Introduction to Information Security

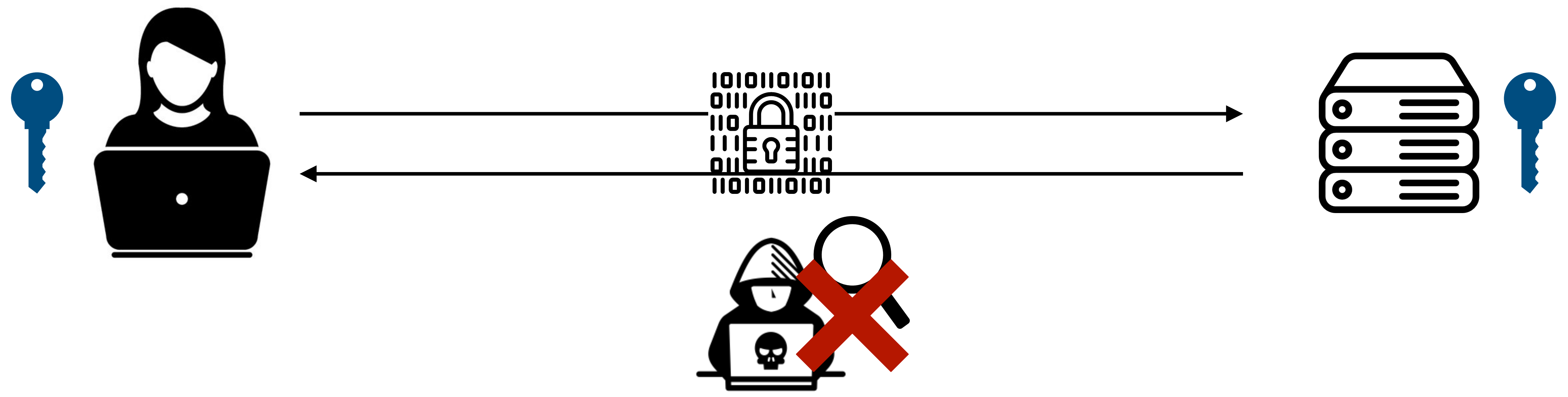
5. Message Integrity

Kihong Heo

KAIST

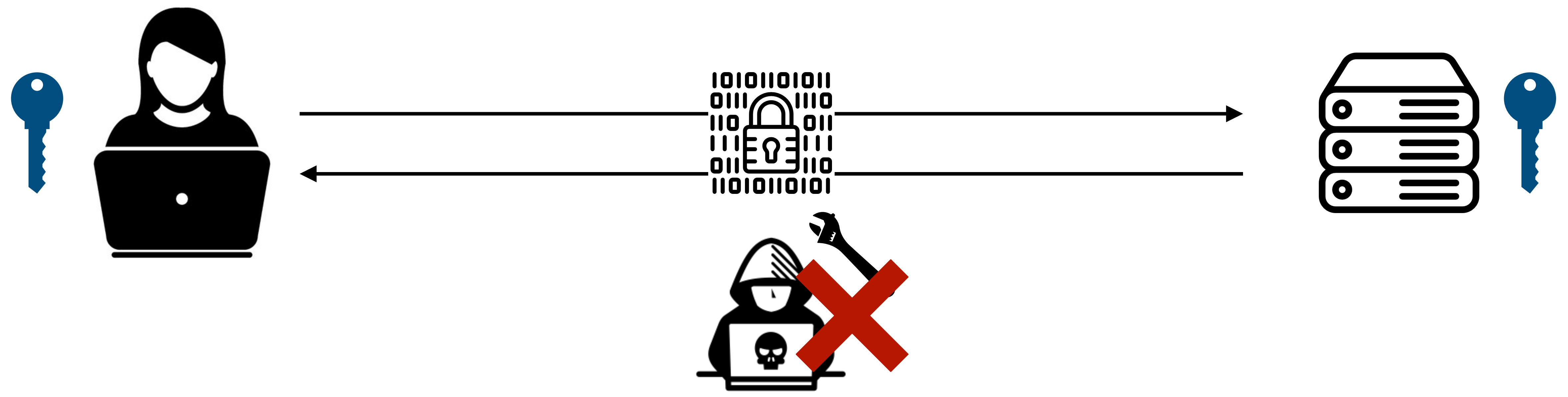
Confidentiality

- Goal: Attackers cannot learn anything about the plaintext w/o the key



Integrity

- Goal: Attackers cannot generate a valid message

