Hash Function

Viet Tung Hoang

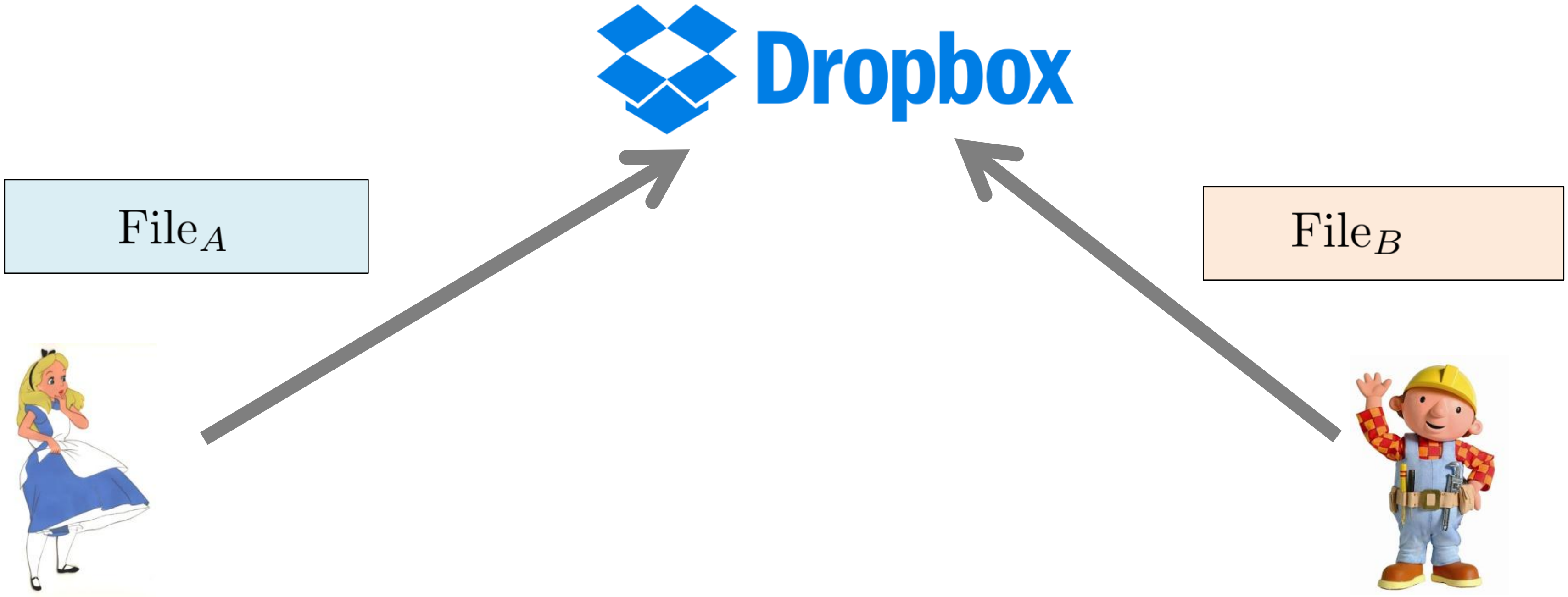
Agenda

1. **Security Modeling for Hash Functions**

2. Building Hash Function: MD Transform

3. Application: Password Storage

Motivating Application: Data Deduplication



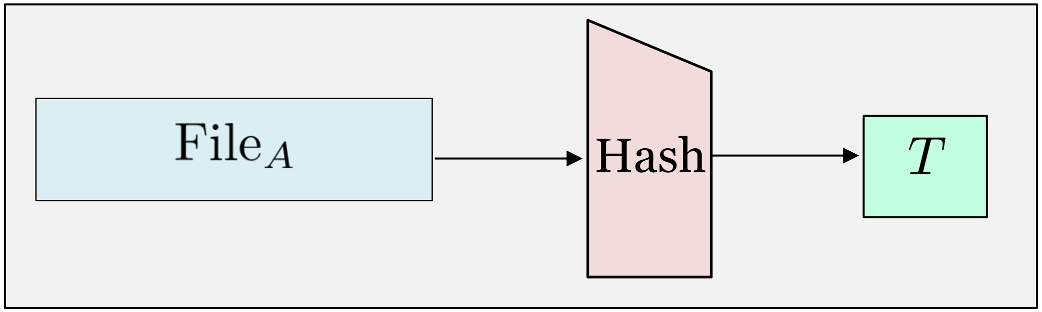
File_A

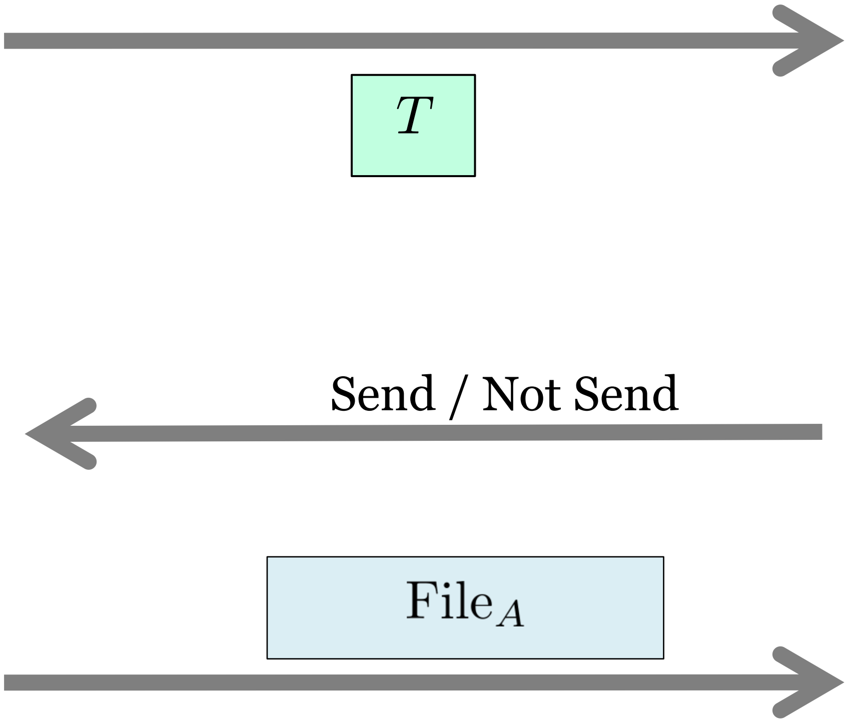
File_B

Dropbox's goals:

- If many users store the same file, keep only a **single** copy
- Minimize bandwidth usage

