

---

CSE 3400/ CSE 5850 - Introduction to Computer & Network  
Security  
/ Introduction to Cybersecurity

Lecture 6  
Hash Functions – Part I

Ghada Almashaqbeh  
UConn

Adapted from textbook slides by Prof. Amir Herzberg  
UConn

---

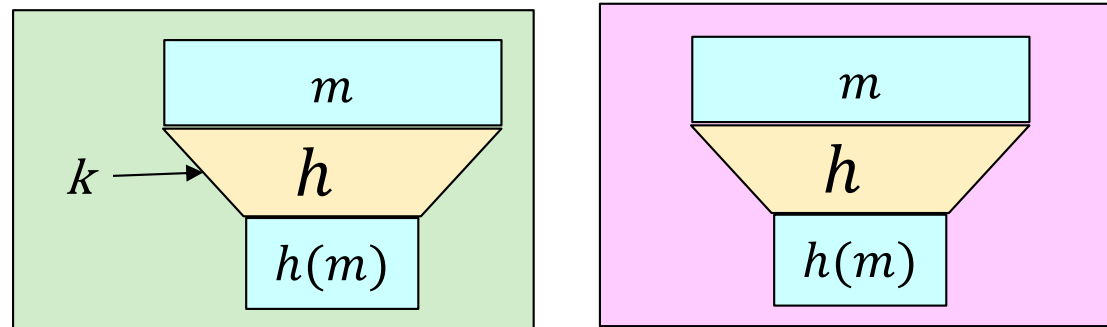
---

# Outline

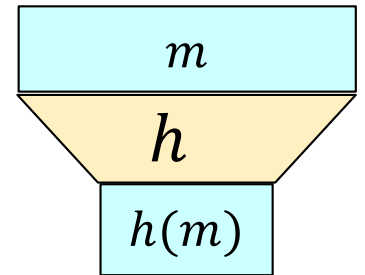
- ❑ Introduction and motivation.
- ❑ Collision resistant hash functions (CRHF).
- ❑ CRHF applications.
- ❑ Other notions of security.
  - ❑ TCR, SPR, OWF.
- ❑ Randomness extraction.
- ❑ The random oracle model.

# Hash Functions

- Input  $m$ : binary strings
- Output  $h(m)$  :
  - 'Short' (n-bit) binary strings
    - Aka **message digest**
- Efficiently computable
- Applications: cryptography, security, efficiency
- Keyed  $h_k(m)$ , where the key is public, or unkeyed  $h(m)$



# Hash functions: simple examples

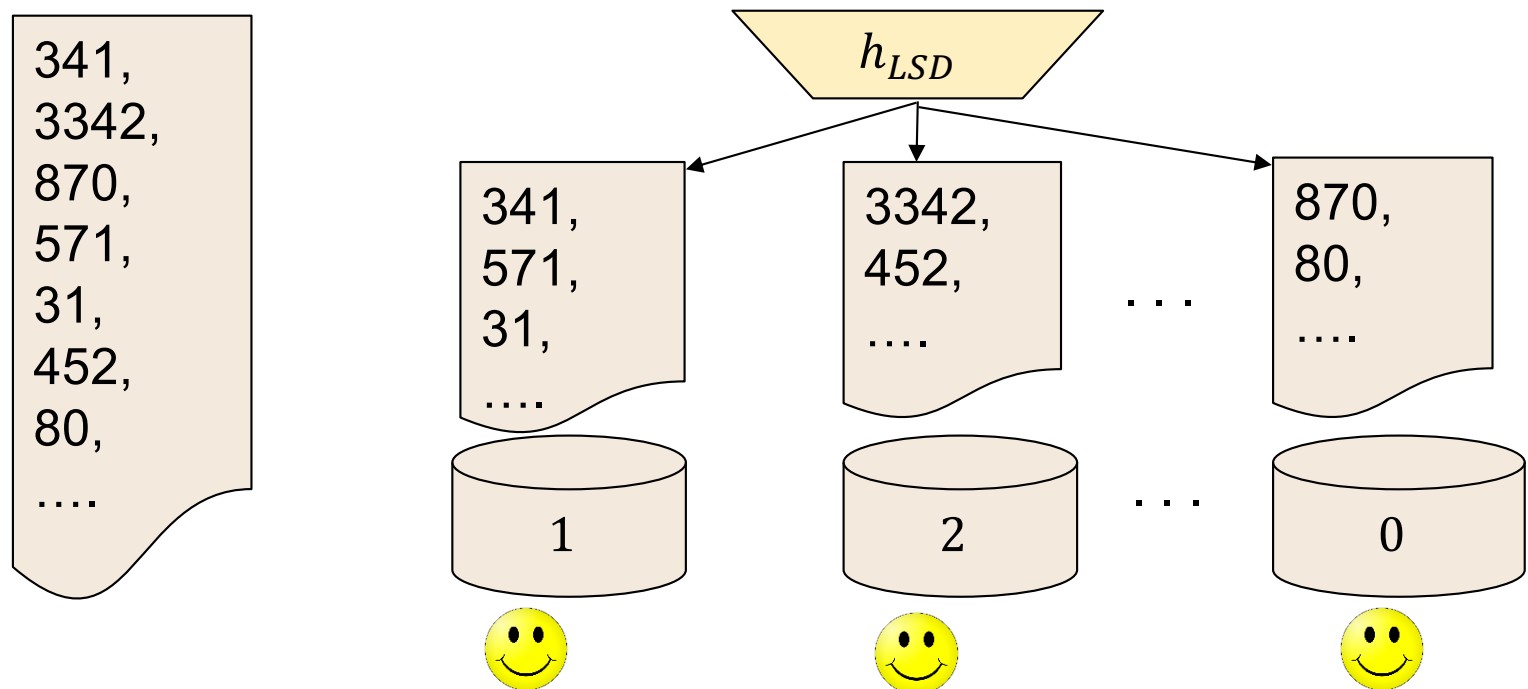


- For simplicity: input  $m$  is decimal integer
  - View as string of (three) digits
  - For example,  $m = 127 \rightarrow m_1 = 1, m_2 = 2, m_3 = 7$
- Least Significant Digit hash:
$$h_{LSD}(m) = m_3$$
- Sum hash: 
$$h_{Sum}(m) = (m_1 + m_2 + m_3) \bmod 10$$
- Exercise:
$$h_{LSD}(117) = \underline{7}$$
$$h_{Sum}(117) = \underline{9}$$

Note: the above are insecure hash functions, these are just toy examples to grasp the concept of hashing.