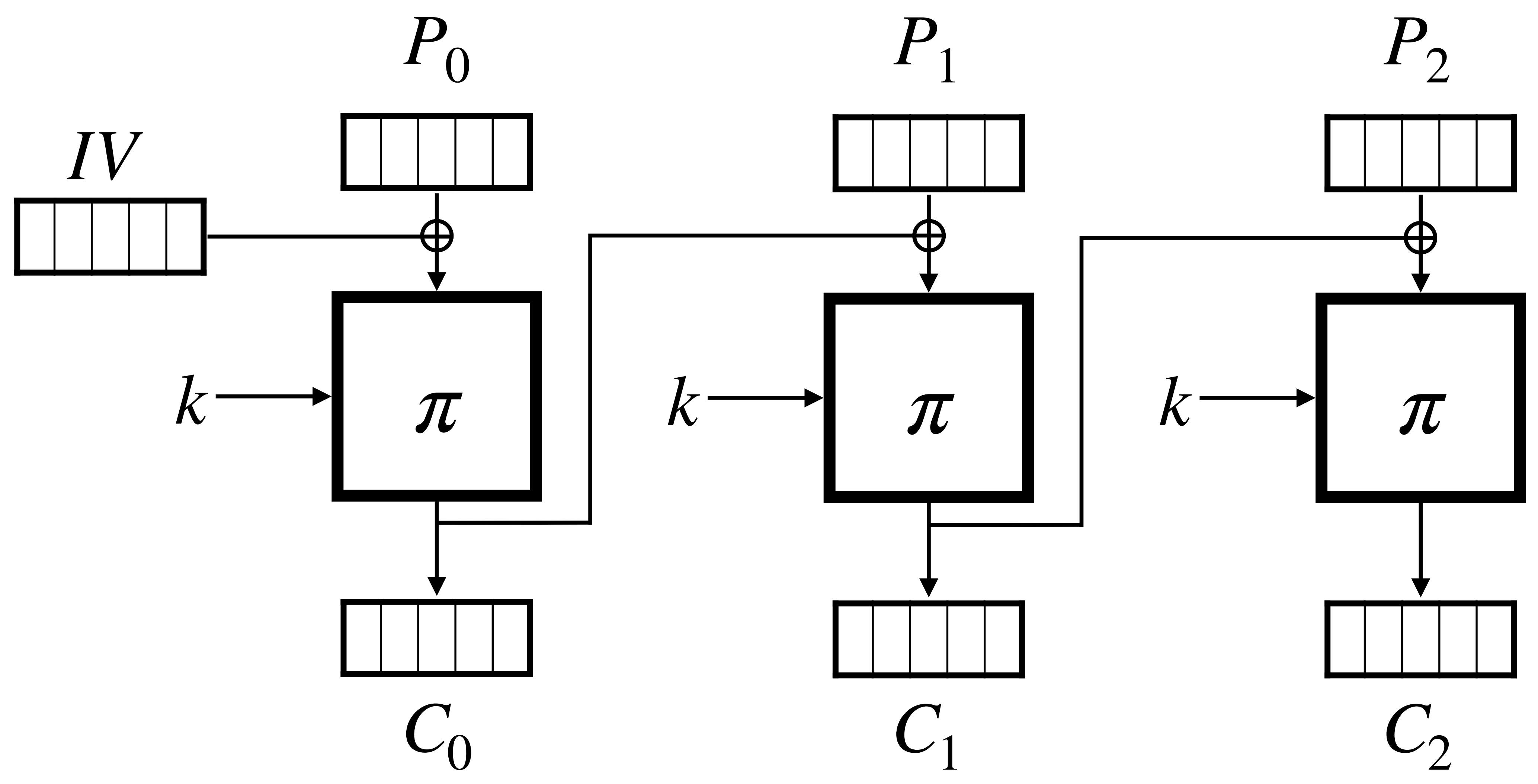


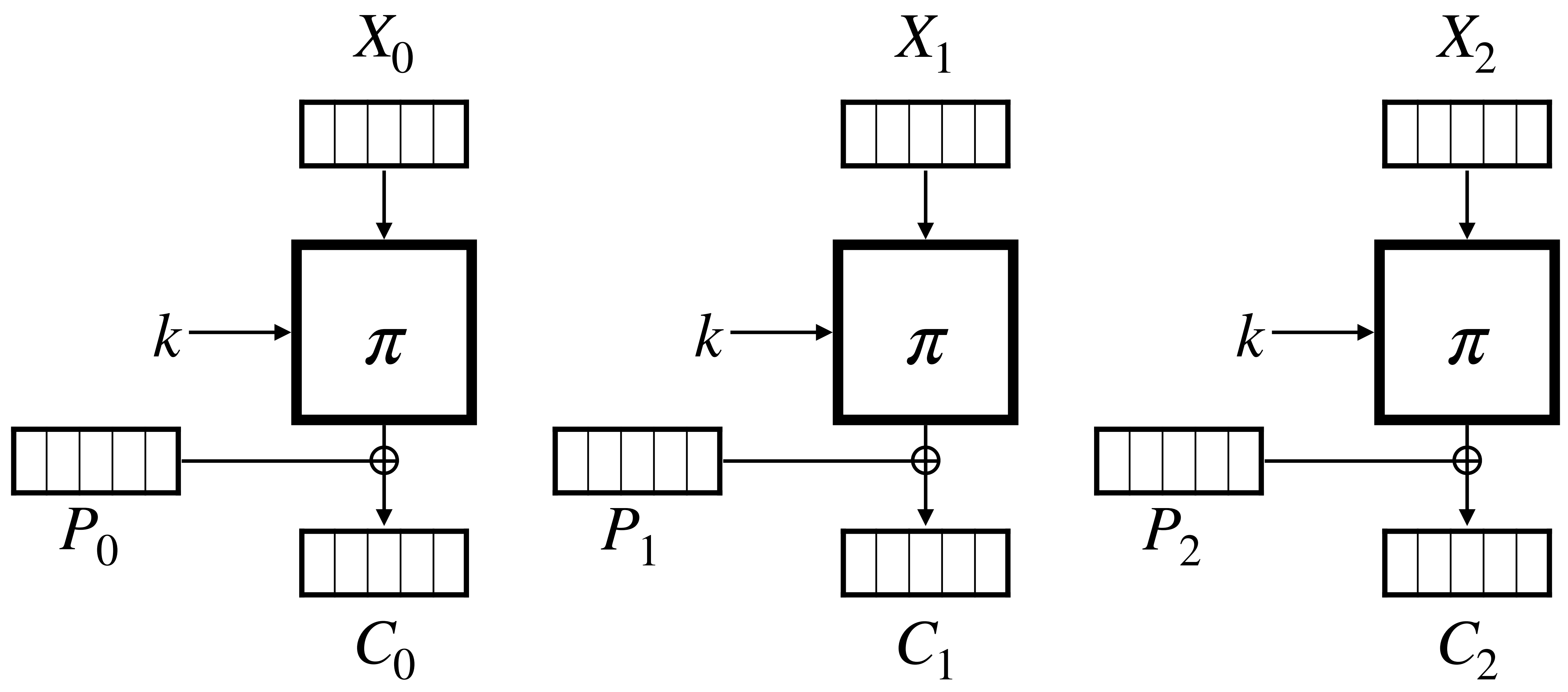
Confidentiality vs Integrity

- What does “secure communication” entail?
- Confidentiality: secret communication
  - Ensured by **encryption** schemes
- Integrity: authenticated communication
  - Ensured by **authentication** schemes