Motivating Application: Data Deduplication





Send only when requested

Dropbox

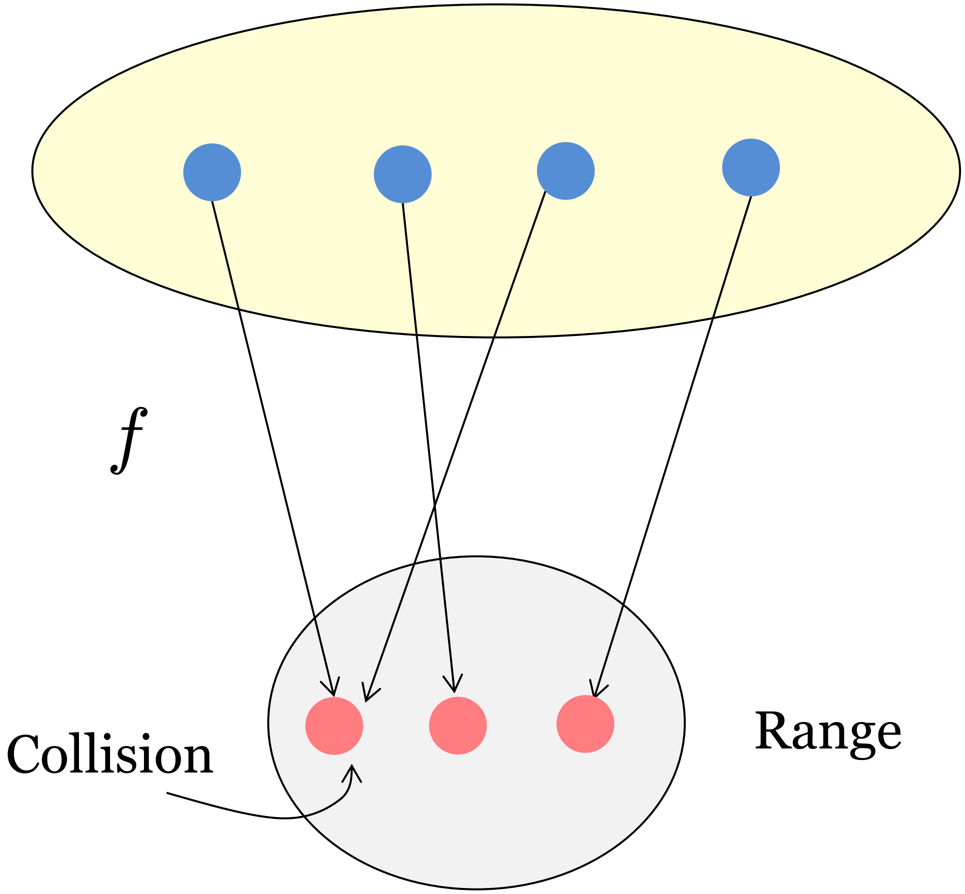
Check tags of existing files for duplication

What property

do we need for the hash?

Collision-Resistance

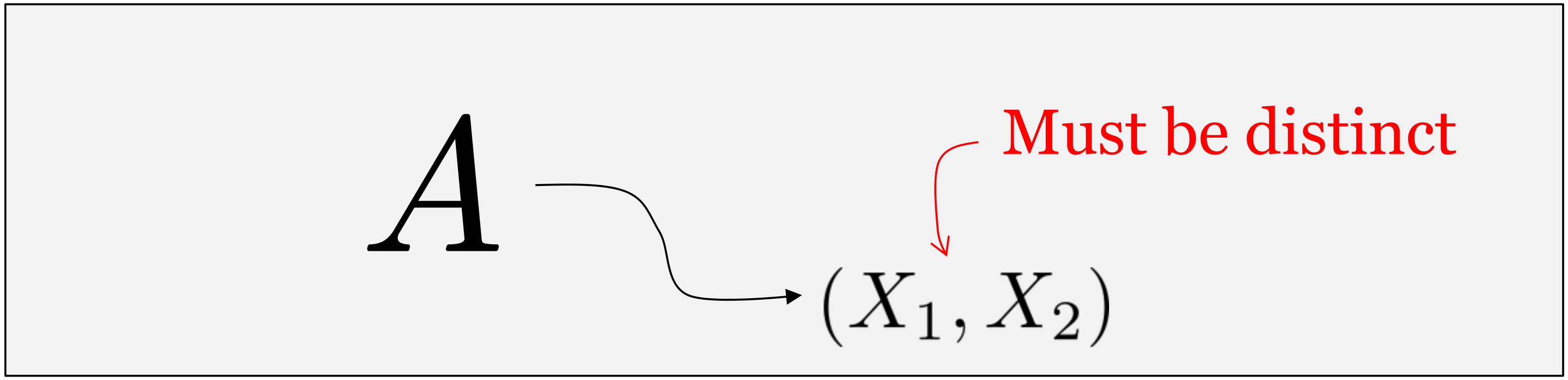
$f : \text{Domain} \rightarrow \text{Range}$



By Pigeonhole Principle, if $|\text{Domain}| > |\text{Range}|$ then collision exists

Want: collisions are hard to find, although they exist

Defining Collision-Resistance



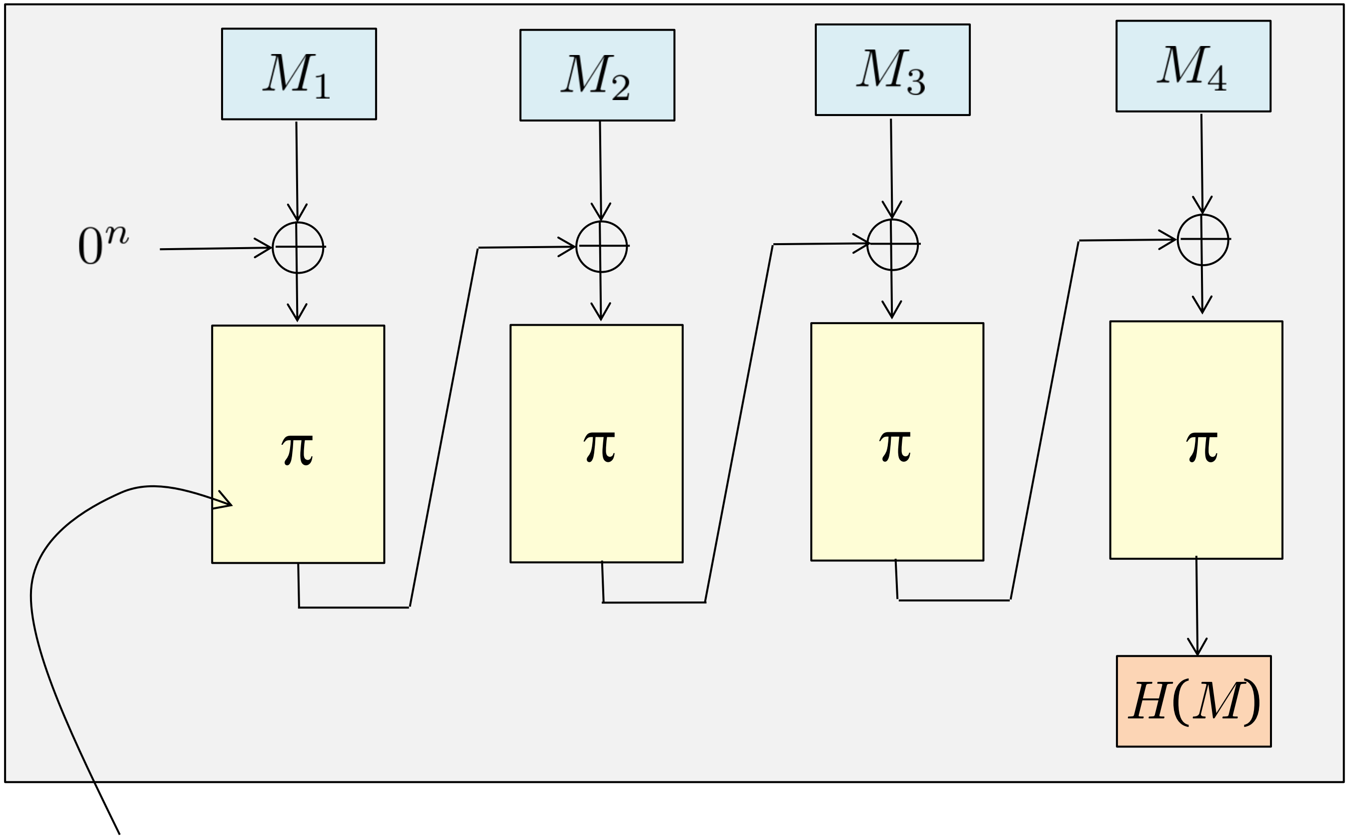
A

Must be distinct

(X_1, X_2)

