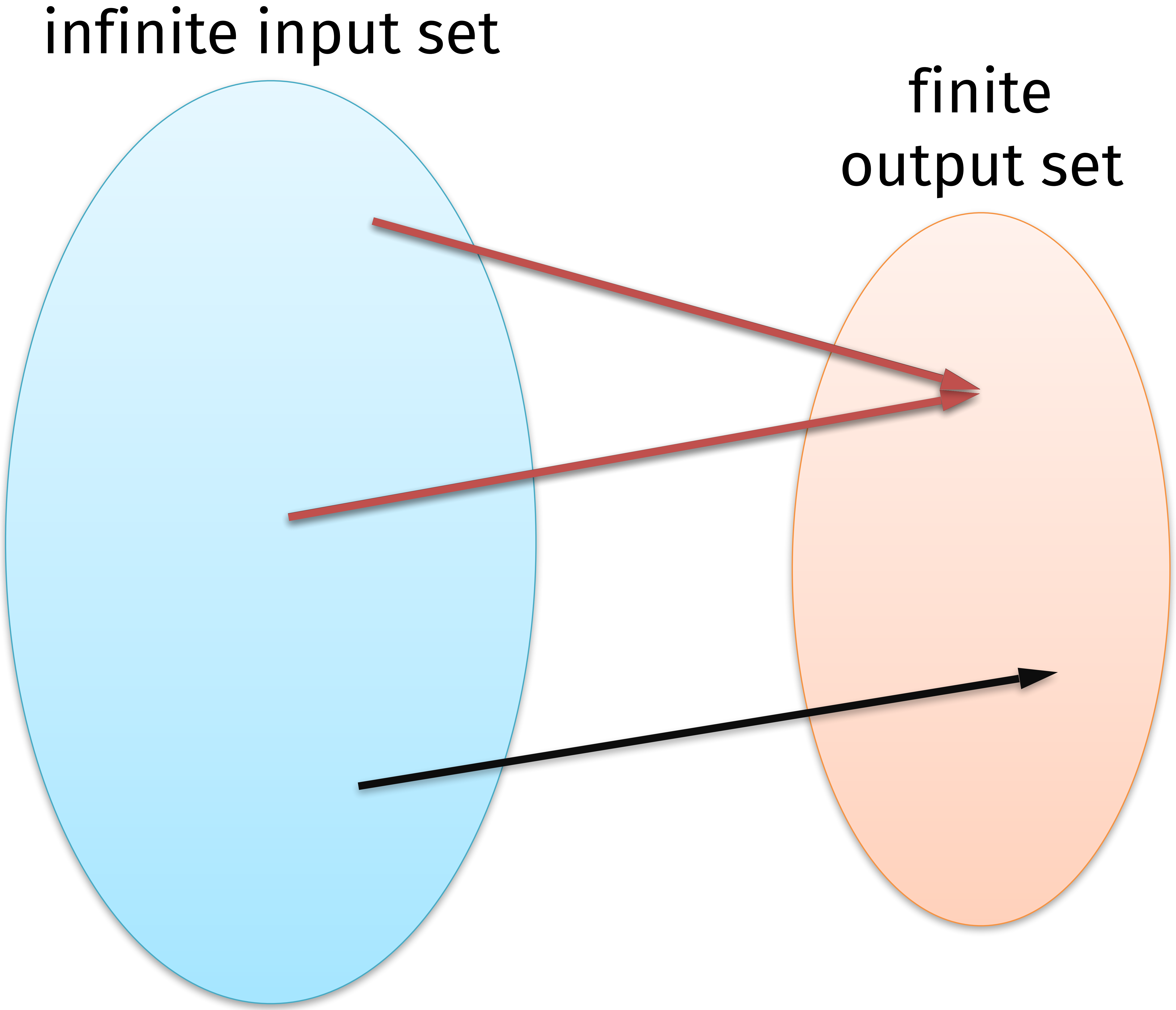
- Hash function $H : \{0,1\}^\infty \rightarrow \{0,1\}^{out}$
- Compresses long messages into short digests
- Most popular example in use today: SHA-256
- *Random oracle* is an ideal public codebook



X	*Y*
aba	nr
abs	mb
ace	yd
act	wv
add	je
ado	hg
aft	uv
age	zm
ago	ds
aha	ae
aid	kf
⋮	⋮
zip	cy
zoo	dx