- In many cases, message integrity is equally (or more) important
  - Any examples?

Encryption vs Authentication

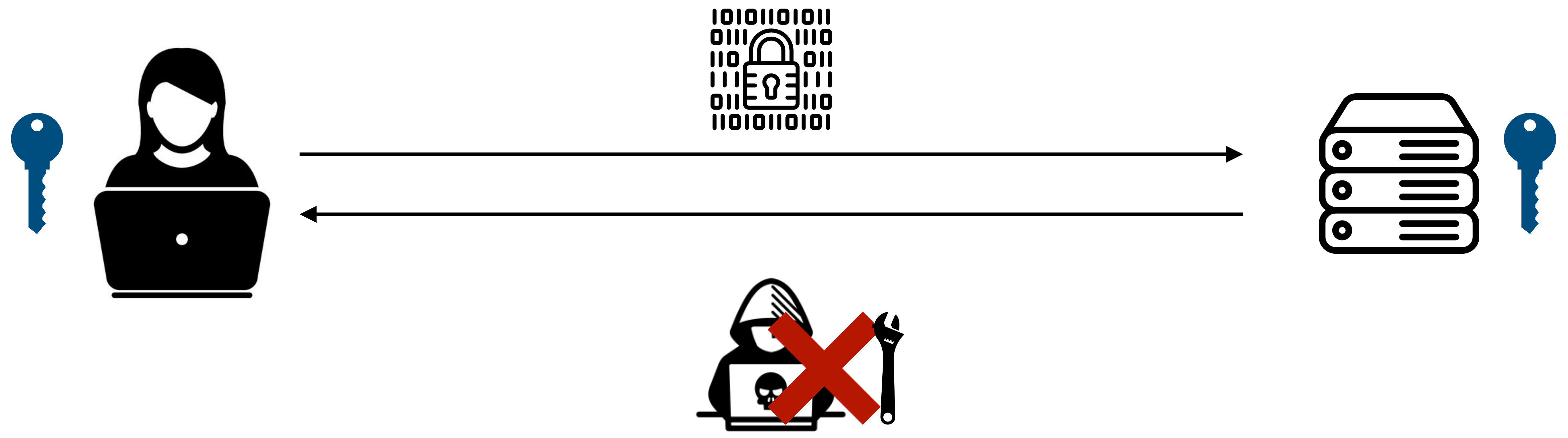
- “Encryption hides message contents and thus adversary cannot modify the encrypted message in any meaningful way” [T / F]?
- Example:
    - CBC-mode encryption
    - CTR-mode encryption