$$\mathbf{Adv}^{\mathrm{cr}}_{H}(A) = \Pr[H(X_1) = H(X_2)]$$

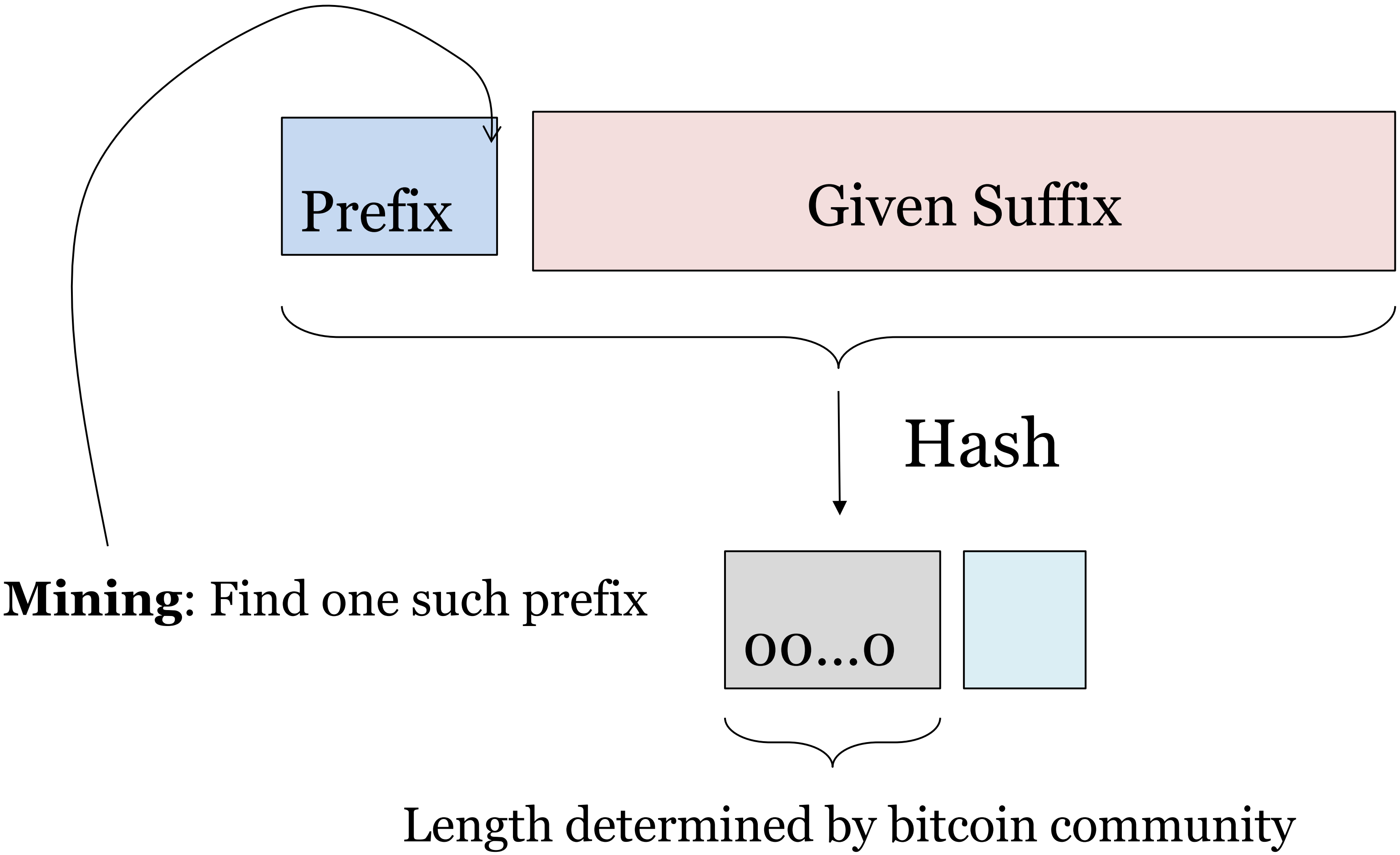
Exercise: Break Collision Resistance



π, π^{-1} are public

Public permutation

CR Is Not Enough: Bitcoin Mining



Prefix

Given Suffix

Hash

Mining: Find one such prefix

00...0

Length determined by bitcoin community

Want: Can’t mine faster than brute-force

Modeling Security of Hash Functions

The Random Oracle Model

RO

Maintain a secret random $f : \{0,1\}^n \rightarrow \{0,1\}^m$

Everybody, including the adversary, has access to RO

Agenda