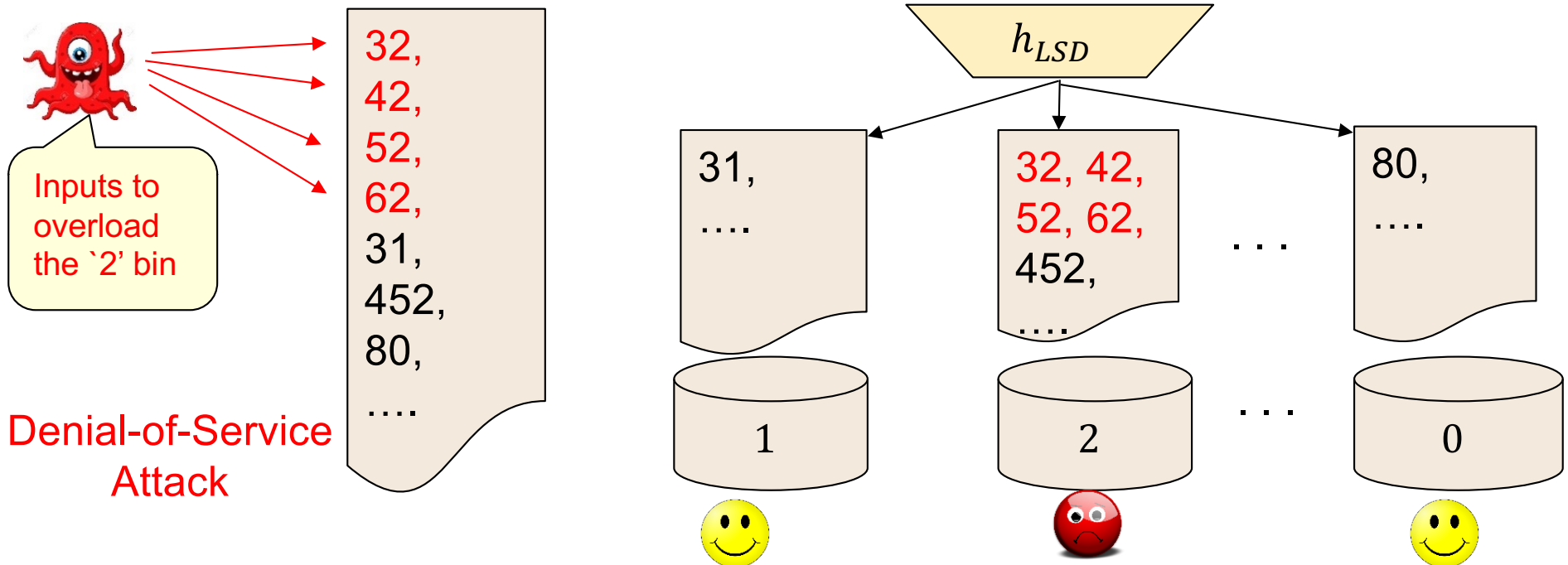
# Motivation: Hashing for efficiency

- Input: large set (e.g., integers or strings)
- Goal: map 'randomly' to few bins
  - E.g., to ensure efficiency – load balancing, etc.



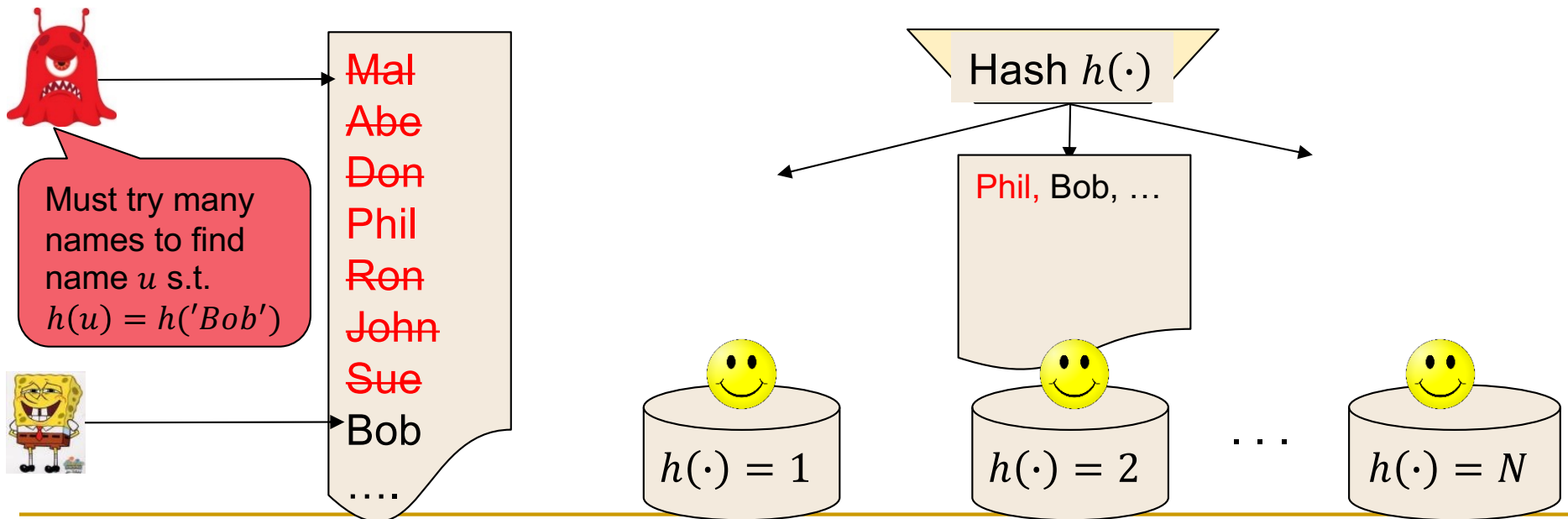
# Collisions?

- Input: large set (e.g., integers or strings)
- Goal: map 'randomly' to few bins
  - E.g., to ensure efficiency – load balancing, etc.
  - Adversary chooses inputs that hash to same bin



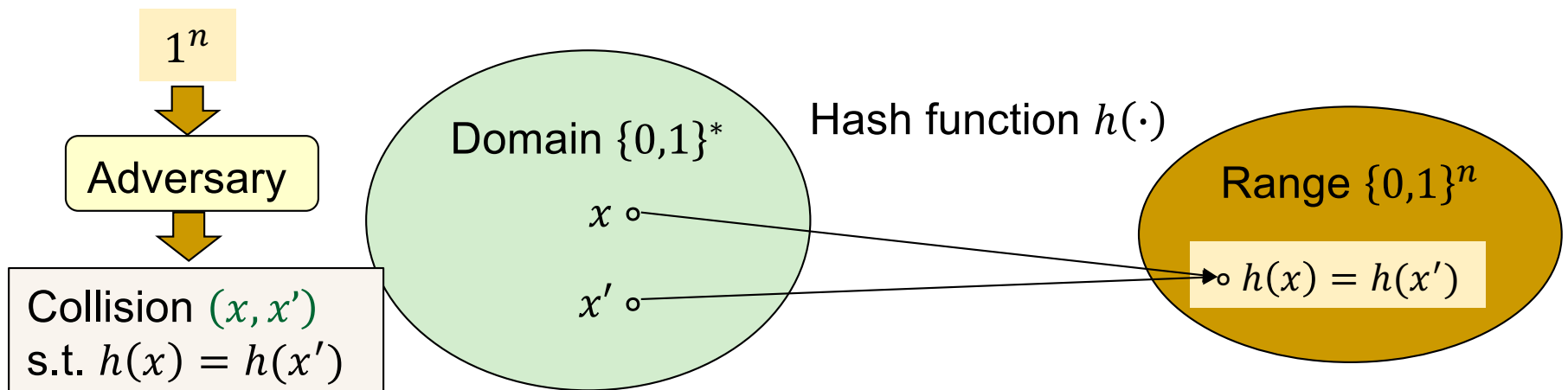
# Security Goal: Collision Resistance

- A **collision**: two inputs (names) with same hash:  
 $h('Bob') = h('Phil')$
- Every hash has collisions, since  $|\text{input}| \gg |\text{output}|$  !
- Collision resistance: hard to **find** these collisions
  - ❑ Note: attacker can always try names randomly until a collision is found
  - ❑ But this should be ineffective: must try about (on average)  $N$  names (number of bins)



# Collision Resistant Hash Function (CRHF)

- $h$  is CRHF if it is hard to **find** collisions  $h(x)=h(x')$ 
  - Note: attacker can always try inputs randomly till finding collisions
  - But this should be ineffective: must try about  $|Range|$  values
- Hard means that the probability that the attacker succeeds in finding a collision is negligible.