Message Authentication Codes (MAC)

- "Cryptographic checksum" to prevent an adversary from modifying a message
- Assume k is the shared symmetric key
- Sender: send (m, t) where m is a message and $t = Mac_k(m)$
- Receiver: receive (m, t) and check whether $t = Mac_k(m)$
- Example: HMAC (hash-based MAC), CBC-MAC



Authenticated Encryption Scheme

- Goal: Attackers cannot learn anything about m and cannot modify m
- Secure encryption scheme: (Enc, Dec)
- Message authentication code: Mac
- IMPORTANT: each cryptographic primitive should always use **independent keys**
  - $k_E \neq k_M$
- How to achieve a secure authenticated encryption scheme?

Secure Authenticated Encryption Scheme

- Encrypt-and-authenticate: $c = Enc_{k_E}(m)$, $t = Mac_{k_M}(m)$
  - Not secure: t is deterministic because Mac is deterministic
- Authenticate-then-encrypt: $t = Mac_{k_M}(m)$, $c = Enc_{k_E}(m \| t)$
  - May or may not be secure (e.g., CBC-mode-with-padding)
- Encrypt-then-authenticate: $c = Enc_{k_E}(m)$, $t = Mac_{k_M}(c)$
  - Secure regardless of the choice of Enc and Mac
  - Common practice (e.g., TLS)

Cryptographic Hash Function

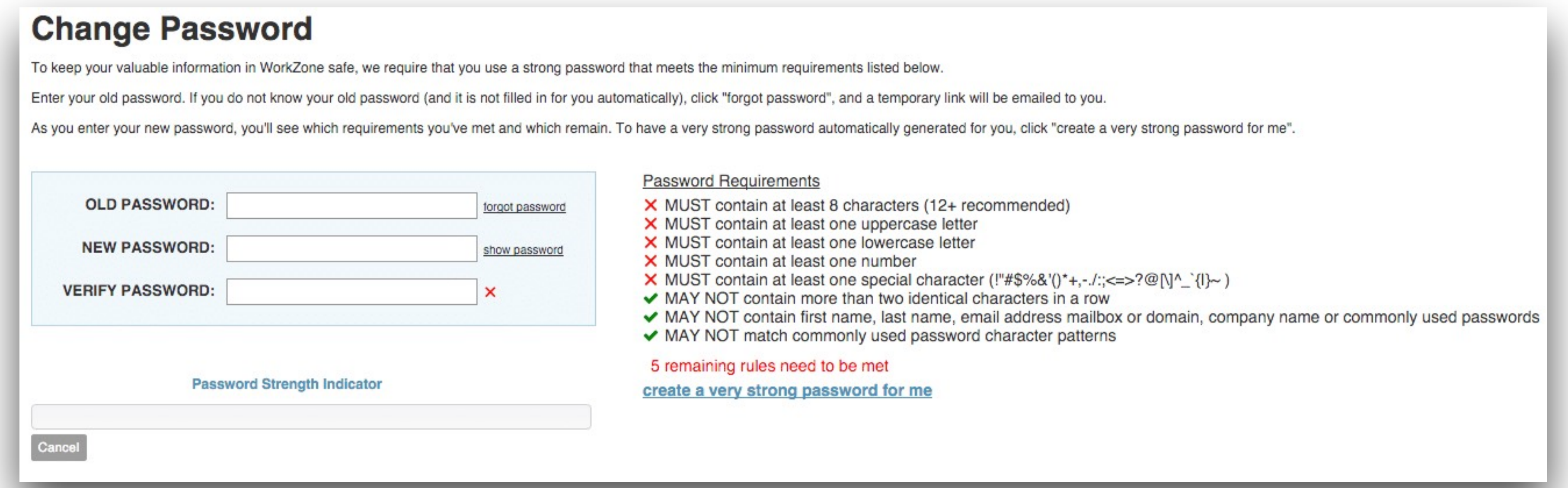
- The most common implementation scheme for MAC
- Hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^l$
    - Take inputs of arbitrary length
    - Compress them into short fixed-length (l) outputs
    - Efficient evaluation and public implementation
- Cryptographic hash function: varying properties required across applications
    - Preimage resistance, second preimage resistance, collision resistance
    - Example: MD5, SHA-1, etc
- $HMAC_K(m) = H((K' \oplus opad) \,||\, H((K' \oplus ipad) \,||\, m))$