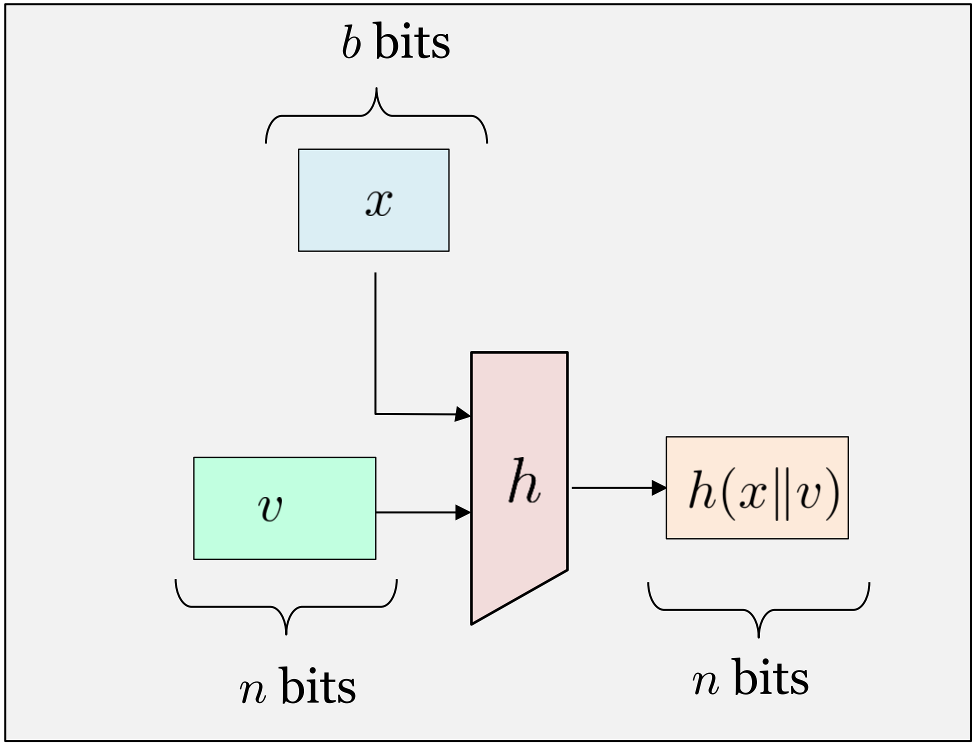
1. Security Modeling for Hash Functions

2. **Building Hash Function: MD Transform**

3. Application: Password Storage

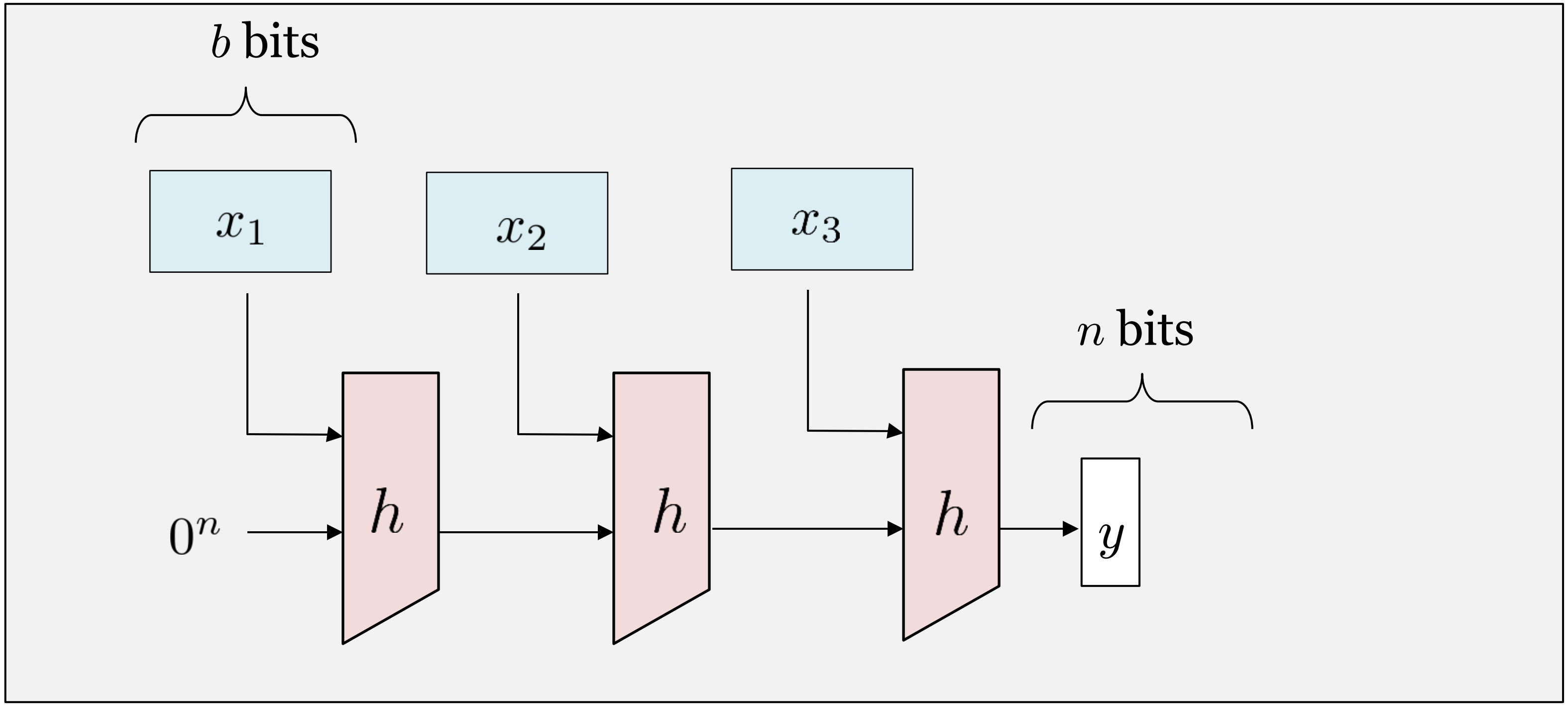
Compression Functions

$$h : \{0,1\}^{b+n} \to \{0,1\}^n$$



For SHA-2, b = 512 and n = 256

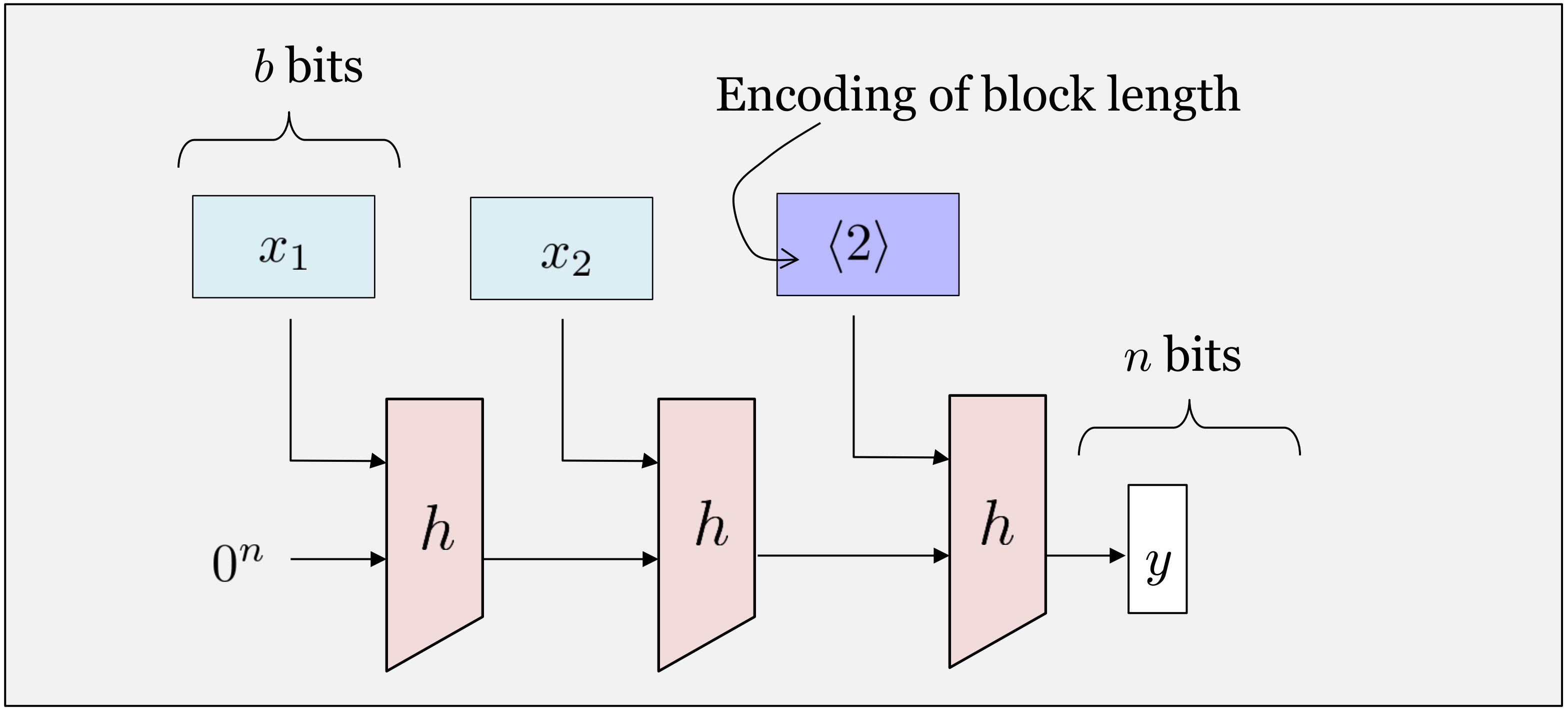
First Attempt



Question: Suppose that $h(0^b \| 0^n) = 0^n$

Break the collision resistance of H

Second Attempt: Plain Merkle-Damgard



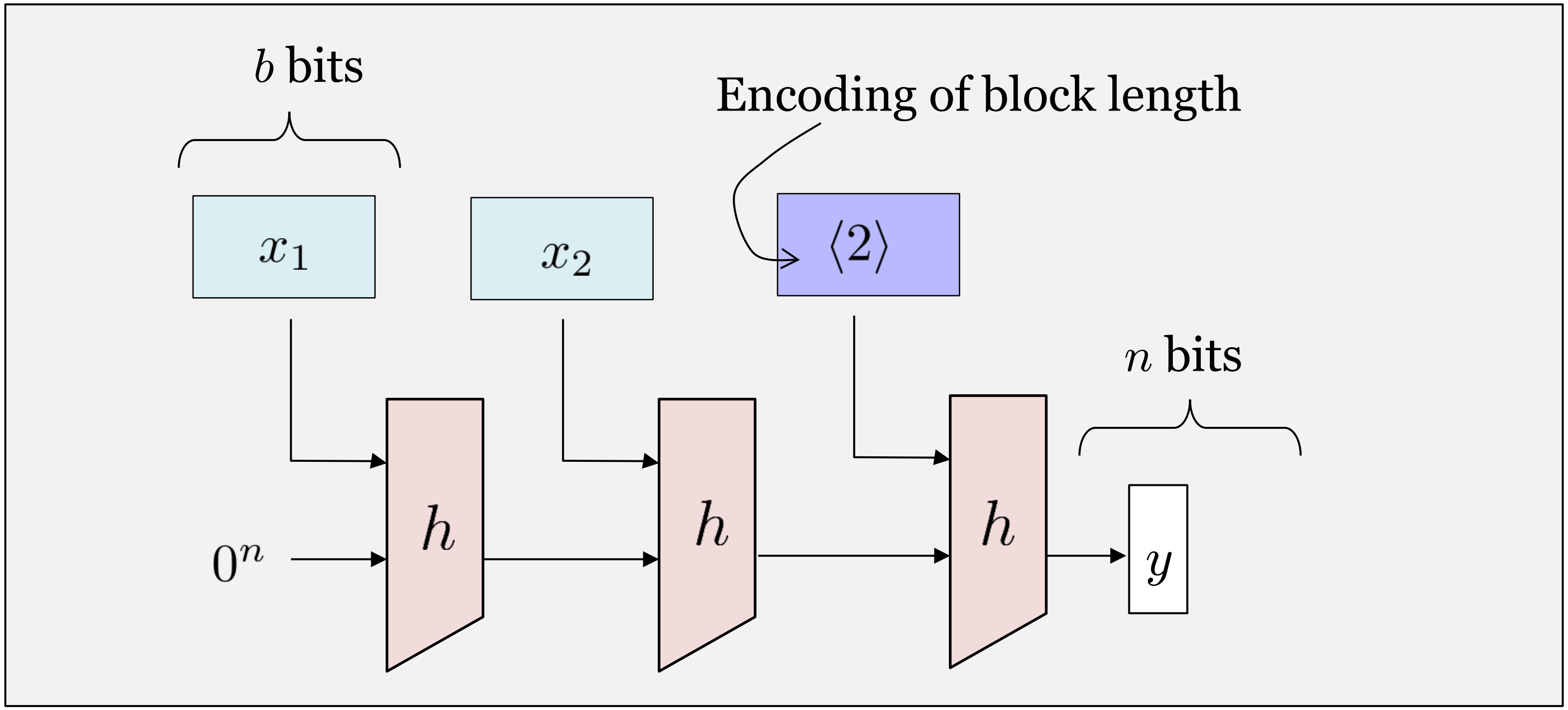
This is the structure of SHA-256

Theorem: If h is CR then H = **MD**(h) is also CR