# Collision Resistant Hash Function (CRHF)

- $h$  is CRHF if it is hard to **find** collisions  $h(x)=h(x')$ 
  - Note: attacker can always try inputs randomly till finding collisions
  - But this should be ineffective: must try about  $|Range|$  values
- Hard means that the probability that the attacker succeeds in finding a collision is negligible.

**Definition** . . . (Keyless Collision Resistant Hash Function (CRHF)). A *keyless hash function*  $h^{(n)}(\cdot) : \{0, 1\}^* \rightarrow \{0, 1\}^n$  is collision-resistant if for every efficient (PPT) algorithm  $\mathcal{A}$ , the advantage  $\varepsilon_{h,\mathcal{A}}^{CRHF}(n)$  is negligible in  $n$ , i.e., smaller than any positive polynomial for sufficiently large  $n$  (as  $n \rightarrow \infty$ ), where:

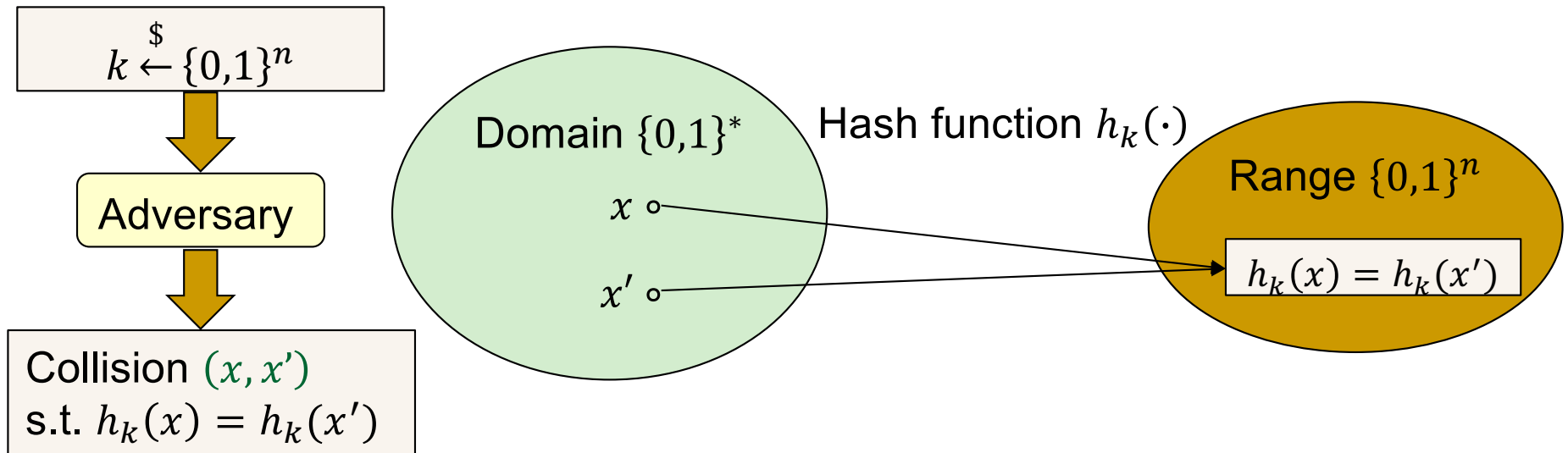
$$\varepsilon_{h,\mathcal{A}}^{CRHF}(n) \equiv \Pr \left[ (x, x') \leftarrow \mathcal{A}(1^n) \text{ s.t. } (x \neq x') \wedge (h^{(n)}(x) = h^{(n)}(x')) \right]$$

Where the probability is taken over the random coin tosses of  $\mathcal{A}$ .

# Keyless CRHF Do Not Exist!

- $|\text{Range}| \ll |\text{Domain}|$  so there is a collision where  $h(x') = h(x)$ ,  $x \neq x'$
- For a keyless CRHF there is a PPT algorithm  $A$  that can always output a collision:  $A(1^n) = \{\text{return } x, x'\}$ 
  - Proof: in textbook.
    - Intuitively, since the function is fixed (same input-output mapping), a collision instance can be hardcoded in the attacker algorithm and just output that collision and win the security game.
- **Solutions:**
  - keyed CRHF,
  - use weaker collision notions,
  - or ignore! (more like asking if the collision is useful for the attacker?)

# Keyed CRHF



Adversary knows  $k$  but **not in advance** – cannot `know` a collision

Often referred to as **ACR**-hash (**ANY**-collision resistance)

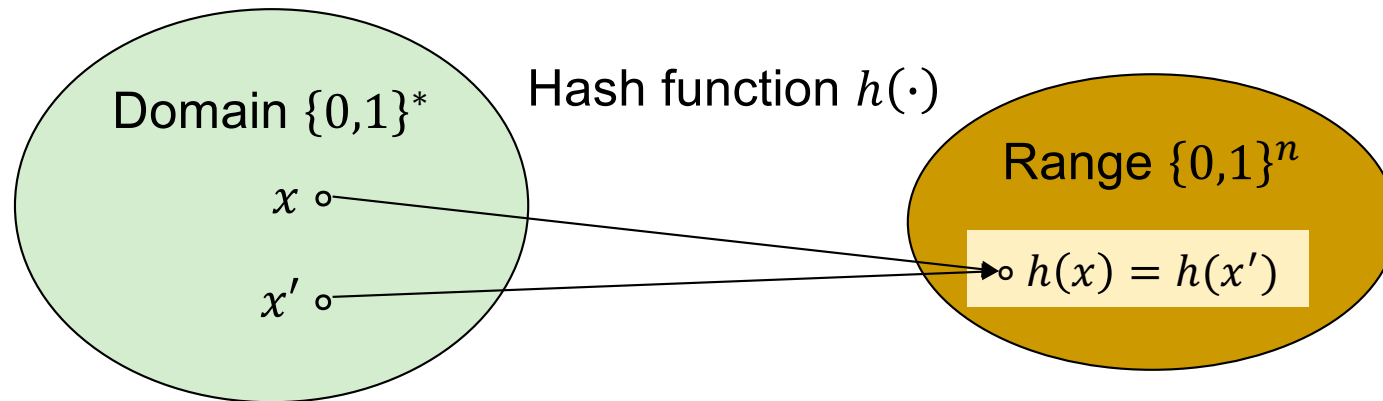
# Keyed CRHF - Definition

**Definition** (Keyed Collision Resistant Hash Function (CRHF)). Consider a keyed hash function  $h_k(\cdot) : \{0, 1\}^n \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ , defined for any  $n \in \mathbb{N}$ . We say that  $h$  is collision-resistant if for every efficient (PPT) algorithm  $\mathcal{A}$ , the advantage  $\varepsilon_{h, \mathcal{A}}^{CRHF}(n)$  is negligible in  $n$ , i.e.,  $\varepsilon_{h, \mathcal{A}}^{CRHF}(n) \in \text{NEGL}(n)$ , where:

$$\varepsilon_{h, \mathcal{A}}^{CRHF}(n) \equiv \Pr_{k \leftarrow \{0, 1\}^n} [(x, x') \leftarrow \mathcal{A}(k) \text{ s.t. } (x \neq x') \wedge (h_k(x) = h_k(x'))]$$

Where the probability is taken over the random coin tosses of the adversary  $\mathcal{A}$  and the random choice of  $k$ .