1. Preimage Resistance

- Given y, computationally infeasible to find x such that $H(x) = y$.
  - So-called one-way property
- How much work is needed to break this resistance?
- Example:
  - Factoring: $H(x_1, x_2) = x_1 \times x_2$ where x_1, x_2 are prime numbers
  - Discrete logarithm: $H(x) = k^x \mod p$

Application: Password Hasing

- Passwords must not be stored in plaintext but hashed and stored
- BTW, why do we need strong password requirements?
- Pre-computation attack: password space is often limited (e.g., dictionary)
- Mitigations
  - Slow hash functions (e.g., bcrypt)
  - Salt: store (h, s) when $h = H(pw \,||\, s)$ and s is a short random string



Change Password

To keep your valuable information in WorkZone safe, we require that you use a strong password that meets the minimum requirements listed below.

Enter your old password. If you do not know your old password (and it is not filled in for you automatically), click "forgot password", and a temporary link will be emailed to you.

As you enter your new password, you'll see which requirements you've met and which remain. To have a very strong password automatically generated for you, click "create a very strong password for me".

OLD PASSWORD: forgot password

NEW PASSWORD: show password

VERIFY PASSWORD: ✗

Password Strength Indicator

Cancel

Password Requirements

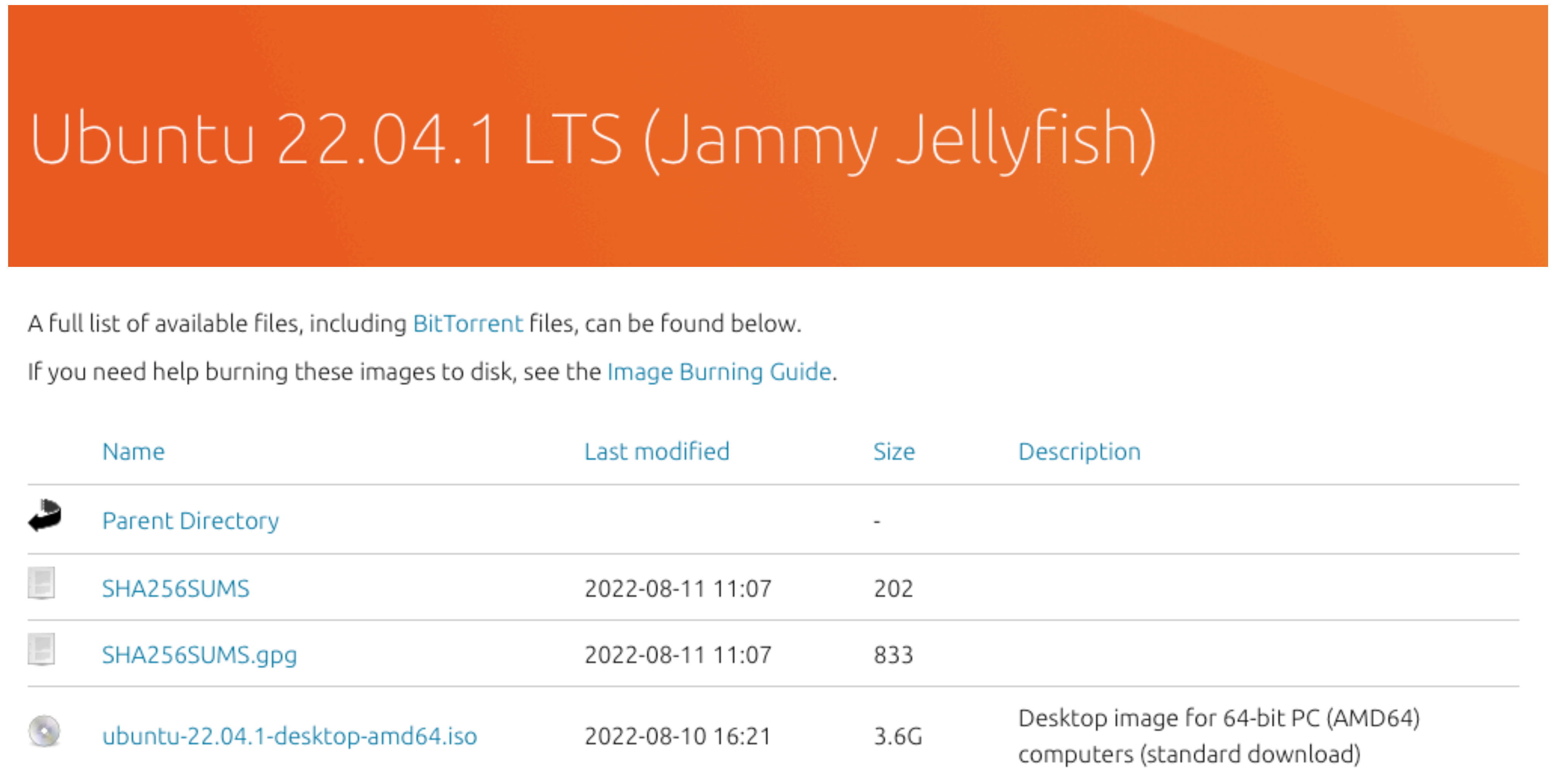
✗ MUST contain at least 8 characters (12+ recommended)
✗ MUST contain at least one uppercase letter
✗ MUST contain at least one lowercase letter
✗ MUST contain at least one number
✗ MUST contain at least one special character (!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~)
✓ MAY NOT contain more than two identical characters in a row
✓ MAY NOT contain first name, last name, email address mailbox or domain, company name or commonly used passwords
✓ MAY NOT match commonly used password character patterns