Can't attack H if h has no weakness

Plain MD Is Not Enough for All Applications

Length-Extension Attack



Question: Consider the following MAC F

$$F_K(x) = H(K \| x)$$

Break the MAC security of F using a single Tag query

The Damage of Length Extension Attack

Hacking Trick: Bypass Authentication

K

Bank of America

bank.com

K

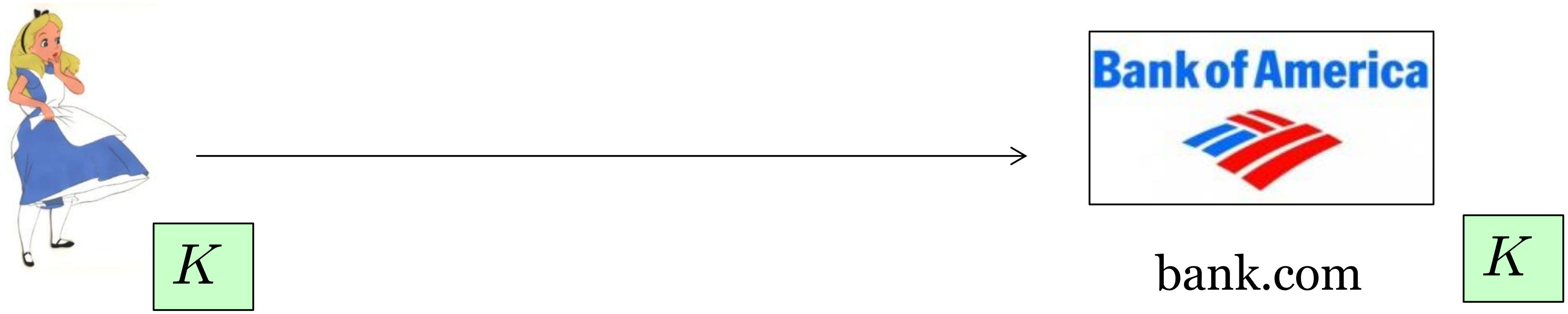
bank.com/api?token=ad6613c382&user=alice&cmd=NoOp

$H(K \| \text{“user=alice\&cmd=NoOp”})$

Adversary tricks Alice to perform a harmless command to learn an authentication token