# Generic Collision Attacks



- An attacker that runs in exponential time can always find a collision (i.e., non PPT attacker)
  - Easy: find collisions in  $2^n$  time by trying  $2^n + 1$  distinct inputs (compute their hash and locate a collision).
- An attacker finds a collision with  $2^{-n}$  probability (negligible probability).
  - Choose  $x$  and  $x'$  at random and check if they produce a collision.

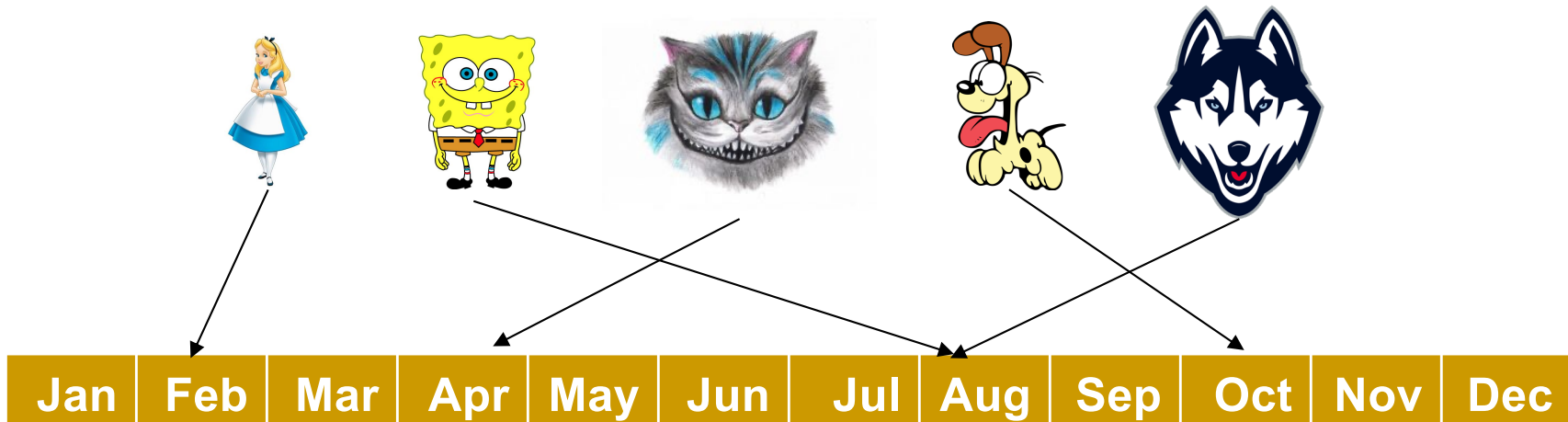
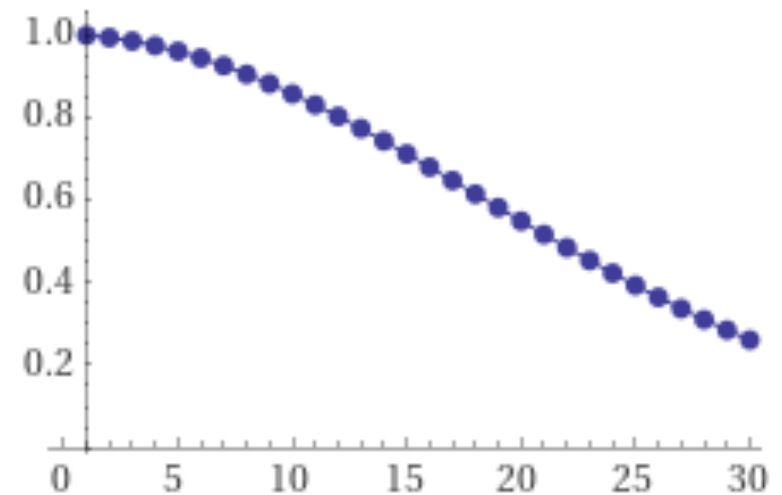
# The Birthday Paradox

- **The birthday paradox** states that the expected number  $q$  of hashes until a collision is found in  $O(2^{n/2})$  not  $O(2^n)$ .
  - It is  $q \lesssim 2^{n/2} \cdot \sqrt{\frac{\pi}{2}} \lesssim 1.254 \cdot 2^{n/2}$
- For 80 bit of effective security, use  $n=160$  !
  - So to defend against an attacker who can perform  $2^{80}$  hashes set the digest length to be at least 160 bits.
    - So the range has a size of  $2^{160}$  digests.
- Why? Intuition?

# The Birthday Attack ('Paradox')

## ■ Probability of NO birthday-collision:

- ❑ Two persons:  $(364/365)$
- ❑ Three persons:  $(364/365) \cdot (363/365)$
- ❑ ...
- ❑  $n$  persons:  $\prod_{i=1}^{n-1} \frac{365-i}{365}$
- ❑ So for 23 individuals, the probability of birthday collision is about 0.5