5 remaining rules need to be met

create a very strong password for me

2. Second-preimage Resistance

- Given x, computationally infeasible to find x' such that $x \neq x'$ and $H(x) = H(x')$
- Example: integrity of software distribution, fingerprinting (e.g., virus, deduplication)
- How much work is needed to break this resistance?



Ubuntu 22.04.1 LTS (Jammy Jellyfish)

A full list of available files, including BitTorrent files, can be found below.

If you need help burning these images to disk, see the Image Burning Guide.

Name	Last modified	Size	Description
Parent Directory		-	
SHA256SUMS	2022-08-11 11:07	202	
SHA256SUMS.gpg	2022-08-11 11:07	833	
ubuntu-22.04.1-desktop-amd64.iso	2022-08-10 16:21	3.6G	Desktop image for 64-bit PC (AMD64) computers (standard download)

3. Collision Resistance

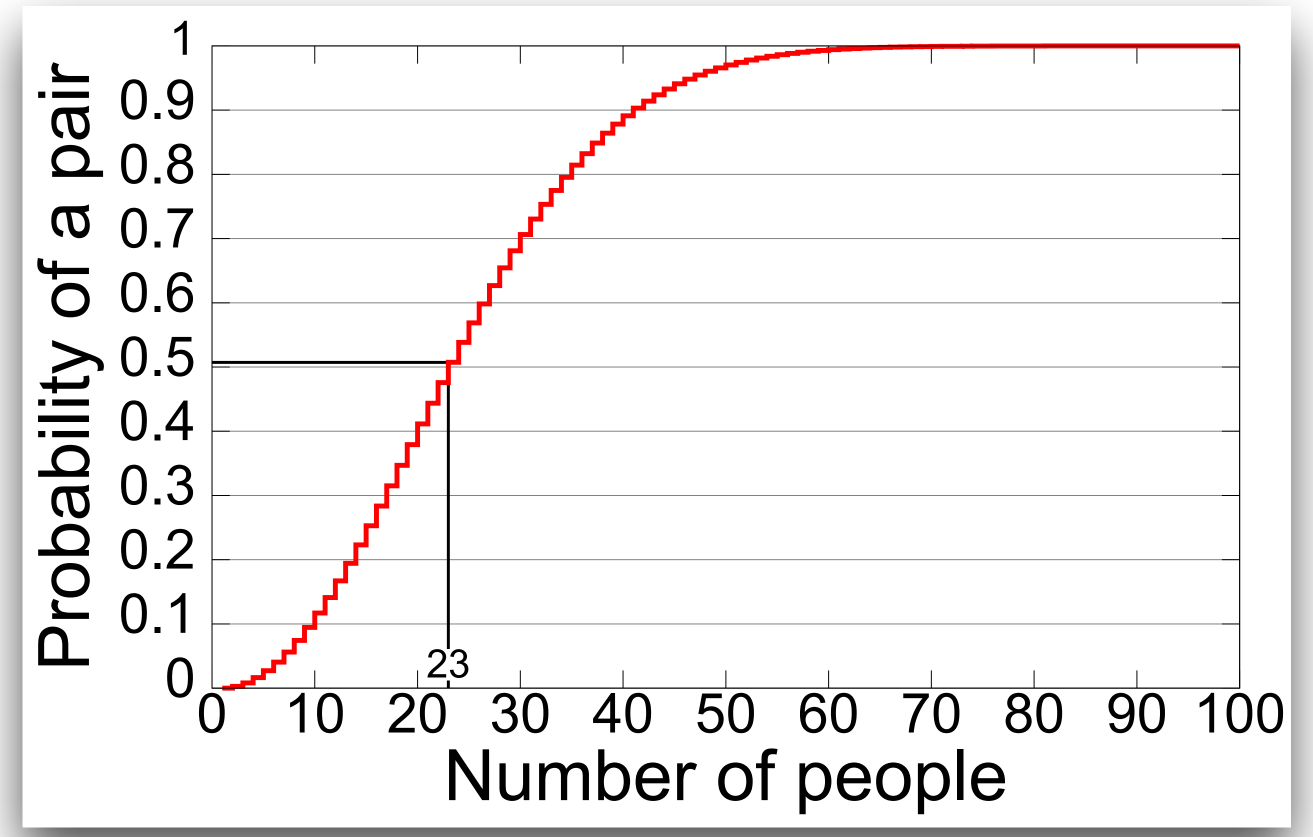
- Computationally infeasible to find x, x' such that $x \neq x'$ and $H(x) = H(x')$
- Example: auction bidding
  - Alice wants to bid B and sends $H(B)$
  - Rival bidders should not recover B (one-wayness)
  - Alice should not be able to change her mind to bid B' such that $H(B) = H(B')$
- How much work is needed to break this resistance?
  - $2^{n/2}$ (not 2^n, birthday paradox)
  - Example (MD5): $2^{128} \approx 3 \times 10^{38}$ vs $2^{64} \approx 2 \times 10^{19}$

Birthday Paradox

- What is the prob. that in a set of n random people, at least two will share a birthday?
  - If $n = 1$, then 0%
  - If $n = 366$, then 100%
  - What is $n = 23, 50$ or 60?
- Let $p(n)$ be the prob. and $\bar{p}(n)$ be that all n birthdays are different, i.e., $p(n) = 1 - \bar{p}(n)$

$$\bar{p}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \times \left(1 - \frac{n-1}{365}\right)$$

$$= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}$$



Birthday Paradox Challenge

- 61 submissions
- How many student submissions was the first collision detected?
- How many collisions were detected?