The Damage of Length Extension Attack

Hacking Trick: Bypass Authentication

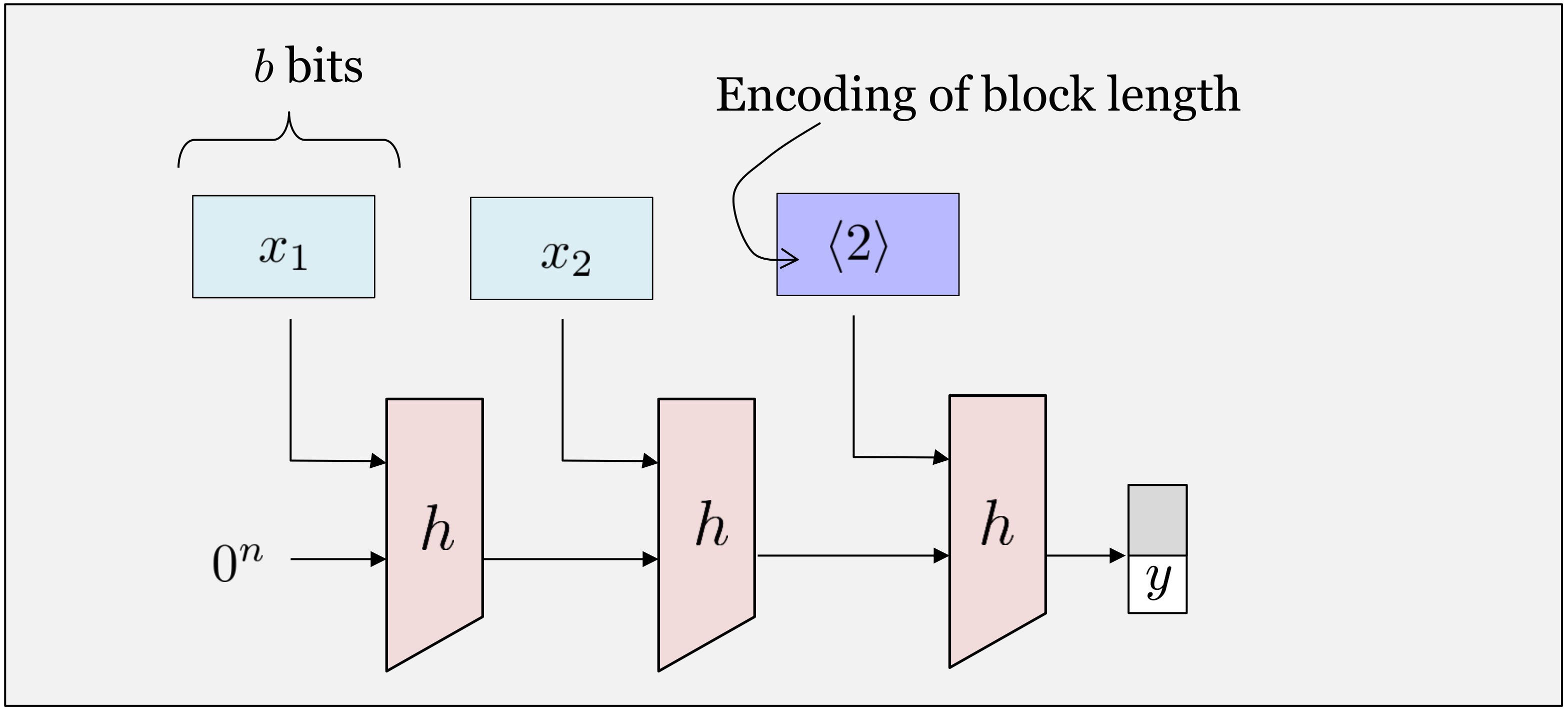


bank.com/api?token=dbb78b593f&user=alice&cmd=NoOp&cmd=OpenSafe

$H(K \| \text{“user=alice\&cmd=NoOp\&cmd=OpenSafe”})$

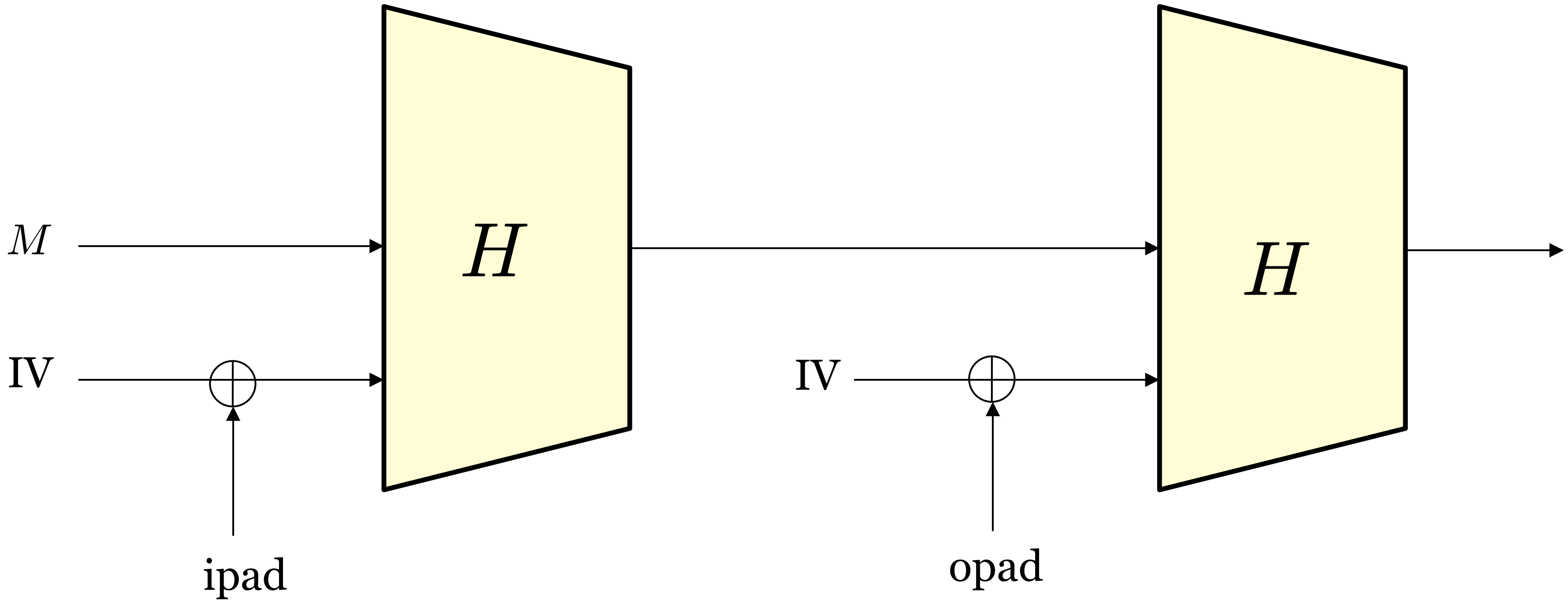
Adversary can compute the authentication token for a damaging command

The (Strengthened) MD Transform



The output needs to be truncated

How To Have Large Output: HMAC



On large input, HMAC is only a bit more expensive than SHA-256

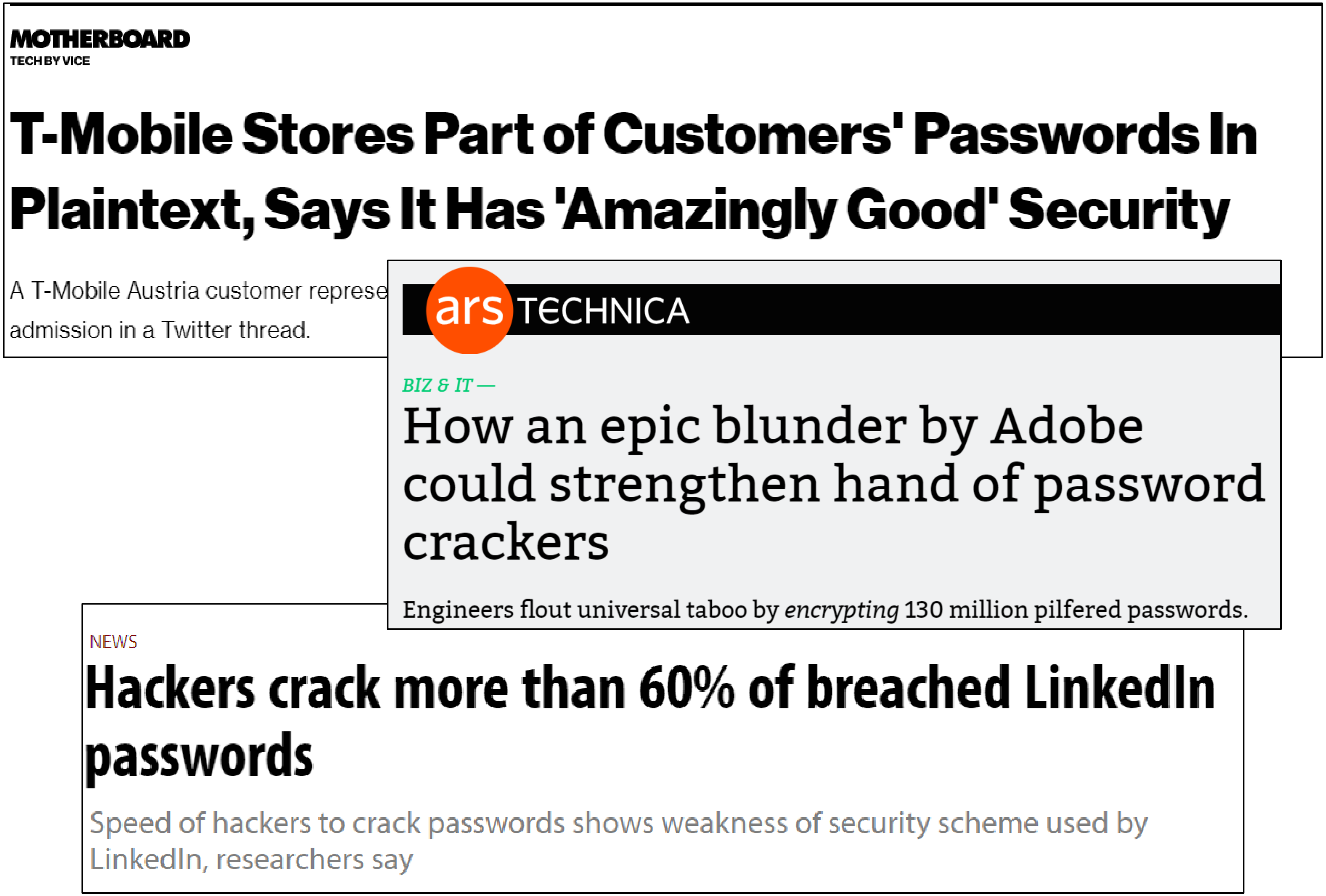
Agenda

1. Security Modeling for Hash Functions

2. Building Hash Function: MD Transform

3. **Application: Password Storage**

Password Storage



MOTHERBOARD
TECH BY VICE

T-Mobile Stores Part of Customers' Passwords In Plaintext, Says It Has 'Amazingly Good' Security

A T-Mobile Austria customer represe
admission in a Twitter thread.

ars TECHNICA

BIZ & IT —

How an epic blunder by Adobe could strengthen hand of password crackers

Engineers flout universal taboo by *encrypting* 130 million pilfered passwords.