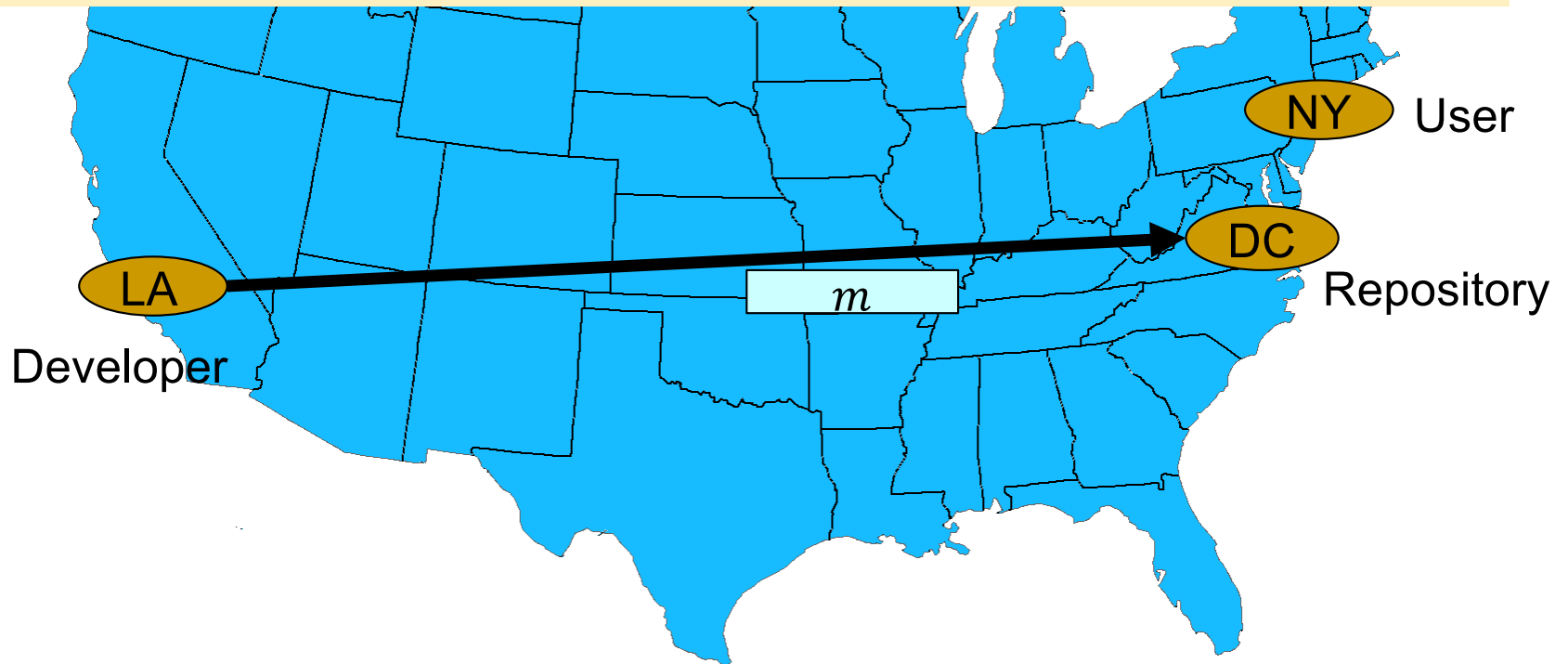
# Collision-Resistance: Applications

- Integrity (of object / file / message )
  - Send  $hash(m)$  securely to validate  $m$
  - Later we will see how a hash function can be used to construct a MAC (called HMAC).
- Hash-then-Sign
  - Instead of signing  $m$  sign  $hash(m)$ 
    - More efficient!
    - We will explore this in detail once we study digital signatures.
- Blockchains
  - Later



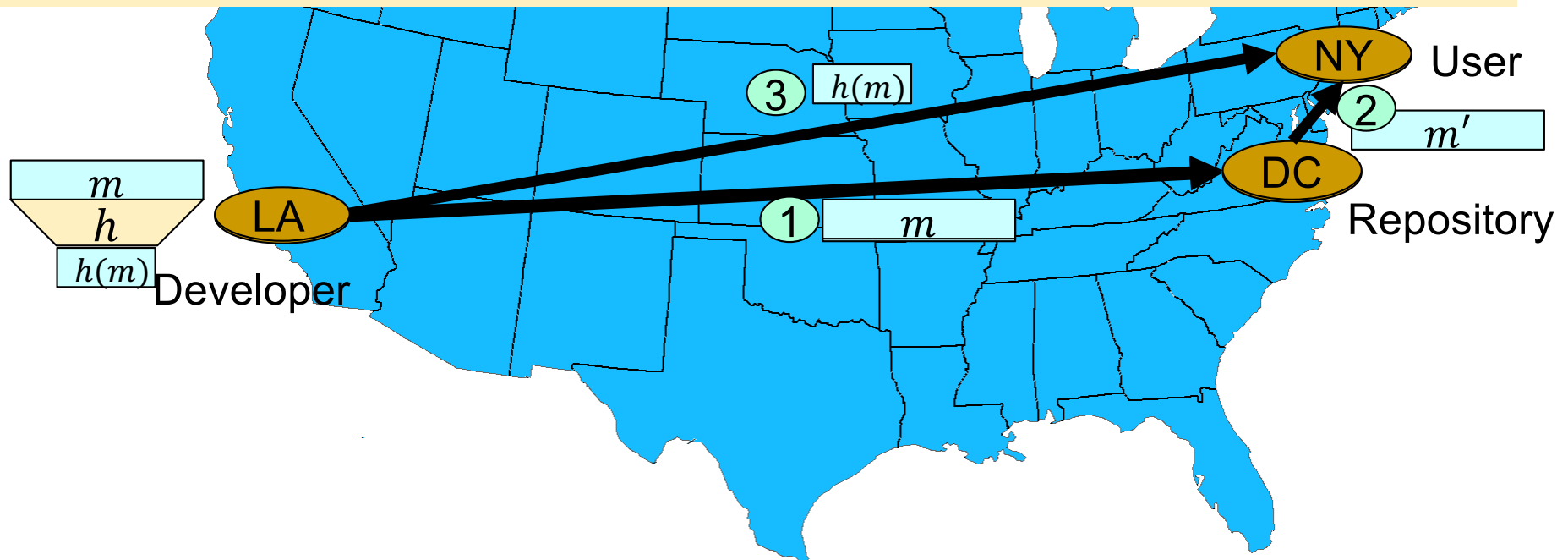
# CRHF and Software Distribution

- ❑ Developer in LA develops a large software  $m$
- ❑ Repository in DC obtains a copy of  $m$
- ❑ User in NY wants to obtain  $m$  – securely and efficiently
  - Don't send  $m$  from LA to **both** NY and DC
- ❑ How?



# CRHF: secure, efficient SW distribution

1. Repository in DC downloads software  $m$  from developer in LA
2. User downloads from (nearby) repository; receives  $m'$ 
  - Is  $m' = m$  ? User should validate! How?
3. User securely downloads  $h(m)$  directly from developer
  - Digest  $h(m)$  is short – much less overhead than downloading  $m$
4. User validates:  $h(m) = h(m') \rightarrow m = m'$

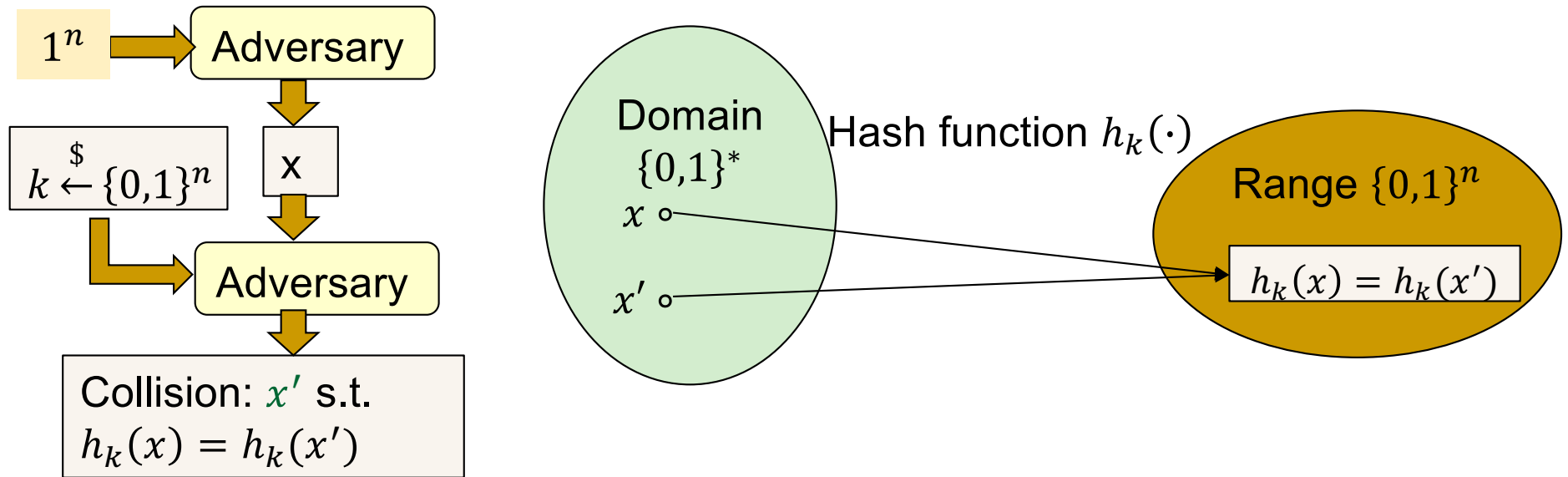


---

# Other Notions of Security

- Collision resistance provides the strongest guarantee.
    - ❑ Gives more freedom to the adversary; the adversary wins if it finds any two inputs with the same digest.
      - No conditions on these two inputs other than being in the domain of the hash function.
  - Other security notions (but sufficient for many applications):
    - ❑ Target collision resistance (TCR).
    - ❑ Second preimage resistance.
    - ❑ First preimage resistance.
  - Birthday paradox (or attack) does not work against these weaker notions.
    - ❑ It is for collision resistance; find **any** two inputs that collide!
-