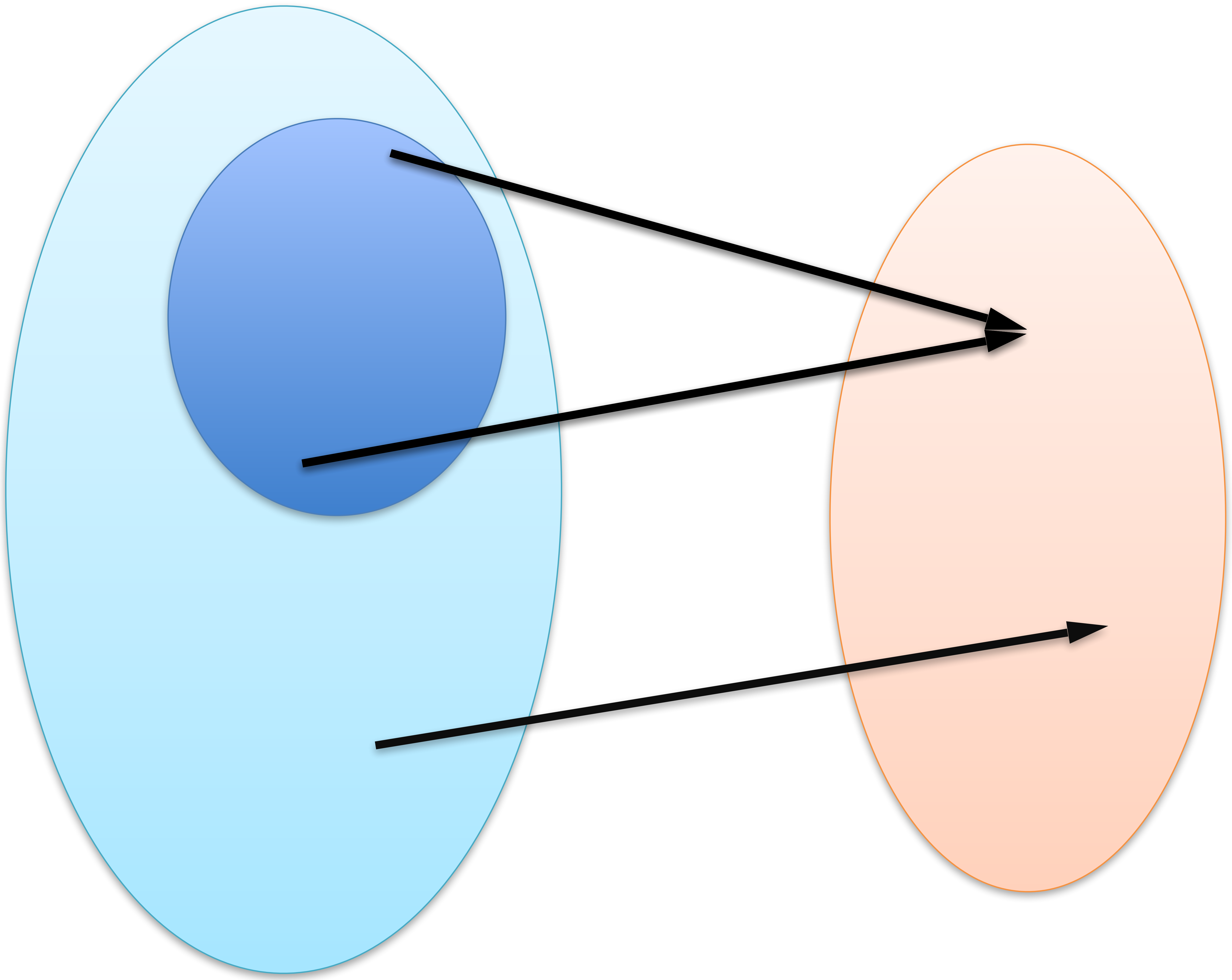
Hash function



- Cannot invert! Given y = H(x) for randomly chosen x, tough to find any preimage x’

- Cannot collide! Given only H, difficult to find two messages with the same digest