NEWS

Hackers crack more than 60% of breached LinkedIn passwords

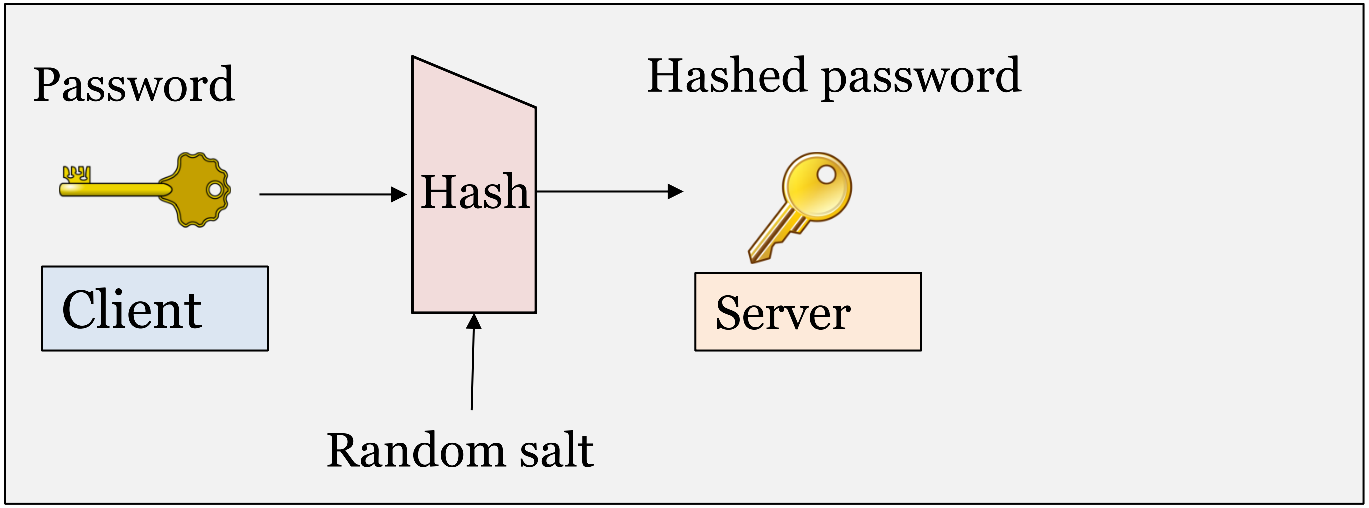
Speed of hackers to crack passwords shows weakness of security scheme used by LinkedIn, researchers say

How Should Servers Store Users' Passwords?

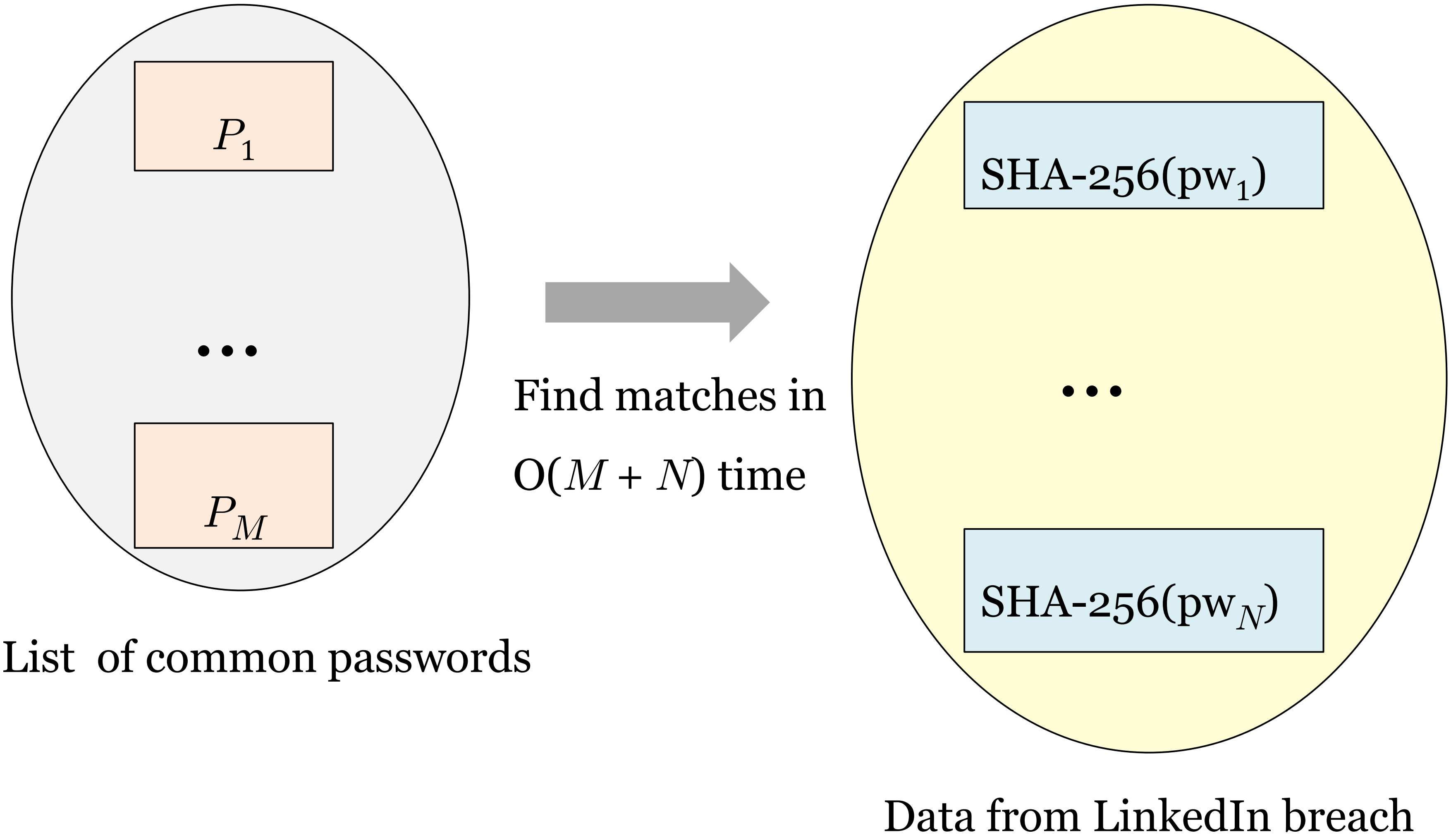
Rule 1: Only store hash outputs of passwords

Even server can't recover the passwords

Rule 2: Use a random salt for each user



Why Salts: Dictionary Attacks



P_1

…

P_M

List of common passwords

Find matches in

O(M + N) time

SHA-256(pw$_1$)