Application: Fingerprinting

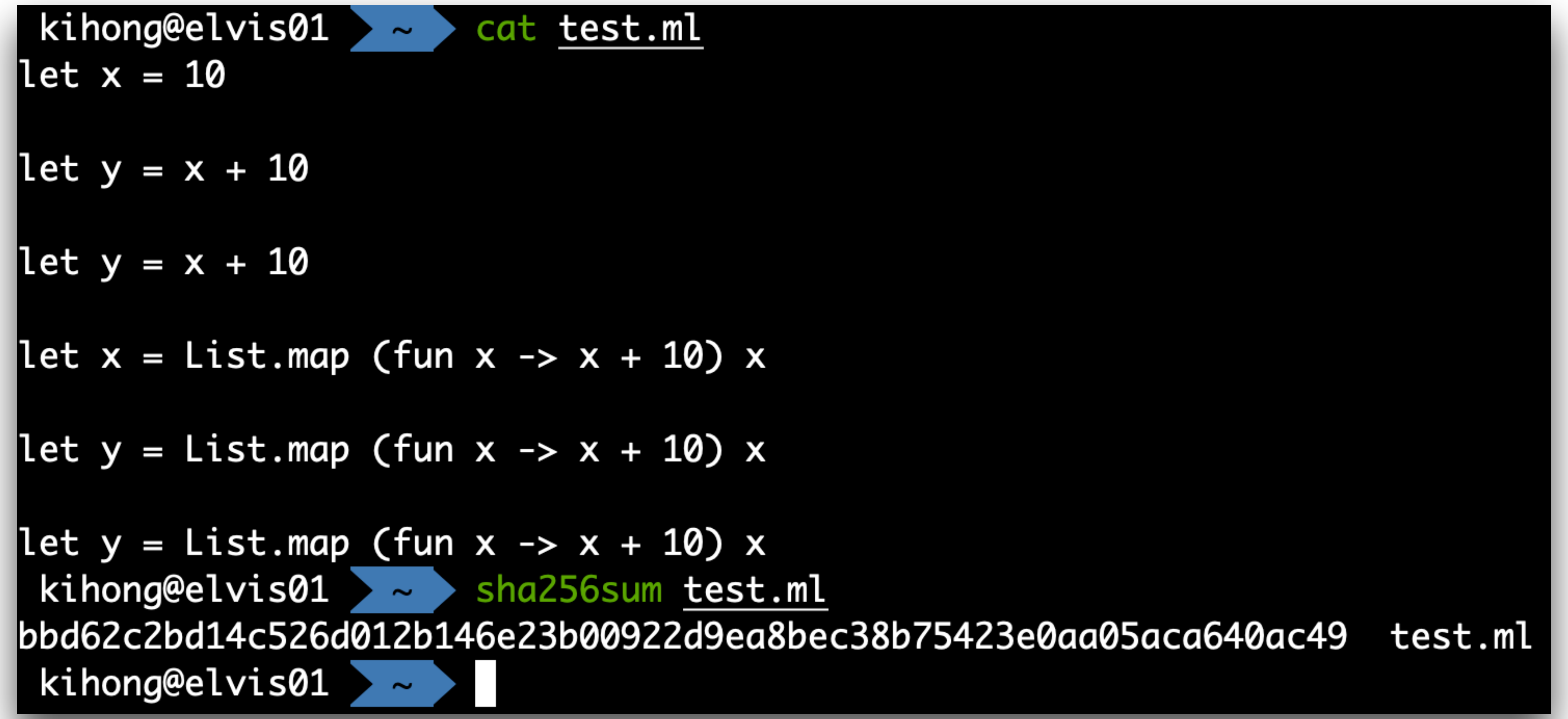
- Instead of storing the original data, simply store a shot hash digest
- Examples:
  - Virus fingerprinting
  - Deduplication
  - File sharing
- What happens if there exist a lot of collisions?

Informal Analysis of Cryptographic Hash

- Collision resistance → second-preimage resistance
    - If an adversary can find $x' \neq x$ s.t. $H(x') = H(x)$, then it can clearly find a collision
- Second-preimage resistance → preimage resistance
    - Assume H is second preimage resistant but not preimage resistant, then contradiction
    - For given x and $y = H(x)$, one can find x' that satisfies $y = H(x')$ (by assumption)
    - When the domain is infinitely large, one can find $x' \neq x$ with high probability

Practical Cryptographic Hash Functions

- MD5 (1991): 128-bit output length
  - Collisions found in 2004. Insecure.
- SHA-1 (1995): 160-bit output length
  - Very commonly used. Yet, collisions found in 2017.
  - Current trend to migrate to SHA-2
- SHA-2 (2001): 256 or 512-bit output lengths