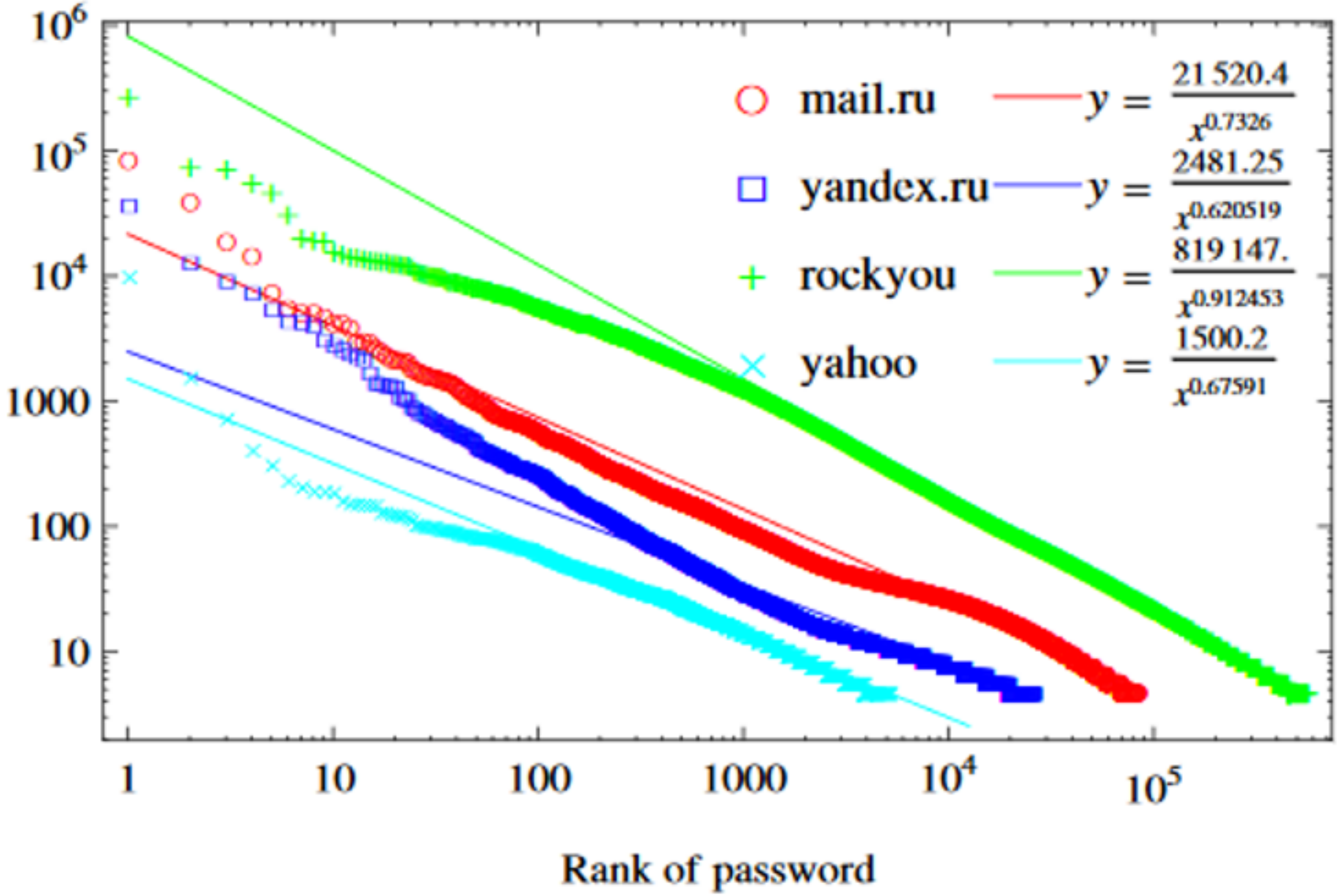
Password hashing



- Can use to protect passwords!
  - Don't store pwd
  - Instead store H(pwd)
- Collision resistance → all incorrect password guesses will have different hashes
- Problem: preimage resistance only applies to random x, and passwords are anything but random…

Concentration of passwords

we estimate that passwords provide fewer than 10 bits of security against an online, trawling attack, and only about 20 bits of security against an optimal offline dictionary attack.



- Passwords tend to follow a Zipf's law distribution

  $Pr[k^{\text{th}}$ most common password$] \propto \frac{1}{k}$

- *Countermeasure:*
  If # passwords is small, then we must make the time to compute each hash large

Sources: eprint.iacr.org/2014/631.pdf and jbonneau.com/doc/B12-IEEESP-analyzing_70M_anonymized_passwords.pdf

Password-based key derivation function

- Threat we are trying to mitigate: a well-funded attacker who either
  - Brute forces the (not too large) password space
  - Obtains your personal phone or organization's `/etc/passwd` file
- Initial approach: generate key on the fly, don't write it down anywhere
- Crypto primitives
  - PBKDF2: NIST standard, requires substantial CPU time to compute
  - Recently, a new wave of hash functions (scrypt, bcrypt, argon2) attempt to reduce parallelization by requiring substantial CPU + RAM to run