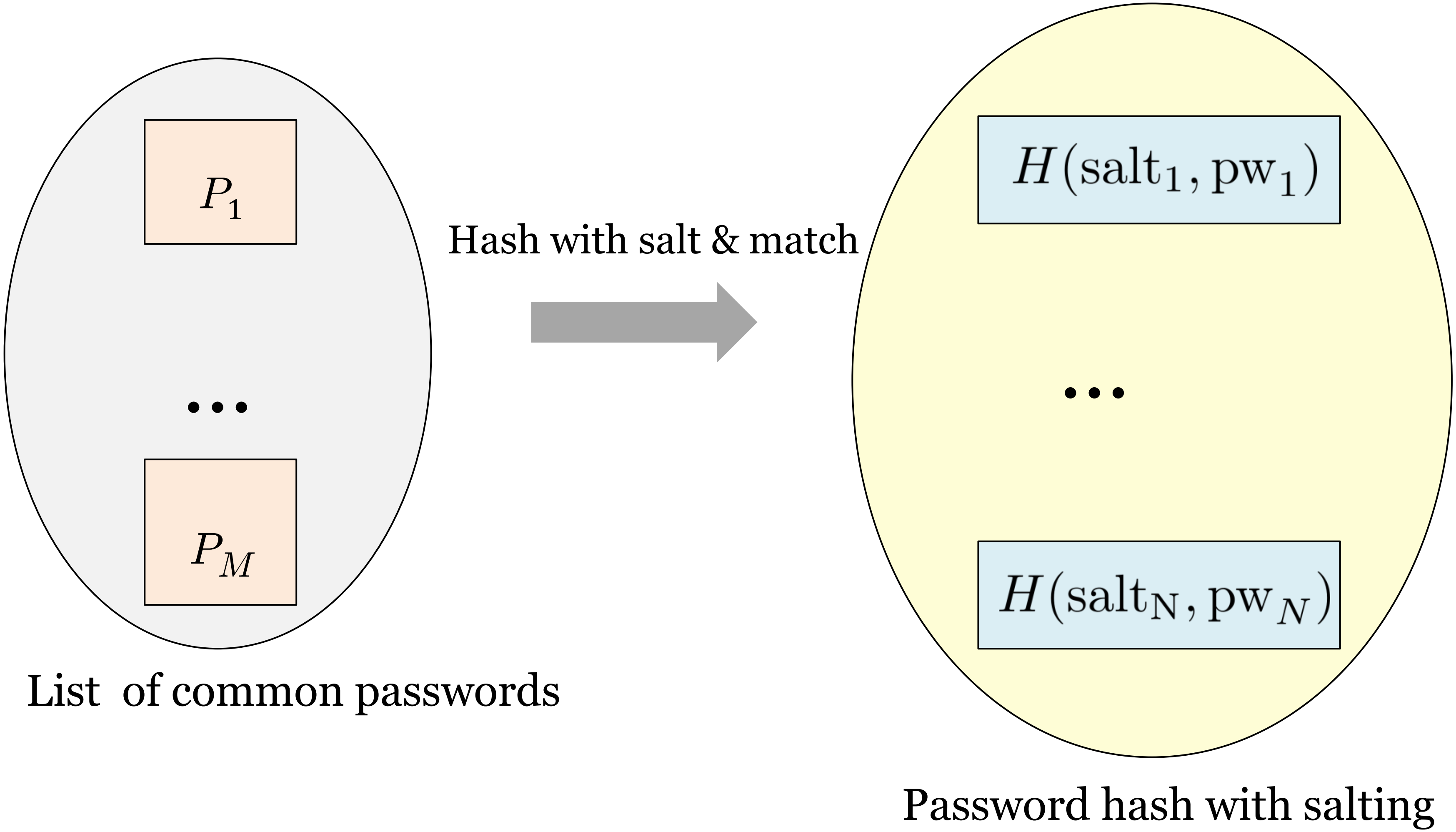
…

SHA-256(pw$_N$)

Data from LinkedIn breach

Cost of Dictionary Attacks on Salting

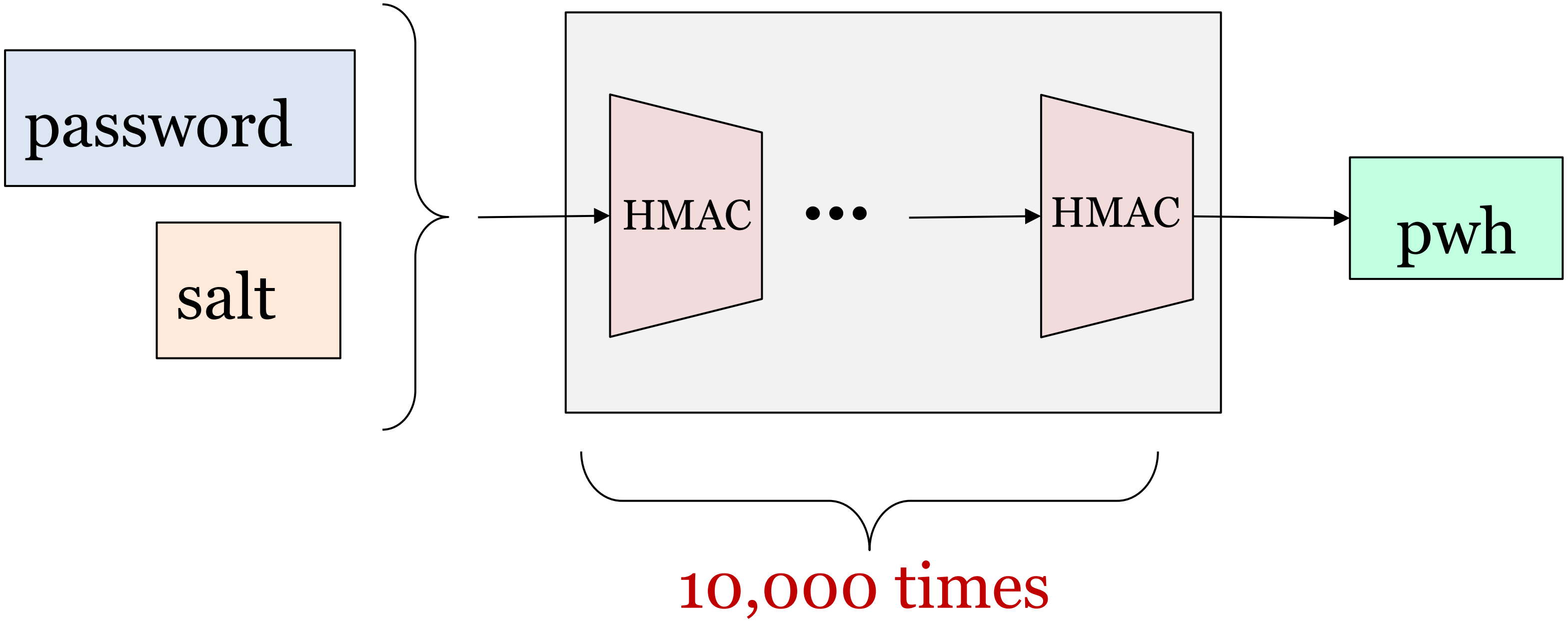
Need $\Theta(Mq)$ calls to H to recover q passwords



P_1
$\dots$
P_M
List of common passwords
Hash with salt & match
$H(\mathrm{salt}_1, \mathrm{pw}_1)$
$\dots$
$H(\mathrm{salt}_\mathrm{N}, \mathrm{pw}_N)$
Password hash with salting

Make It Even More Expensive

Deliberately Slow Hashing



- Makes no difference for human users.
- Increase the cos of attackers for 10,000 times