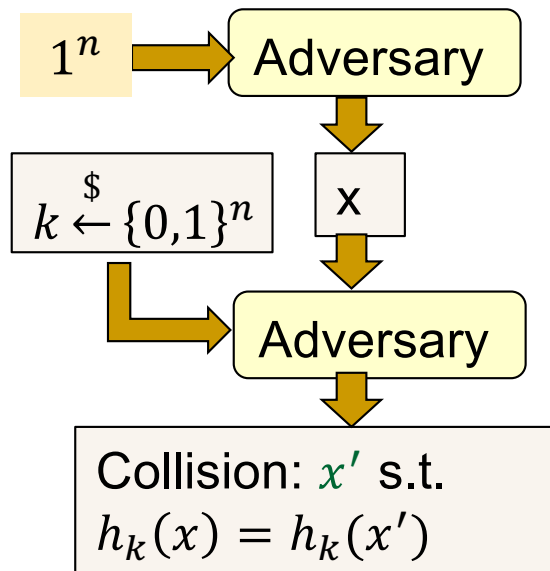
# Target CRHF (TCR Hash Function)



Adversary has to select target **before** knowing key

$$\varepsilon_{h,\mathcal{A}}^{TCR}(n) \equiv \Pr_{k \leftarrow \{0,1\}^n} \left[ \left\{ \begin{array}{l} x \leftarrow A(1^n); \\ x' \leftarrow A(x, k) \end{array} \right\} \text{ s.t. } (x \neq x') \wedge (h_k(x) = h_k(x')) \right]$$

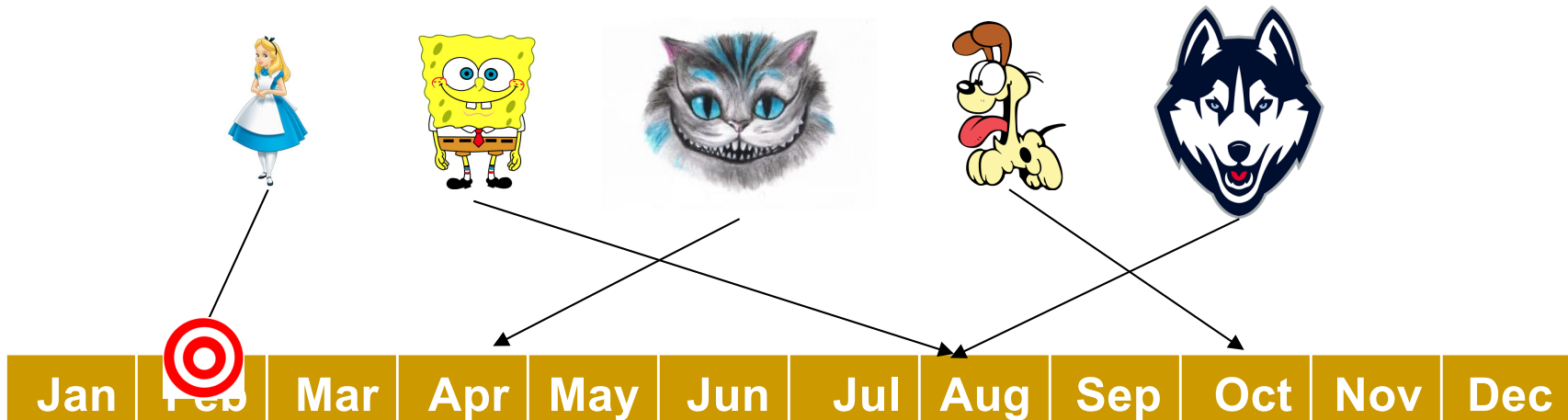
# TCR and Birthday Paradox?



- **First:** adversary selects  $x$
- Probability for NO birthday-collision **with  $x$ :**

- Two persons:  $(364/365)$
- Three persons:  $(364/365) \cdot (363/365)$
- ...

- $n$  persons:  $\prod_{i=1}^{n-1} \frac{364}{365} = \left(\frac{364}{365}\right)^{n-1}$

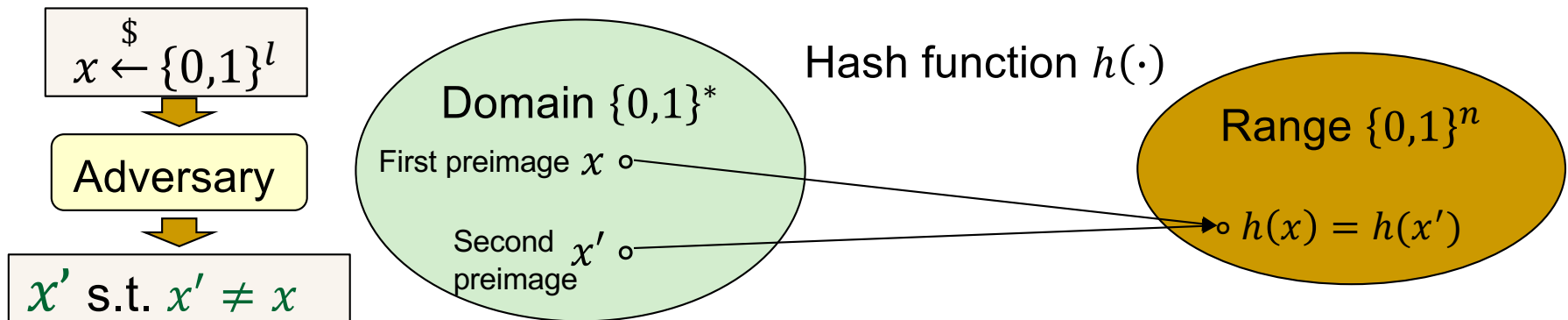


# We (mostly) focus on keyless hash...

- Although there are no CRHFs
- And theory papers focus on keyed hash
- But...
  - It's a bit less complicated and easier to work with.
  - No need to consider both ACR (this stands for Any Collision Resistance, which is the same as CRHF) and TCR
    - Why?
  - Modifying to CRHF is quite trivial
    - Just make it keyed!
  - Usually used in practice: libraries, standards, ...

# 2<sup>nd</sup>-Preimage-Resistant Hash (SPR)

- Hard to find collision with a specific random  $x$ .