PBKDF2 inputs

Data

- P: Password
- S: Salt, aka nonce
- L: Output length, in blocks
- C: Iteration count

B(K, M): keyed pseudorandom function like a block cipher or HMAC (just as we used in Signal)

PBKDF2 construction

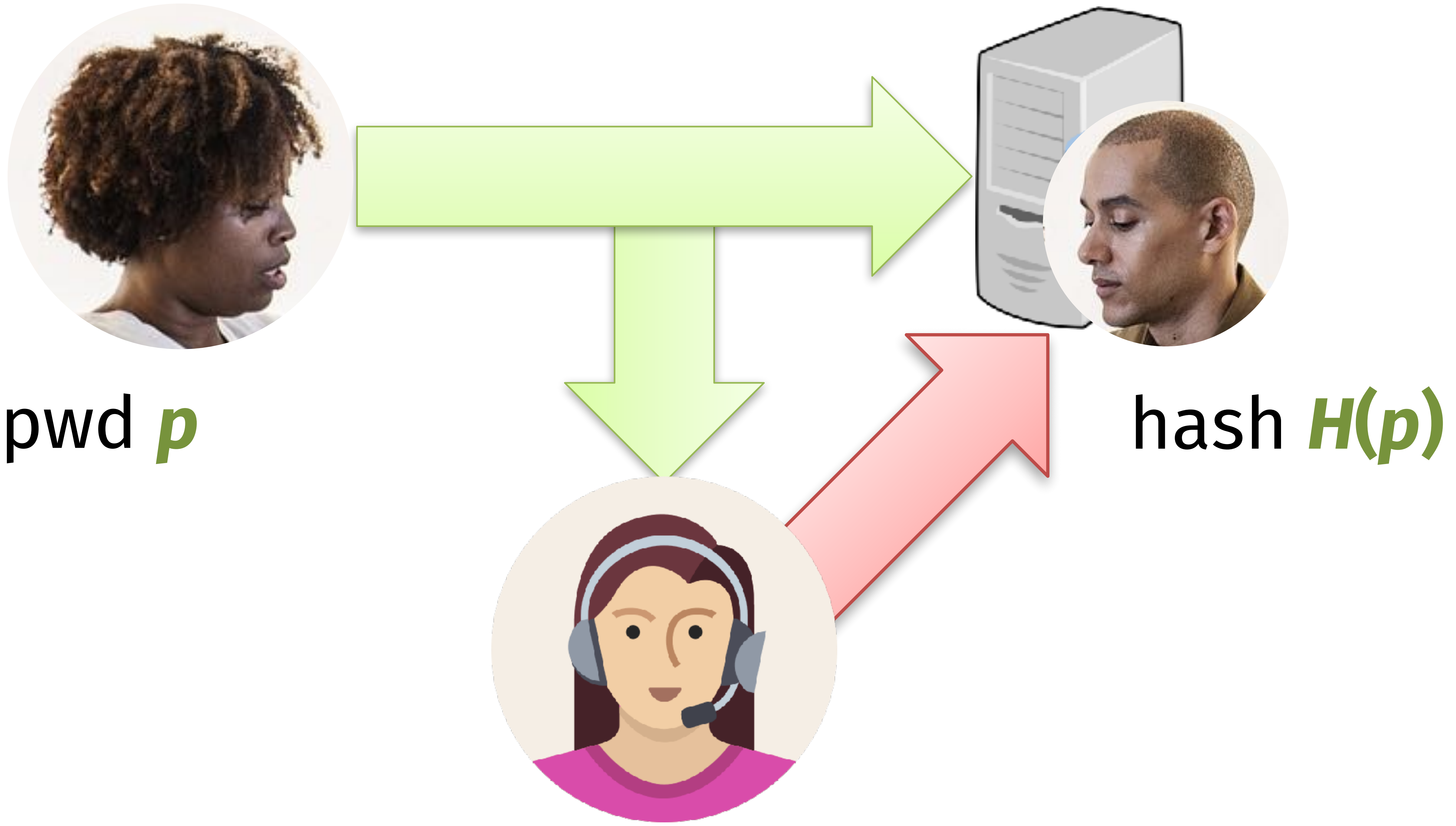
- Output $T_1 \,||\, T_2 \,||\, \ldots \,||\, T_L$
  - Definitely L blocks of something!
- Each block $T_i = U_1 \oplus U_2 \oplus \ldots \oplus U_\mathrm{C}$
  - Must compute all U_j to learn T_i
- For each i, initial $U_1 = S \,||\, i$
  - Depends on seed
- Subsequent $U_\mathrm{j} = B(\mathrm{P}, U_{\mathrm{j}-1} \,||\, \mathrm{i})$
  - Must compute sequentially