  - Often called SHA256 or SHA512
  - No known significant weaknesses
- SHA-3 (2012): 224, 256, 384, or 512-bit output lengths



```
 kihong@elvis01  ~  cat test.ml
let x = 10

let y = x + 10

let y = x + 10

let x = List.map (fun x -> x + 10) x

let y = List.map (fun x -> x + 10) x

let y = List.map (fun x -> x + 10) x
 kihong@elvis01  ~  sha256sum test.ml
bbd62c2bd14c526d012b146e23b00922d9ea8bec38b75423e0aa05aca640ac49  test.ml
 kihong@elvis01  ~ 
```

SHA-1 Broken?

- SHA-1: 160-bit output length
- SHAttered attack by CWI and Google (2017)
  - Two different PDF files f_1 and f_2 such that $H(f_1) = H(f_2)$
  - $2^{63.4}$ operations (<< 2^{80})
  - 6,500 years of single-CPU, 110 years of single-GPU
- DO NOT USE SHA-1 for any security-critical systems!

Common Mistakes

- MAC vs checksum?
- MAC vs cryptographic hash function?

Summary

- Integrity: attackers cannot generate a valid message
    - As important as confidentiality
- MAC (message authentication code): prevent an adversary from modifying a message
- Cryptographic hash function: a common method to implement MAC