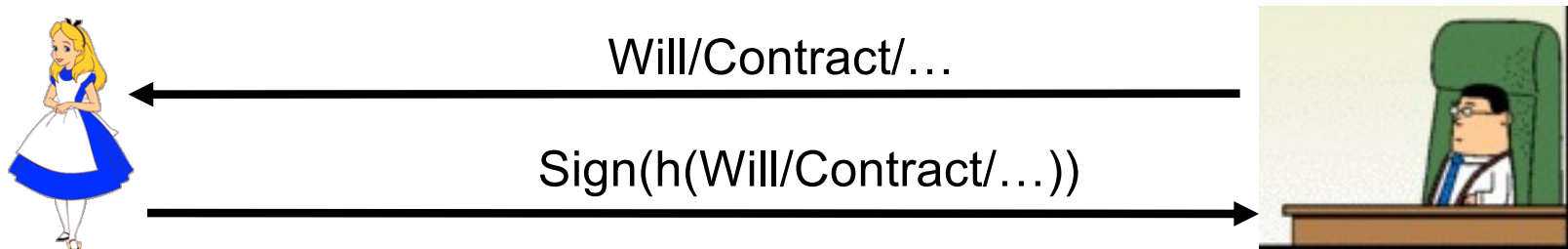


$$\epsilon_{h,\mathcal{A}}^{SPR}(n) \equiv \Pr_{x \leftarrow \{0,1\}^{A(1^n)}} [x' \leftarrow \mathcal{A}(x) \text{ s.t. } x \neq x' \wedge h(x) = h(x')]$$

Use with care!  
(think carefully about the security you want to achieve and see if SPR suffices)

# CRHF/SPR vs. Applications

- CRHF secure for signing, SW-distribution
- How about SPR hash (weak-CRHF)?
  - ❑ SW-distribution? **YES**
  - ❑ Hash-then-sign? **NO**
- Why?
  - ❑ Attacker can't impact SW to be distributed
  - ❑ But... attacker may be able to impact signed msg!





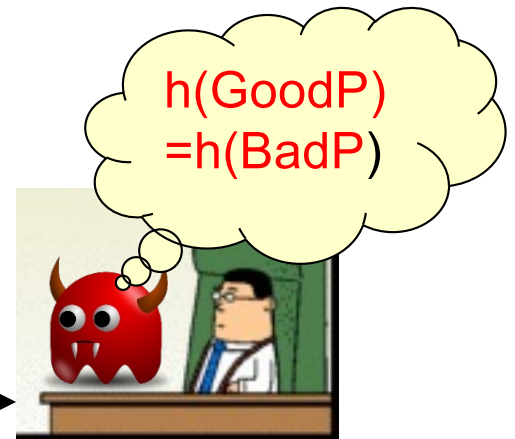
# SPR: Collisions to Chosen Messages

- Or: Alice and Mal, the corrupt lawyer
- Mal finds two `colliding wills', GoodW and BadW:
  - ❑ GoodW: contents agreeable to Alice
  - ❑  $h(\text{GoodW}) = h(\text{BadW})$
  - ❑ Alice Signs good will:  $\text{Sign}_A(h(\text{GoodW}))$



GoodW: 'I leave all to Bob'

$\text{Sign}_A(h(\text{GoodW}))$

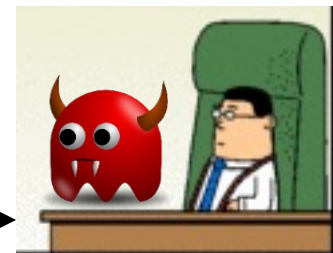


- Later... Mal presents to the court:



BadW: 'I leave all to Mal',  $\text{Sign}_A(h(\text{BadW}))$

\$\$\$\$



# SPR: collisions to **chosen** message

- Or: Alice and Mal, the corrupt lawyer
- Mal finds two `colliding wills', GoodW and BadW:
  - GoodW: contents agreeable to Alice

Is such attack realistic?  
Or SPR is enough 'in practice'?



# SPR & Chosen-prefix Vulnerability

- Chosen-prefix vulnerability:
  - Mal selects 'prefix string'  $p$
  - Efficient algorithm finds:
$$x \neq x' \text{ s.t. } h(p||x) = h(p||x')$$
  - Or, also for any suffix:  $(\forall s) h(p||x||s) = h(p||x'||s)$
- Hash may be SPR yet allow chosen-prefix attacks
- Such attacks found for several proposed, standard cryptographic hash function, e.g., MD5 and SHA1
- We show chosen prefix attack on HtS
  - Example of possible attack on HtS with SPR

# Chosen-prefix Attack

- Let  $x < x'$  be collision for prefix:  $p = \text{'Pay Mal \$'}$
- Mal tricks Alice into signing him an IOU for  $\$x$
- Alice signs, sends  $s = S_s^h(m)$  where  $m = \text{'Pay Mal \$'} \parallel x$ 
  - $S_s^h(m) = S_s(h(p \parallel x)) = S_s(h(p \parallel x')) = S_s^h(m')$
  - $m' = \text{'Pay Mal \$'} \parallel x'$
- Mal sends  $s, m'$  to Alice's bank
  - Bank validates " $Ok$ " =  $Verify_{Alice.v}(m', s)$
- Bank gives  $\$x'$  of Alice to Mal!!
- This attack is simplified:
  - Mal has to find 'good' collision (high profit, convince Alice to sign)
  - People sign (PDF) files, not plain text...
- In reality, attacker also chooses suffix → stronger attack

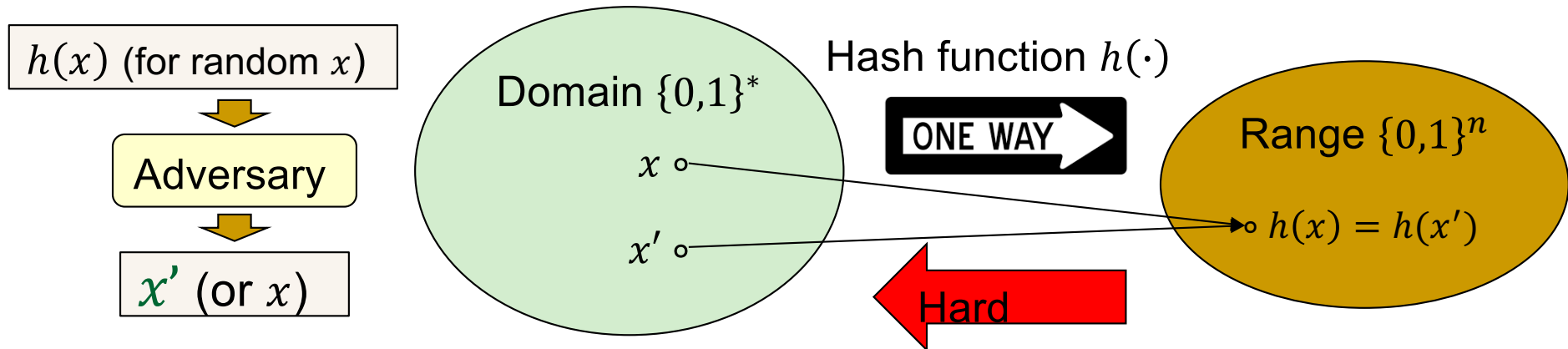
# Examples

- 1) Let  $h_k$  be a keyed CRHF. Is  $h_k' = h_k(h_k(x))$  a CRHF? Why?
- 2) For  $x$  parsed as  $x = x_1 || x_2 || x_3$ , let  $h(x_1 || x_2 || x_3) = x_1 + x_2 + x_3 \bmod p$ , is  $h$  a CRHF? Why? Is it an SPR? Why?
- 3) Let  $h_k(m)$  be a TCR function. Construct  $h_k'(m) = 0^n$  if  $m[1: |k|] = k$  and  $h_k(m)$  otherwise (recall  $m$ , the hash function input, can be of any length).
  - Is  $h_k'$  a CRHF? Why?
  - Is  $h_k'$  a TCR? Why?