The limits of introversion

- Key derivation functions like PBKDF2 are the "best possible" solution for a non-interactive login process, like to your own laptop or phone
  - They eliminate any 1-bit "is pwd correct?" check; instead use key to encrypt device
- But they are not perfect! An attacker with enough CPU/RAM can eventually find the password, especially if the pwd is easy to guess
- We call this attack an *offline dictionary attack*
  - Dictionary attack → attacker finds a dictionary of common passwords to guess
  - Offline → attacker can conduct this attack on *her own machine*, and then only make 1 guess on the real victim's machine
  - (Note: can try to use hardware to localize computing to target device)

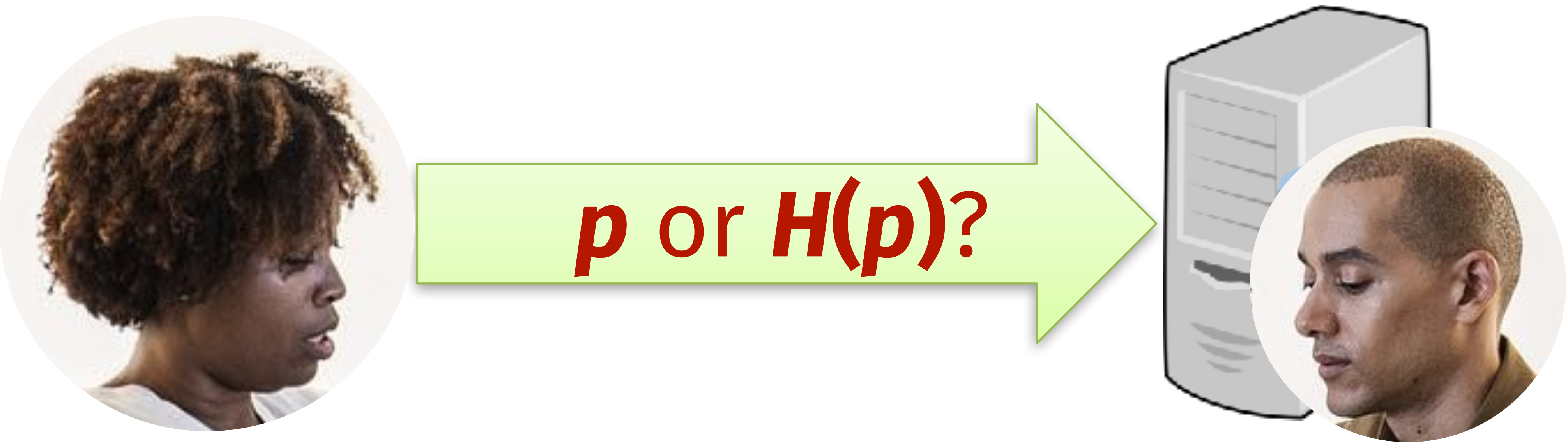
The power of interaction

- Countermeasure: make each guess require network communication, and eventually cut off the attacker if she makes too many attempts
- Here, we say that the attacker must run *online dictionary attack*
- Even if Mallory compromises the contents of Bob's hard drive, then security reverts to an offline dictionary attack



The necessity of interaction

password $\boldsymbol{p}$



database of $\boldsymbol{H(p)}$,
where $\boldsymbol{H}$ = pbkdf2, etc

- If Alice wants to authenticate to bob.com, does she send p or $H(p)$?
- Alice sends $H(p)$ $\rightarrow$ the stored hashed database is very sensitive
- Alice sends p $\rightarrow$ the transmission itself is very sensitive

KrebsonSecurity
In-depth security news and investigation

21 MAR 19 **Facebook Stored Hundreds of Millions of User Passwords in Plain Text for Years**

Hundreds of millions of **Facebook** users had their account passwords stored in plain text and searchable by thousands of Facebook employees — in some cases going back to 2012, KrebsOnSecurity has learned. Facebook says an ongoing investigation has so far found no indication that employees have abused access to this data.

“**Cryptography** is how people get things done when they need one another, don’t fully trust one another, and have adversaries actively trying to screw things up.”

–Ben Adida

Source: benlog.com/2018/01/07/crypto-as-in-crypto/