# One-Way Function (OWF)



- ❑ **One-way function or first preimage resistance:** given  $h(x)$  for a random  $x$ , it is hard to find  $x$ , or any  $x'$  s.t.  $h(x')=h(x)$

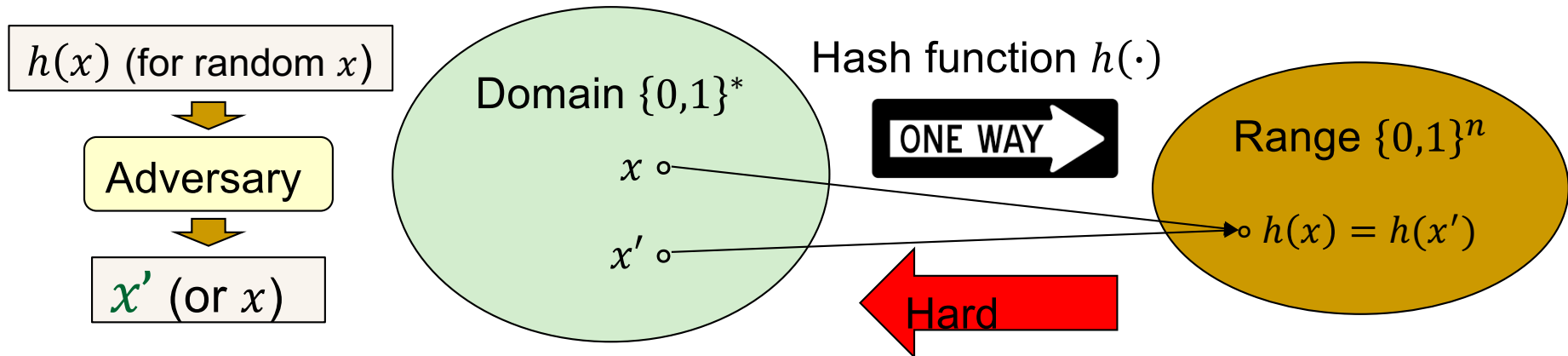
Compare to:

- ❑ **Collision-Resistance (CRHF):** hard to find any collision, i.e., any  $(x, x')$  s.t.  $h(x')=h(x)$ ,  $x \neq x'$
- ❑ **Second-preimage resistance (SPR):** hard to find a collision with random  $x$ , i.e.,  $x'$  s.t.  $h(x')=h(x)$ ,  $x \neq x'$

**Note:** the textbook has a typo in formula 3.17, page 201. The attacker  $A$  takes  $h(x)$  as input not  $x$ .



# Application: One-time Password Authentication



## ❑ One-time password authentication:

- Select random  $x$  : 'one-time password' (keep secret!)
- Validate using non-secret 'one-time validation token':  $h(x)$

*How about a one-time password chain?*

## Not an Application: One-time Password Chain

- Alice computes a hash chain instead of one hash:
  - Select random  $x_0$  then compute a chain of length  $l$  of hashes:  $x_{i+1} = h(x_i)$
  - This allows Alice to authenticate  $l$  times instead of one.
    - Alice gives the server  $x_l$  then each time she wants to authenticate she sends  $x_{i-1}$
    - The server can check by verifying that  $x_i = h(x_{i-1})$
- A one-way function property alone may not be sufficient,  $h$  has also to be a permutation.
  - $x_i$  need to be uniformly distributed.

## Example

- Let  $h(x)$  be a OWF, construct  $g(x)$  as:
  - $g(x) = 0^{2n}$  if  $x \bmod 2^n = 0$
  - $g(x) = h(x) \parallel 0^n$  otherwise
- $g(x)$  is a OWF.
  - Why?
- But  $f(x) = g(g(x))$  is not a OWF.
  - Why?
- And recall that a one time password chain is a nested calls of the hash function.
  - So  $g(x)$  cannot be used to construct such a chain.
  - Why?



---

# Exercise

- Let  $h_1, h_2$  be both CRHF and OWF
- Use them to construct:
  - $h_{CRHF}$  - CRHF but not OWF
  - $h_{OWF}$  - OWF but not CRHF

# Exercise

- Let  $h_1, h_2$  be both CRHF and OWF

- Use them to construct:

- $h_{CRHF}$  - CRHF but not OWF

- $h_{OWF}$  - OWF but not CRHF

- One possible solution:

- $$h_{CRHF}(m) = \begin{cases} 1 ||m & \text{if } |m| = n \\ 0 ||h_1(m) & \text{otherwise} \end{cases}$$

- $$h_{OWF}(m) = \begin{cases} h_1(m) & \text{if } |m| = n \\ h_1(m_{1..n} \oplus h_2(m')) & \text{if } m = m_{1..n} || m' \end{cases}$$



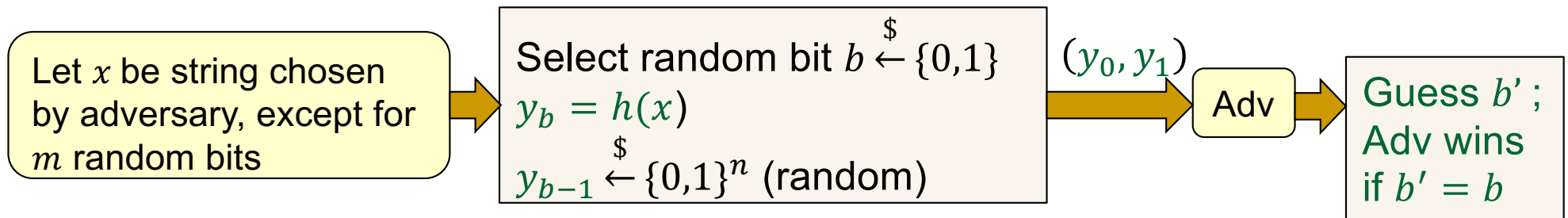
---

# What does all of that tell you about ...

- ... *the relation between CRHF, SPR, and OWF?*
  - CRHF vs. SPR?
  - CRHF vs. OWF?
  - SPR vs. OWF?



# Randomness Extraction



- ❑ 'If input is sufficiently random, then output is random'
- ❑ Multiple 'sufficiently random' models
- ❑ **Randomness extraction:** if any  $m$  input bits are random → all  $n$  output bits are pseudorandom
  - For sufficiently large  $m$
  - **Pseudorandom:** it is not computationally-feasible to distinguish between these bits and truly random bits

# Random Oracle Model (ROM)

- Use a fixed, keyless hash function  $h$
- Analyse security as if hash  $h()$  is a *random function*
  - An invalid assumption:  $h()$  is fixed!
  - Whenever  $h()$  is used, use oracle (black box) for random function
- Good for screening insecure solutions
  - Random oracle security  $\rightarrow$  many attacks fail
- In practice: assume random oracle and use a standard hash function
  - It was shown that in some cases the construction will become insecure.
- Better to have security with standard assumption than the non-standard ROM.

---

# Covered Material From the Textbook

- ❑ Chapter 3
  - ❑ Section 3.1,
  - ❑ Section 3.2 (except 3.2.6 and 3.2.7),
  - ❑ Section 3.3,
  - ❑ Section 3.4 (except 3.4.2),
  - ❑ Section 3.5 (only its introduction, i.e., all text before 3.5.1),
  - ❑ Section 3.6

---

# Thank You!

